RX23W Group

BLE Module Firmware Integration Technology

Introduction

This application note describes the BLE module which uses Firmware Integration Technology (FIT). This module controls Bluetooth® Low Energy communication. In this document, this module is referred to as the BLE FIT module.

Target Device

RX23W Group

Related Documents

- Bluetooth Core Specification (https://www.bluetooth.com)
- RX23W Group User's Manual: Hardware (R01UH0823)
- Firmware Integration Technology User's Manual (R01AN1833)
- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)
- RX Family Renesas FreeRTOS(R01AN4307)
- Renesas e2 studio Smart Configurator User Guide (R20AN0451)
- RX23W Group Tuning procedure of Bluetooth dedicated clock frequency (R01AN4762)
- RX23W Group Bluetooth Low Energy Profile Developer's Guide (R01AN4553)
- Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205)
- RX23W Group Bluetooth Low Energy Application Developer's Guide(R01AN5504)
- Bluetooth Low Energy MCU Bluetooth Test Tool Suite operating instructions(R01AN4554)

Contents

1. Overview

The BLE FIT module provides BLE features which are compliant with Bluetooth Core Specification version 5.0. The BLE FIT module supports the following features.

Bluetooth 5.0 features :

- LE 2M PHY
- LE Coded PHY
- LE Advertising Extensions
- LE Channel Selection Algorithm #2
- High Duty Cycle Non-Connectable Advertising

Bluetooth 4.2 features :

- LE Secure Connections
- Link Layer Privacy
- Link Layer Extended Scanner Filter policies
- LE Data Packet Length Extension

Bluetooth 4.1 features :

- LE L2CAP Connection Oriented Channel Support
- Low Duty Cycle Directed Advertising
- 32-bit UUID Support in LE
- LE Link Layer Topology
- LE Ping

1.1 How to Use

The BLE FIT module is implemented in a project and used as the API. Refer to Section 2.11 - Adding the FIT Module to Your Project - for details on implementing the module to the project.

1.2 Software Structure

Figure 1.1 shows the software structure of the BLE FIT module.
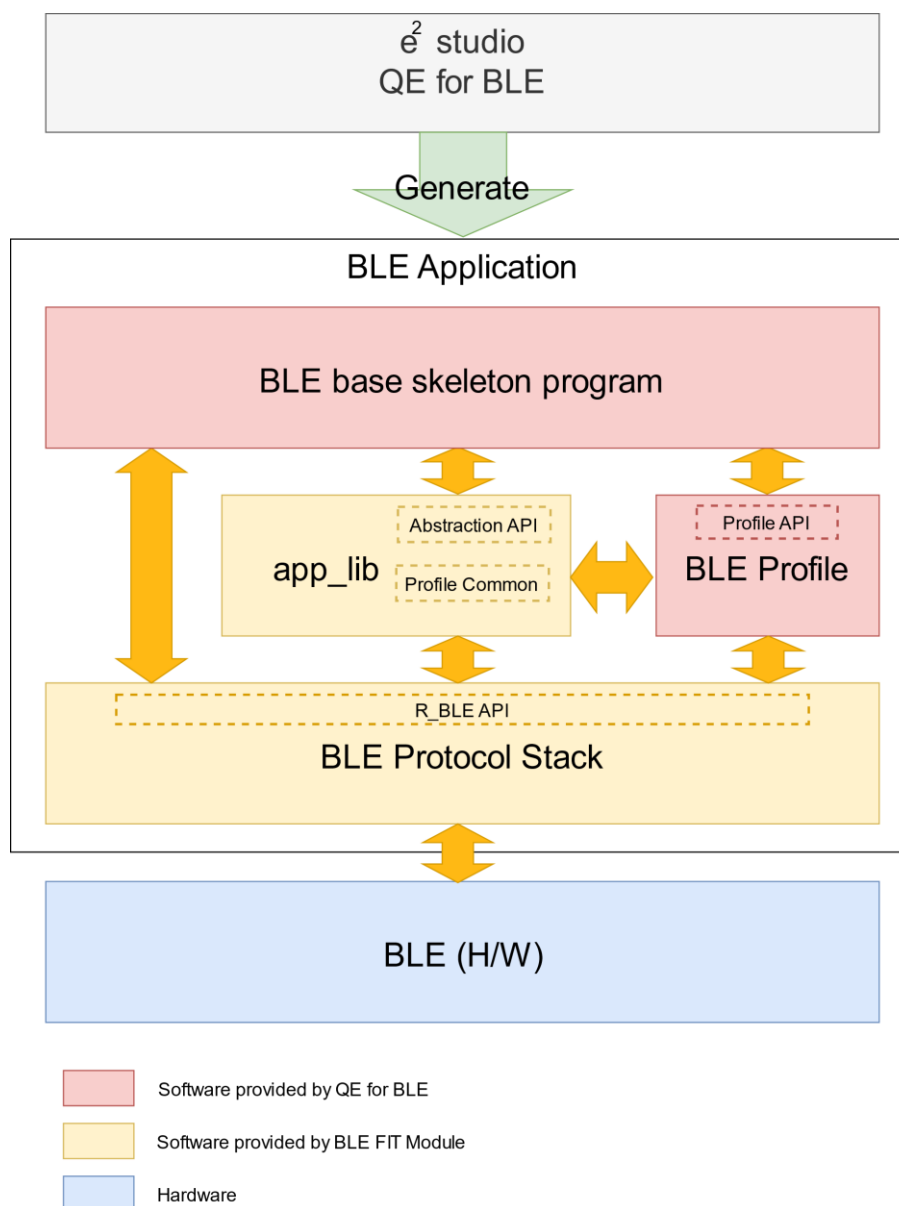
Figure 1.1 : Software structure

The BLE FIT module consists of the BLE Protocol Stack and app_lib.

The BLE Application uses the BLE functions via the R_BLE API provided by the BLE Protocol Stack.

The app_lib includes optional functions available for the BLE Application. The abstraction API of the R_BLE API also is an interface for the BLE functions.

The QE for BLE generates the source codes (BLE base skeleton program) as a base for the BLE Application and the BLE Profile. Renesas recommends using the QE for BLE for the development of the BLE Application.

1.3   Directory / File structure

Table 1.1 shows the directory / file structure of the BLE FIT module.

Table 1.1 : Directory / File structure

Directory /File structure				Description
r_ble_rx23w	doc	en	r01an4860ej0220-rx23w-ble.pdf	Application Note(English)
		jp	r01an4860jj0220-rx23w-ble.pdf	Application Note(Japanese)
		r_ble_api_spec.chm		R_BLE API document
	lib	ble_fit_lib_selector.bat		Library selector
		lib_ble_ps_ccrx_a.lib		BLE Protocol Stack(ALL Features)
		lib_ble_ps_ccrx_b.lib		BLE Protocol Stack(Balance)
		lib_ble_ps_ccrx_c.lib		BLE Protocol Stack(Compact)
		lib_ble_ps_hci_ccrx_a.lib		HCI mode library (ALL Features)
		lib_ble_ps_hci_ccrx_b.lib		HCI mode library (Balance)
		lib_ble_ps_hci_ccrx_c.lib		HCI mode library (Compact)
	ref	r_ble_rx23w_config_reference.h		Configuration reference file
	src	app_lib	abs	Abstraction API(GAP)
			board	Control the LED and SW on the board
			cli	Command Line(input/output)
			cmd	Command Line(command)
			discovery	Abstraction API(GATT)
			logger	Logger
			profile_cmn	Profile common
			rtos	BLE task control
			sec_data	Security data management
			timer	Software timer
		platform	driver dataflash	Data Flash driver for BLE
			r_ble_pf_config_private.h	BLE configuration control
			r_ble_pf_configs.c	
			r_ble_pf_functions.c	RF driver dependent on platform
			r_ble_pf_lowpower.c	Low power consumption program
	r_ble_rx23w_if.h			BLE interface file
	readme.txt			readme
r_config	r_ble_rx23w_config.h			Configuration option file

2. API Information

The BLE FIT module API follows the Renesas API naming standards.

2.1 Hardware Requirements

The MCU used must support the following functions:

- BLE

2.2 Software Requirements

The BLE FIT module is dependent upon the following FIT modules:

- Renesas Board Support Package(r_bsp)
  If you use the BLE FIT module ver 1.01 or later, use BSP version 5.40 or later.
  If you select R5F523W8CxLN or R5F523W8DxLN, use BSP version 5.62 or later.
- CMT(r_cmt_rx, version 4.10 later)
  The BLE Protocol Stack uses CMT2, CMT3. If the CMT FIT module version is prior to 4.50, modify the CMT code (Refer "4.6.1 Changing CMT for BLE"). If the version is 4.50 or later, you do not need change the CMT code. When the software timer(app_lib/timer) is in use, it uses another one CMT channel.
- LPC(r_lpc_rx)

Also, the following FIT modules are required according to the configuration option of the BLE FIT module.

Command Line Interface(BLE_CFG_CMD_LINE_EN=1):

- SCI (r_sci_rx)
- byte queues/circular buffers(r_byteq_rx)

Security Data Management(BLE_CFG_EN_SEC_DATA=1) or
device-specific data (data area) (BLE_CFG_DEV_DATA_DF_BLOCK=0 – 7):

- Data Flash (r_flash_rx)

LED and Switch control (BLE_CFG_BOARD_LED_SW_EN=1):

- GPIO (r_gpio_rx)
  If you select R5F523W8CxLN or R5F523W8DxLN, use GPIO version 3.80 or later.
- IRQ (r_irq_rx)

2.3   Supported Toolchain

The BLE FIT module has been confirmed to work with the toolchain listed in "9.1 Confirmed Operation Environment".

2.4   Header Files

All API calls and their supporting interface definitions are located in r_ble_rx23w_if.h.

2.5   Integer Types

This project uses ANSI C99 "Exact width integer types" in order to make the code clearer and more portable. These types are defined in stdint.h.

2.6 Compile Configuration

The configuration options of the BLE FIT module are located in the r_ble_rx23w_config.h. The options can be configured in Software component configuration screen in case of using Smart Configurator. The changed options are automatically reflected when adding the BLE FIT module to the project. The option names and setting values are listed in the table below:

Table 2.1 : Configuration options

Configuration options (r_ble_rx23w_config.h)	

BLE_CFG_LIB_TYPE
Default : "0"
	Type of the BLE Protocol Stack. Select one of the followings.
	    0: All features
	    1: Balance
	    2: Compact
	
	Refer to "2.7 Feature of BLE Protocol Stack " for details.

BLE_CFG_RF_DBG_PUB_ADDR Default : "{0xFF,0xFF,0xFF,0x50,0x90,0x74}"	Initial Public Address. If the public addresses in the Code Flash and the Data Flash are all 0x00 or 0xFF, the device adopts this public address. If all 0x00 or 0xFF is set, the device uses 74:90:50:FF:FF:FF as public address.
BLE_CFG_RF_DBG_RAND_ADDR Default : "{0xFF,0xFF,0xFF,0xFF,0xFF,0xFF}"	Initial Static Address. If the static addresses in the Code Flash and the Data Flash are all 0x00 or 0xFF, the device adopts this static address. If all 0x00 or 0xFF is set, the device uses the value generated with the device specific value the static address.
BLE_CFG_RF_CONN_MAX Default : "7"	Maximum number of simultaneous connections. Range : 1 to 7
BLE_CFG_RF_CONN_DATA_MAX Default : "251"	Maximum packet data length (bytes). Range : 27 to 251
BLE_CFG_RF_ADV_DATA_MAX Default : "1650"	Maximum advertising data length (bytes). Range : 31 to 1650 The maximum advertising data length of the BLE Protocol Stack libraries other than "All features" is fixed to 31bytes.
BLE_CFG_RF_ADV_SET_MAX Default : "4"	Maximum number of the advertising set. Range : 1 to 4 The number of the advertising set of the BLE Protocol Stack libraries other than "All features" is fixed to one.
BLE_CFG_RF_SYNC_SET_MAX Default : "2"	Maximum number of periodic sync set. Range : 1 to 2 If the BLE Protocol Stack library is other than "All features", this option is not used.

Configuration options (r_ble_rx23w_config.h)	

BLE_CFG_EVENT_NOTIFY_CONN_START
Default : "0"
	Enable or disable start interrupt notification of a connection complete event.
	
	  0: Disable
	  1: Enable
	
	Because the start notification is triggered by the interrupt, it occurs after the actual RF event.

BLE_CFG_EVENT_NOTIFY_CONN_CLOSE
Default : "0"
	Enable or disable end interrupt notification of a connection complete event.
	
	  0: Disable
	  1: Enable

BLE_CFG_EVENT_NOTIFY_ADV_START
Default : "0"
	Enable or disable the advertising event start interrupt notification.
	  0: Disable
	  1: Enable
	
	The notification occurs at the following timings.
	- Start Primary Advertising channel.
	- Start Secondary Advertising Channel
	- Start Periodic Advertising. (When the Extended Advertising is enabled.)
	
	Because the start notification is triggered by the interrupt, it occurs after the actual RF event.

BLE_CFG_EVENT_NOTIFY_ADV_CLOSE
Default : "0"
	Enable or disable the advertising event complete interrupt notification.
	  0: Disable
	  1: Enable
	
	The notification occurs at the following timings.
	- Complete Primary Advertising channel.
	- Complete Secondary Advertising Channel
	- Complete Periodic Advertising. (When the Extended Advertising is enabled.)
	
	If the advertising is terminated by a command, the notification doesn't occur.

BLE_CFG_EVENT_NOTIFY_SCAN_START
Default : "0"
	Enable or disable the scan start interrupt notification.
	
	  0: Disable
	  1: Enable
	
	If the scan interval is equal to the scan window, this notification doesn't occur.
	Because the start notification is triggered by the interrupt, it occurs after the actual RF event.

Configuration options (r_ble_rx23w_config.h)	

BLE_CFG_EVENT_NOTIFY_SCAN_CLOSE
Default : "0"
	Enable or disable the scan complete interrupt notification
	
	   0: Disable
	   1: Enable
	
	If the scan interval is equal to the scan window, this notification doesn't occur.
	If the scan is terminated by a command, the notification doesn't occur.

BLE_CFG_EVENT_NOTIFY_INIT_START
Default : "0"
	Enable or disable the notification that the scan start interrupt has occurred in sending a connection request.
	
	   0: Disable
	   1: Enable
	
	If the scan interval is equal to the scan window, this notification doesn't occur.
	Because the start notification is triggered by the interrupt, it occurs after the actual RF event.

BLE_CFG_EVENT_NOTIFY_INIT_CLOSE
Default : "0"
	Enable or disable the notification that the scan complete interrupt has occurred in sending a connection request.
	
	   0: Disable
	   1: Enable
	
	If the scan interval is equal to the scan window, this notification doesn't occur.
	If the connection request is terminated by a command, the notification doesn't occur.

BLE_CFG_EVENT_NOTIFY_DS_START
Default : "0"
	Enable or disable the RF_DEEP_SLEEP start notification.
	   0: Disable
	   1: Enable

BLE_CFG_EVENT_NOTIFY_DS_WAKEUP
Default : "0"
	Enable or disable the RF_DEEP_SLEEP wakeup notification.
	
	   0: Disable
	   1: Enable

BLE_CFG_RF_CLVAL Default : "6"	Adjustment value of the 32MHz crystal oscillator. Set this option according to the board environment. Range : 0 to 15 Refer to"RX23W Group Tuning procedure of Bluetooth dedicated clock frequency(R01AN4762)" for details. If you select R5F523W8CxLN or R5F523W8DxLN, set "7".

BLE_CFG_RF_DDC_EN
Default : "0"
	Enable or disable the DC-DC on the RF.
	
	   0: Disable
	   1: Enable
	
	If you use Target Board, set "0".

Configuration options (r_ble_rx23w_config.h)	

BLE_CFG_RF_EXT32K_EN
Default : "0"
	Slow clock source to the RF.
	Range : 0 to 1
	
	  0: RF_LOCO
	  1: External 32.768kHz
	
	If this option is set to 1, the sub clock is required to be enabled in the Smart Configurator clock configuration screen (**Figure 4.11**).

BLE_CFG_RF_MCU_CLKOUT_PORT
Default : "0
	Port of the MCU CLKOUT.
	Range : 0 to 1
	
	  0: PE3
	  1: PE4
	
	If the BLE_CFG_RF_EXT32K_EN option is 0, this option is ignored.

BLE_CFG_RF_MCU_CLKOUT_FREQ
Default : "0"
	Output frequency from the MCU CLKOUT.
	Range : 0 to 1
	
	  0: MCU CLKOUT frequency 32.768kHz
	  1: MCU CLKOUT frequency 16.384kHz
	
	If the BLE_CFG_RF_EXT32K_EN option is 0, this option is ignored.

BLE_CFG_RF_SCA Default : "250"	Sleep Clock Accuracy(SCA) for the RF slow clock. Range : 0 to 500 If the BLE_CFG_RF_EXT32K_EN option is 0, the SCA is fixed to more than 250ppm and this option is ignored.

BLE_CFG_RF_MAX_TX_POW
Default : "1"
	Maximum transmit power configuration.
	Range : 0 to 1
	
	  0: max +0dBm
	  1: max +4dBm

BLE_CFG_RF_DEF_TX_POW Default : "0"	Default transmit power level. Range : 0 to 2 This option depends on the BLE_CFG_RF_MAX_TX_POW option. If the BLE_CFG_RF_MAX_TX_POW option is 0(0dBm), the BLE_CFG_RF_DEF_TX_POW is as follows. 0(High) : 0dBm 1(Mid) : 0dBm 2(Low) : -18dBm If the BLE_CFG_RF_MAX_TX_POW option is 1(+4dBm), the BLE_CFG_RF_DEF_TX_POW is as follows. 0(High) : +4dBm 1(Mid) : 0dBm 2(Low) : -20dBm
Configuration options (r_ble_rx23w_config.h)	

Configuration options (r_ble_rx23w_config.h)	

BLE_CFG_RF_CLKOUT_EN
Default : "0"
	CLKOUT_RF output.
	Select one of the followings.
	
	  0: No output
	  5: 4MHz output
	  6: 2MHz output
	  7: 1MHz output

BLE_CFG_RF_DEEP_SLEEP_EN
Default : "1"
	Enable or disable the RF Deep Sleep.
	  0: Disable
	  1: Enable

BLE_CFG_MCU_MAIN_CLK_KHZ Default : "4000"	MCU main clock frequency (kHz). This option needs to be configured according to the board environment. Set the clock frequency configured in the Smart Configurator clock configuration. If the HOCO is used, this option is ignored. If the Main Clock is used, set a value within the range between 1000 and 20000. If the PLL Circuit is used, set a value within the range between 4000 and 12500. Set the frequency input in Smart Configurator Clock configuration.
BLE_CFG_DEV_DATA_CF_BLOCK Default : "16"	The Code Flash(ROM) block stored the device specific data. Range : -1 to 255 If this option is set to -1, the device specific data in the Code Flash isn't used. The blocks from "0" to "15" are the Start-Up Program Protection block. If the Start-Up Program Protection is used, don't use the blocks from "0" to "15".
BLE_CFG_DEV_DATA_DF_BLOCK Default : "-1"	The E2 Data Flash block stored the device specific data. Range : -1 to 7 If this option is set to -1, the device specific data in the E2 Data Flash isn't used. Set the block number different from the one specified by BLE_CFG_SECD_DATA_DF_BLOCK.
BLE_CFG_GATT_MTU_SIZE Default : "247"	The MTU size (bytes) for the GATT communication. Range : 23 to 247 The Profile Common responds to the MTU request from the remote device to BLE_CFG_GATT_MTU_SIZE.
BLE_CFG_NUM_BOND Default : "7"	Maximum number of the bonding information stored in the Data Flash. Range : 1 to 7 When you change this value during development, after writing the firmware, delete the bonding information by R_BLE_GAP_DeleteBondInfo() or "gap auth del remote all" command.

Configuration options (r_ble_rx23w_config.h)	
BLE_CFG_EN_SEC_DATA Default : "0"	Enable or disable the security data management. The bonding information is stored in the Data Flash block specified by "BLE_CFG_SECD_DATA_DF_BLOCK" by this option. 0: Disable 1: Enable If this option is enabled, add the Data Flash FIT module.
BLE_CFG_SECD_DATA_DF_BLOCK Default : "0"	The Data Flash block for the security data management to store the bonding information. Range : 0 to 7 Set the block number different from the one specified by BLE_CFG_DEV_DATA_DF_BLOCK.
BLE_CFG_CMD_LINE_EN Default : "0"	Enable or disable the command line function. 0: Disable 1: Enable If this option is enabled, add the SCI FIT module.
BLE_CFG_CMD_LINE_CH Default : "1"	SCI Channel for the command line function. Enable the SCI channel for the command line in the SCI FIT module configuration. If the BLE_CFG_CMD_LINE_EN is 0, this option is ignored.
BLE_CFG_BOARD_LED_SW_EN Default : "0"	Enable or disable support the board LED & Switch control. 0: Disable 1: Enable If the option is enabled, add the IRQ FIT module and the GPIO FIT module.
BLE_CFG_BOARD_TYPE Default : "0"	Board type. Range : 0 to 3 0 : Customer board 1 : Target Board 2 : RSSK 3: Evaluation board
BLE_CFG_LOG_LEVEL Default : "3"	Log level. Range : 0 to 3 0 : disable 1 : Error 2 : Error & Warning 3 : Error & Warning & Debug

Configuration options (r_ble_rx23w_config.h)	

BLE_CFG_ABS_API_EN
Default : "1"
	Enable or disable support the Abstraction API.
	
	  0: Disable
	  1: Enable

BLE_CFG_SOFT_TIMER_EN
Default : "1"
	Enable or disable support the software time in app_lib.
	
	  0: Disable
	  1: Enable
	
	If you use the Abstraction API, enable the software timer.

BLE_CFG_MCU_LPC_EN
Default : "1"
	Enable or disable support the MCU low power consumption control.
	
	  0: Disable
	  1: Enable

BLE_CFG_HCI_MODE_EN
Default : "0"
	Select start in HCI mode or not.
	
	  0: Not start in HCI mode
	  1: Start in HCI mode

2.7 Feature of BLE Protocol Stack

The features of the BLE Protocol Stack depends on the BLE_CFG_LIB_TYPE option (r_ble_rx23w_config.h). Table 2.2 shows the features of each type of the BLE Protocol Stack.

See "Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205) 3.1.2 Library of BLE Protocol Stack" for the details of each BLE Features.

Table 2.2 : Features of the BLE Protocol Stack

Bluetooth LE Feature	BLE Protocol Stack		
	All features	Balance	Compact
LE 2M PHY	Yes	Yes	No
LE Coded PHY	Yes	Yes	No
LE Advertising Extensions	Yes	No	No
LE Channel Selection Algorithm #2	Yes	Yes	No
High Duty Cycle Non-Connectable Advertising	Yes	Yes	Yes
LE Secure Connections	Yes	Yes	Yes
Link Layer privacy	Yes	Yes	Yes
Link Layer Extended Scanner Filter policies	Yes	Yes	No
LE Data Packet Length Extension	Yes	Yes	Yes
LE L2CAP Connection Oriented Channel Support	Yes	No	No
Low Duty Cycle Directed Advertising	Yes	Yes	Yes
LE Link Layer Topology	Yes	Yes	No
LE Ping	Yes	Yes	Yes
GAP Role	Central Peripheral Observer Broadcaster	Central Peripheral Observer Broadcaster	Peripheral Broadcaster
GATT Role	Sever Client	Sever Client	Sever Client
32-bit UUID Support in LE	Yes	Yes	Yes

2.8   app_lib

The app_lib provides optional functions available for the BLE Application. The functions included in the app_lib are as follows.

2.8.1   Abstraction API

Abstraction API simplifies the procedure used with the BLE Protocol Stack. Refer to "R_BLE API document (r_ble_api_spec.chm)" for details. If the Abstraction API is used, set the BLE_CFG_ABS_API_EN option and the BLE_CFG_SOFT_TIMER_EN option to "1".

2.8.2   Software timer

The bare metal version and the Renesas FreeRTOS version of software timer vary the implementation.

- Bare metal
  Software timer uses the CMT. If the software timer is used, add the CMT FIT module to the application. The CMT channel is dynamically allocated by the CMT FIT module.

- Renesas FreeRTOS
  Software timer uses the FreeRTOS Timer API.

If the Software timer is used, set the BLE_CFG_SOFT_TIMER_EN option to "1".

2.8.3   Security data management

Security data management provides the interfaces which are used for storing or getting the bonding information in the Data Flash. If the security data management is used, add the Flash FIT module and set the BLE_CFG_EN_SEC_DATA option to "1". The BLE_CFG_SECD_DATA_DF_BLOCK option indicates the Data Flash block for the security data management.

2.8.4   Profile common

This function provides the common interfaces in the BLE Profile. The interfaces are called by the code generated by the QE for BLE. Refer to "RX23W Group Bluetooth Low Energy Profile Developer's Guide(R01AN4553)" for the details of the profile development and the profile common.

2.8.5   Logger

This function outputs the message. The BLE_CFG_LOG_LEVEL option sets the output level. If you use CCRX C++ , the logger function is not supported.

2.8.6   Command Line
Command Line control the BLE functions via a serial interface. This function uses the SCI.

1) Component

To configure the channel for command line, follow these steps:

1.   Add the SCI FIT module in the Smart Configurator and select the channel for the command line.
2.   Set the BLE_CFG_CMD_LINE_EN option to 1.
3.   Set the BLE_CFG_CMD_LINE_CH option to the selected channel number.

2) Terminal

Connect the board to a computer and launch Terminate software on the computer with the following settings.

Table 2.3 : Terminal configuration

Item	Settings
New-line (Receive)	LF
New-line (Transmit)	CR
Terminal Mode	VT100
Baud rate	115200
Data	8bit
Parity	none
Stop bits	1bit
Flow Control	none

3) Command

Table 2.4 presents the command supported by the command line function.

Table 2.4 : Command List

Command	Sub command	Description
gap	adv	Start or stop advertising.
	scan	Start scan.
	conn	Send a connection request.
	disconn	Disconnect the link.
	device	Display the addresses of the connected devices.
	priv	Set the privacy feature to the local device.
	conn_cfg	Link configuration command.
	wl	White List operation command.
	auth	Pairing or encryption command.
	sync	Create or Terminate a periodic sync.
	ver	Display the version information.
vs	txp	Set or Get the transmit power.
	scheme	Set the coding scheme of the Coded PHY
	test	DTM test command.
	addr	Set or Get the local device BD_ADDR.
	rand	Generate a random number.
sys	stby	Operate the software standby mode.
ble	reset	Reset the BLE protocol stack.
	close	Stop the BLE protocol stack.

If you use gap command or vs command, set "1" to BLE_CFG_ABS_API_EN with the BLE FIT module component configuration.

For more details, refer to "Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205) ".

2.8.7   LED and Switch control
The LED and Switch Control feature controls LEDs and switches on board. About setting the BLE FIT and the IRQ FIT configuration options to use this feature, see "4.5.4 LED and Switch control".

2.8.8   BLE task control

The BLE task control is enabled if creating a project, "FreeRTOS" is selected as the "RTOS". This feature is used if another task wakes the BLE task up.

2.9   Flash Memory Protection

The Bluetooth Device Address(BD_ADDR) can be written in the User Area (ROM) or Data Area (E2 Data Flash) of the Flash memory.

