

RX23W Group

BLE Module Firmware Integration Technology

Introduction

This application note describes the Bluetooth® Low Energy(BLE) module which uses Firmware Integration Technology (FIT). This module controls BLE 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)
- 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)

The *Bluetooth*® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Renesas Electronics Corporation is under license. Other trademarks and registered trademarks are the property of their respective owners.

Contents

1. Overview.....	4
1.1 How to Use	4
1.2 Software Structure.....	5
1.3 Directory / File structure	6
2. API Information	7
2.1 Hardware Requirements	7
2.2 Software Requirements	7
2.3 Supported Toolchain	8
2.4 Header Files	8
2.5 Integer Types.....	8
2.6 Compile Configuration	9
2.7 Feature of BLE Protocol Stack	16
2.8 app_lib	19
2.8.1 Abstraction API.....	19
2.8.2 Software timer	19
2.8.3 Security data management	19
2.8.4 Profile common	19
2.8.5 Logger	19
2.8.6 Command Line	20
2.9 Flash Memory Protection	22
2.10 Code Size	24
2.11 Adding the FIT Module to Your Project	25
3. R_BLE API functions	26
4. BLE FIT module project.....	27
4.1 Create a new project	27
4.2 Clock configuration.....	30
4.3 Adding FIT modules	31
4.4 Configuration options	33
4.4.1 BSP(r_bsp).....	33
4.4.2 Command Line Interface	34
4.4.3 Security Data Management/Device-specific data (in data flash)	36
4.4.4 LED and Switch control	37
4.5 Configuration after code generation.....	39
4.5.1 Changing CMT for BLE	39
4.5.2 LED and Switch control for customer board.....	40
4.5.3 Add application code	42
4.6 Linker configurations	43

4.6.1	BLE Protocol Stack program section separation.....	43
4.6.2	BLE Protocol Stack Library Configuration.....	46
4.7	Debug configurations	48
4.7.1	Flash ID Code	48
4.7.2	Power	49
4.8	Create a project on the IAR development environments	50
5.	Application Guide	54
5.1	main function	54
5.1.1	main loop	55
5.1.2	Initialization of the BLE Host Stack and Profile	56
5.2	Callback function	58
5.3	Event notification	59
5.4	GATT Database.....	60
5.5	Low power consumption status	61
5.6	Data Flash Block	62
6.	Tools.....	63
6.1	BDAddrWriter	63
6.2	CLVALTune.....	63
7.	Demo Projects	64
7.1	GATT Server demo project	65
7.2	GATT Client demo project.....	68
7.3	HCI mode demo project	70
7.4	Adding the demo project	71
8.	Appendices.....	73
8.1	Confirmed Operation Environment.....	73
8.2	Troubleshooting.....	74
	Revision History	75
	General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products..	77
	Notice	78

1. Overview

The BLE FIT module provides BLE functions which are compliant with Bluetooth 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.