The location in which the BD_ADDR is written is specified by BLE_CFG_DEV_DATA_CF_BLOCK or BLE_CFG_DEV_DATA_DF_BLOCK in the configuration option(r_ble_rx23w_config.h).

If the BD_ADDR is written in the User Area (ROM), it is necessary to specify the block which the program code doesn't use. Also, the BD_ADDR needs to be written in the top address of the specified block.

Table 2.5 shows the format of the written data.

Table 2.5 : Data Format

Offset	Size[byte]	Description
0	4	Data length after the magic number(Fixed 16)
4	4	Magic number(Fixed 0x12345678)
8	6	Public BD_ADDR
14	6	Random BD_ADDR

Data must be written in little endian for each block.

Data can be written to multiple RX23Ws continuously by using the unique code function of Renesas Flash Programmer (RFP).

A setting example of the RFP unique code file (ruc) is shown below.
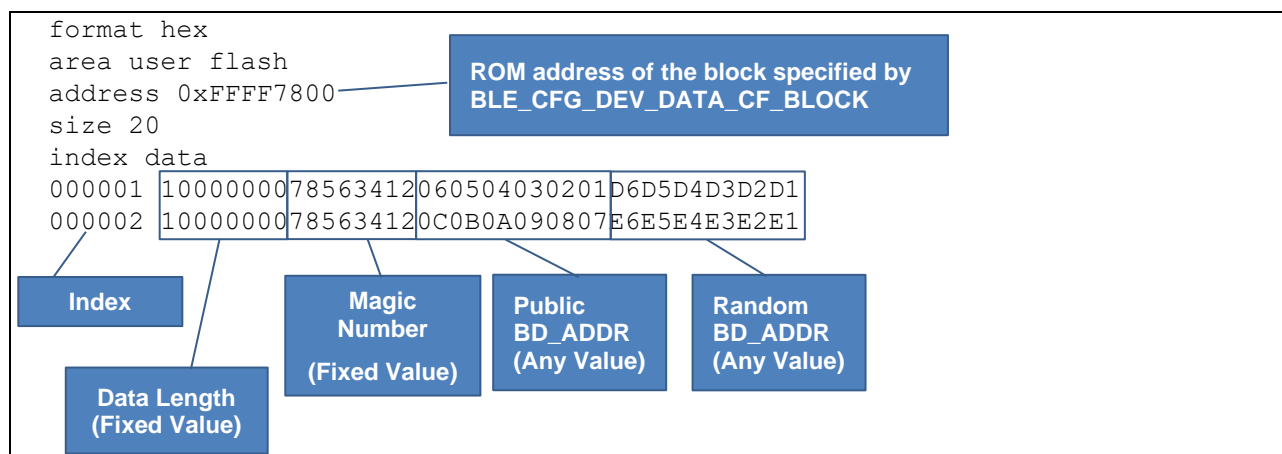
The device searches for the BD_ADDR which are not all zero and not all 0xFF in the following order.

(1)The specified block of the Data Area (E2 Data Flash)

(2)The specified block of the User Area (ROM)

(3)The initial value of the firmware
     (BLE_CFG_RF_DBG_PUB_ADDR or BLE_CFG_RF_DBG_RAND_ADDR)

By default, the flash memory protection of the RX23W is disabled. If the flash memory protection is disabled, all the blocks are erased in using serial programmer connection such as the Renesas Flash Program(RFP) , etc. Therefore, it is necessary to enable the flash memory protection to write a new firmware while remaining the written BD_ADDR in the flash memory. To enable the flash memory protection, configure the r_bsp configuration option and the ID Code following the below sections.

- 4.5.1 BSP(r_bsp)

- 4.8.1 Flash ID Code

2.10 Code Size

Table 2.6 shows the code size of the BLE Protocol Stack of the BLE FIT module. The code size is based on optimization level 2 and optimization type for size for the RXC toolchain in Section 9.1 .

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options set in the BLE FIT module configuration header file. The values in the table below are confirmed under the following conditions.

Table 2.6 : Code Size

Device	Compiler	Category	Type of the BLE Protocol Stack		
			All Features	Balance	Compact
RX23W	CC-RX	ROM	190K bytes	148K bytes	132K bytes
		RAM	38K bytes	23K bytes	23K bytes

Configuration Option

Common :

BLE_CFG_RF_CONN_MAX 7

BLE_CFG_RF_CONN_DATA_MAX 251

BLE_CFG_NUM_BOND 7

All Features :

BLE_CFG_LIB_TYPE 0

BLE_CFG_RF_ADV_DATA_MAX 1650

BLE_CFG_RF_ADV_SET_MAX 4

BLE_CFG_RF_SYNC_SET_MAX 2

Balance :

BLE_CFG_LIB_TYPE 1

Compact :

BLE_CFG_LIB_TYPE 2

NOTE :
- The app_lib and the BLE Profile are not included in the above code size.
- The BLE Protocol Stack libraries which are generated with the CCRX compiler. The IAR development environments uses the same libraries.

2.11 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends using "Smart Configurator" described in (1) or (3).

(1) Adding the FIT module to your project using "Smart Configurator" in e2 studio.
By using the "Smart Configurator" in e2 studio, the FIT module is automatically added to your project.
Refer to "Renesas e2 studio Smart Configurator User Guide (R20AN0451)" for details.

(2) Adding the FIT module to your project using "FIT Configurator" in e2 studio
By using the "FIT Configurator" in e2 studio, the FIT module is automatically added to your project. Refer to "Adding Firmware Integration Technology Modules to Projects (R01AN1723)" for details.

(3) Adding the FIT module to your project using "Smart Configurator" on CS+
By using the "Smart Configurator Standalone version" in CS+, the FIT module is automatically added to your project. Refer to "Renesas e2 studio Smart Configurator User Guide (R20AN0451)" for details.

(4)Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)" for details.

3. R_BLE API functions

Refer to "R_BLE API document (r_ble_api_spec.chm)" for details.

If the r_ble_api_spec.chm cannot be opened, right-click the file, select Properties and check "Unblock" as follows.
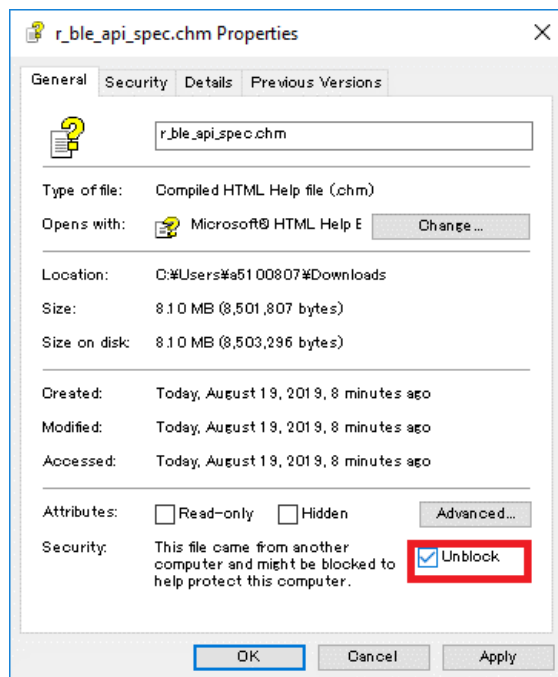

Figure 3.1 : r_ble_api_spec.chm Properties

4. BLE FIT module project

This section describes how to create a new BLE project in the e^{2}studio and the Smart Configurator to build a BLE Application development environment. If you create a project for RX23W module, use e^{2}studio 2021-01 (64bit) or later.

4.1 Create a new project

Select "File"→"C/C++ Project". In "New C/C++ Project" dialog, select "Renesas RX" and "Renesas CC-RX C/C++ Executable Project" and click on the "Next" button.
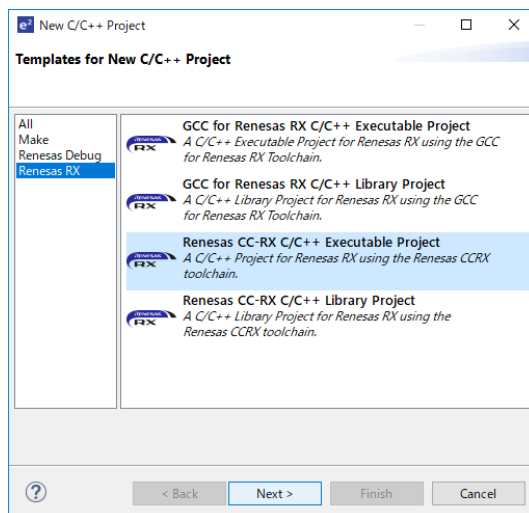
Figure 4.1 : Templates for New C/C++ Project

Enter the project name and click on the "Next" button.
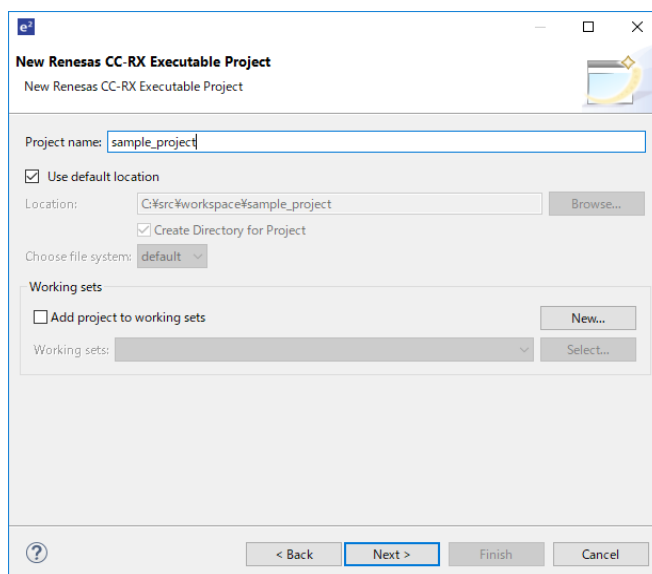
Figure 4.2 : New Renesas CC-RX Executable Project

Set the "Endian" combo box to "Little" and select the RX23W type name from [RX200]→[RX23W] in accordance with the board in the "Target Device".
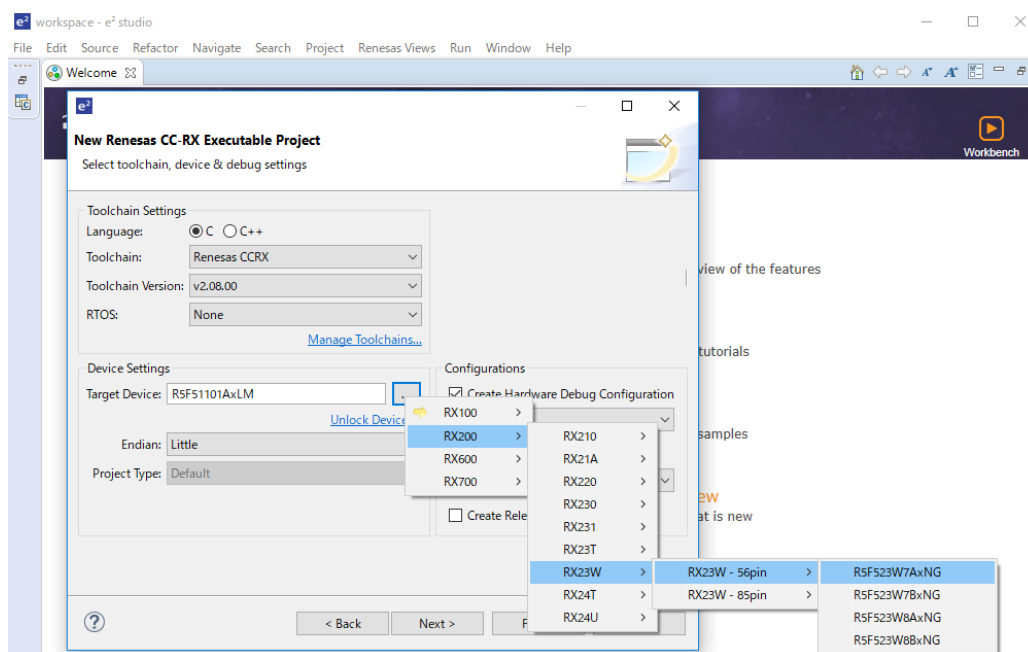
Figure 4.3 : Toolchain, device & debug settings

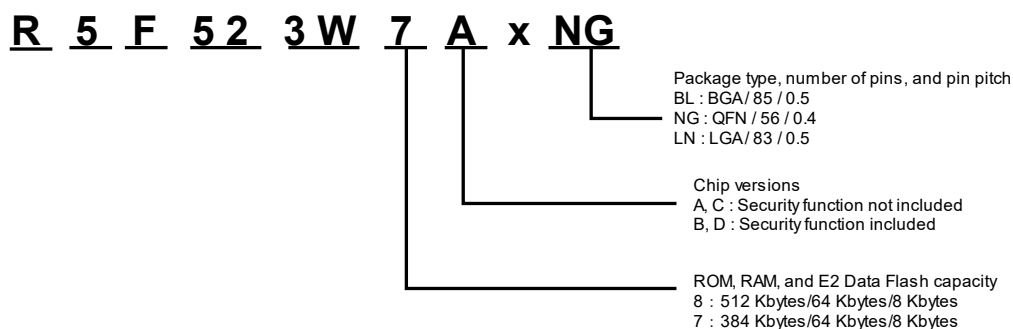Figure 4.4 shows the RX23W Product Part Number.

Figure 4.4 : RX23W Product Part Number

For the RX23W Product Part Number details, see "1.2 List of Products" in "RX23W Group User's Manual: Hardware (R01UH0823)".

If you use the Renesas FreeRTOS, select "FreeRTOS" from the "RTOS". If you don't need RTOS, select "None".

Click "Next" button, the following screen is displayed depending on the "RTOS" item.

- "None"
  "Coding Assistant settings" screen is displayed.
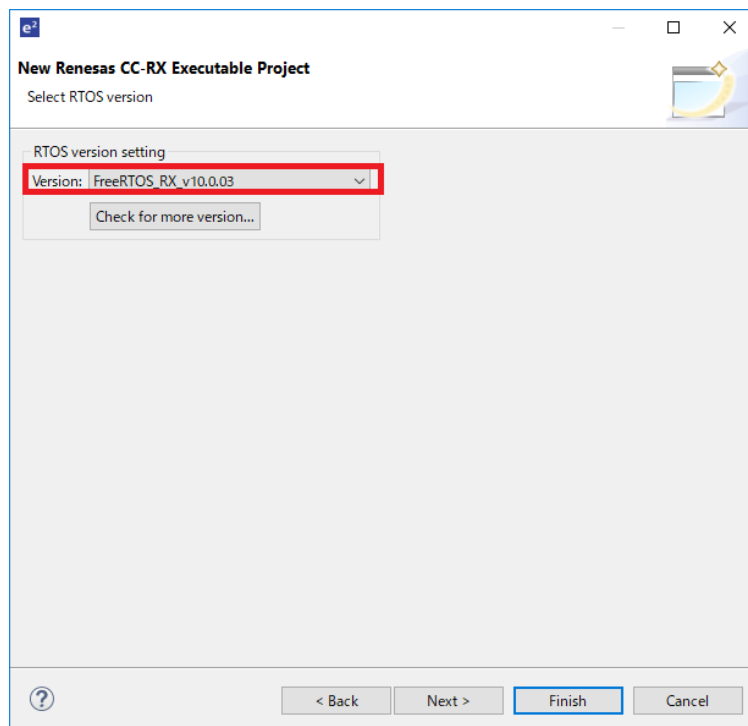
- "FreeRTOS"
  "RTOS version" screen is displayed.

Figure 4.5 : Renesas FreeRTOS version

Set "FreeRTOS_RX_v10.0.0.2" or later to the version. To download the latest Renesas FreeRTOS, click "Check for more version…" and select the latest version.
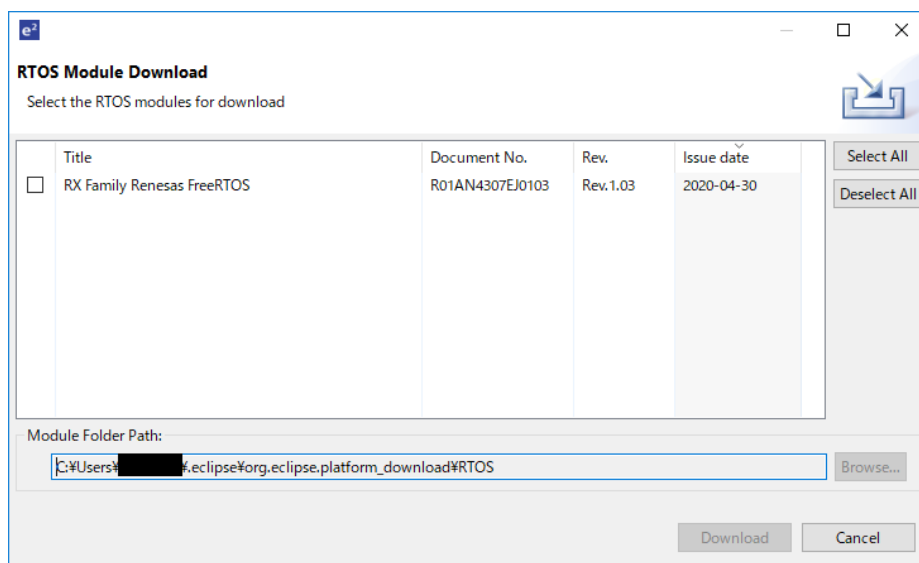

Figure 4.6 : Renesas FreeRTOS Download

Click "Next" button in "Select RTOS version" screen and "Select Coding Assistant settings" dialog is displayed.

Check the "Smart Configurator" in "Select Coding Assistant settings" dialog and click "Finish" button.
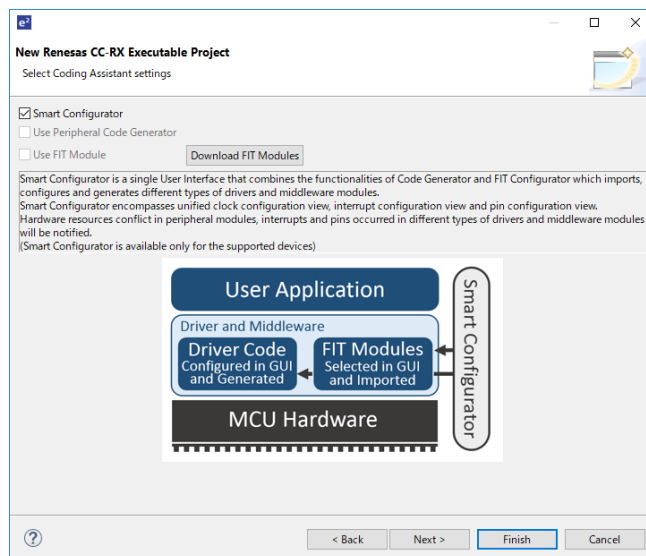

Figure 4.7 : Coding Assistant settings

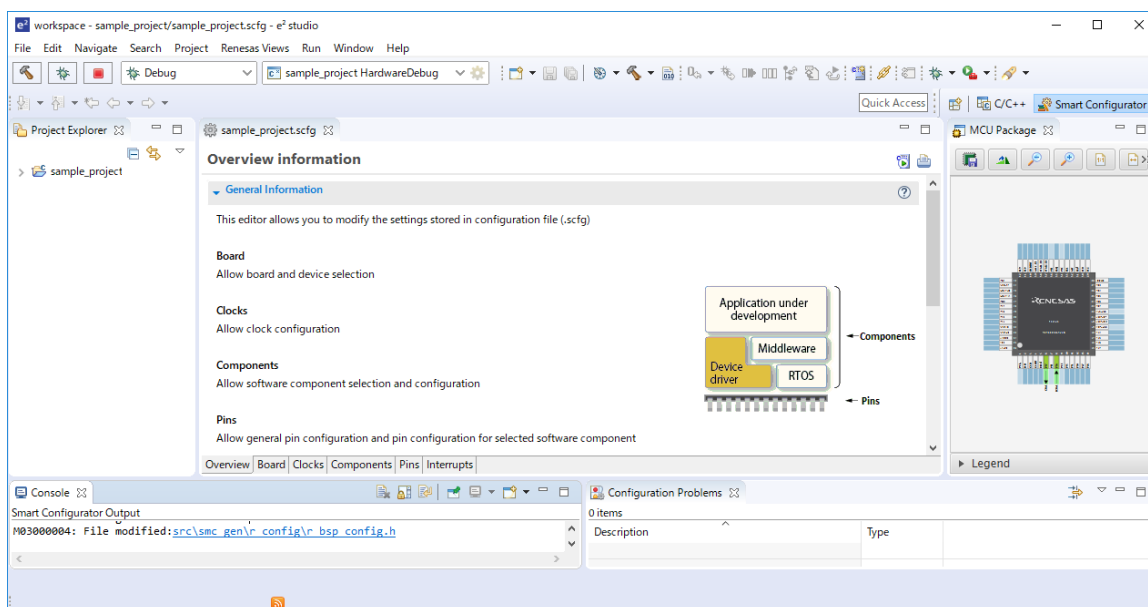After a little while, the project is created.

Figure 4.8 : Project overview

For the details about creating a new e² studio project by the Smart Configurator, see "Renesas e² studio Smart Configurator User Guide (R20AN0451)".

4.2   Confirm BSP version

After creating a project, right-click "r_bsp" on the "Software component configuration" screen and select "Change version".
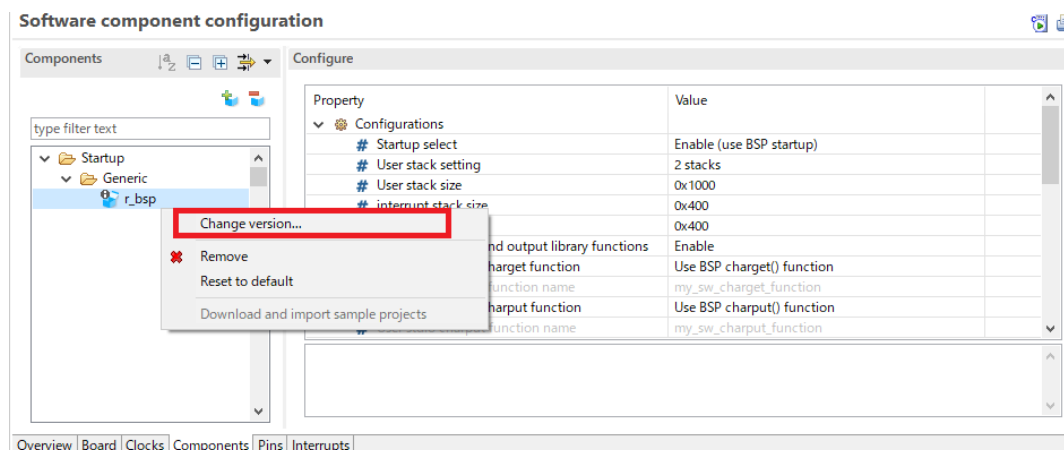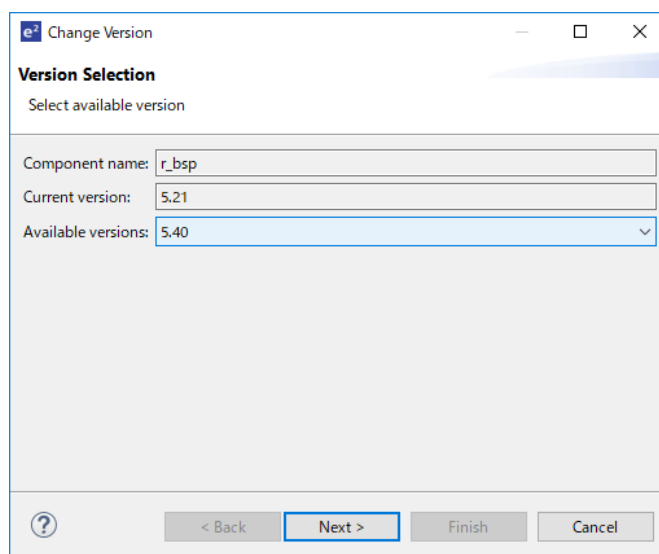

Figure 4.9 : Change version

Figure 4.10 : Change Version dialog

Confirm the "Current version" on the "Change Version" dialog.

If you use the BLE FIT module ver 1.01 or later, the r_bsp needs to be updated to version 5.40 or later.

If the r_bsp update is not made, the following error occurs when the project is built.

../src/smc_gen/r_sci_rx/src/targets/rx23w/r_sci_rx23w_data.c(153):E0520020:Identifier "IR_SCI8_ERI8" is undefined

If you select R5F523W8CxLN or R5F523W8DxLN, use version 5.62 or later.

4.3   Clock configuration

Configure the clock on the "clock" tab in the Smart Configurator.

If the BLE FIT module is used, set the clock within the following range.

- System clock (ICLK) : more than 8MHz
- Peripheral module clock B (PCLKB) : more than 8MHz

The BLE Protocol Stack is optimized with ICLK : 32MHz, PCLK : 32MHz.

Renesas recommends configuring the frequency of ICLK and PCLKB to 32MHz to maximize the BLE performance.

Figure 4.11 : Clock configuration

4.4   Adding FIT modules

Click the scfg file in the project and add the FIT modules on "component" tab in the Smart Configurator.

This section takes the case of using the following functions described in "2.2 Software Requirements" as an example.

- Command Line Interface
- Security Data Management
- LED and Switch control

Click "Add component" button 🔧 , select BLE Profile Creation, r_ble_qe_utility, r_ble_rx23w, r_byteq, r_cmt_rx, r_flash_rx, r_gpio_rx, r_irq_rx, r_lpc_rx, r_sci_rx and click "Finish" button.

NOTE : The QE for BLE refers to the r_ble_qe_utility. Therefore, the BLE Profile Creation is added to after or together with the addition of the r_ble_qe_utility.
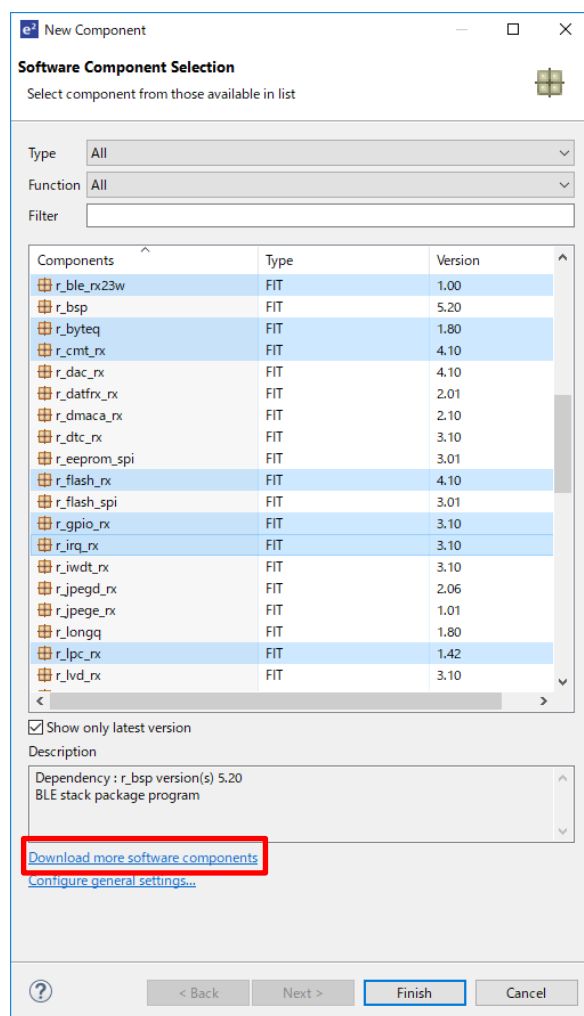

Figure 4.12 : Software Component Selection (e^{2}studio v7.8)

NOTE: When the necessary FIT modules are not found, click [Download more software components] and download them in accordance with the procedure in [FIT Module Download] dialog.
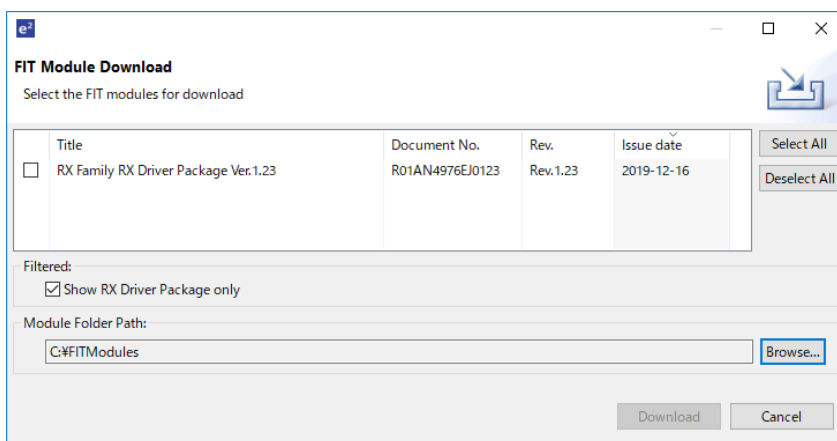

Figure 4.13 : FIT Module Download

NOTE: When downloaded FIT modules are not displayed, change the following item in Smart Configurator.