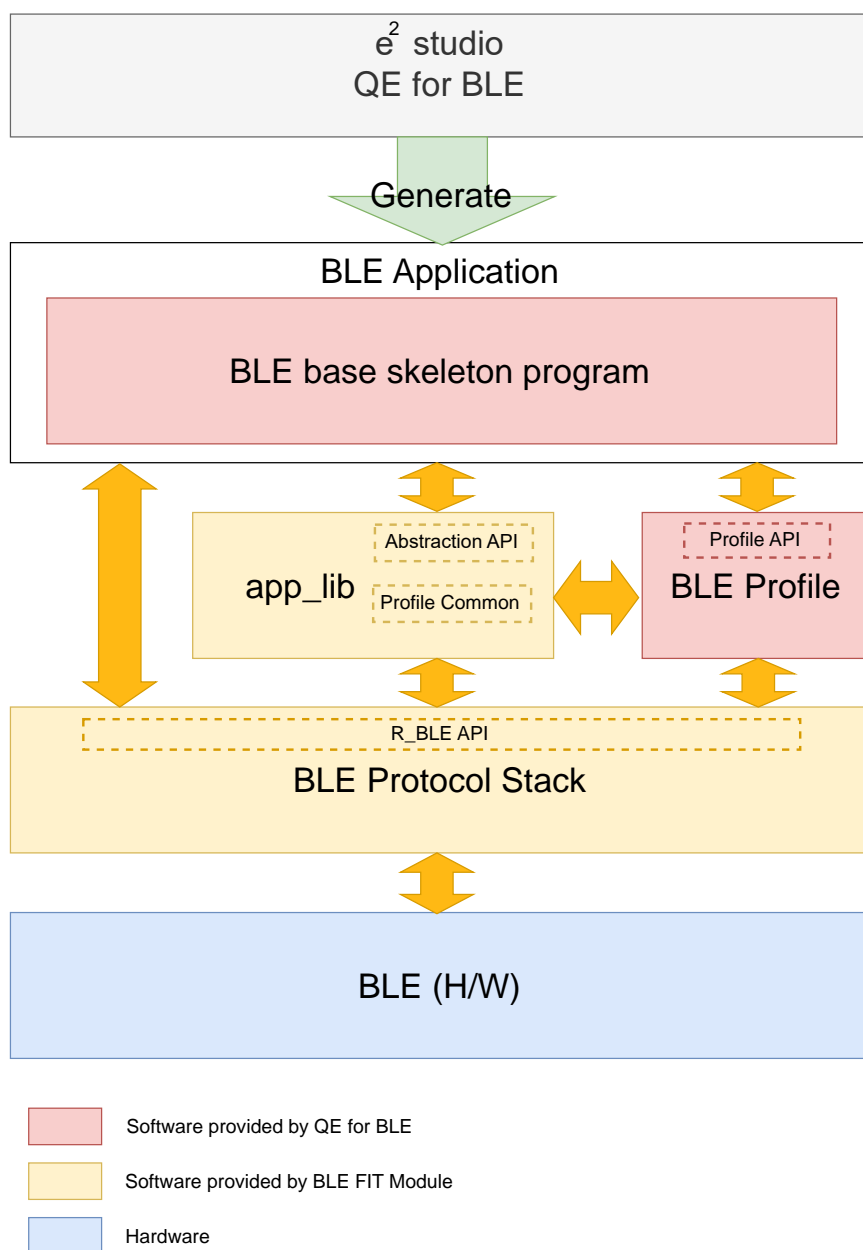


Figure 1.1 : Software structure

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	r01an4860ej0110-rx23w-ble.pdf		Application Note(English)
		jp	r01an4860jj0110-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_cmnn		Profile common
			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)
- CMT(r_cmt_rx, version 4.10 later)
The BLE Protocol Stack uses CMT2, CMT3. Therefore, modify the CMT code (Refer “4.5.1 Changing CMT for BLE”) 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)
- IRQ (r_irq_rx)

2.3 Supported Toolchain

The BLE FIT module has been confirmed to work with the toolchain listed in “8.1 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 are able to be configured in 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"	<p>Enable or disable start interrupt notification of a connection complete event.</p> <p>0: Disable 1: Enable</p> <p>Because the start notification is triggered by the interrupt, it occurs after the actual RF event.</p>
BLE_CFG_EVENT_NOTIFY_CONN_CLOSE Default : "0"	<p>Enable or disable end interrupt notification of a connection complete event.</p> <p>0: Disable 1: Enable</p>
BLE_CFG_EVENT_NOTIFY_ADV_START Default : "0"	<p>Enable or disable the advertising event start interrupt notification.</p> <p>0: Disable 1: Enable</p> <p>The notification occurs at the following timings.</p> <ul style="list-style-type: none"> - Start Primary Advertising channel. - Start Secondary Advertising Channel - Start Periodic Advertising. (When the Extended Advertising is enabled.) <p>Because the start notification is triggered by the interrupt, it occurs after the actual RF event.</p>
BLE_CFG_EVENT_NOTIFY_ADV_CLOSE Default : "0"	<p>Enable or disable the advertising event complete interrupt notification.</p> <p>0: Disable 1: Enable</p> <p>The notification occurs at the following timings.</p> <ul style="list-style-type: none"> - Complete Primary Advertising channel. - Complete Secondary Advertising Channel - Complete Periodic Advertising. (When the Extended Advertising is enabled.) <p>Because the start notification is triggered by the interrupt, it occurs after the actual RF event. If the advertising is terminated by a command, the notification doesn't occur.</p>
BLE_CFG_EVENT_NOTIFY_SCAN_START Default : "0"	<p>Enable or disable the scan start interrupt notification.</p> <p>0: Disable 1: Enable</p> <p>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.</p>

Configuration options (r_ble_rx23w_config.h)	
BLE_CFG_EVENT_NOTIFY_SCAN_CLOSE Default : "0"	<p>Enable or disable the scan complete interrupt notification</p> <p>0: Disable 1: Enable</p> <p>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.</p>
BLE_CFG_EVENT_NOTIFY_INIT_START Default : "0"	<p>Enable or disable the notification that the scan start interrupt has occurred in sending a connection request.</p> <p>0: Disable 1: Enable</p> <p>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.</p>
BLE_CFG_EVENT_NOTIFY_INIT_CLOSE Default : "0"	<p>Enable or disable the notification that the scan complete interrupt has occurred in sending a connection request.</p> <p>0: Disable 1: Enable</p> <p>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.</p>
BLE_CFG_EVENT_NOTIFY_DS_START Default : "0"	<p>Enable or disable the RF_DEEP_SLEEP start notification.</p> <p>0: Disable 1: Enable</p>
BLE_CFG_EVENT_NOTIFY_DS_WAKEUP Default : "0"	<p>Enable or disable the RF_DEEP_SLEEP wakeup notification.</p> <p>0: Disable 1: Enable</p>
BLE_CFG_RF_CLVAL Default : "5"	<p>Adjustment value of the 32MHz crystal oscillator. Set this option according to the board environment. Range : 0 to 15</p> <p>Refer to "RX23W Group Tuning procedure of Bluetooth dedicated clock frequency(R01AN4762)" for details.</p>
BLE_CFG_RF_DDC_EN Default : "0"	<p>Enable or disable the DC-DC on the RF.</p> <p>0: Disable 1: Enable</p>

Configuration options (r_ble_rx23w_config.h)	
BLE_CFG_RF_EXT32K_EN Default : "0"	<p>Slow clock source to the RF. Range : 0 to 1</p> <p>0: RF_LOCO 1: External 32.768kHz</p> <p>If this option is set to 1, the sub clock is required to be enabled in the Smart Configurator clock configuration screen (Figure 4.7).</p>
BLE_CFG_RF_MCU_CLKOUT_PORT Default : "0"	<p>Port of the MCU CLKOUT. Range : 0 to 1</p> <p>0: PE3 1: PE4</p> <p>If the BLE_CFG_RF_EXT32K_EN option is 0, this option is ignored.</p>
BLE_CFG_RF_MCU_CLKOUT_FREQ Default : "0"	<p>Output frequency from the MCU CLKOUT. Range : 0 to 1</p> <p>0: MCU CLKOUT frequency 32.768kHz 1: MCU CLKOUT frequency 16.384kHz</p> <p>If the BLE_CFG_RF_EXT32K_EN option is 0, this option is ignored.</p>
BLE_CFG_RF_SCA Default : "250"	<p>Sleep Clock Accuracy(SCA) for the RF slow clock. Range : 0 to 500</p> <p>If the BLE_CFG_RF_EXT32K_EN option is 0, the SCA is fixed to more than 250ppm and this option is ignored.</p>
BLE_CFG_RF_MAX_TX_POW Default : "1"	<p>Maximum transmit power configuration. Range : 0 to 1</p> <p>0: max +0dBm 1: max +4dBm</p>
BLE_CFG_RF_DEF_TX_POW Default : "0"	<p>Default transmit power level. Range : 0 to 2 This option depends on the BLE_CFG_RF_MAX_TX_POW option.</p> <p>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</p> <p>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</p>

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.
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.
BLE_CFG_GATT_MTU_SIZE Default : "247"	The MTU size (bytes) for the GATT communication. Range : 23 to 247
BLE_CFG_NUM_BOND Default : "7"	Maximum number of the bonding information stored in the Data Flash. Range : 1 to 7
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.