[e^{2}studio 32bit-version] :
Click [Configure general settings…] and put a check in [Allow blocked FIT modules to be displayed] in [Preferences] dialog.

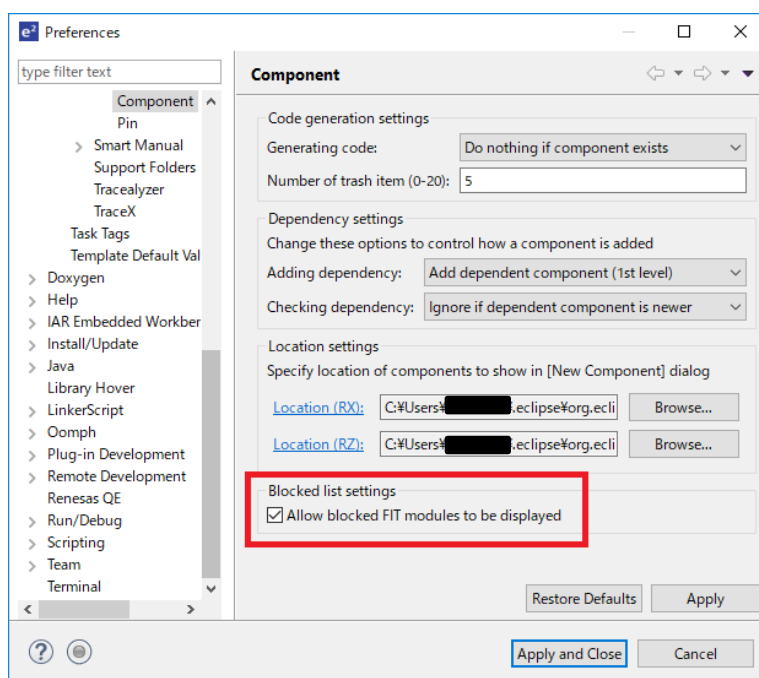

Figure 4.14 : Blocked list settings

[e²studio 64bit-version] :
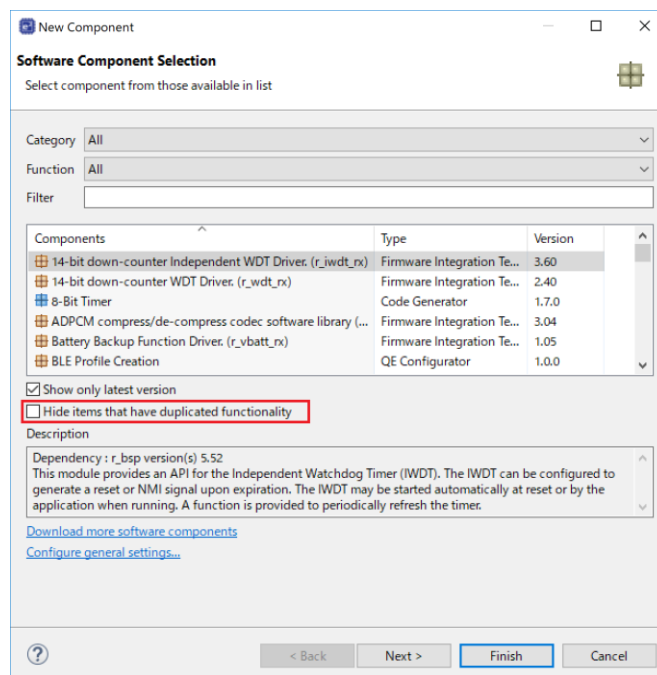Click out [Hide items that have duplicated functionality].

Figure 4.15 : Software Component Selection (e²studio 2021-01)

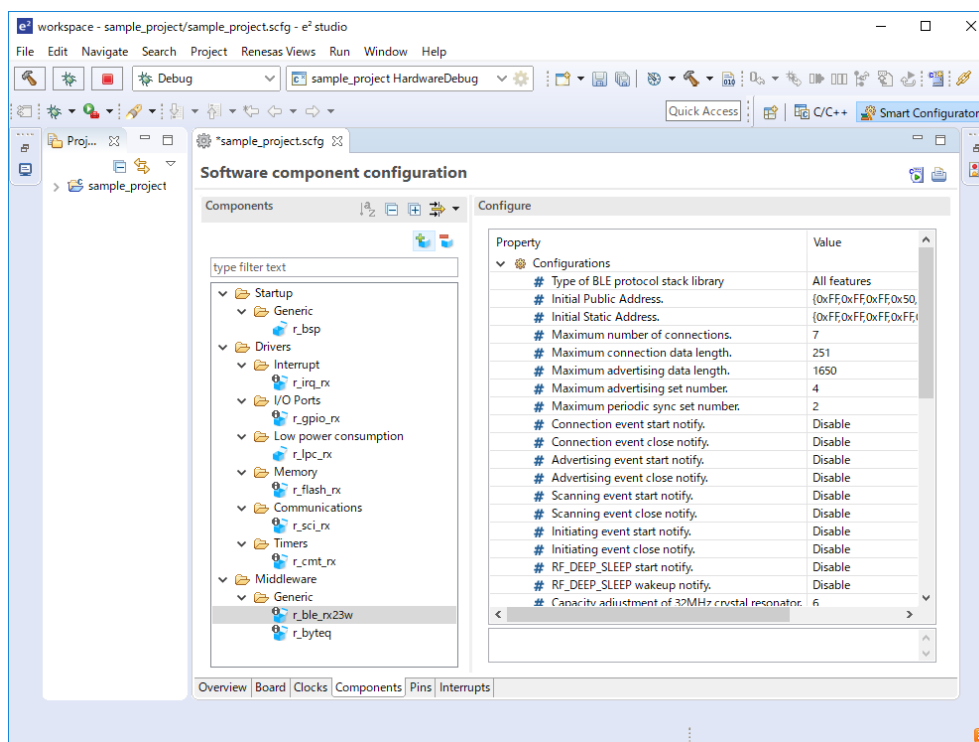After a little while, the selected FIT modules are added.

Figure 4.16 : Software component configuration

4.5 Configuration options

Configure the options according to the BLE features that you want to use after adding the FIT modules.

4.5.1 BSP(r_bsp)

Set the ID Code to enable the Flash Memory Protection described in "2.9 Flash Memory Protection". Select "Components" tab →"r_bsp"→"ID code 1" and set the option to "0x45FFFFFF".
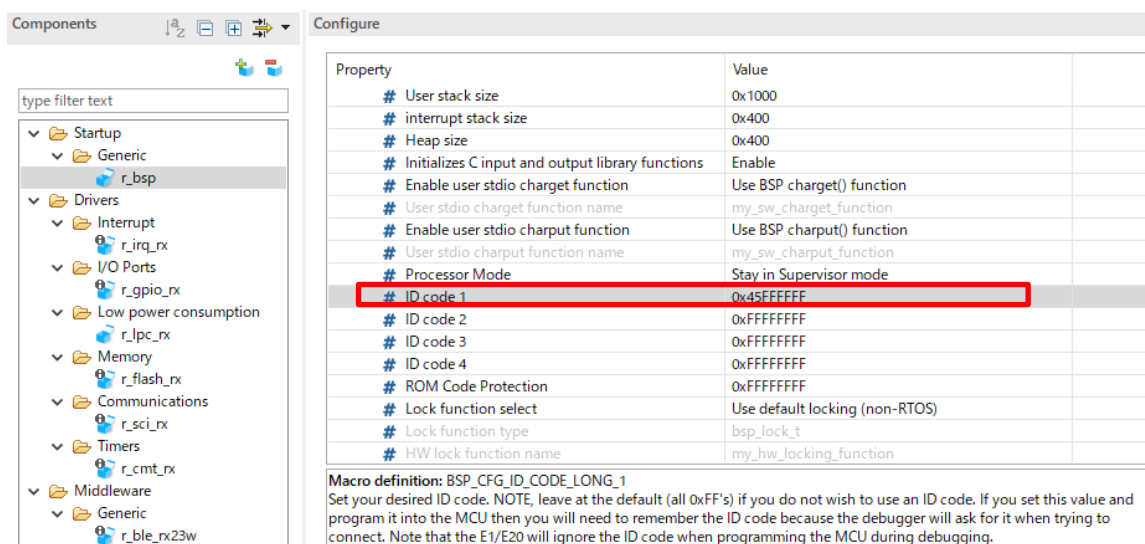

Figure 4.17 : BSP configuration options

4.5.2   Command Line Interface

Set the configuration options for the BLE FIT Module(r_ble_rx23w) and the SCI FIT Modules(r_sci_rx) for the Command Line Interface. The byte queues/circular buffers (r_byteq_rx) need not be set the configuration options.

(1)   BLE(r_ble_rx23w)

Select "Components" tab →"r_ble_rx23w" →"Enabled/Disabled command line function", set the option to "Enabled". Configure the "SCI CH for command line function" option to the SCI channel number for command line interface. In case of the Target Board(TB) and the RSSK board, set the option to "8".
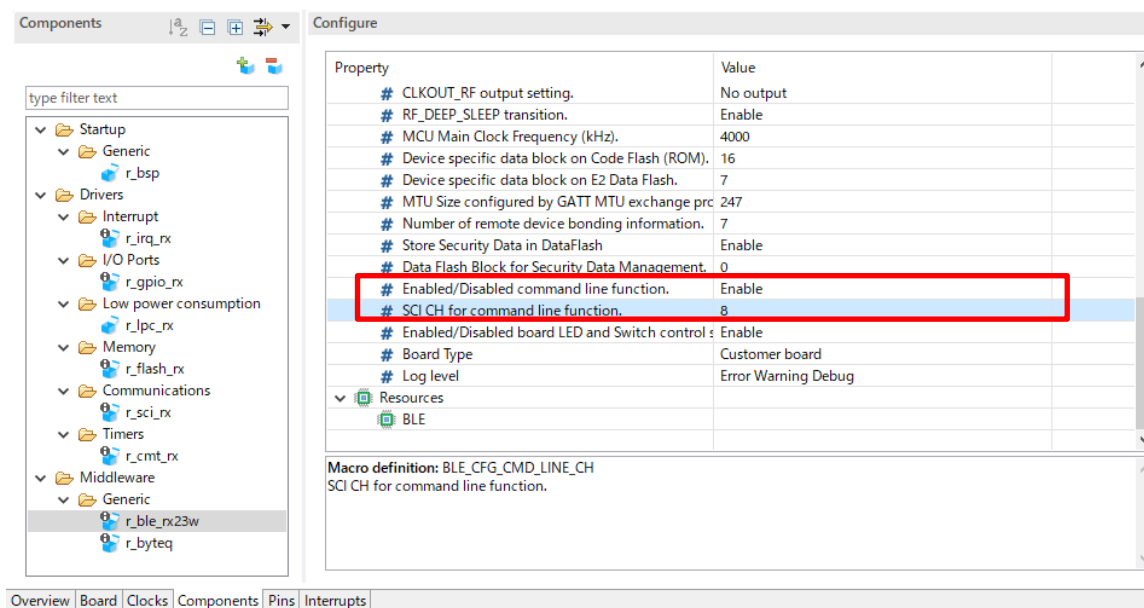

Figure 4.18 : BLE configuration options for Command Line Interface

(2) SCI(r_sci_rx)

Select "Components" tab→"r_sci_rx" and set the following configuration options.

- Select "Include software support for channel n" (n is the SCI channel number for command line) and set the option to "Include".

Property	Value
# Include software support for channel 6	Not
# Include software support for channel 7	Not
# Include software support for channel 8	Include
# Include software support for channel 9	Not
# Include software support for channel 10	Not
# Include software support for channel 11	Not
# Include software support for channel 12	Not

Macro definition: SCI_CFG_CH8_INCLUDED
SPECIFY CHANNELS TO INCLUDE SOFTWARE SUPPORT FOR 1=included, 0=not
NOTE: If using ASYNC mode, adjust BYTEQ_CFG_MAX_CTRL_BLKS in r_byteq_config.h to provide 2 queues per channel (static mode only).
- * = port connector RDKRX63N, RSKRX210, RSKRX11x
- u = channel used by the USB-UART port (G1CUSB0)
- a = this channel is used only for RX130-512KB
- n = this channel is not available for RX65N-64pin.
- s = this channel is not available in simple SPI mode.
RX MCU supported channels (refer to Hardware Manual for information)

Figure 4.19 : SCI configuration options for Command Line Interface(1)

- Select " ASYNC mode TX queue buffer size for channel n" (n is the SCI channel number for command line) and set the option to "160".

Property	Value
# ASYNC mode TX queue buffer size for channel 5	80
# ASYNC mode TX queue buffer size for channel 6	80
# ASYNC mode TX queue buffer size for channel 7	80
# ASYNC mode TX queue buffer size for channel 8	160
# ASYNC mode TX queue buffer size for channel 9	80

Macro definition: SCI_CFG_CH8_TX_BUFSIZ
SPECIFY ASYNC MODE TX QUEUE BUFFER SIZES (will not allocate if chan not enabled)

Figure 4.20 : SCI configuration options for Command Line Interface(2)

- Select "Transmit end interrupt" and set the option to "Enable".

Property	Value
# ASYNC mode RX queue buffer size for channel 11	80
# ASYNC mode RX queue buffer size for channel 12	80
# Transmit end interrupt	Enable
# GROUP12 (Receive error) interrupt priority	3
# GROUPBL0 (ERI, TEI) interrupt priority	3

Macro definition: SCI_CFG_TEI_INCLUDED
ENABLE TRANSMIT END INTERRUPT (ASYNCHRONOUS)
This interrupt only occurs when the last bit of the last byte of data has been sent and the transmitter has become idle. The interrupt calls the user's callback function specified in R_SCI_Open() and passes it an SCI_EVT_TEI event. A typical use of this feature is to disable an external transceiver to save power. It would then be up to the user's code to re-enable the transceiver before sending again. Not including this feature reduces code space used by the interrupt. Note that this equate is only for including the TEI code. The interrupt itself must be enabled using an R_SCI_Control(hdl, SCI_CMD_EN_TEI, NULL) call.

Figure 4.21 : SCI configuration options for Command Line Interface(3)

- Check the SCI channel number , "RXDn/SMISOn" and "TXDn/SMOSIn" (n is the SCI channel number for command line) in "SCI" resources.

Property	Value
∨ Resources	
∨ SCI	
> SCI1	☐
> SCI5	☐
∨ SCI8	☑
SCK8 Pin	☐ Unused
RXD8/SMISO8/SSCL8 Pin	☑ Used
TXD8/SMOSI8/SSDA8 Pin	☑ Used
CTS8#/RTS8#/SS8# Pin	☐ Unused

Figure 4.22 : SCI configuration options for Command Line Interface(4)

4.5.3 Security Data Management/Device-specific data (in data flash)

Set the BLE configuration options for the Security Data Management. It does not need to configure the Data Flash FIT module(r_flash_rx) options.

(1) BLE(r_ble_rx23w)

Select "Components" tab →"r_ble_rx23w"→"Store Security Data in DataFlash" and set the option to "Enabled". And configure the "Data Flash Block for Security Data Management" option to the Data Flash block number that you want to use.

In addition, configure the "Device Specific data block on E2 Data Flash" option to the block number for device-specific data in Data Flash.
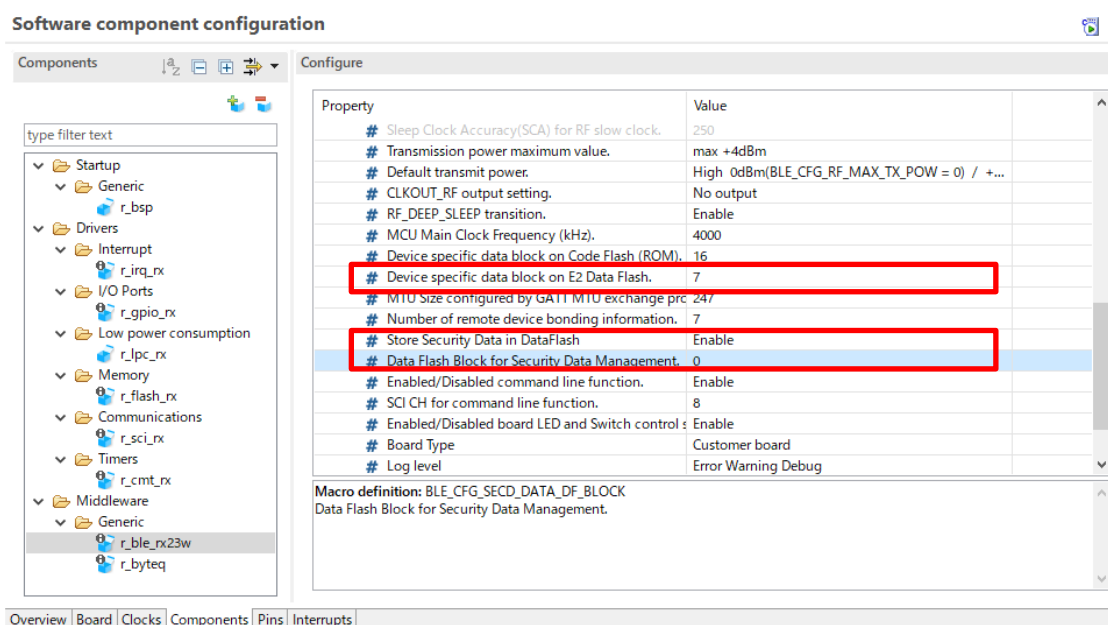

Figure 4.23 : BLE configuration options for Security Data Management

4.5.4　LED and Switch control

Set the BLE(r_ble_rx23w) and the IRQ(r_irq_rx) configuration options for LED and Switch control. It does not need to configure the GPIO options.

Table 4.1 shows the pins that connects to the LED and the Switch on the Target Board and the RSSK.

If you use a customer board, the values need to be set in accordance with the board.

Table 4.1 : Pins connected to the LED and the Switch.

Board	LED	Switch
Target Board	LED1 : PC0 LED2 : PB0	SW1 : IRQ5
RSSK	LED1 : P42 LED2 : P43	SW1 : IRQ1 SW2 : IRQ0
Customer board	The user can arbitrarily change the followings. LED1 : PC5(default) LED2 : PC6(default)	The user can arbitrarily change the followings. SW1 : IRQ7(default) SW2 : IRQ5 (default)

(1)　BLE(r_ble_rx23w)

Select "Components" tab →"r_ble_rx23w"→" Enabled/Disabled board LED and Switch control support" and set the option to "Enabled". And configure the "Board Type" option to the type of the board that you want to use. If selecting "Customer board, modify the codes described in "4.6.2 LED and Switch control for customer board".
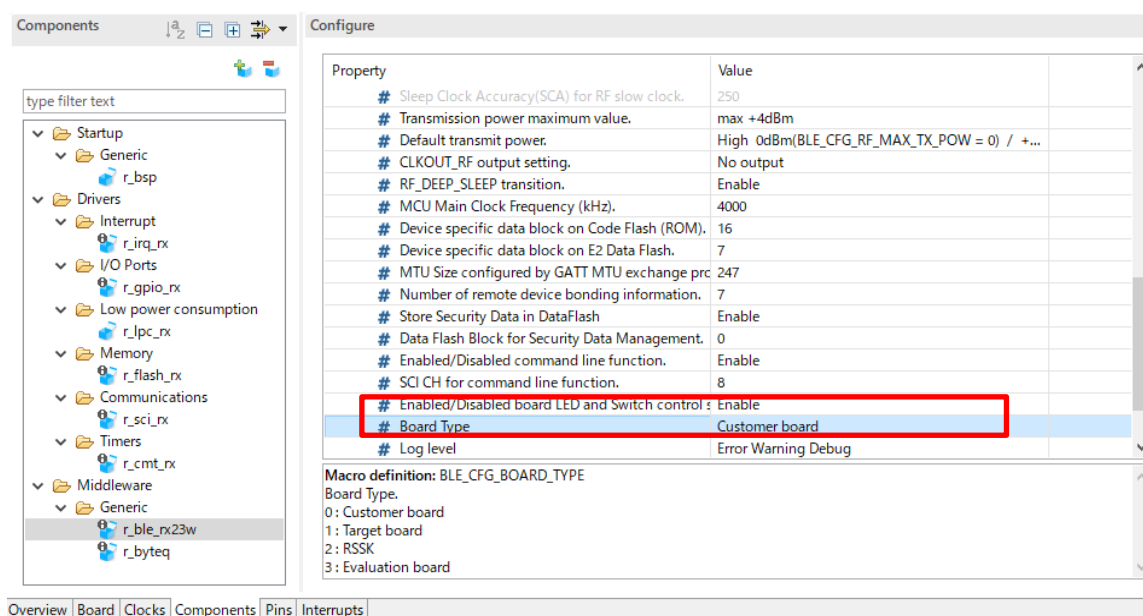

Figure 4.24 : BLE configuration options for LED and Switch control

(2) IRQ(r_irq_rx)

Select "Components" tab →"r_irq_rx" and set the following configuration options.

- Set the "Filter for IRQn" option (n is the IRQ terminal number for the board switch) to "Enabled" and the "Filter clock divisor for IRQn" option to "Divisor1". Figure 4.25 shows the default values for the default Customer Board. If you use a Customer Board, configure the "Filter for IRQn" option and the "Filter clock divisor for IRQn" in accordance with the board.

Figure 4.25 : IRQ configuration options for LED and Switch control (1)

- Check the "IRQn Pin" connected to the board switch in "ICU" resources. Figure 4.26 shows the default for the default Customer Board. Check the boxes according to your board IRQ pins.

Figure 4.26 : IRQ configuration options for LED and Switch control (2)

4.5.5   Renesas FreeRTOS

(1)   FreeRTOS Kernel

Select "FreeRTOS Kernel" in Smart Configurator "component" tab to configure the FreeRTOS Kernel. If a RX23W device communicates with Bluetooth LE, change the following configuration option from the default setting.

Because the RF interrupt has the maximum priority (14), change the priority to the following.

- configMAX_PRIORITIES (Maximum number of priorities to the application task)
  7 -> 16

- configMAX_SYSCALL_INTERRUPT_PRIORITY (Maximum syscall interrupt priority)
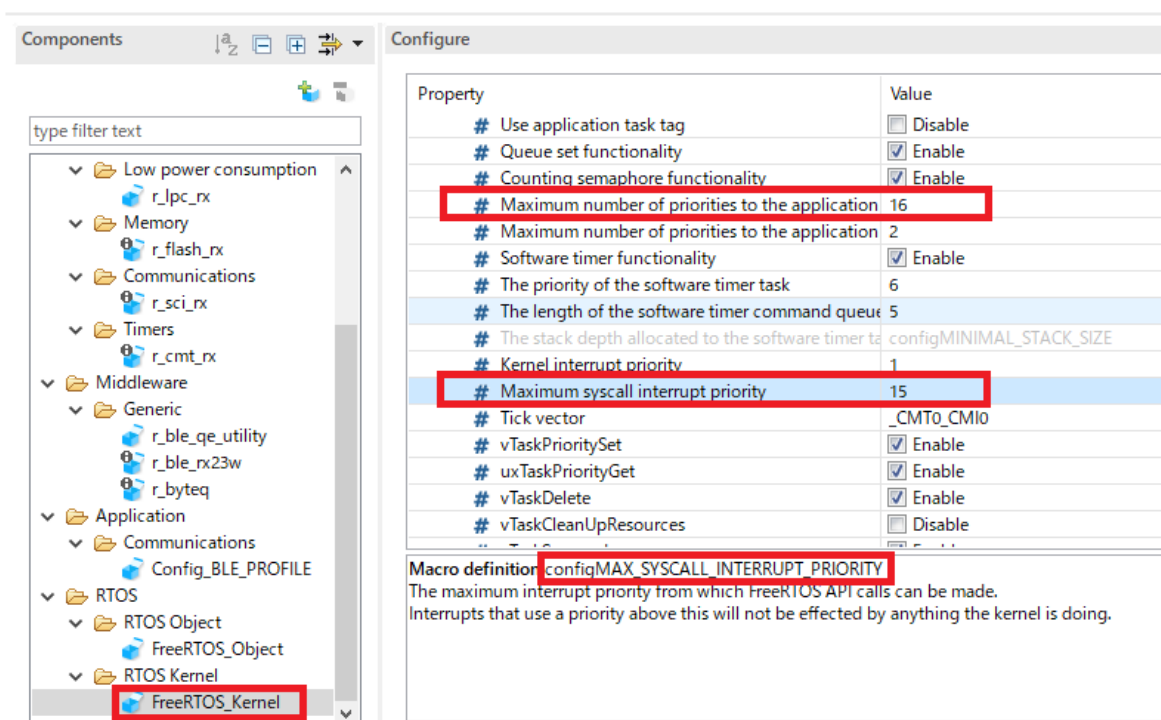 4 -> 15

Figure 4.27 FreeRTOS Kernel Configuration (1)

Configure the following option to change the heap size. It recommends that this option is set to about 10 for Bluetooth LE communication.

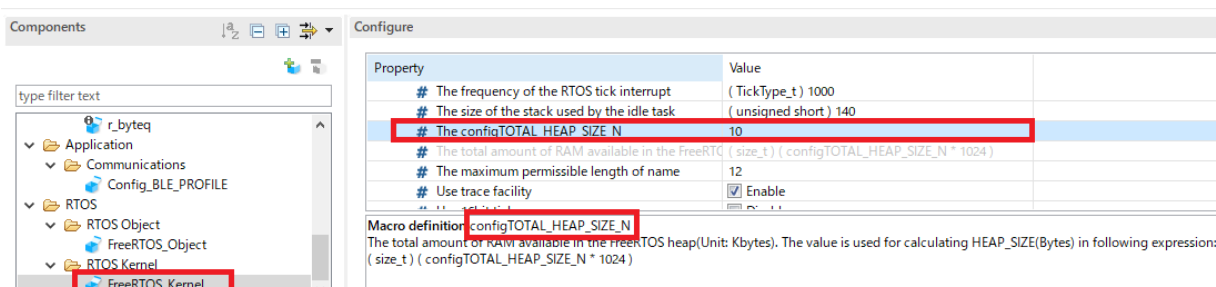- configTOTAL_HEAP_SIZE_N (The configTOTAL_HEAP_SIZE_N)

Figure 4.28 : FreeRTOS Kernel Configuration (2)

Use the following configurations as default setting.