Configuration options (r_ble_rx23w_config.h)	
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
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
BLE_CFG_ABS_API_EN Default : "1"	Enable or disable support the Abstraction API. 0: Disable 1: Enable

Configuration options (r_ble_rx23w_config.h)	
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.

Table 2.2 : Features of the BLE Protocol Stack

BLE 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

- LE 2M PHY
Support the BLE communication with 2 Msym/s PHY.
- LE Coded PHY
Support the BLE communication with Coded PHY.
LE Coded PHY makes a longer communication distance than 1M PHY and 2M PHY available.
- LE Advertising Extensions
The Advertising Extensions have the following features.
 - Simultaneous advertisings up to 4.
(The BLE_CFG_RF_ADV_SET_MAX option sets the maximum number of simultaneous advertisings)
 - Advertising Data / Scan Response data is extended up to 1650 bytes
(The BLE_CFG_RF_ADV_DATA_MAX option sets the maximum length (bytes) of the advertising data / scan response data.)
 - Periodic advertising.
- LE Channel Selection Algorithm #2
The Advertising Extensions have the following features.
- High Duty Cycle Non-Connectable Advertising
Support the non-connectable advertising of which the lower limit of the interval is 20 msec.
- LE Secure Connections
Support the pairing with the Elliptic curve Diffie-Hellman key exchange for passive eavesdropping protection.
- Link Layer privacy
Change the Bluetooth Device Address at stated periods to avoid the tracking from other BLE devices.
- Link Layer Extended Scanner Filter policies
Support the scan filter compatible with resolvable private address.
- LE Data Packet Length Extension
Extend the maximum transmission packet payload size (up to 251 byte) and maximum packet transmission time and transmit time.
- LE L2CAP Connection Oriented Channel Support
Support the L2CAP credit-based flow control channel communication.
- Low Duty Cycle Directed Advertising
Support Low Duty Cycle advertising to reconnect a known device.
- LE Link Layer Topology
Support both Master and Slave role. Act as a master in one connection and as a slave in another connection.

- LE Ping
After encrypting a link, check whether the link is lost by requesting a packet including MIC.

- GAP Role
Support the following GAP roles.
 - Central : Central device sends a connection request to a Peripheral device.
 - Peripheral : Peripheral device accepts the connection request from a Central device.
 - Observer : Observer device scans advertising packets.
 - Broadcaster : Broadcaster device sends advertising packets.

- GATT Role
Support the following GATT roles.
 - Server : GATT Server has a GATT Database including services, characteristics and responds for request from GATT Client.
 - Client : GATT Client requests for the service provided by GATT Server.

- 32-bit UUID Support in LE
Support 32-bit UUID. A 32-bit UUID is extended to a 128-bit UUID in GATT.

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_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

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. 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 call 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.

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.

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

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.

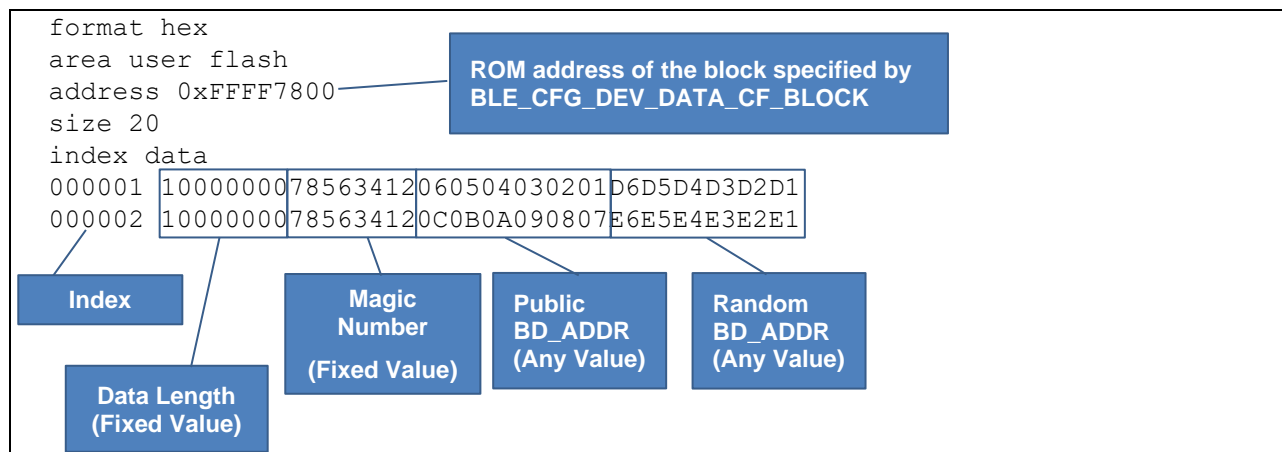
Table 2.5 : Data Format

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.

- 4.4.1 BSP(r_bsp)
- 4.7.1 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 2.3 . 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	187K bytes	146K bytes	130K bytes
		RAM	38K bytes	23K bytes	23K bytes
Configuration Option					
<u>Common :</u>					
BLE_CFG_RF_CONN_MAX 7					
BLE_CFG_RF_CONN_DATA_MAX 251					
BLE_CFG_NUM_BOND 7					
 <u>All Features :</u>					
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					
 <u>Balance :</u>					
BLE_CFG_LIB_TYPE 1					
 <u>Compact :</u>					
<u>BLE_CFG_LIB_TYPE 2</u>					

[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_spec.chm)” for details.

If the r_ble_spec.chm cannot be opened, right-click the file, select Properties and check “Unblock” as follows.

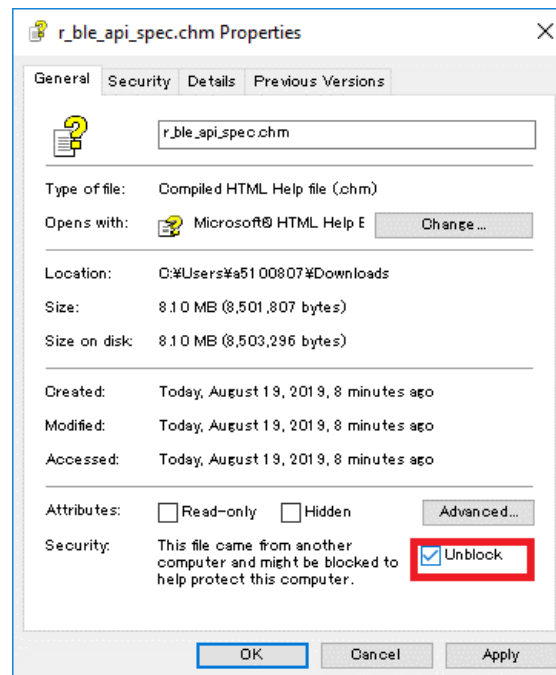


Figure 3.1 : r_ble_api_spec.chm Properties

4. BLE FIT module project

This section describes how to create a new BLE project in the e²studio and the Smart Configurator to build a BLE Application development environment.

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.