- configUSE_MUTEXES(Mutex functionality)
  1(Enable)

- configUSE_TASK_NOTIFICATIONS (Use task notifications)
  1(Enable)

For more information about the other FreeRTOS Kernel configurations, refer to the following documents with the "configXXX" word described in Smart Configurator "Macro definition".

- FreeRTOS customization : https://www.freertos.org/a00110.html
- "RX Family Renesas FreeRTOS (R01AN4307) 3.2 Custom Configuration"

The configuration options are output to src/frtos_config/FreeRTOSConfig.h .

(2) Subtask Declaration

If your application use tasks other than main task, select "FreeRTOS_Object" in Smart Configurator "component" tab and input the subtask information and the resource that the application uses. This section describes the subtask declaration.
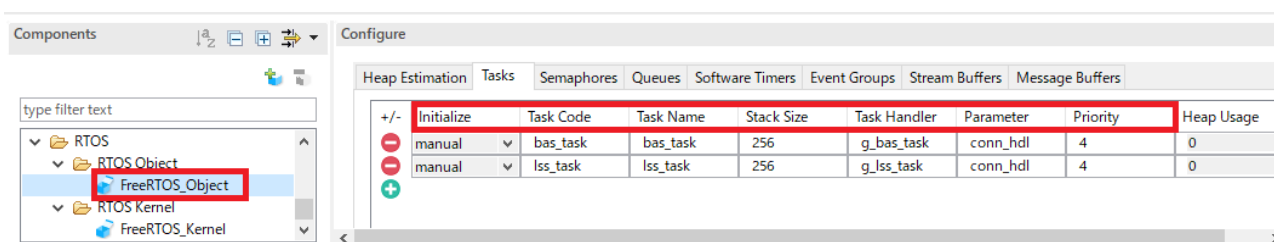
Figure 4.29 : FreeRTOS Object Configuration

Table **4.2** shows the parameters shown in the FreeRTOS Object Configuration (Figure 4.29).

Table 4.2 : Parameter regarding subtask

Parameter	Description
Initialize	Specify where the subtasks are generated. If "manual" is selected, Object_init_manual() in freertos_object_init.c generates the subtasks. This function needs to be called by application. If "kernel" is selected, Kernel_Object_init() in freertos_object_init.c generates the subtasks. The Renesas FreeRTOS code calls this function.
Task Code	Subtask function.
Task Name	Subtask name for debug
Stack Size	This parameter multiplied by 4 bytes are allocated for the subtask stack size.
Task Handler	A variable for storing the task handle.
Parameter	Parameter is passed to the task function in generating the task.
Priority	Task priority.

| | If Bluetooth LE communication is performed, set less than the BLE task priority to the subtask priority. |

After declaring subtasks in FreeRTOS Object Configuration, the files in Table **4**.**3** are output with code generation.

Table 4.3 : Codes regarding subtask output in code generation

File	Description
src/frtos_skeleton/"subtask name".c	Subtask template
src/frtos_startup/freertos_object_init.c	Code included in subtask generation function.

4.6 Configuration after code generation

Output the codes by click the code generation button 🖼 . Configure the followings after code generation.

4.6.1 Changing CMT for BLE(if the CMT FIT module version is prior to 4.50)

Because the BLE FIT module use the two CMT channels, add the following after the definition of CMT_RX_NUM_CHANNELS in the r_cmt_rx.c in the CMT FIT module.(Figure 4.30)

If the CMT FIT module is v4.50 or later, you do not need to change the source code.

```
#if defined(BSP_MCU_RX23W)
#undef CMT_RX_NUM_CHANNELS
#define CMT_RX_NUM_CHANNELS            (2)
#endif /* BSP_MCU_RX23W */
```

```
/********************************************************************************************************
Macro definitions
********************************************************************************************************/
/* Define the number of CMT channels based on MCU type. */
#if defined(BSP_MCU_RX62_ALL) || defined(BSP_MCU_RX63_ALL) || defined(BSP_MCU_RX21_ALL) || \
    defined(BSP_MCU_RX61_ALL) || defined(BSP_MCU_RX64_ALL) || defined(BSP_MCU_RX113)    || \
    defined(BSP_MCU_RX71_ALL) || defined(BSP_MCU_RX231)     || defined(BSP_MCU_RX23_ALL) || \
    defined(BSP_MCU_RX24_ALL) || defined(BSP_MCU_RX65_ALL)  || defined(BSP_MCU_RX66_ALL) || \
    defined(BSP_MCU_RX72_ALL)
    #define CMT_RX_NUM_CHANNELS       (4)
#elif defined(BSP_MCU_RX111)  || defined(BSP_MCU_RX110)     || defined(BSP_MCU_RX130)
    #define CMT_RX_NUM_CHANNELS       (2)
#else
    #error "Error! Number of channels for this MCU is not defined in r_cmt_rx.c"
#endif

#if defined(BSP_MCU_RX23W)
#undef CMT_RX_NUM_CHANNELS
#define CMT_RX_NUM_CHANNELS           (2)
#endif /* BSP_MCU_RX23W */
```

Figure 4.30 : Changing the r_cmt_rx.c for BLE

4.6.2 LED and Switch control for customer board

In case of using a customer board, modify the followings in app_lib/board/r_ble_board.c.

(1) Macros of LED and Switch

Modify the below macros which define the IRQ and the GPIO pin number in accordance with the customer board.

> BLE_BOARD_SW1_IRQ
>
> BLE_BOARD_SW2_IRQ
>
> BLE_BOARD_LED1_PIN
>
> BLE_BOARD_LED2_PIN

```
#if (BLE_CFG_BOARD_TYPE == 1)   /* for RX23W Target Board(TB) */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_LED1_PIN (GPIO_PORT_C_PIN_0)
#define BLE_BOARD_LED2_PIN (GPIO_PORT_B_PIN_0)
#elif (BLE_CFG_BOARD_TYPE == 2) /* for RX23W RSSK board */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_1)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_0)
#define BLE_BOARD_LED1_PIN (GPIO_PORT_4_PIN_2)
#define BLE_BOARD_LED2_PIN (GPIO_PORT_4_PIN_3)
#else /* BLE_CFG_BOARD_TYPE */  /* for Customer board */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_7)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_5)            If you use a customer board,
#define BLE_BOARD_LED1_PIN (GPIO_PORT_C_PIN_5)     these values need to be set in
#define BLE_BOARD_LED2_PIN (GPIO_PORT_C_PIN_6)     accordance with the board.
#endif /* BLE_CFG_BOARD_TYPE */
```

Figure 4.31 : LED, SW macros

If SW1 connects to IRQ7, SW2 connects to IRQ5, LED1 pin connects to PortC pin5 and LED2 pin connects to PortC pin6, make modifications shown in Figure 4.31.

(2) Configure the registers in irq_pin_set()

Modify the register settings in irq_pin_set() for the IRQ pins connected to the SW on the customer board.

```
#if (BLE_CFG_BOARD_TYPE == 1)   /* for RX23W Target Board(TB) */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_C_PIN_0)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_B_PIN_0)
#elif (BLE_CFG_BOARD_TYPE == 2) /* for RX23W RSSK board */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_1)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_0)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_4_PIN_2)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_4_PIN_3)
#else /* BLE_CFG_BOARD_TYPE */   /* for Custom board */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_7)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_C_PIN_5)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_C_PIN_6)
#endif /* BLE_CFG_BOARD_TYPE */
```

If you use a customer board, these values need to be set in accordance with the board.

Figure 4.32 : The changes in irq_pin_set()

4.6.3 Add application code

When a project is created, the application code("{Project Name}.c") is generated. But it is a blank template. Therefore, modify main() to call app_main() in app_main.c generated by the QE for BLE. If you use Renesas FreeRTOS, modify main_task() to call app_main(). Refer to "6.2.1 Main Task (2)call app_main()". Here is a sample main().

```c
#include "r_smc_entry.h"

void main(void);
void app_main(void);

void main(void)
{
        app_main();
}
```

Figure 4.33 : Sample main() calling app_main()

(1) QE for BLE

Select and configure profiles, services, characteristics in the QE for BLE, output the base code for the application and profile development. For the QE for BLE details, see "Bluetooth Low Energy Profile Developer's Guide(R01AN4553)".

4.7   Linker configurations

4.7.1   BLE Protocol Stack program section separation

Configure the section for the BLE.

Table 4.4 shows the section name of the BLE Protocol Stack.

Table 4.4 : Section Name

No.	Name	Default Section Name	BLE Protocol Stack Section Name	Description
1	Program area (ROM)	P	BLE_P	Stores machine code
2	Constant area (ROM)	C	BLE_C	Stores const type data
		C_2	BLE_C_2	
		C_1	BLE_C_1	
3	Initialized data area (ROM)	D	BLE_D	Stores data with initial values in ROM
		D_2	BLE_D_2	
		D_1	BLE_D_1	
4	Initialized data area (RAM)	R	BLE_R	Stores data with initial values in RAM
		R_2	BLE_R_2	
		R_1	BLE_R_1	
5	Uninitialized data area (RAM)	B	BLE_B	Stores data without initial values
		B_2	BLE_B_2	
		B_1	BLE_B_1	
6	Switch statement branch table area (ROM)	W	BLE_W	Stores branch tables for switch statements
		W_2	BLE_W_2	
		W_1	BLE_W_1	
7	Literal area (ROM)	L	BLE_L	Stores string literals and initializers used for dynamic initialization of aggregates

Set the section name shown in Table 4.4 to the BLE Application project in the following procedure.

4.7.1.1   Add the section name

(1) Click [Project] → [Properties] →[C/C++ build] → [Settings].

(2) Select "Linker" → "Section" in the "Tool Settings" tab and click the "…" button.

(3) Input the following in the "Section Viewer".

RAM	BLE_B*
	BLE_R*

ROM	BLE_C*
	BLE_D*
	BLE_W*
	BLE_L
	BLE_P

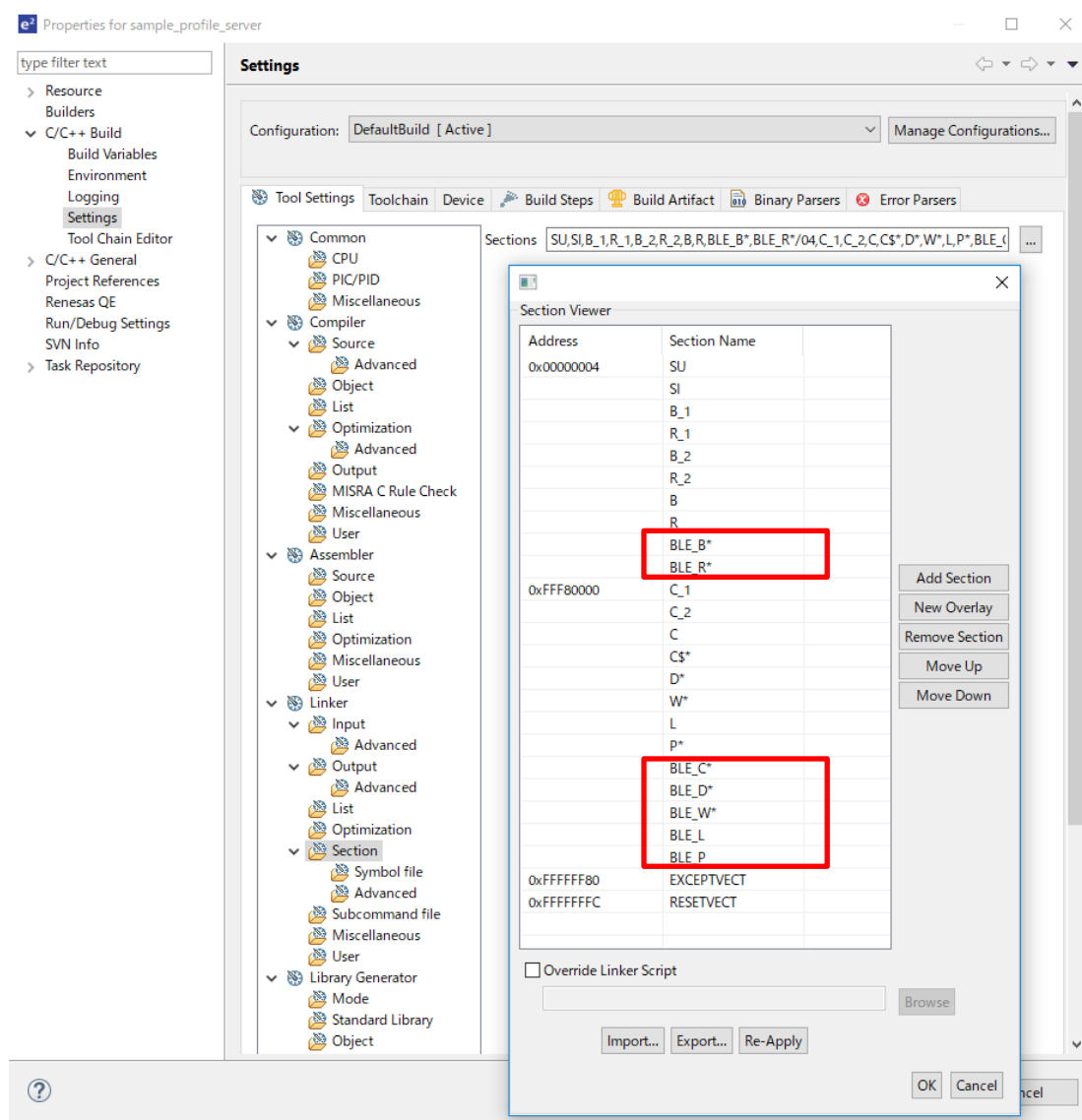

Figure 4.34 : Section Configuration

4.7.1.2 Rom to RAM mapped section

(1) Select "Settings" → "Tool Settings" → "Linker" → "Section" → "Symbol file".

(2) Input the following to the "ROM to RAM mapped section". (Figure 4.35)
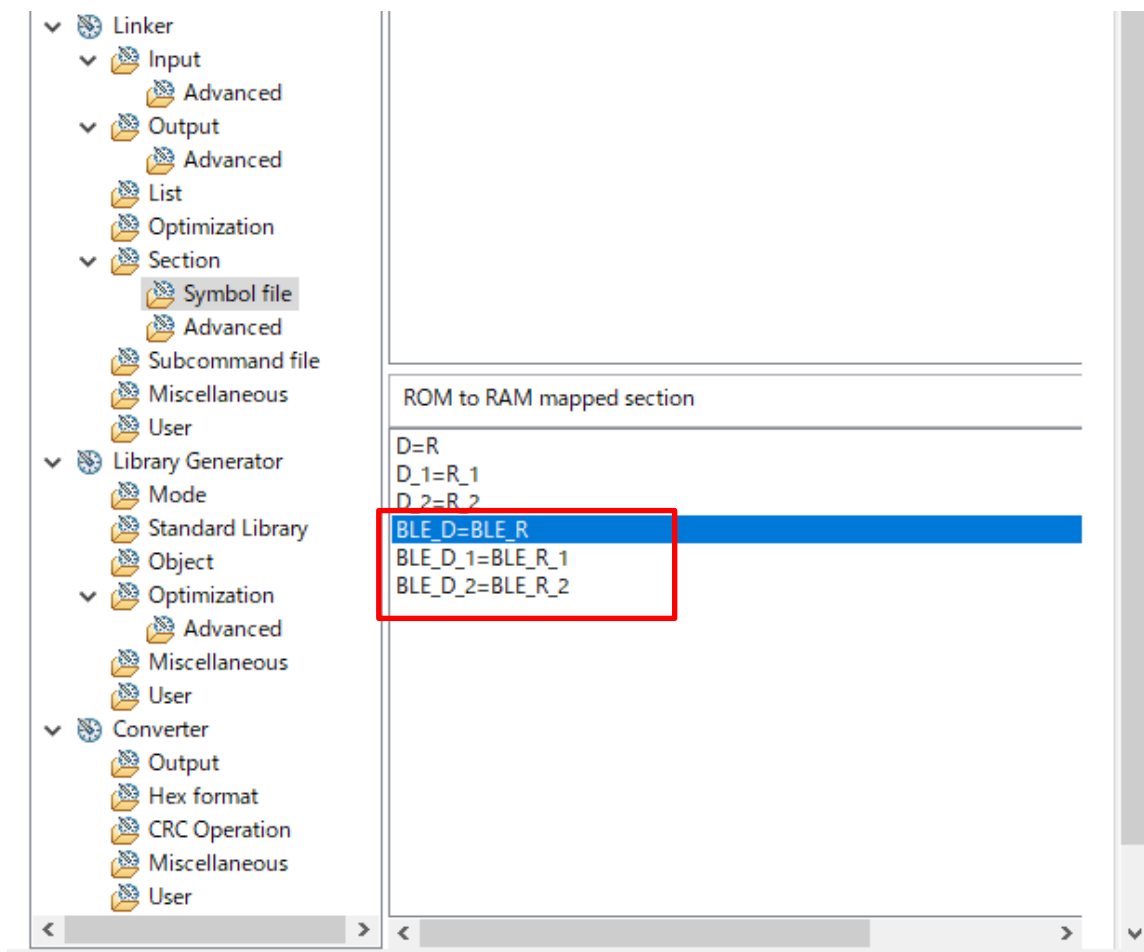
BLE_D=BLE_R

BLE_D_1=BLE_R_1

BLE_D_2=BLE_R_2

Figure 4.35 : Rom to RAM mapped section configuration

NOTE :
Because the R_BLE_Open() initialize the BLE section, it is not necessary to change the DTBL/BTBL table.

4.7.2 BLE Protocol Stack Library Configuration

The BLE Protocol Stack is provided as static library. Three types of libraries shown in Table 4.5 are included in the lib directory of the BLE FIT module.

Table 4.5 : BLE Protocol Stack libraries

Type of the BLE Protocol Stack	Library
All features	lib_ble_ps_ccrx_a.lib
Balance	lib_ble_ps_ccrx_b.lib
Compact	lib_ble_ps_ccrx_c.lib

Configure the followings to make the BLE Application project available to the BLE Protocol Stack library.

1) Linker configuration

Click [Project] → [Properties] on e²studio.

Select "C/C++ Build" → "Settings" → "Tool Settings" → "Linker" → "Input".

Confirm that the following library is added to the "Relocatable files, object files and library files" (Figure 4.36). If not, add the library.

"${workspace_loc:/${ProjName}/src/smc_gen/r_ble_rx23w/lib/lib_ble_ps_ccrx.lib}"
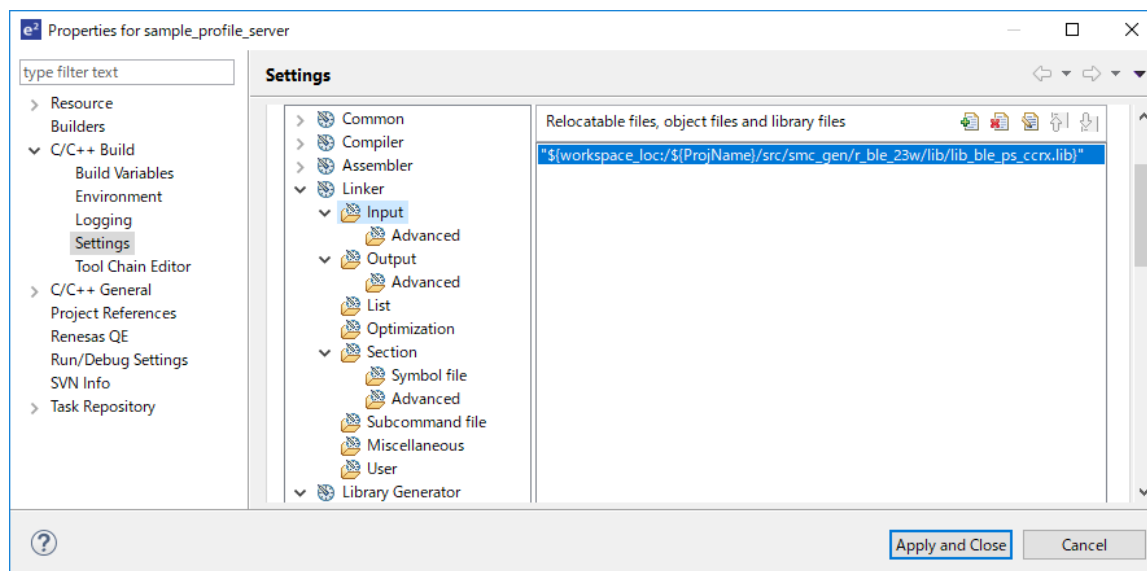
Figure 4.36 : Linker configuration

2) Register the bat file

Register the bat file to switch the referred library in conjunction with the configuration option of the BLE FIT module.

Click [Project] → [Properties] on e^2 studio.

Select "C/C++ Build" → "Settings" → "Build Steps".

Input the following to the "Command(s)" in the "Pre-build steps". (see Figure 4.37)

..\src\smc_gen\r_ble_rx23w\lib\ble_fit_lib_selector.bat

Figure 4.37 : Build Steps configuration

4.8   Debug configurations

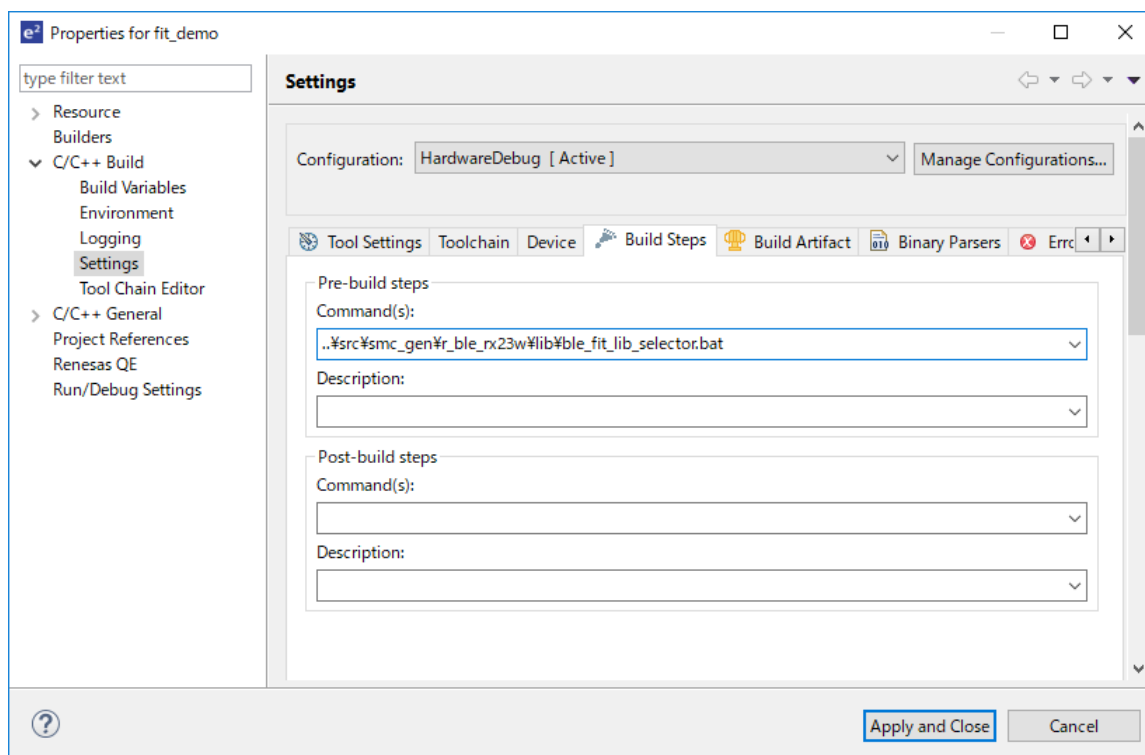Select "Run" → "Debug Configurations…" → "Renesas GDB Hardware Debugging" → the application project and configure the followings. Build the project after setting the debug configurations. If the build is successfully finished, write the firmware to the board.

4.8.1   Flash ID Code

Click [RUN]→[Debug Configuration] on the e^{2}studio.

Select the BLE Application project from the "Renesas GDB Hardware Debugging" and input "45FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF" to "ID Code" to enable the ID Code protection described in "2.9 Flash Memory Protection".

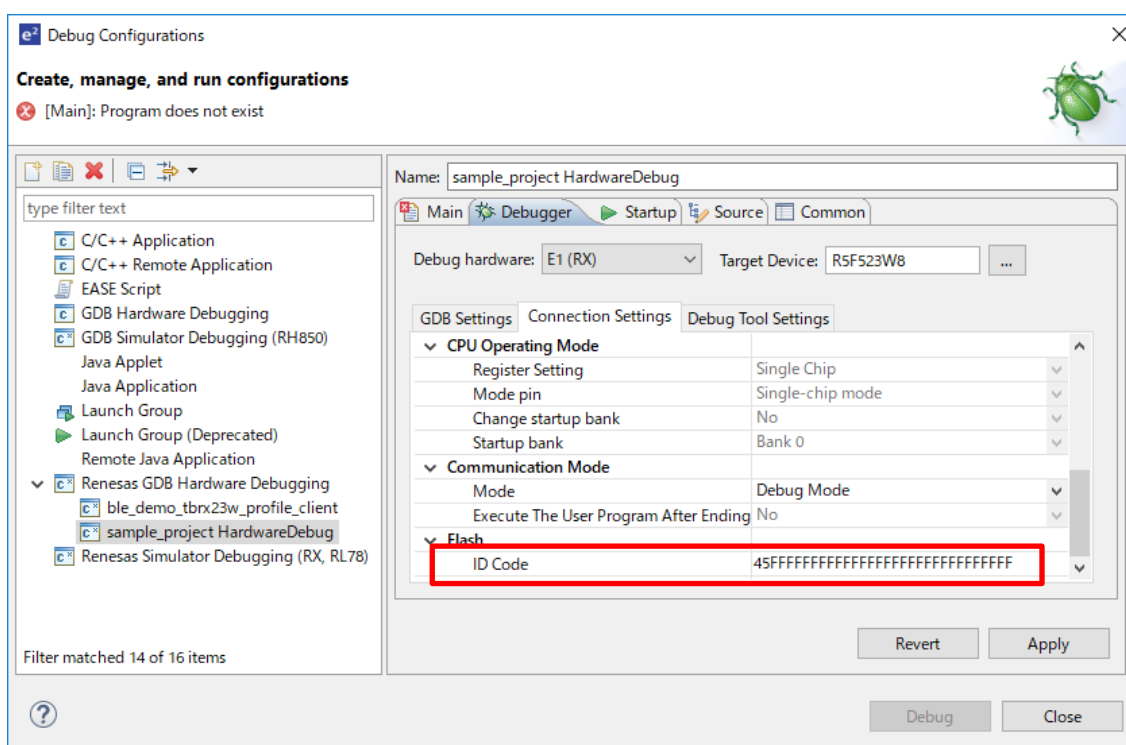If writing a firmware failed, confirm the "ID Code".

Figure 4.38 : ID Code protection configuration

4.8.2 Power

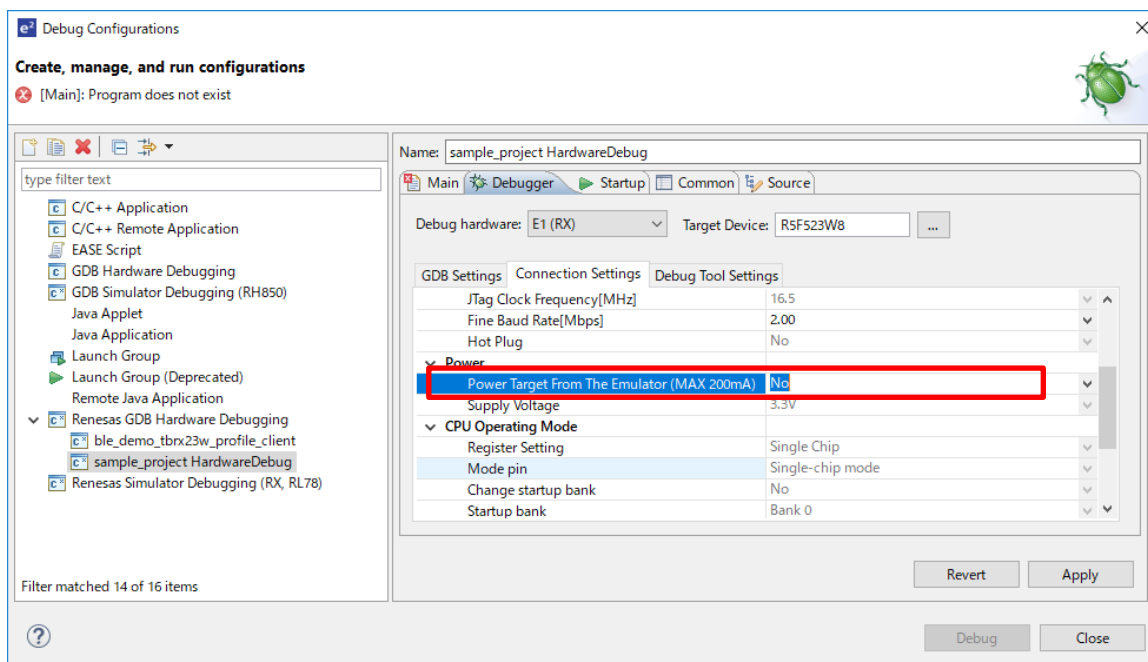Configure "Power Target From the Emulator (MAX 200mA)" to "No".

Figure 4.39 : Power supply from the emulator

4.9 IAR development environments

This section describes the project creation on the IAR Embedded Workbench for Renesas RX.
If a BLE project includes Renesas FreeRTOS, it does not support IAR development environments.

4.9.1 Create a project

Create a new CCRX project on e^2 studio according to Section 4.1 to 4.7.

4.9.2 Add iodefines.h for IAR

Generate the iodefines.h by the way (1) or (2) and add it in the CCRX project. The iodefines.h needs to be added to the CCRX project previous to "4.9.3 Project conversion".

(1) Generate from the RX Smart Configurator

- Install the RX Smart Configurator from https://www.renesas.com/software-tool/smart-configurator.

- Start the RX Smart Configurator.

- Select "File >> New…", set the following items and click "Finish".

  "Platform" : RX23W device
  "Toolchain" : IAR EWRX Toolchain
  "File name" : [Project name]
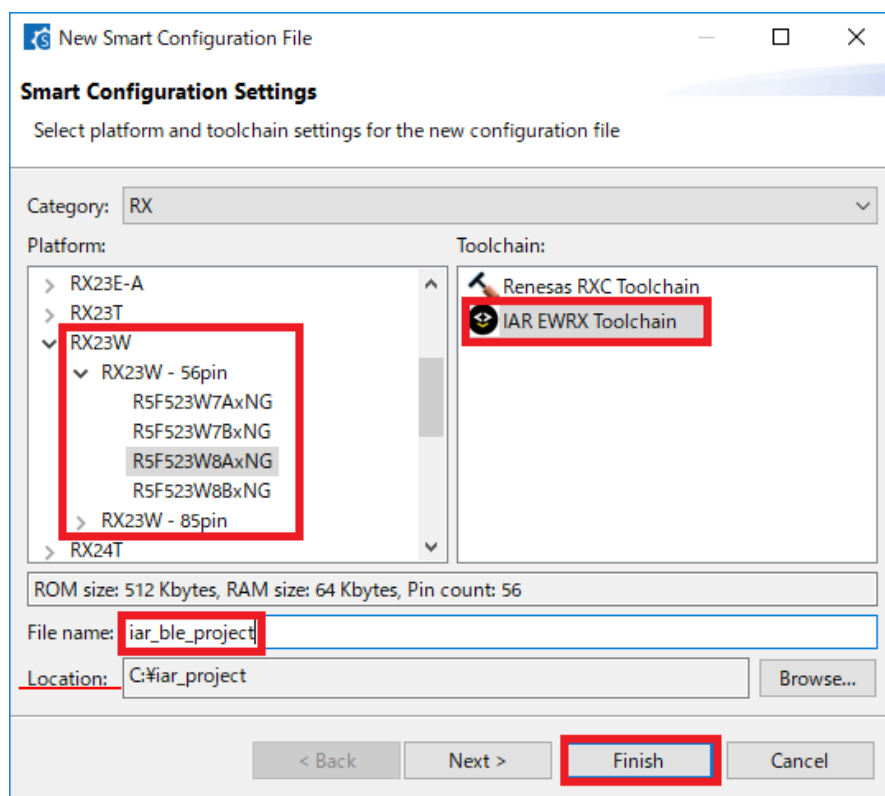

Figure 4.40 : RX Smart Configurator New Project settings

- Click "Add component" on the "Component" tab, select "r_bsp" on the "New component" window and click "Finish".
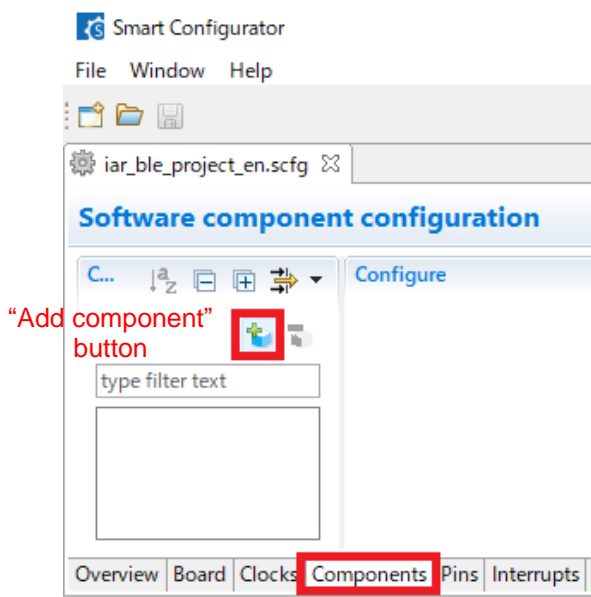
Figure 4.41 : Add component (1)
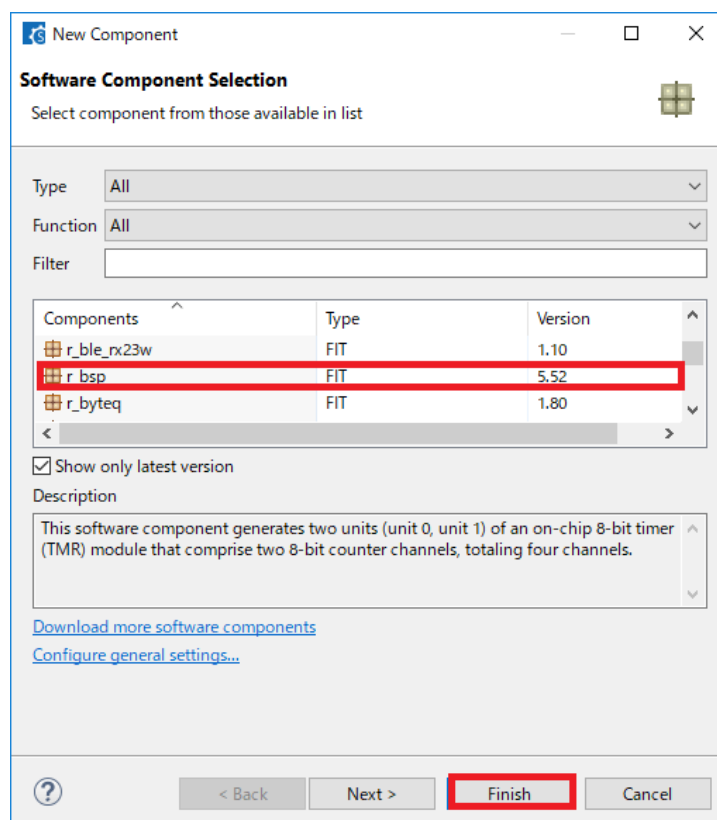
Figure 4.42 : Add component (2)

If the "r_bsp" is not shown, start CS+ and run Smart Configurator to install the "r_bsp". Refer to "RX Smart Configurator User's Guide: CS+ (R20AN0470)"

- Click "Add component" on the "Component" tab, select "r_bsp" on the "New component" window and click "Finish".
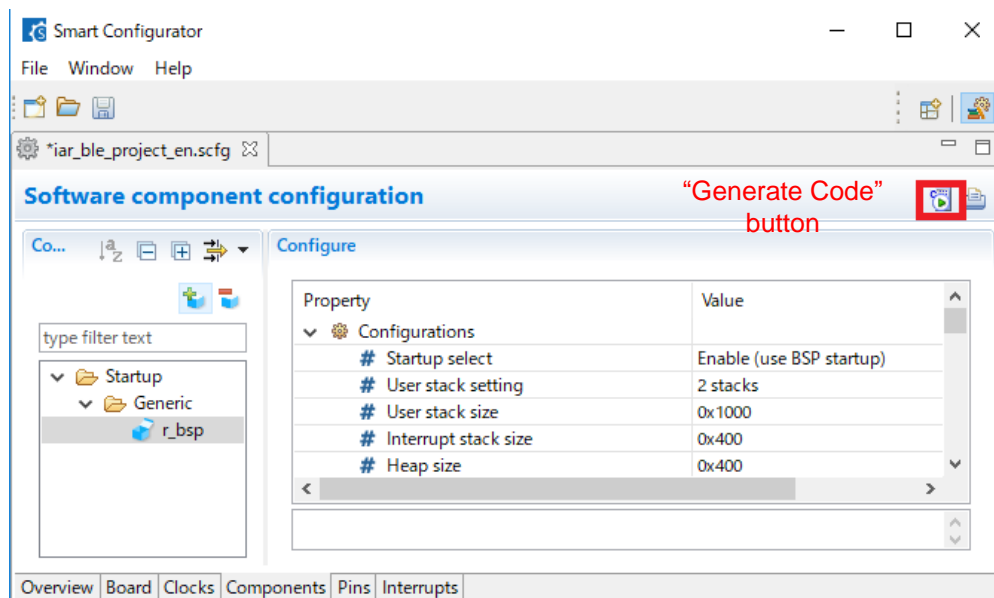
Figure 4.43 : Generate Code

- Copy the "iccrx" directory included in \src\smc_gen\r_bsp\mcu\rx23w\register_access which is in the RX Smart Configurator project location directory (refer Figure 4.40) to \src\smc_gen\r_bsp\mcu\rx23w\register_access which is generated according to 4.9.1.

(2) Copy from the FITDemos
   Copy the "iccrx" directory included in
   <<Demo Project Name>>\src\smc_gen\r_bsp\mcu\rx23w\register_access which is in the Table 8.1 demo project to \src\smc_gen\r_bsp\mcu\rx23w\register_access which is generated according to 4.9.1.

4.9.3   Project conversion

Convert a CCRX project to an IAR project according to the following procedure.

(1)   Start the IAR Embedded Workbench for Renesas RX.

(2)   Click "Tools" >> "Convert To IAR for RX…".

(3)   Input the project created by 4.9.1 into "Root directory of source project".

(4)   Check the "Enable" box in "Project file conversion".

(5)   Select "Renesas e2studio for RX" in "Project type".

(6)   Uncheck the "Enable" box in "Source code substitution".

(7)   Click the "Execute" button.
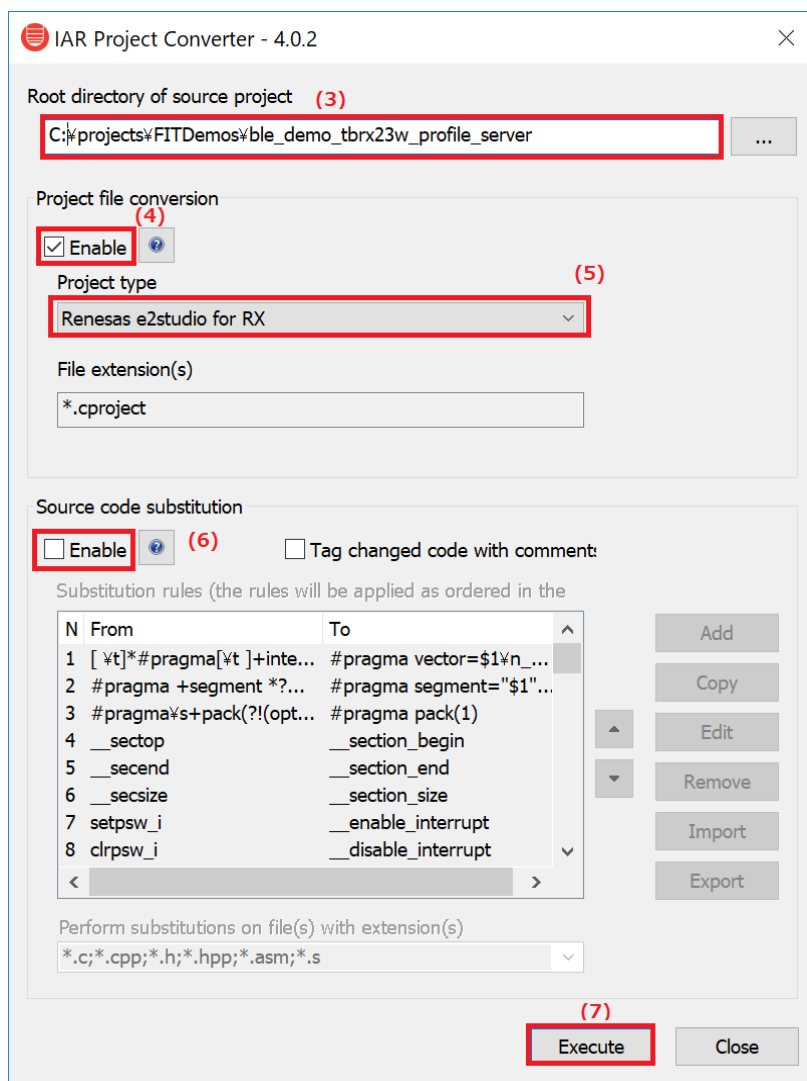
Figure 4.44 : IAR Project Converter settings screen (1)

(8)   Select the "Operate on a copy of the source code in the following directory" button in the "Choose destination directory" window.

(9) Input the directory that the converted IAR project are located in and click "OK".
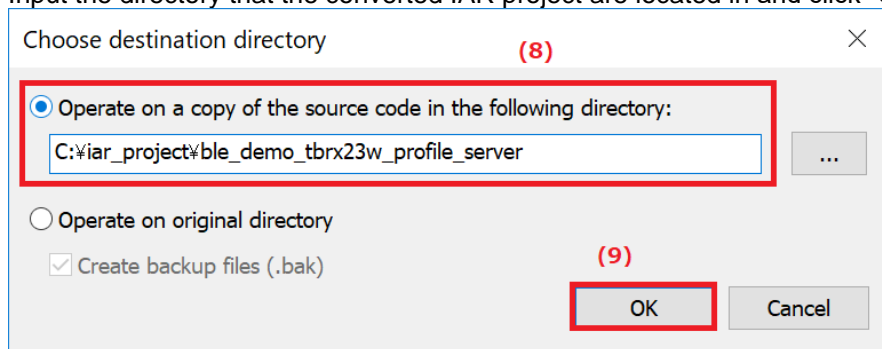
Figure 4.45 : IAR Project Converter settings screen (2)

(10) Click "OK" in the "Report" window.
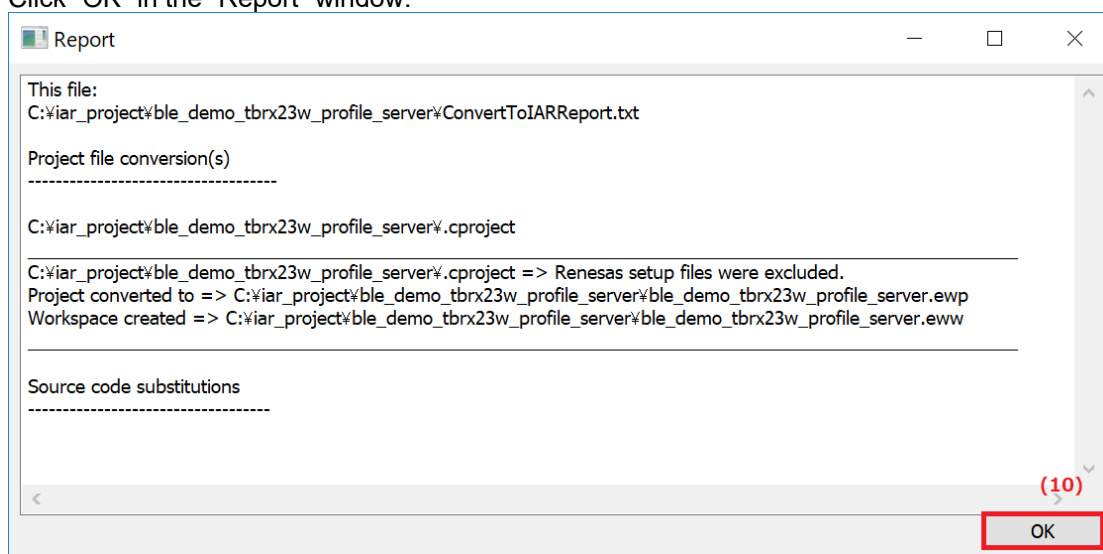
Figure 4.46 ： IAR Project Converter settings screen (3)

4.9.4 Project options configuration

Click "Project >> Add Existing Project…" in the IAR Embedded Workbench for Renesas RX and open the project file (.ewp) converted in 4.9.3. Click "Project >> Option" to change the following options.

(1) Stack/Heap

    Click "General Options >> Stack/Heap". Set "User mode stack size" and "Heap size" and "Supervisor mode stack size" same as the sizes in the CCRX project r_bsp configuration options. Table **4.6** and Figure **4.47** show the stack/heap relationship between CCRX Project and IAR Project.

<div align="center">Table 4.6 : Project stack/heap option name</div>

CCRX Project	IAR Project
User stack size	User mode stack size
interrupt stack size	Supervisor mode stack size
Heap size	Heap size
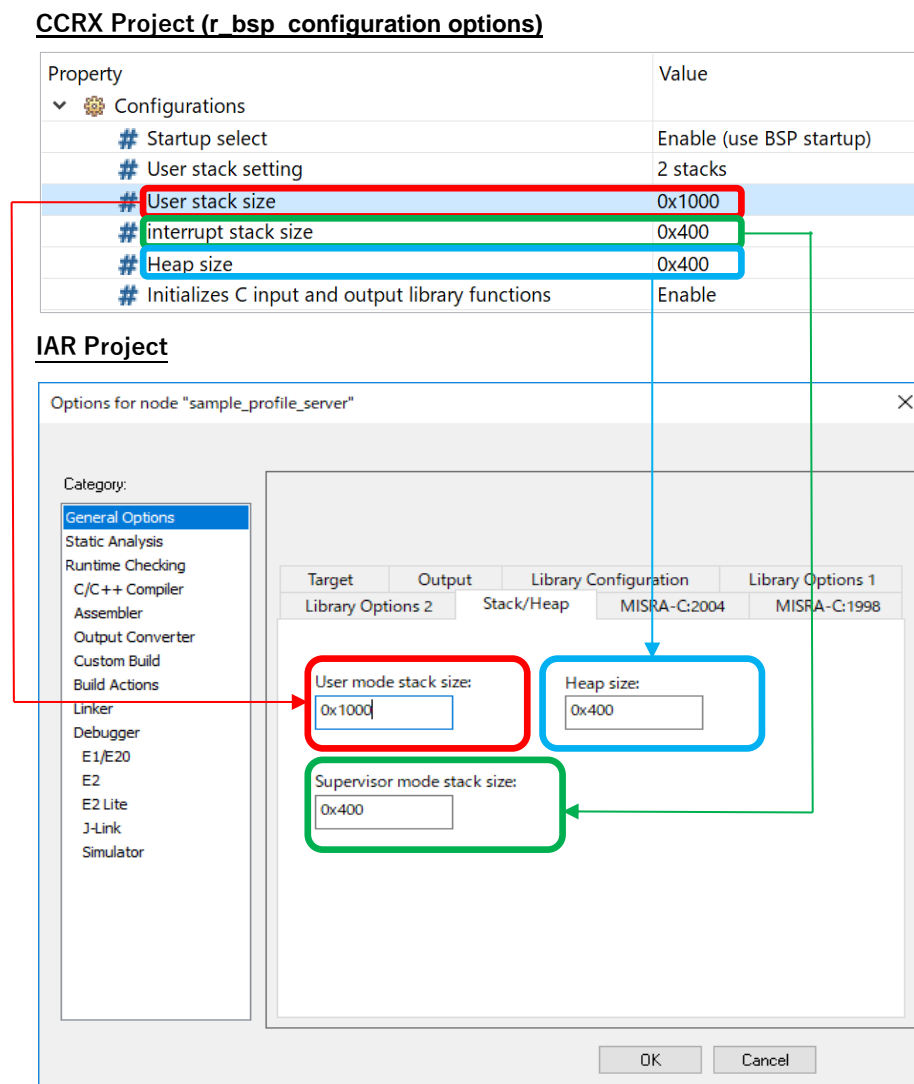
Figure 4.47 : The stack/ heap relationship between CCRX Project and IAR Project

(2) Target

Click "General Options >> Target" and select a "RX23W Group" device in the "Device".
Click the "32 bits" button as the "Size of type 'double' " in the "Floating-point".
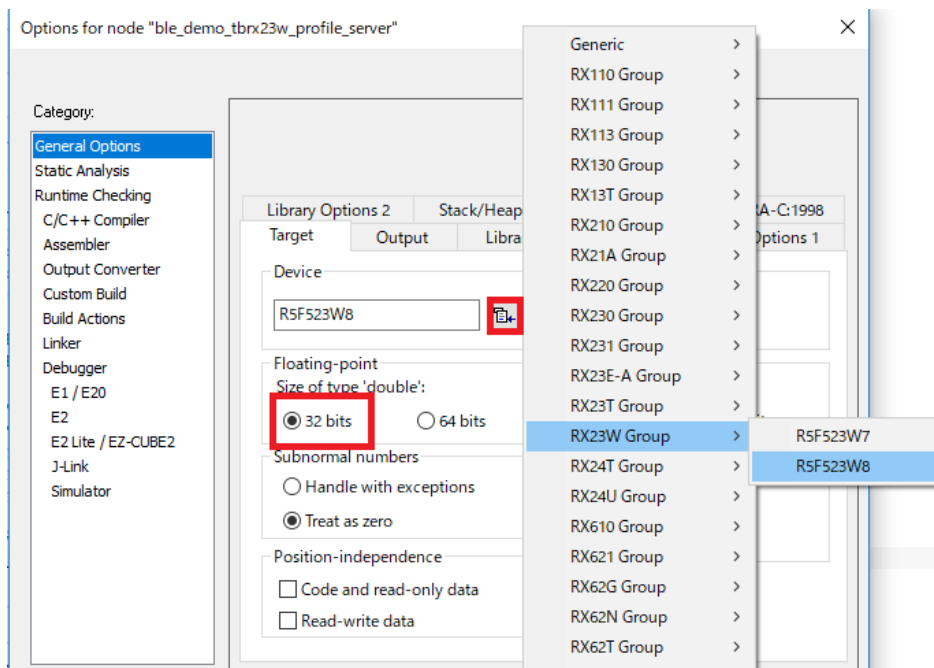
Figure 4.48 : Target setting

(3) Library Configuration

Click "General Options >> Library Configuration". Select "Normal DLIB" on "Library".
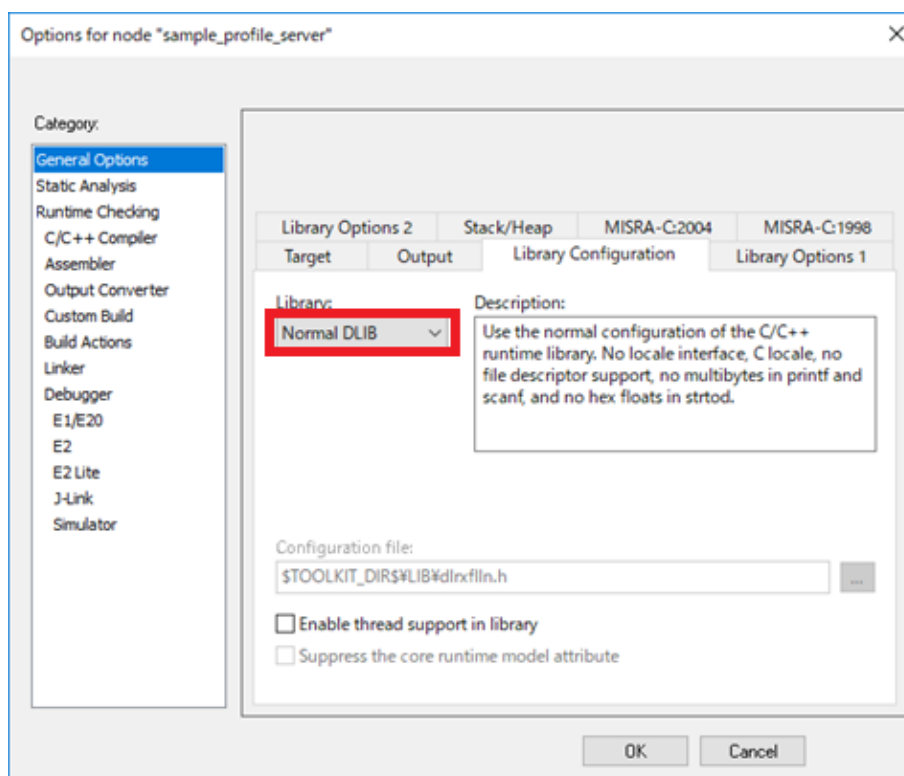
Figure 4.49 : Library Configuration