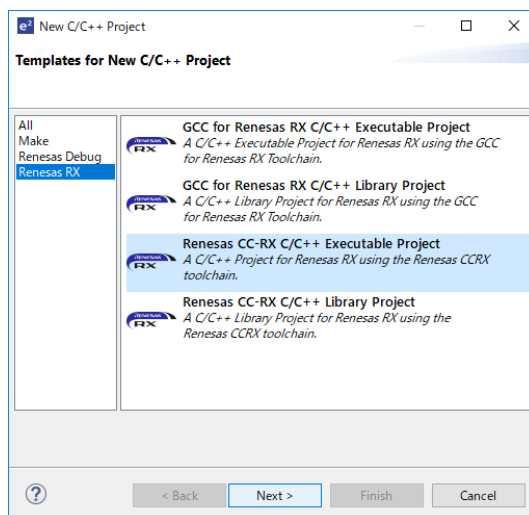


Figure 4.1 : Templates for New C/C++ Project

Enter the project name and click on the “Next” button.

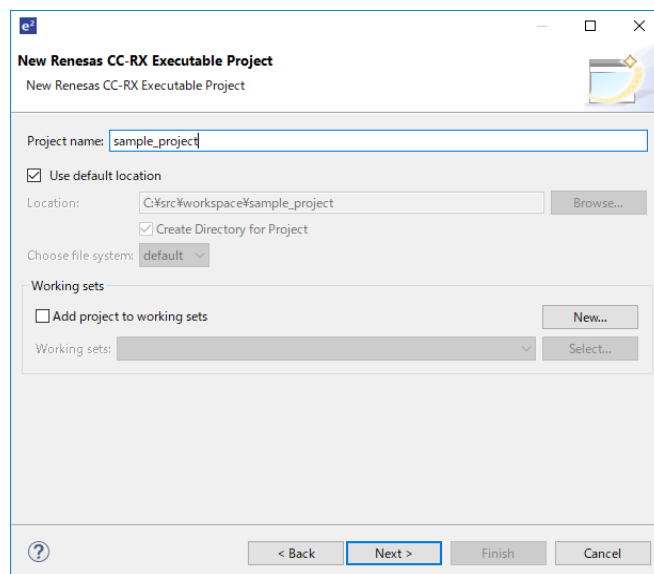


Figure 4.2 : New Renesas CC-RX Executable Project

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”.

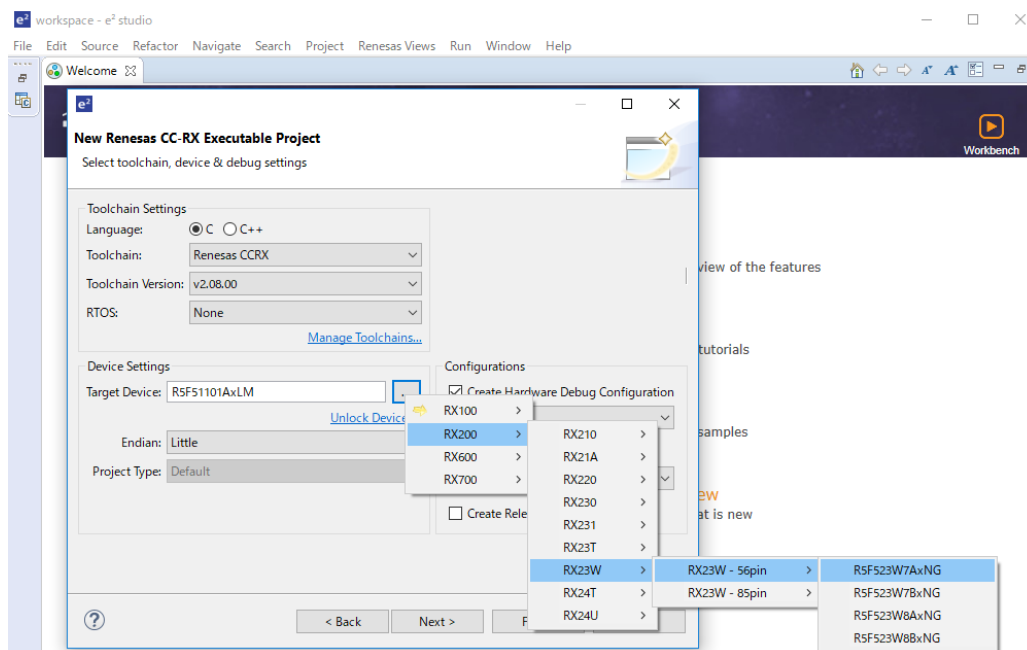


Figure 4.3 : Toolchain, device & debug settings

Figure 4.4 shows the RX23W Product Part Number.