(4) Preprocessor

Click "C/C++ Compiler >> Preprocessor" and confirm that the same include paths as the CCRX Project are input in the "Additional include directories".
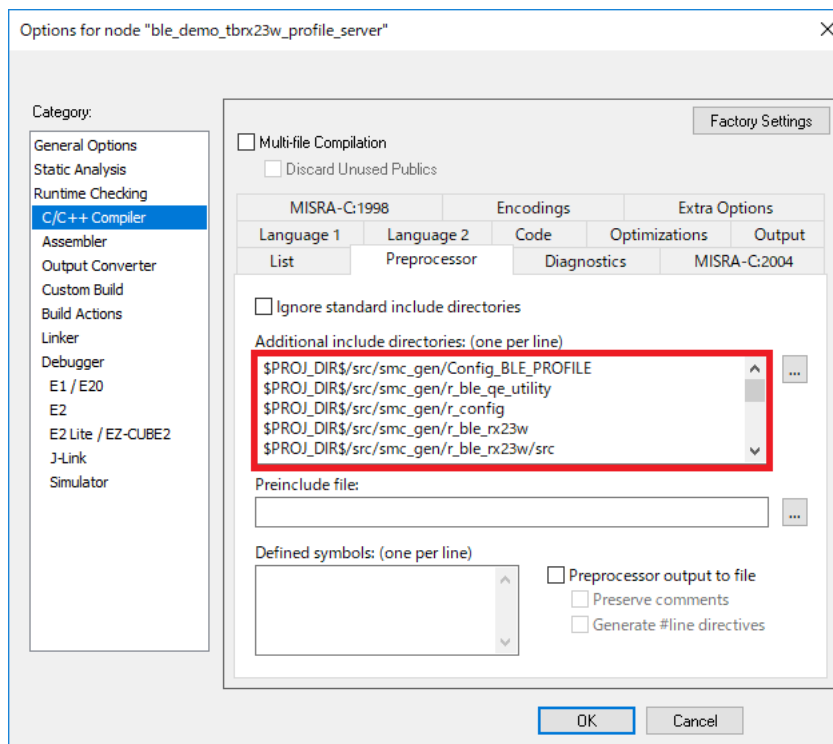
Figure 4.50 : Additional include path setting

(5) Optimizations

Click "C/C++ Compiler >> Optimizations". Select optimization level. If you don't debug the project, the ROM size can be reduced by configuring the optimization level to "High" and "Size".
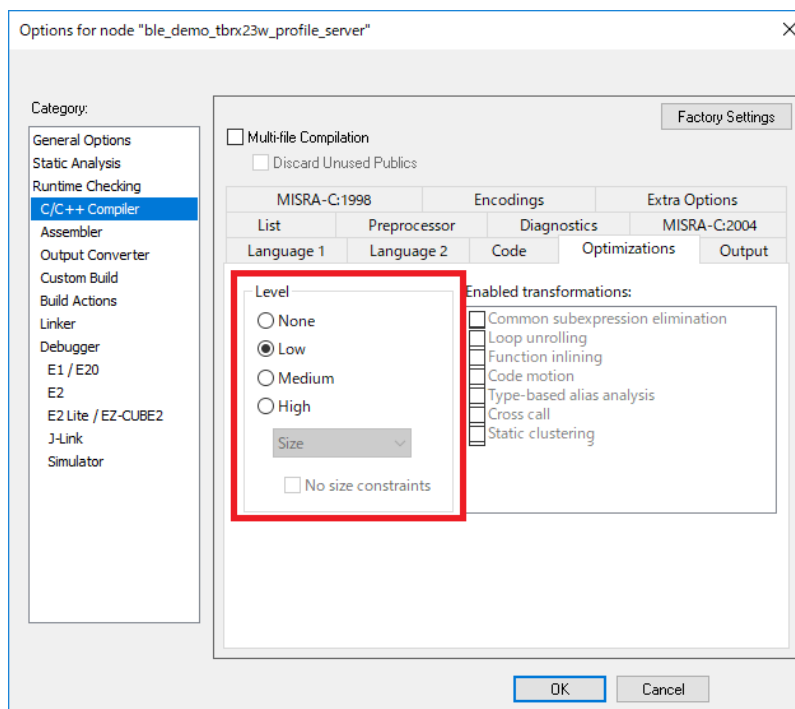
Figure 4.51 : Optimization setting

(6) Output Converter
Click "Output Converter". Check the "Generate additional output" box and select the "Motorola S-records" in the "Output format".
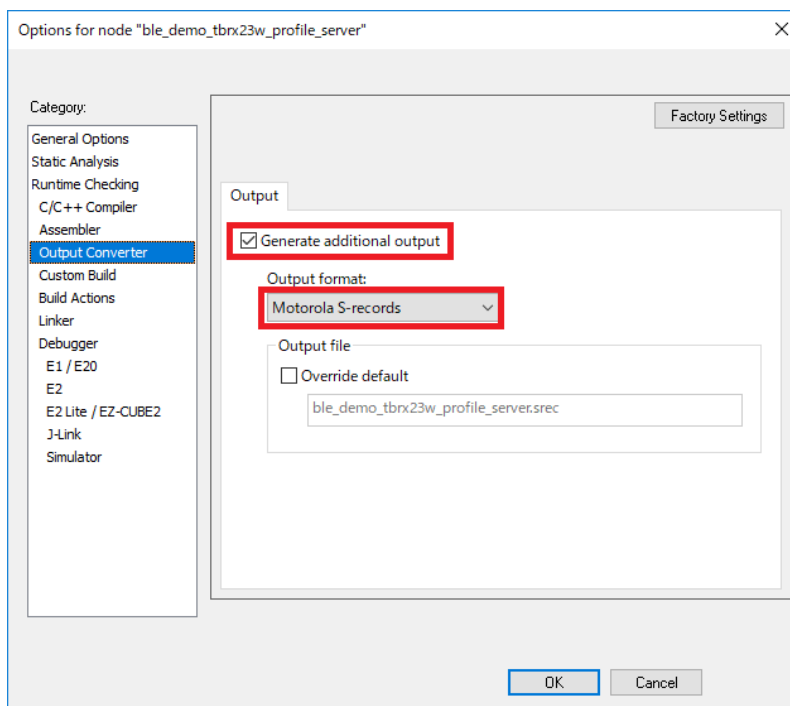
Figure 4.52 : Output Converter setting

(7) Pre-build command line
Click "Build Actions". Enter the file path to the ble_fit_lib_selector.bat file on "Pre-build command line".

Example:
The bat file path is $PROJ_DIR$\src\smc_gen\r_ble_rx23w\lib\ble_fit_lib_selector.bat.
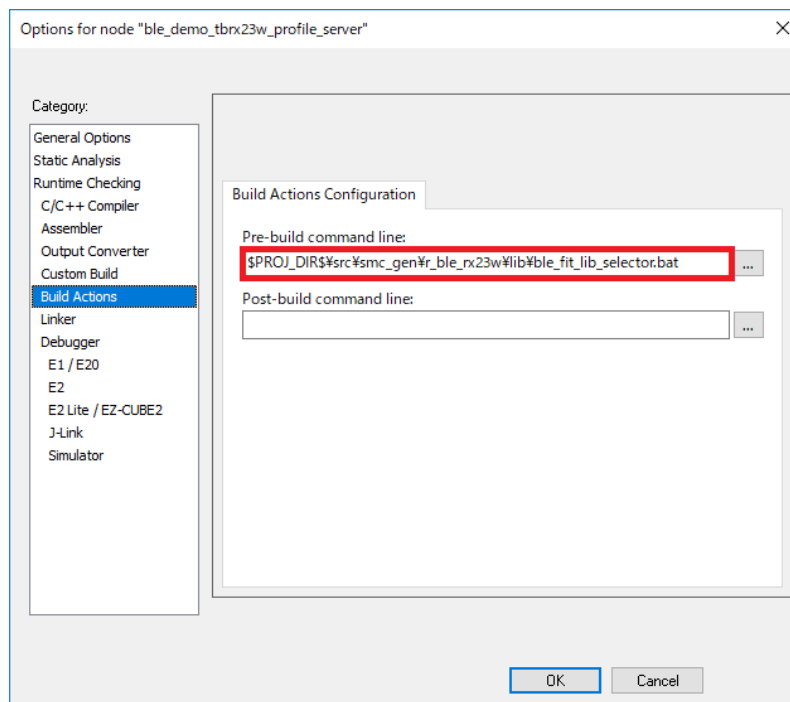
Figure 4.53 : Pre-build command line setting

(8)   Linker configuration file

Click "Linker >> Config". Check "Override default" and enter the file path to lnkr5f523w8.icf or lnkr5f523w7.icf.

Example :
The lnkr5f523w8.icf file path is $TOOLKIT_DIR$\CONFIG\lnkr5f523w8.icf.
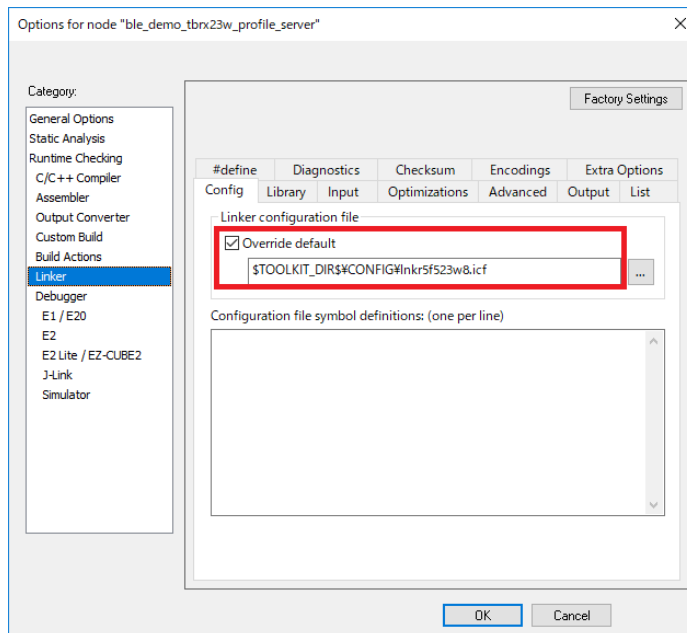
Figure 4.54 : Linker configuration file

(9)   Additional libraries
Click "Linker >> Library". Enter the library file path on "Additional libraries".

Example :
The library file path is $PROJ_DIR$/src/smc_gen/r_ble_rx23w/lib/lib_ble_ps_ccrx.lib.

Figure 4.55 : Additional libraries setting

(10) Debugger
Click "Debugger >> Setup". Select the emulator on "Driver".

Example :
If you use the E2 Lite as the emulator, select "E2 Lite/EX-CUBE2".
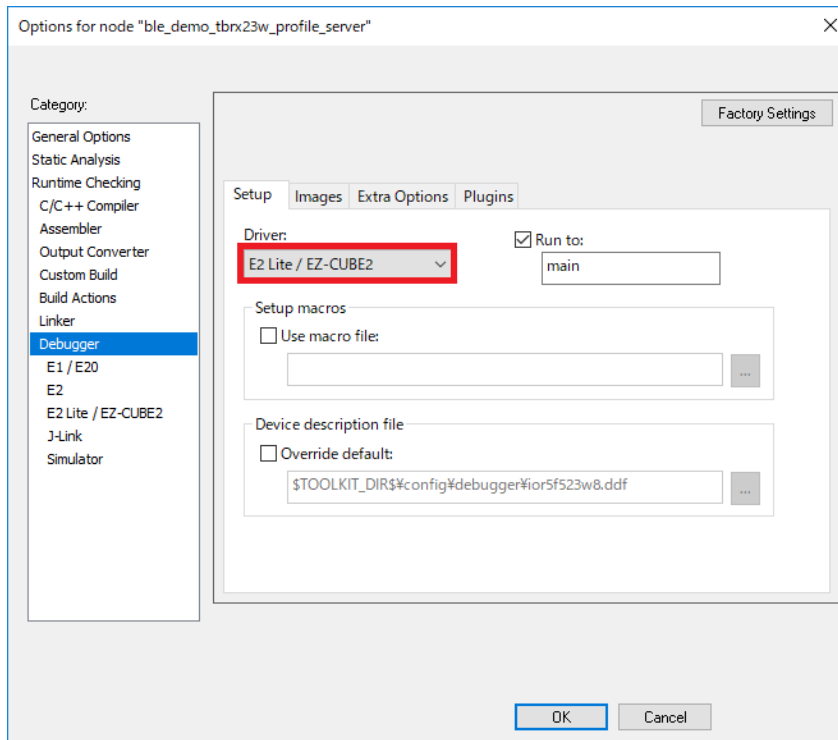
Figure 4.56 : Debugger setting

4.9.5 Exclude some directories from the build target

Remove the following directories included in the build target due to project conversion.

(1) The lib directory included in the BLE FIT module

Exclude the BLE FIT module lib directory from the build target on GUI.
Right-click on the "src/smc_gen/r_ble_rx23w/lib" directory and select "Options…".
On the Option window, check the "Exclude from build" box.
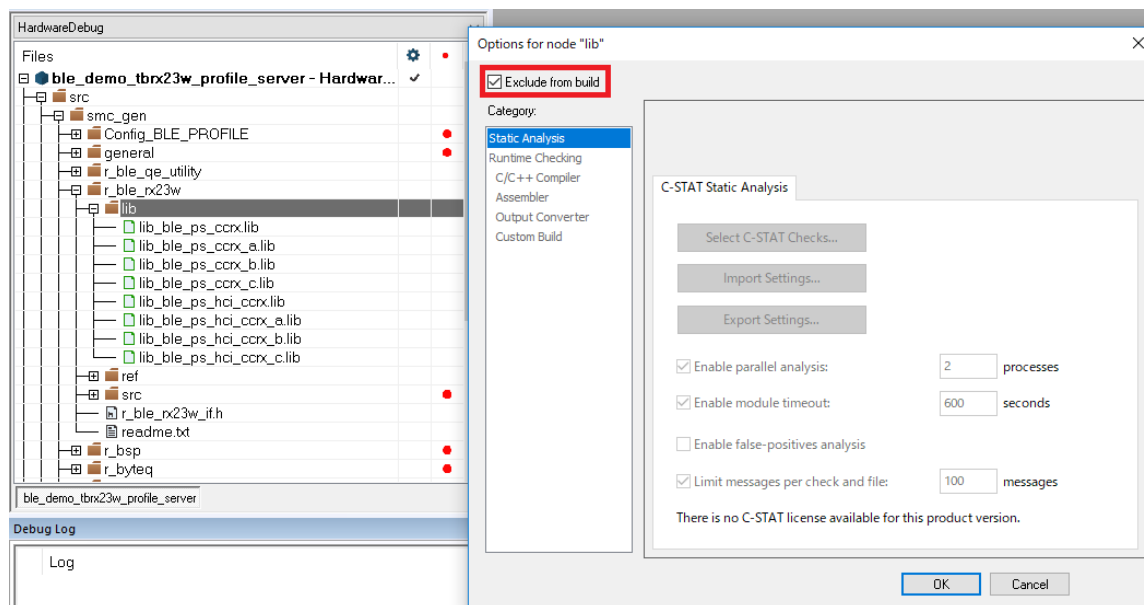
Figure 4.57 : Lib directory option

(2) Trash directory

If the build target includes the "trash" directory, right-click the directory and select "Options…".

On the Option window, check the "Exclude from build" box.
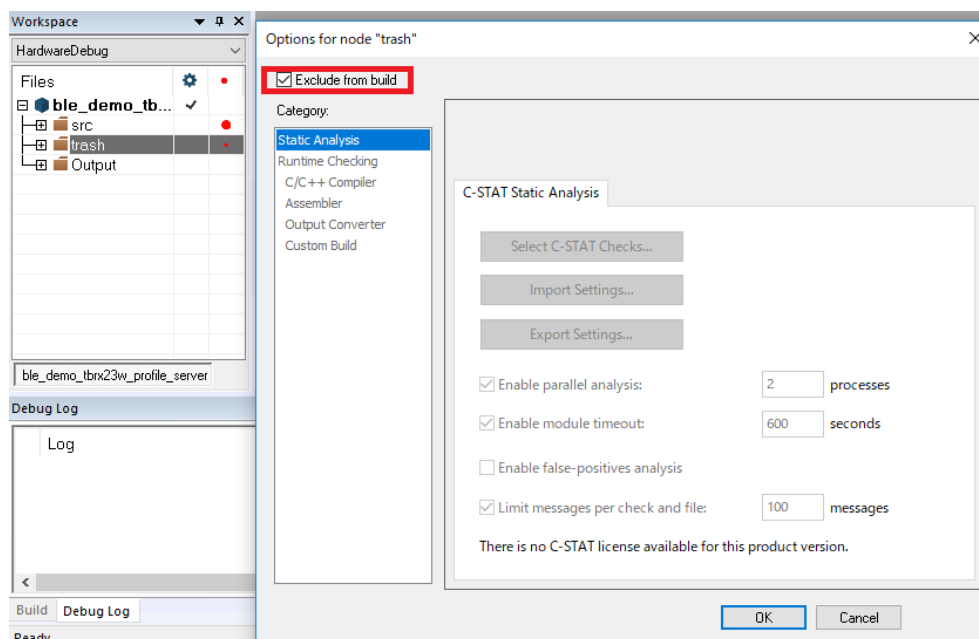
Figure 4.58 : Trash directory option

4.9.6 Build the project and download the firmware

Click "Project >> Rebuild All" to build the project.

After completion of the build, select "[emulator name] >> Hardware setup …".

The following sample use the "E2/E2 Lite" as emulator.
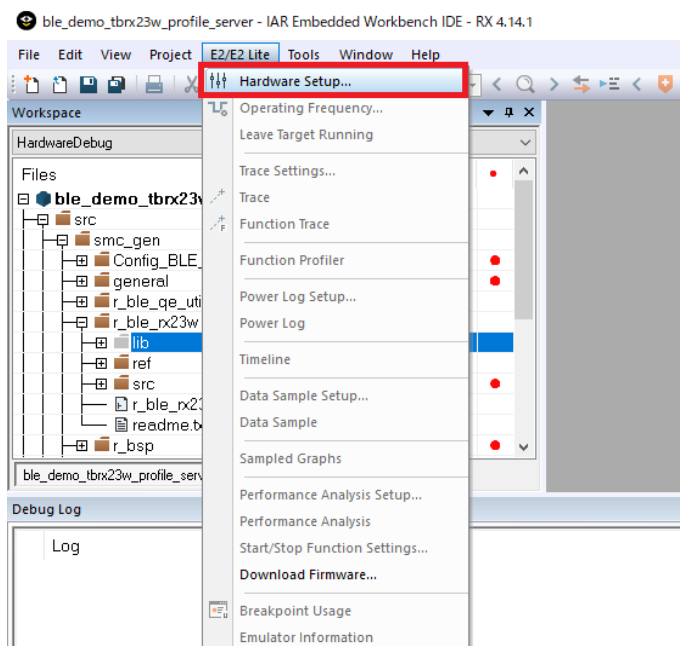
Figure 4.59 : Hardware setup (1)

Configure the hardware settings on the "Hardware Setup" window. If there is no need to change from the default settings, just click "OK".
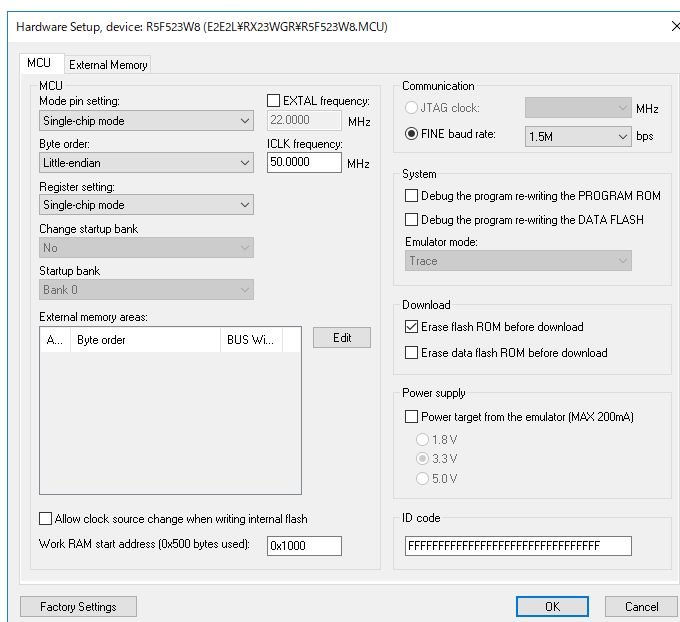
Figure 4.60 : Hardware setup (2)

Click "Project >> Download" and download the firmware.

5. Application Guide

For more information about how to create the BLE application, refer to "RX23W Group Bluetooth Low Energy Application Developer's Guide (R01AN5504)".

6. Renesas FreeRTOS BLE task

It assumes that Renesas FreeRTOS BLE application has the task configuration as Figure **6**.**1**. The BLE task consists of the BLE Core Stack features, the Bluetooth LE communication control and the application processing callbacks. The BLE GATT App task consists of the GATT service applications.
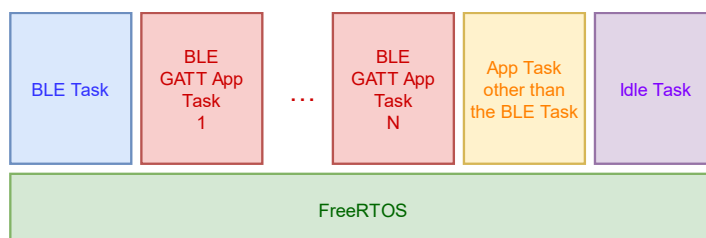
Figure 6.1 : Task configuration on Renesas FreeRTOS

6.1 Call available API

Call available APIs vary between the BLE task and the BLE GATT App task.

■ Call available APIs from the BLE task

- GAP API (R_BLE_GAP_XXX)
- GATT Server API (R_BLE_GATTS_XXX)
- GATT Client API   (R_BLE_GATTC_XXX)
- Vendor Specific API (R_BLE_VS_XXX)
- L2CAP API (R_BLE_L2CAP_XXX)
- Abstraction API (R_BLE_ABS_XXX)
- Discovery API   (R_BLE_DISC_XXX)
- R_BLE_[GATT service abbreviation + S(server selected) or C(client selected)]_YYY

■ Call available APIs from the BLE GATT App task

- R_BLE_[GATT service abbreviation + S(server selected) or C(client selected)]_YYY

All tasks can call BLE_LOG_XXX, APIs using the FIT module other than BLE (R_BLE_LPC_XXX、 R_BLE_TIMER_XXX、R_BLE_SECD_XXX、R_BLE_CLI_XXX、R_BLE_CMD_XXX、 R_BLE_BOARD_XXX).

When the BLE GATT App task calls the GATT service API, the GATT communication is processed with the Core Stack in the BLE task. Figure **6**.**2** shows Bluetooth LE communication that two BLE GATT App tasks control the GATT services and send notification of the GATT service characteristic on the multi-task environment including the BLE task and the BLE GATT App task.
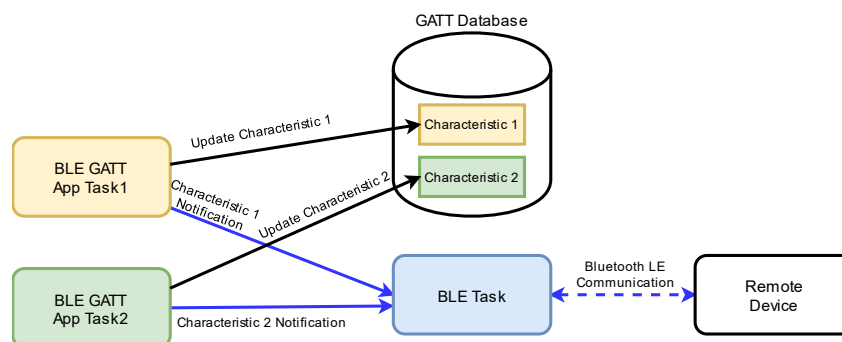
Figure 6.2 : BLE task and BLE GATT App task

6.2 BLE task generation

The BLE task is generated as main task or a task other than main task (hereafter referred to as the subtask).

6.2.1 Generation as main task

If the BLE task is generated as main task, change the following code.

(1) Change the parameters of main task generation

The main task generation code by xTaskCreate() is output to Processing_Before_Start_Kernel() in src/frtos_startup/freertos_start.c . Include rtos/r_ble_rtos.h that is header for the BLE task control in freertos_start.c and change the parameters (Table **6.1**) for main task generation as Figure **6.3**.

Table 6.1 : Main task generation parameters

Parameter	Description
Stack size	1024
	Stack size is multiplied by 4 (4096bytes).
Priority	6 (higher than the subtask)
Task handle	g_ble_task_hdl

```c
#include "rtos/r_ble_rtos.h"

/* some code is omitted.*/

/************** task creation **************************/
/* Main task. */
ret = xTaskCreate(main_task, "MAIN_TASK", 1024, NULL, 6, &g_ble_task_hdl);
if (pdPASS != ret)                         Stack Size    Priority    Task handle
{
    while(1)
    {
        /* Failed! Task cannot be created. */
    }
}
```

Figure 6.3 : BLE task generation code as main task

(2) Call to app_main()

Change main_task() in src/"{Project Name}".c output by code generation to call app_main().

```c
#include "FreeRTOS.h"
#include "task.h"

void app_main(void);

void main_task(void *pvParameters)
{
    app_main();

    vTaskDelete(NULL);
}
```

Figure 6.4 : Calling app_main() in main_task()

(3)   Changes to app_main.c

Include "FreeRTOS.h" and "task.h" to call FreeRTOS API and change the main loop in app_main() to pass control to other tasks when the BLE Protocol Stack is idle.

```c
#include "FreeRTOS.h"
#include "task.h"

/* some code is omitted. */

/* main loop */
while (1)
{
    /* some code is omitted. */

    /* Process Event */
    R_BLE_Execute();

    if(0 != R_BLE_IsTaskFree())
    {
      ulTaskNotifyTake(pdFALSE, portMAX_DELAY);
    }
}
```

Figure 6.5 : Changes to the main loop

(4)   Subtask generation

If subtask is generated by "4.5.5(2) Subtask Declaration", call Object_init_manual() from the application code such as "{Project Name}.c" or app_main.c.

```c
void app_main(void)
{
    /* some code is omitted. */

    extern void Object_init_manual (void);
    Object_init_manual();

    /* main loop */
    while (1)
    {
        /* some code is omitted. */
    }
}
```

Figure 6.6 : Subtask generation

6.2.2 Generation as subtask

This section describes how to generate the BLE task as subtask according to "4.5.5(2) Subtask Declaration".

(1) Register BLE task

If the BLE task is registered by FreeRTOS Object configuration, configure the parameters in Table **6.1** as Figure **6.7**. The stack size is multiplied by 4 (4096 bytes).

+/-	Initialize	Task Code	Task Name	Stack Size	Task Handler	Parameter	Priority	Heap Usage
⊖	manual ∨	ble_task	ble_task	1024	g_ble_task_hdl	NULL	6	0
⊕								

(Tabs: Heap Estimation | **Tasks** | Semaphores | Queues | Software Timers | Event Groups | Stream Buffers | Message Buffers)

Figure 6.7 : BLE task registration in FreeRTOS Object configuration

(2) Call to app_main()

Change ble_task() in src/rtos_skeleton/ble_task.c output by code generation to call app_main() and include "FreeRTOS.h" and "task.h" to call FreeRTOS API. An example of ble_task.c is shown below.

```c
/* Start user code for import. Do not edit comment generated here */
#include "FreeRTOS.h"
#include "task.h"

void app_main(void);

/* End user code. Do not edit comment generated here */

void ble_task(void * pvParameters)
{
    /* Start user code for function. Do not edit comment generated here */
    app_main();

    vTaskDelete(NULL);
    /* End user code. Do not edit comment generated here */
}
```

Figure 6.8 : Sample of ble_task.c

(3) Changes to app_main.c

Refer to "6.2.1(3) Changes to app_main.c".

(4)   Include the BLE task control header and delete task handle

Delete the BLE task handle definition in src/frtos_startup/freertos_object_init.c and include the BLE task control header rtos/r_ble_rtos.h .

```
#include "FreeRTOS.h"
#include "freertos_start.h"
#include "../frtos_skeleton/task_function.h"
#include "task.h"
/* Include the BLE task control header */
#include "rtos/r_ble_rtos.h"

/* Some code is omitted. */

void Kernel_Object_init (void);
void Object_init_manual (void);
BaseType_t ret;
/* Delete the BLE task handle definition.  */
/* TaskHandle_t g_ble_task_hdl = NULL; */
```

Figure 6.9 : Changes to freertos_object_init.c

(5)   Subtask generation

Call Object_init_manual() from the main task code such as "{Project Name}.c" . An example of starting subtasks and do nothing in main loop is shown below.

```
#include "FreeRTOS.h"
#include "task.h"

void main_task(void *pvParameters)
{
    /* Create all other application tasks here */
    void Object_init_manual (void);
    Object_init_manual();
    while(1)
    {
    }
}
```

Figure 6.10 : Subtask generation

6.3   BLE GATT App task generation

This section describes how to generate according to "4.5.5(2) Subtask Declaration".

(1)   Register BLE GATT App task

Figure **6.11** shows a sample of BLE GATT App task registration in FreeRTOS Object configuration.

Heap Estimation	Tasks	Semaphores	Queues	Software Timers	Event Groups	Stream Buffers	Message Buffers		

+/-	Initialize	Task Code	Task Name	Stack Size	Task Handler	Parameter	Priority	Heap Usage
⊖	manual ∨	bas_task	bas_task	256	g_bas_task	conn_hdl	4	0
⊖	manual ∨	lss_task	lss_task	256	g_lss_task	conn_hdl	4	0
⊕								

Figure 6.11 : BLE GATT App task registration in FreeRTOS Object configuration

(2)   Header files included in BLE GATT App task file

BLE GATT App task is implemented in src/rtos_skeleton/"Task Code".c output by code generation. BLE GATT App task needs the following header files.

- GATT service API header   ../Config_BLE_Profile/r_ble_[GATT service abbreviation].h
- BLE task control header   rtos/r_ble_rtos.h
- FreeRTOS API header   FreeRTOS.h and task.h
- Subtask header   ../ frtos_skeleton/task_function.h

```
/* FreeRTOS API header */
#include "FreeRTOS.h"
#include "task.h"
/* GATT service API header */
#include "../Config_BLE_Profile/r_ble_bas.h"
/* BLE task control header */
#include "rtos/r_ble_rtos.h"
```

Figure 6.12 : Headers included in BLE GATT App task

The event definitions for task communication to other subtask are implemented in the subtask header (task_function.h). A sample of subtask header is shown below.

```
/* some code is omitted. */

/* bas */
#define BAS_WAIT_EN_CCCD        (0x0001)
/* lss */
#define LSS_NOTIF_DIS_CCCD      (0x0000)
#define LSS_WAIT_EN_CCCD        (0x0001)
#define LSS_WAIT_DIS_CCCD       (0x0002)
#define LSS_WAIT_PUSH_SW        (0x0004)
#define LSS_WAIT_WR_BLINK       (0x0008)
```

Figure 6.13 : Sample of subtask header

(3)   Edit the task generation API (Object_init_manual)

If the parameter is passed to the task function when the task is generated according to (1), prepare a static variable for storing the parameter and specify the static variable in xTaskCreate. An example of task generation API passing a connection handle when establishing a connection is shown below.

```c
static uint16_t gs_conn_hdl;

/* some code is omitted. */
void Object_init_manual (uint16_t conn_hdl)
{
    gs_conn_hdl = conn_hdl;

    /************** task creation **************************/
    ret = xTaskCreate(bas_task, "bas_task", 256, &gs_conn_hdl, 4, &g_bas_task);
    /* some code is omitted. */
    ret = xTaskCreate(lss_task, "lss_task", 256, &gs_conn_hdl, 4, &g_lss_task);
    /* some code is omitted. */
} /* End of function Object_init_manual()*/
```

Figure 6.14 : Sample of passing a parameter to the generated task function

(4)   Call to the task generation API (Object_init_manual)

Refer to "6.2.1(4) Subtask generation".

6.4 Wake up BLE task from another task

If other task wakes the BLE task up, include the BLE task control header rtos/r_ble_rtos.h and call
R_BLE_RTOS_WakeTask(). An example of task switching from BLE GATT App task (LED and Switch
service) to BLE task is shown below.

```c
/* Include BLE task control header */
#include "rtos/r_ble_rtos.h"

/* some code is omitted. */
static uint16_t gs_conn_hdl;

void lss_task(void * pvParameters)
{
    uint8_t push_state;
    ble_status_t retval;

    /* some code is omitted. */

    retval = R_BLE_LSS_NotifySwitchState(gs_conn_hdl, &push_state);
    if(BLE_SUCCESS == retval)
    {
        R_BLE_CLI_Printf("R_BLE_LSS_NotifySwitchState \n");
        R_BLE_RTOS_WakeTask();
    }
    /* some code is omitted. */
}
```

Figure 6.15 : Sample of task switching from BLE GATT App task to BLE task

6.5 Wake up BLE task from interrupt other than RF

The BLE FIT module provides mechanisms that task is waked up from registered function by the SCI interrupt using the console function or the IRQ interrupt using LED & SW.

6.5.1 Register task waked up by command input

If a task is waked up with the SCI interrupt by command input, included rtos/r_ble_rtos.h and call R_BLE_CLI_RegisterEventCb() in application initialization to register a callback for waking up the task. An example of registering a function that wakes the BLE task up from ISR to communicate with Bluetooth LE by command input is shown below.

```c
#include "rtos/r_ble_rtos.h"

/* some code is omitted */

void app_main(void)
{
    /* Initialize BLE */
    R_BLE_Open();

    /* some code is omitted */

    /* Configure Command Line */
    R_BLE_CLI_Init();
    R_BLE_CLI_RegisterCmds(gsp_cmds, ARRAY_SIZE(gsp_cmds));
    /* Register task waked up by command input */
    R_BLE_CLI_RegisterEventCb(R_BLE_RTOS_WakeTaskFromIsr);
    R_BLE_CMD_SetResetCb(ble_host_stack_init);
```

Figure 6.16 : Sample of registering a task waked up by command input

6.5.2 Wake BLE task up using LED & SW

If a task is waked up with the IRQ interrupt by pressing SW, included rtos/r_ble_rtos.h and call R_BLE_CLI_RegisterEventCb() in application initialization to register a callback for waking up the task. An example of registering a function that wakes the BLE task up from ISR to communicate with Bluetooth LE by pressing SW is shown below.

```c
#include "rtos/r_ble_rtos.h"

/* some code is omitted */

void app_main(void)
{
    /* Initialize BLE */
    R_BLE_Open();

    /* Configure the board */
    R_BLE_BOARD_Init();
    R_BLE_BOARD_RegisterSwitchCb(BLE_BOARD_SW2, sw_cb);
    /* Register task waked up by pressing SW */
    R_BLE_BOARD_RegisterSwitchEventCb(R_BLE_RTOS_WakeTaskFromIsr);
```

Figure 6.17 : Sample of registering a task waked up by pressing SW

7. Tools

The BLE FIT module package available for download from the RX23W Group BLE Module Firmware Integration Technology Application Note site (https://www.renesas.com/us/en/document/scd/rx23w-group-ble-module-firmware-integration-technology-application-note) provides the tools for the HCI mode firmware. For more details centering the HCI mode, see "Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205), 6. HCI Mode".

REN_r01an4860xxXXXX-rx23w-ble-fit_SCD_yyyymmdd.zip	XXXX: Revision number, yyyymmdd: release date
├── FITDemos\	Sample application folder
├── FITModules\	FIT module folder
├── utilities\	Utility folder
├── **BDAddrWriter.zip**	Public BD address writing tool for HCI mode
└── **CLVALTune.zip**	Operation tool for calibration for HCI mode

Figure 7.1 : HCI mode tool

7.1 BDAddrWriter
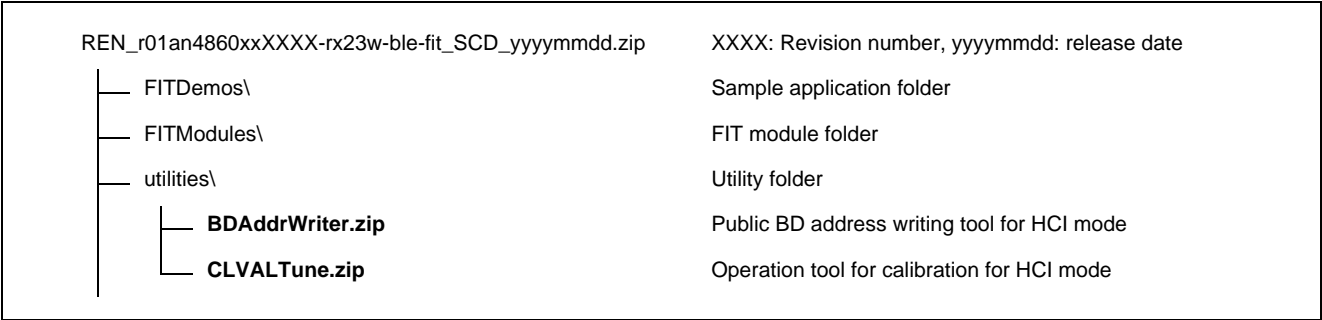
BDAddrWriter configures BD_ADDR in the board for HCI mode. For more details, see "Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205), 5.6.4.2 Write to data area using BDAddrWriter tool".

7.2 CLVALTune

CLVALTune is a tool used for calibration operation. Refer to "RX23W Group Tuning procedure of Bluetooth dedicated clock frequency(R01AN4762)" for details.

8. Demo Projects

The applications shown in Table **8.1** are provided as the BLE demo projects.

The demo projects include a main function that uses the BLE FIT module and the related FIT modules (e.g., r_bsp).

Table 8.1 : Demo Projects

Demo Project	Description
ble_demo_rsskrx23w_profile_client	GATT Client demo application for RSSK.
ble_demo_rsskrx23w_profile_server	GATT Server demo application for RSSK.
ble_demo_rsskrx23w_uart_hci	HCI mode demo application for RSSK.
ble_demo_tbrx23w_profile_client	GATT Client demo application for Target Board for RX23W.
ble_demo_tbrx23w_profile_server	GATT Server demo application for Target Board for RX23W.
ble_demo_tbrx23w_uart_hci	HCI mode demo application for Target Board for RX23W.
ble_demo_tbrx23wmodule_profile_client	GATT Client demo application for Target Board for RX23W module.
ble_demo_tbrx23wmodule_profile_server	GATT Server demo application for Target Board for RX23W module.
ble_demo_tbrx23wmodule_uart_hci	HCI mode demo application for Target Board for RX23W module.
ble_demo_tbrx23w_FreeRTOS_multi_services	GATT Server demo application on Renesas FreeRTOS for Target Board.

The above FITDemos use the same version of the r_ble_rx23w as this document and the r_ble_qe_utility version 1.10. The ble_demo_tbrx23wmodule_profile_xxx use the r_gpio_rx version 3.80. The version of other FIT modules is the same as the one included in RX Driver Package v1.29 (https://www.renesas.com/software-tool/rx-driver-package).

The GATT server and the GATT Client demo projects support the command line interface. By connecting the board and PC with serial interface, it allows to input commands and output debug messages on a serial terminal.

8.1　GATT Server demo project

The GATT Server demo works as below.

● After starting, it starts advertising and waits for a command.

● By scanning from a remote device, it is detected by the "RBLE-DEV" device name.

Figure 8.1 : Scan result example
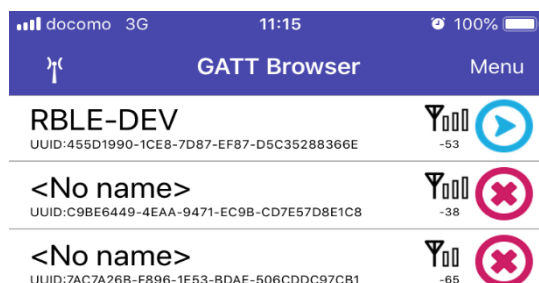

● When connected, it stops advertising.

● The followings are detected by searching GATT services from a remote device.

➢ LED Switch Service(LSS, UUID : 58831926-5F05-4267-AB01-B4968E8EFCE0)

➢ Switch State Characteristic(UUID : 58837F57-5F05-4267-AB01-B4968E8EFCE0)

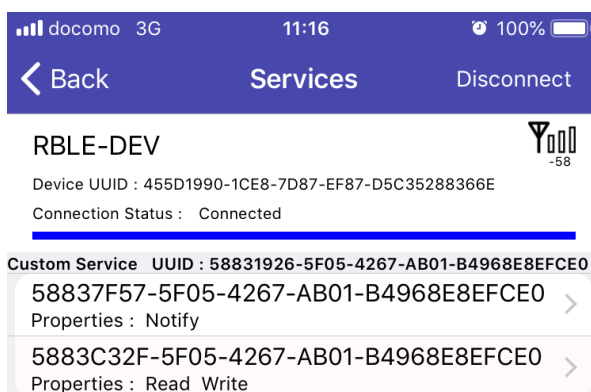➢ LED Blink Rate Characteristic(UUID : 5883C32F-5F05-4267-AB01-B4968E8EFCE0)

Figure 8.2 : Service Discovery result

- If the LED Switch Service second parameter in the gs_gatt_service variable in the gatt_db.c is set to **BLE_GATT_DB_SER_SECURITY_UNAUTH │ BLE_GATT_DB_SER_SECURITY_ENC**, the GATT Server requests pairing to access to the Characteristic in the LED Switch Service. If **0** is set, pairing is not necessary.

```
static const st_ble_gatts_db_serv_cfg_t gs_gatt_service[] =
{

    /* some code is omitted. */

    /* LED Switch */
    {
        /* Num of Services */
        {
            1,
        },
        /* Description */
        BLE_GATT_DB_SER_SECURITY_UNAUTH | BLE_GATT_DB_SER_SECURITY_ENC,     Pairing is necessary.
        /* Service Start Handle */
        0x0010,
        /* Service End Handle */
        0x0015,
        /* Characteristic Start Index */
        6,
        /* Characteristic End Index */
        7,
    },
};
```

```
static const st_ble_gatts_db_serv_cfg_t gs_gatt_service[] =
{

    /* some code is omitted. */

    /* LED Switch */
    {
        /* Num of Services */
        {
            1,
        },
        /* Description */
        0,                                            Pairing is not necessary.
        /* Service Start Handle */
        0x0010,
        /* Service End Handle */
        0x0015,
        /* Characteristic Start Index */
        6,
        /* Characteristic End Index */
        7,
    },
};
```

Figure 8.3 : The security setting of the access to the LED Switch Service.

- After enabling the notification in the Switch State characteristic, the notification packet is sent to the remote device by pushing the SW1 on the board.

- By writing a number to the LED Blink Rate characteristic, the LED blinks at the number x 100ms interval. The LED turns off by writing zero to the characteristic.

- When disconnected, it restarts advertising.

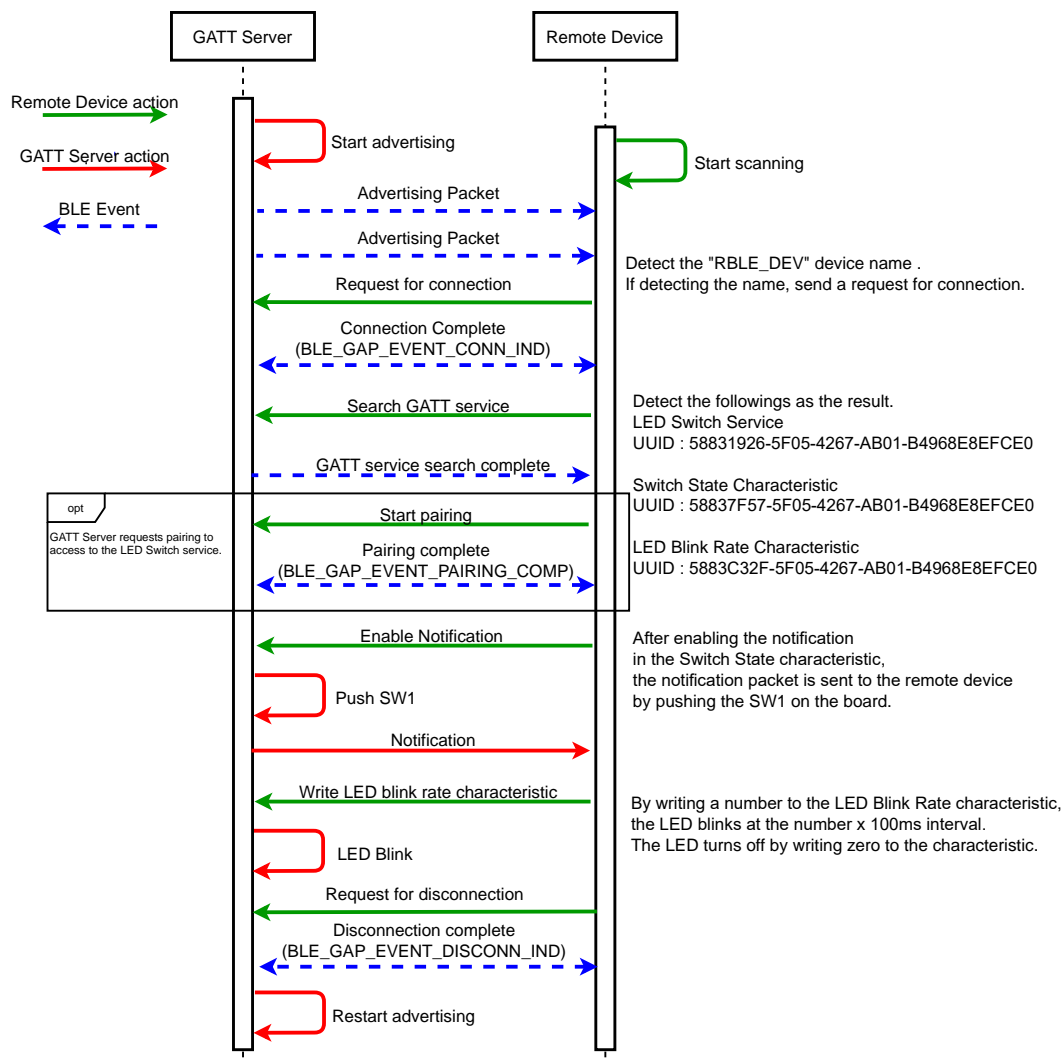Figure **8.4** shows usage example for the GATT Server demo project and a remote device.

Figure 8.4 : Usage example for the GATT Server demo project and a remote device

8.2   GATT Client demo project

The GATT Client demo works as below.

- After starting, it waits for a command.

- If you would like to confirm the presence of the GATT Server demo project, execute the "gap scan" command by your terminal software and the advertising packet are received. Refer to "Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205), 5.1.1.1 (2) Scan command" for the details of the "gap scan" command. After receiving the advertising, stop scan by press CTRL + C on the terminal software.

- Execute the "gap conn" command by the terminal software to connect with the GATT Server demo project. Refer to "Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205), 5.1.1.1 (3) Connection command" for the details of the "gap conn" command.

- When connected, packet data length exchange is done.

- When the packet data length changed, it sends the MTU exchange request.

- Receiving the MTU exchange response, it starts searching GATT services in the remote device.

- If the GATT Server demo project is set to request pairing (BLE_GATT_DB_SER_SECURITY_UNAUTH | BLE_GATT_DB_SER_SECURITY_ENC: Figure **8.3**), pairing needs to be done previous to the LED Switch client command execution. Input the "gap auth" command to start pairing. Refer to "Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205), 5.1.1.1 (9) Authentication command" for the details of the "gap auth" command.

- After the completion of the above procedure, it allows to use the following LED Switch client commands.

  - ➢ Enable / Disable Notification
    lsc set_switch_state_ntf [connection handle] [0(disabled) or 1(enabled)]

  - ➢ Write the LED blink rate
    lsc write_led_blink_rate [connection handle] [blink_rate]
    The LED blinks at the [blink_rate] x 100ms. Valid range is 0-255.
    The LED turns off by writing 0.

- Execute the "gap disconn" command by the terminal software to disconnect the GATT Server demo project. Refer to "Bluetooth Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205), 5.1.1.1 (4) Disconnection command" for the details of the "gap disconn" command.

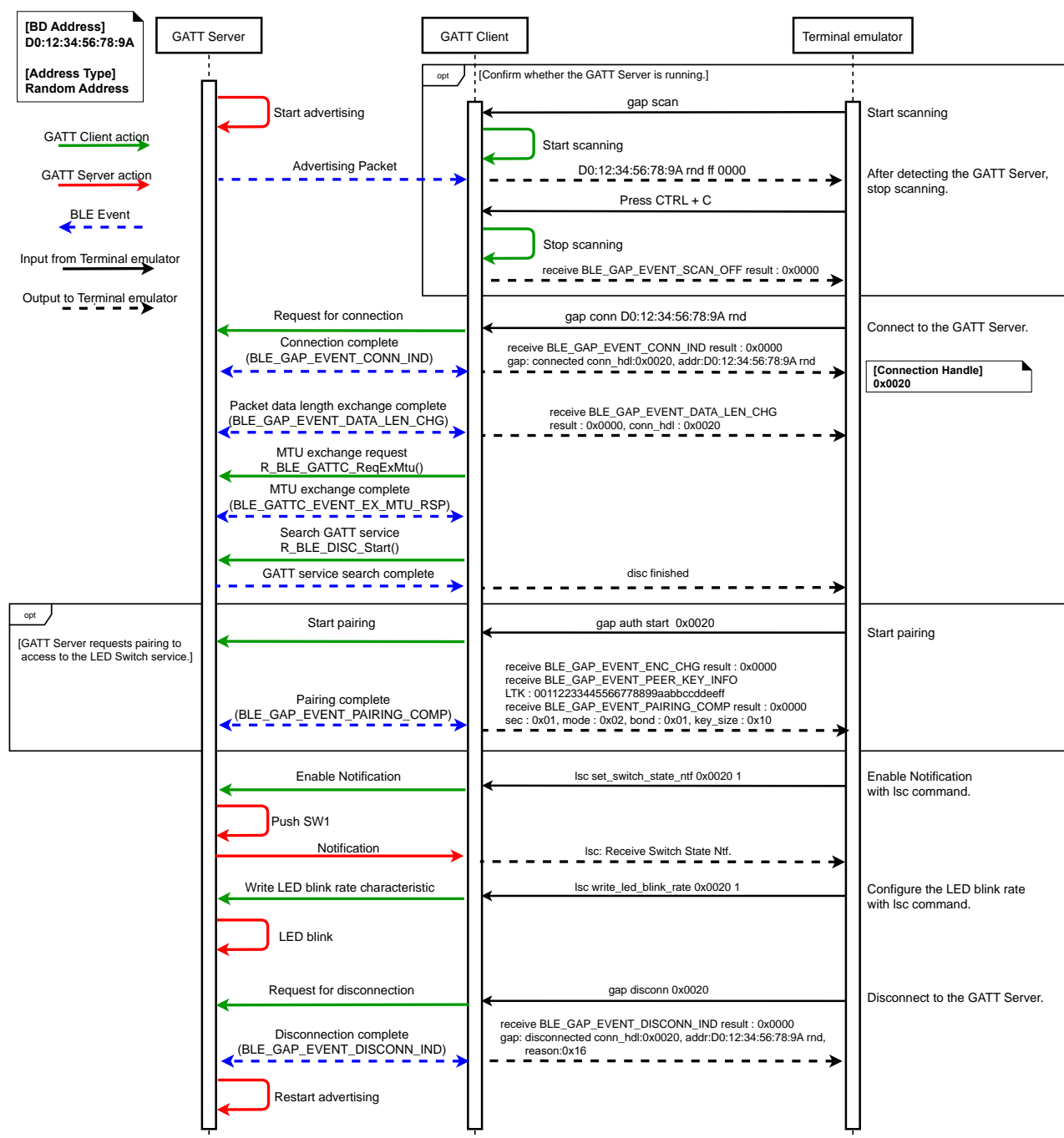Figure **8.5** shows usage example for the GATT Client demo project and the GATT Server demo project.

Figure 8.5 : Usage example for the GATT Client demo project and the GATT Server demo project

8.3   Renesas FreeRTOS BLE demo project

The GATT Server demo works as below. This project consists of Battery Service task (BAS task) and LED & SW Service task (LSS task).

● After starting, it starts advertising and waits for a command.

● By scanning from a remote device, it is detected by the "RBLE-DEV" device name.

Figure 8.6 : Scan result example

● When connected, it stops advertising.

● The followings are detected by searching GATT services from a remote device.

  ➢ Battery Service (BAS, UUID : 180F)

  ➢ Battery Level Characteristic (UUID : 2A19)

  ➢ LED Switch Service(LSS, UUID : 58831926-5F05-4267-AB01-B4968E8EFCE0)

  ➢ Switch State Characteristic(UUID : 58837F57-5F05-4267-AB01-B4968E8EFCE0)

  ➢ LED Blink Rate Characteristic(UUID : 5883C32F-5F05-4267-AB01-B4968E8EFCE0)
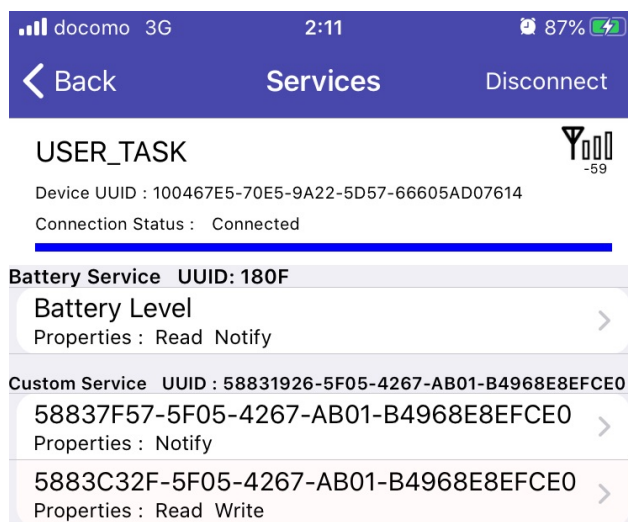
Figure 8.7 : Service Discovery result

● After enabling the notification in the Battery Level characteristic, the notification packet including a dummy Battery Level is sent to the remote device from RX23W every 1 second.

● After enabling the notification in the Switch State characteristic, the notification packet is sent to the remote device by pushing the SW1 on the board.

● By writing a number to the LED Blink Rate characteristic, the LED blinks at the number x 100ms interval. The LED turns off by writing zero to the characteristic.

● When disconnected, it restarts advertising.