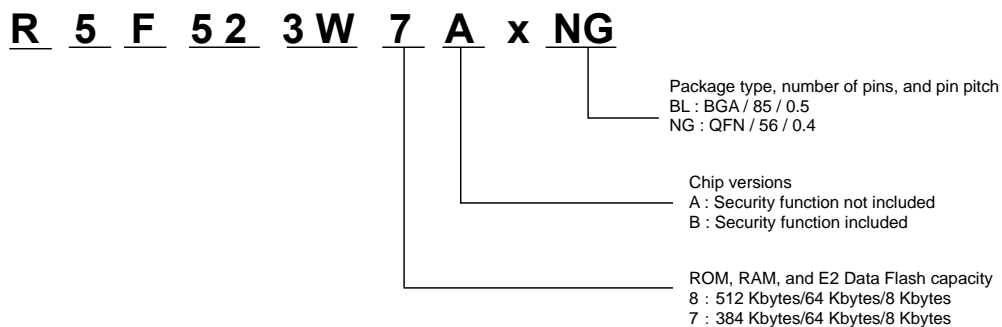


Figure 4.4 : RX23W Product Part Number

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

Click “Next” button and “Select Coding Assistant settings” dialog appears. Check in the “Smart Configurator” and click “Finish” button.

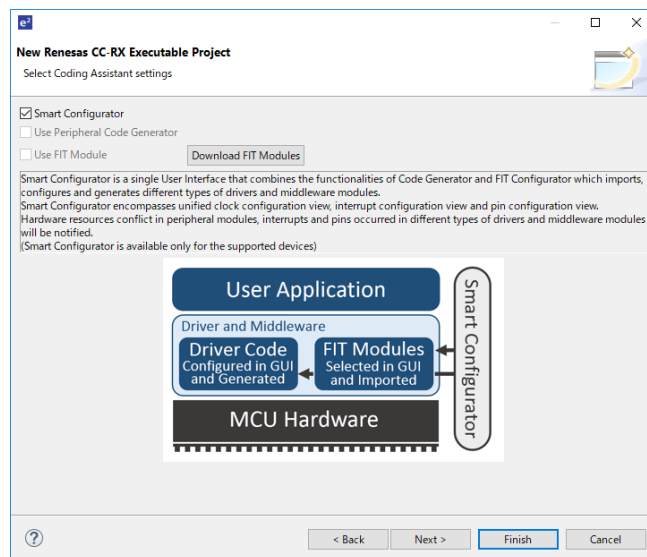


Figure 4.5 : Coding Assistant settings

After a little while, the project is created.