Figure **8.8** shows usage example for the Renesas FreeRTOS BLE demo project and a remote device.
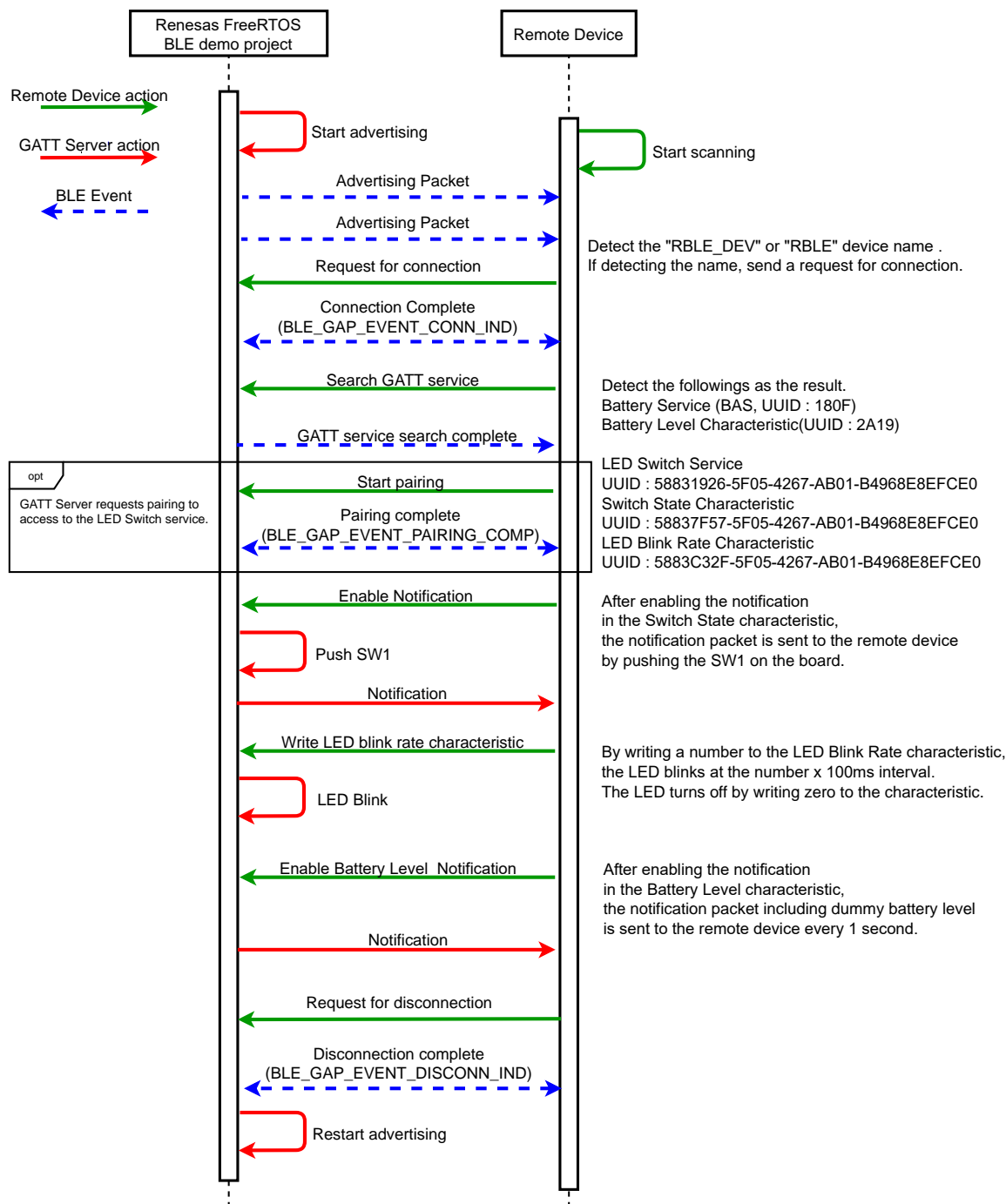
Figure 8.8 : Usage example for the Renesas FreeRTOS BLE demo project and a remote device

8.4   HCI mode demo project

The HCI mode demo project waits for an HCI command after starting. Input an HCI command for the RF characteristic evaluation or connect to BTTS(Bluetooth Test Tool Suite : R01AN4554).

8.5   Operation confirmation on CS+

Using the Renesas common project file (rcpc file) in demo project, you can do an operation confirmation of the demo projects on CS+. The version of CS+ used for operation confirmation is V8.05 .
Open the demo project on CS+ according to the following procedure.

1. Run CS+.

2. Click "File >> Open File", set the filename extension to "Project File foe MCU Simulator Online/ e^2 studio (*.rcpc) and select the demo project rcpc file.
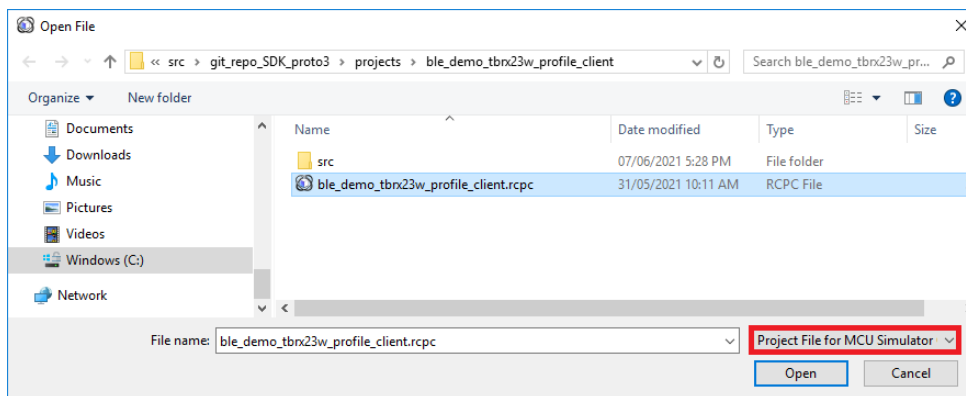
Figure 8.9 : Open the demo project rcpc file

3. Press "OK" on "Project Convert Settings" screen.
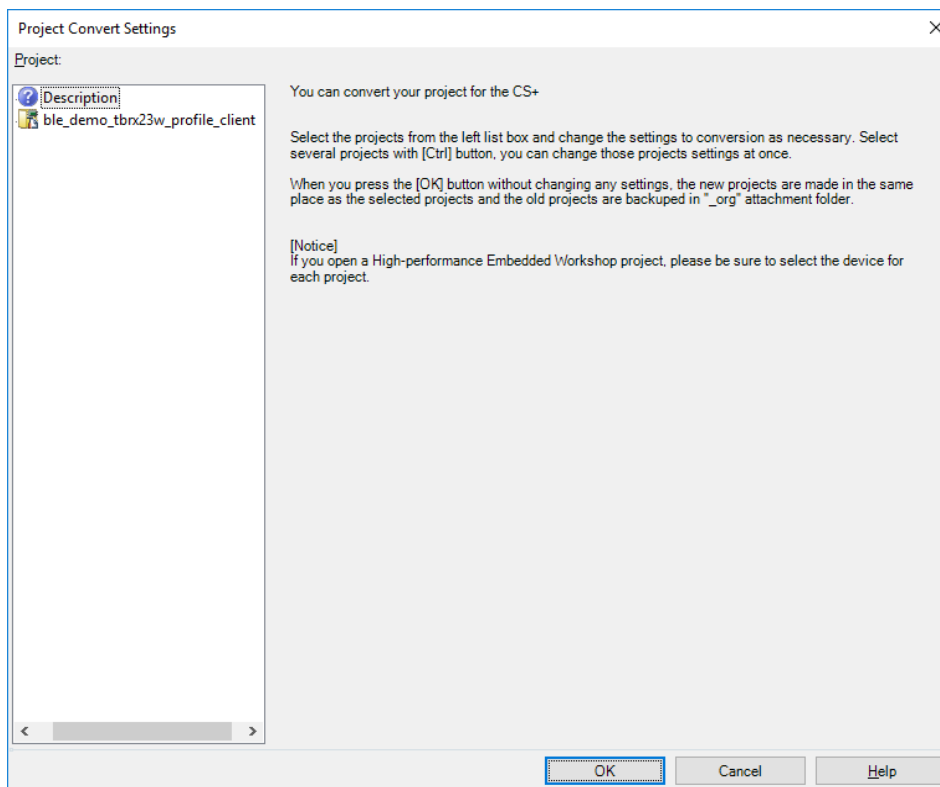
Figure 8.10 ： Project Convert Settings

4. After the project conversion, you can do an operation confirmation on CS+.
   For the CS+ operation, refer to the following Users Manual.
   https://www.renesas.com/software-tool/cs
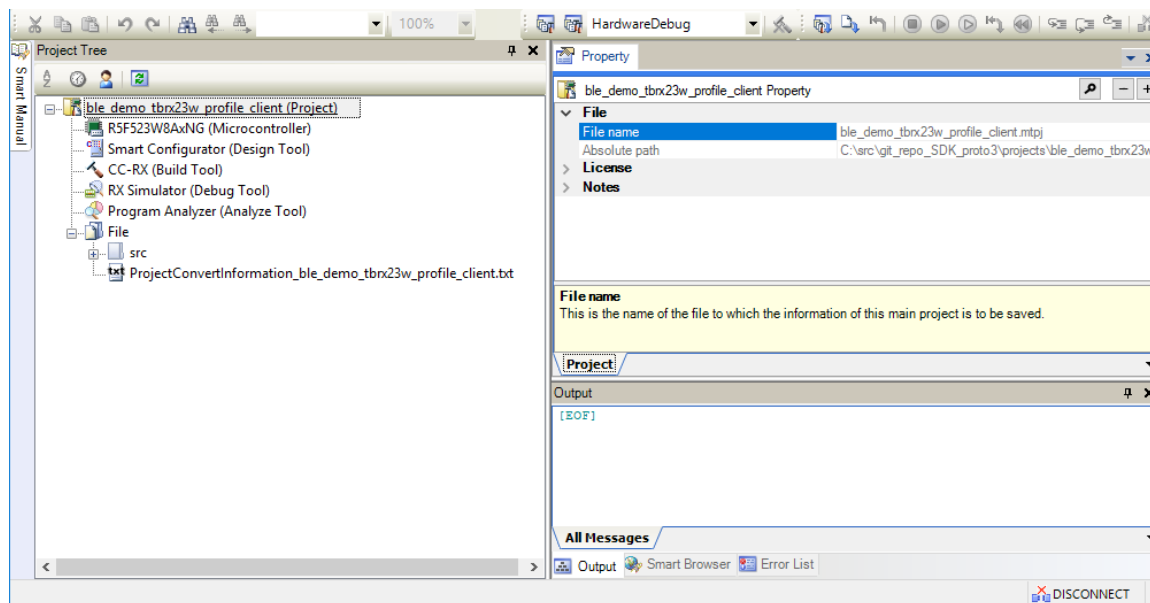
Figure 8.11 : Demo project on CS+

8.6   Importing the demo project

The sample application provided with this document may be imported into e² studio using the steps in this section.

(1)   Create a RX23W project including the BLE FIT module(r_ble_rx23w), referring to 4.1 Create a new project and 4.4 Adding FIT modules. Or prepare an existing RX23W project including the BLE FIT module.

(2)   Select "Components" tab, right-click "r_ble_rx23w" and click "Download and import sample projects".
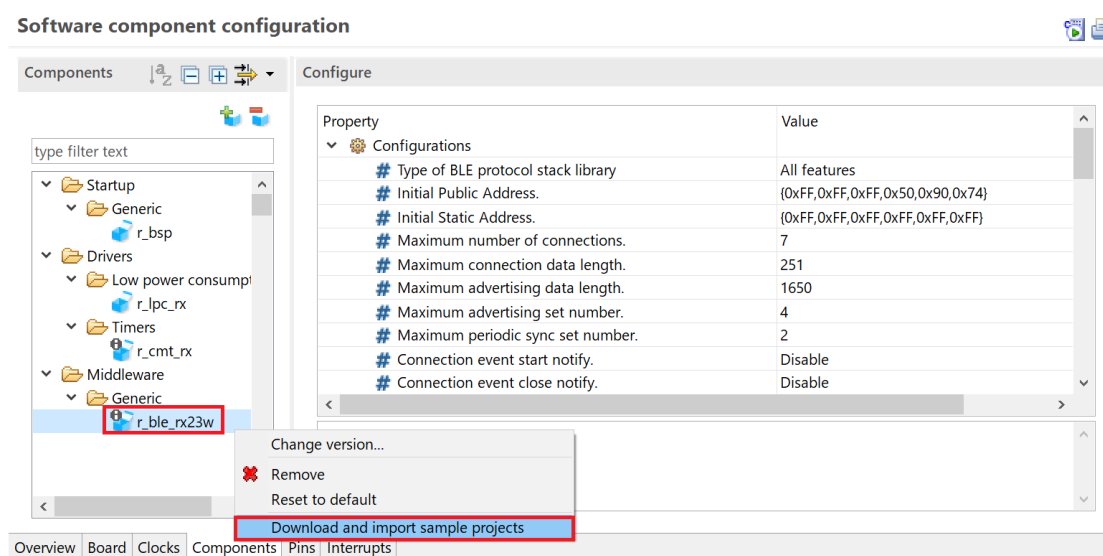
Figure 8.12 : Download and import sample projects (1)

(3)   After Smart Browser displays the BLE FIT module application note(Title : RX23W Group BLE Module Firmware Integration Technology Application Note, Document No : R01AN4860EJYYYY (YYYY : version)), right-click this and select "Sample Code (import projects).
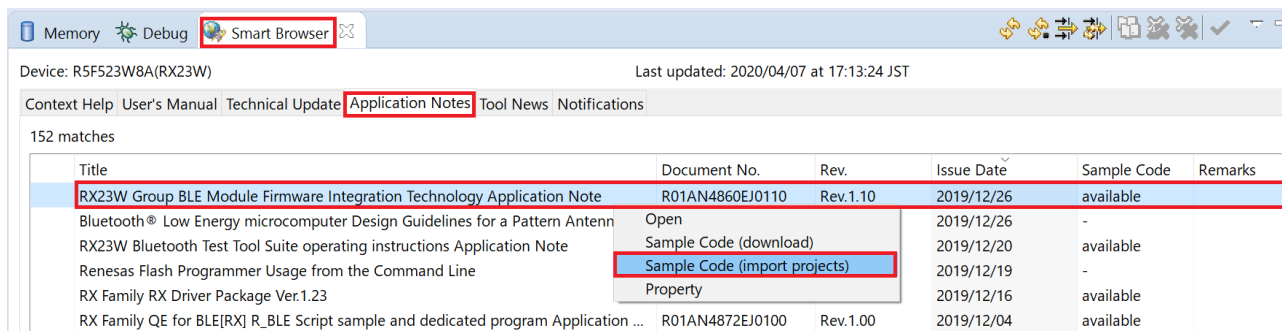
Figure 8.13 : Download and import sample projects (2)

(4) Input the location where r01an4860xxYYYY-rx23w-ble-fit.zip(YYYY : version) is downloaded. After the download has been completed, "Select import package" window is displayed. Select the BLE demo project.
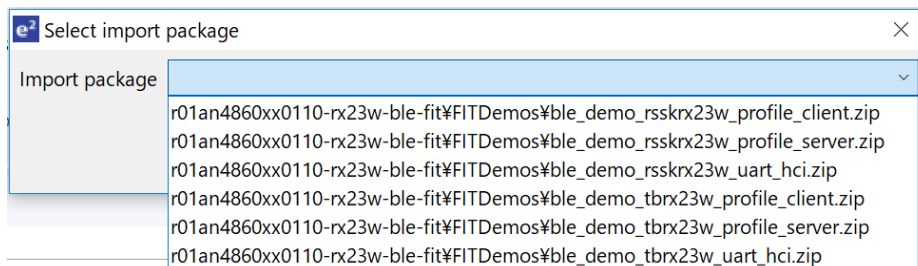
Figure 8.14 ： Download and import sample projects (3)

(5) When "Finish" button on "Import" window has been pressed, the demo project is imported.
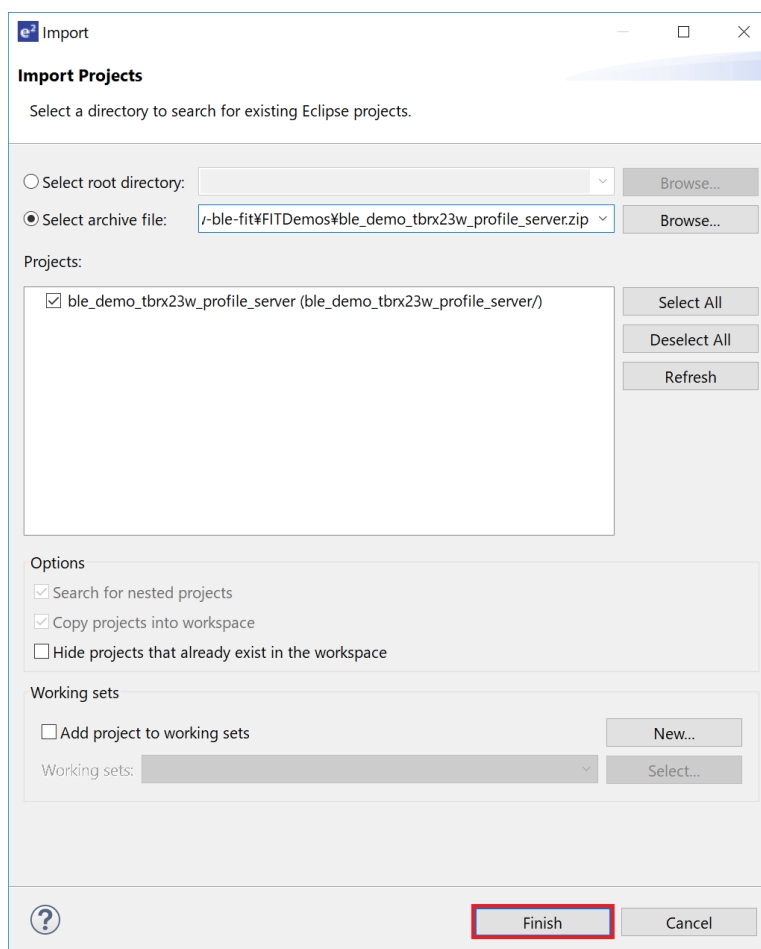
Figure 8.15 : Download and import sample projects (4)

9. Appendices

9.1 Confirmed Operation Environment

This section describes confirmed operation environment for the BLE FIT module.

Table 9.1 : Confirmed Operation Environment (Rev.2.20)

Item	Contents
Integrated development environment[*1]	Renesas Electronics e^2 studio Version 7.8.0 (32bit) Renesas Electronics e^2 studio 2021-04 (64bit) IAR Embedded Workbench for Renesas RX 4.14.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.08.00 IAR C/C++ Compiler for Renesas RX version 4.14.1 Compiler option: The default settings of the integrated development environment.
Endian	little endian
Revision of the module	Rev 2.20
RTOS	Renesas FreeRTOS version 10.0.0.4
Board used	Target Board for RX23W (RTK5RX23W0C00000BJ) Target Board for RX23W module (RTK5RX23W0C01000BJ) Renesas Solution Starter Kit for RX23W (RTK5523W8AC00001BJ)

*1 : If your project is for RX23W module, use e^2 studio 2021-01 or later.

9.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file "platform.h".

   A: The FIT module may not be added to the project properly. When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "Board Support Package Module Using Firmware Integration Technology (R01AN1685)".

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_ble_rx23w module.

   A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

(3) Q: I have added the FIT module to the project and built it. Then I got an error for when the configuration setting is wrong.

   A: The setting in the file "r_ble_rx23w_config.h" may be wrong. Check the file "r_ble_rx23w_config.h". If there is a wrong setting, set the correct value for that. Refer to 2.6, Configuration Overview for details.

(4) Q: I have added the FIT module to the project and built it. Then I got an error : Could not open source file .

   A: The file path may be too long. Shorten the file path.

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.00	Aug.23.2019	—	First edition issued.
1.01	Oct.31.2019	20	Move the description of the command to "Bluetooth® Low Energy Protocol Stack Basic Package: User's Manual (R01UW0205) ".
		26	Update "4. How to use".
		58	Add how to use the demo application to "7. Demo Project".
		Library	Add Mesh support.
		Library	Support the pairing when changing the BD_ADDR by R_BLE_VS_SetBdAddr() with RBLE_VS_ADDR_AREA_REG(0x00).
1.10	Dec.26.2019	—	Supported the following compilers. - IAR C/C++ Compiler for Renesas RX
		Library	- Fixed the issue that pairing fails with the max bonding information after restart. - Fixed the issue that the BLE_GAP_EVENT_PAIRING_COMP event is not notified where R_BLE_GAP_StartPairing() is called after the local device deleted the bonding information. - Fixed the issue where a connection with the second and following paired remote device which of the address was resolved is not established after restart. - Fixed the issue that the bonding information is not deleted where the remote parameter of R_BLE_GAP_DeleteBondInfo() is specified as BLE_GAP_SEC_DEL_REM_NOT_CONN(0x02). - Changed the host stack to return error when adding or deleting a device from Resolving List / White List / Periodic Advertiser List in clearing the list. - Fixed an issue where the link is disconnected in high throughput communication when a request from the remote device is accepted. - Fixed the RSSI value error correction. - Changed the MTU exchange request minimum size to 23 bytes. - Fixed an issue that local device using the balance library couldn't send a connection request in adverting.

		Program	- Added the BSP macro as the BLE interrupt function declaration and the section.
			- Changed the Abstraction advertising API to set the random own address type.
			- Moved the response for a connection parameter update request from the command line to application layer.
			- Disabled the code in the cmd directory when BLE_CFG_CMD_LINE_EN is set to 0.
			- Fixed the memory access violation where BLE_CFG_RF_ADV_SET_MAX is other than 4.
			- Added the scan process where the ad type is specified as 0.
			- Fixed the other FIT modules dependencies and changed the following configuration option default value.
			  BLE_CFG_EN_SEC_DATA (r_flash_rx)
			  BLE_CFG_CMD_LINE_EN (r_sci_rx, r_byteq_rx)
			  BLE_CFG_BOARD_LED_SW_EN (r_gpio_rx, r_irq_rx)
			  BLE_CFG_DEV_DATA_DF_BLOCK (r_flash_rx)
			- Added the following configuration option.
			  BLE_CFG_ABS_API_EN (enable/disable abstraction API)
			  BLE_CFG_SOFT_TIMER_EN (enable/disable software timer)
			  BLE_CFG_MCU_LPC_EN (enable/disable MCU low power consumption control)
			  BLE_CFG_HCI_MODE_EN (enable/disable HCI mode)

2.00	Jul.27.2020	8, 44	- Added the description that the CMT code does not need to be changed if the version is 4.50 or later.
		14	- Added the notice about BLE_CFG_NUM_BOND.
		17	- Move the description of the features in Table 2.2 to the User's Manual (R01UW0205).
		30	- Added 4.2 Confirm BSP version.
		32	- Modified the SCI FIT Module name. r_sic_rx → r_sci_rx
		33	- Added the description about a method for displaying all FIT modules.
		36-37	- Added the description about the SCI FIT Module configuration when the command line is used.
		39-40	- Added the initial value for the customer board to Table 4.1. - Added the IRQ configuration option settings.
		47	- Modified the code changes in "4.6.3 Add application code".
		55-68	- Added the description about the optimization to "4.9 Create a project on the IAR development".
		69	- Move the contents in "5. Application Guide" to the Application.
		81	- Added the HCI mode tool location to "7. Tools". - Added the reference to the User's Manual (R01UW0205) to "7.1 BDAddrWriter".
		86-87	- Modified the description about the GATT Client demo project and Figure 8.5 .
		91-92	- Modified "8.5 Importing the demo project" to download and import from the Smart Configurator.
		94	- Delete the description about a method for adding FIT modules with the FIT configurator from 9.2 (1) answer.
		—	- Change the code sample style to the e²studio editor style. - Corrected the API specification name. r_ble_spec.chm --> r_ble_api_spec.chm

		Library	- Added Scan Channel Map API. - Fixed the RF event notification in Balance and Compact libraries. - Fixed the transmission power change feature by R_BLE_VS_SetTxPower(). - Fixed the long characteristic write completion notification. - Added the measures of the vulnerability to invalid packets. - Fixed the problem regarding slave latency. - Fixed the problem regarding bonding.
		Program	- Added the support for Renesas FreeRTOS. - Modified the condition for moving to the software standby state. - Fixed the CCCD confirmation for GATT Notification and Indication.
2.10	Dec.24.2020	6	- Modified "Figure 1.1 : Software structure".
		Library	Fixed the following bugs. - The RF wake-up process cannot be conducted normally, if the API execution process in R_BLE_Execute() and a RF wake-up interrupt compete in limited timing (a few cycles). - Scan cannot continue depending on the timing of repeated advertising turning on and off. - Software timer fails to expire after a specified interval if R_BLE_TIMER_UpdateTimeout() was continuously called. - A Central(Master) not using RPA fails to connect to a Peripheral(Slave) using RPA. - A GATT server returns "Insufficient Authentication Error" during service discovery, if check "Encryption" of "Included Service" on the QE for BLE.

		Program	Modified the followings.
			- The scan and create connection abstraction APIs can specify the local device address type.
			- The software timer callback function wakes-up the BLE task when a project includes Renesas FreeRTOS.
			- The updating process of the bonding information in the security data management.
			
			Added the following options to the Command Line.
			   - Local device address type option for adv, scan, conn command.
			   - Use White List option for adv, scan, conn command.
			   - Remove an advertising set option for adv command.
			   - Delete an entry in the Resolving List for priv command.

2.11	Mar.30.2021	-	- Support RX23W module (R5F523W8CxLN, R5F523W8DxLN).
		8	- Add the version of the BSP and the GPIO FIT when selecting the RX23W module.
		12	- Add BLE_CFG_RF_CLVAL value when selecting RX23W module. - Add BLE_CFG_RF_DDC_EN value when using Target Board.
		18	- Add the note that the logger feature is not supported when using CCRX C++.
		20	- Add the note when using gap command and vs command. - Add "2.8.7 LED and Switch control"

		26	- Add the note that using e²studio 2021-01 or later when creating a project for RX23W module.
		27	- Add the RX23W module type information to Figure 4.4.
		30	- Add the BSP version when selecting RX23W module.
		32	- Add BLE Profile Creation and r_ble_qe_utility to the components.
		33-34	- Add the configuration of Smart Configurator if the downloaded FIT modules are not found.
		78	- Change the BLE FIT APN URL. - Add the BLE FIT package name.
		79	- Add the FITDemos for RX23W module to Table 8.1 . - Add the FIT version included in FITDemos.

		92	- Add the following in Table 9.1 .
			  - Add "e²studio 2021-01(64bit)" to "Integrated development environment".
			  - Add "Target Board for RX23W module" and "Renesas Solution Start Kit for RX23W to Board used".

		Library	- Fixed that the changed address is reflected in advertising packet if static address is changed. - Fixed the scanner address included in a Scan Request Receive event (BLE_GAP_EVENT_SCAN_REQ_RECV) when using RPA feature. - Fixed that the BLE_GAP_EVENT_ADV_REPT_IND event is not notified when scan is executed only 1M PHY after scan is executed only Coded PHY. - Fixed that R_BLE_Execute() API does not return when the non-maskable interrupt status register (NMISR) is not 0x00.

		Program	- Fixed that variable length macro is not used when selecting CCRX C++.
			- Fixed the following about Command Line.
			  - Command line has come to recognize a passkey starting from 0 as a decimal number.
			  - The filter of the gap scan command has come to be able to include 0x00.

2.20	Jun.30.2021	88	- Added "8.5 Operation confirmation on CS+".
		Library	- Fixed for impersonation in the Passkey Entry Protocol (CVE-2020-26558).

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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. 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 the 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.

5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.

5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard":   Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1)   "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2)   "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics
Corporation. All trademarks and registered trademarks are the property
of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date
version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.