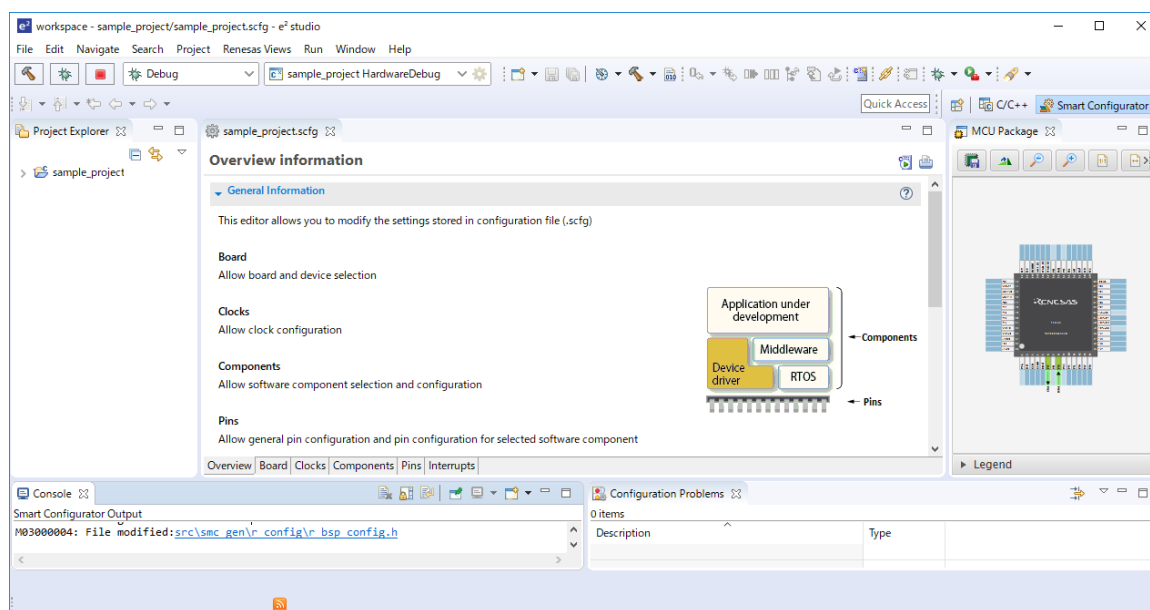


Figure 4.6 : Project overview


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

4.3 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 r_ble_rx23w, r_byteq, r_cmt_rx, r_flash_rx, r_gpio_rx, r_irq_rx, r_lpc_rx, r_sic_rx and click “Finish” button.

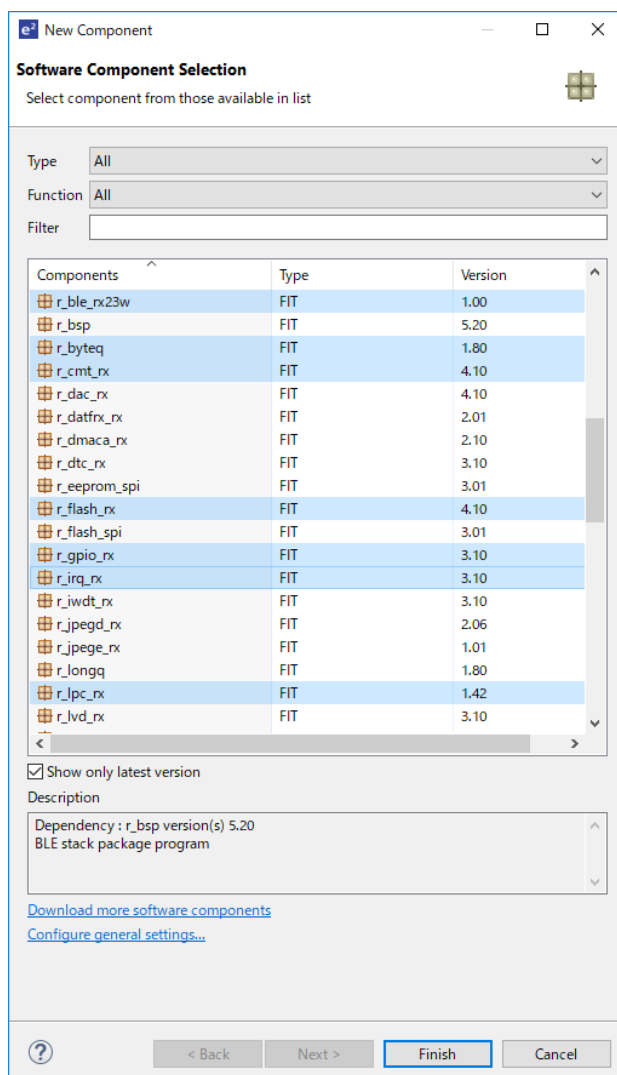


Figure 4.8 : Software Component Selection

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

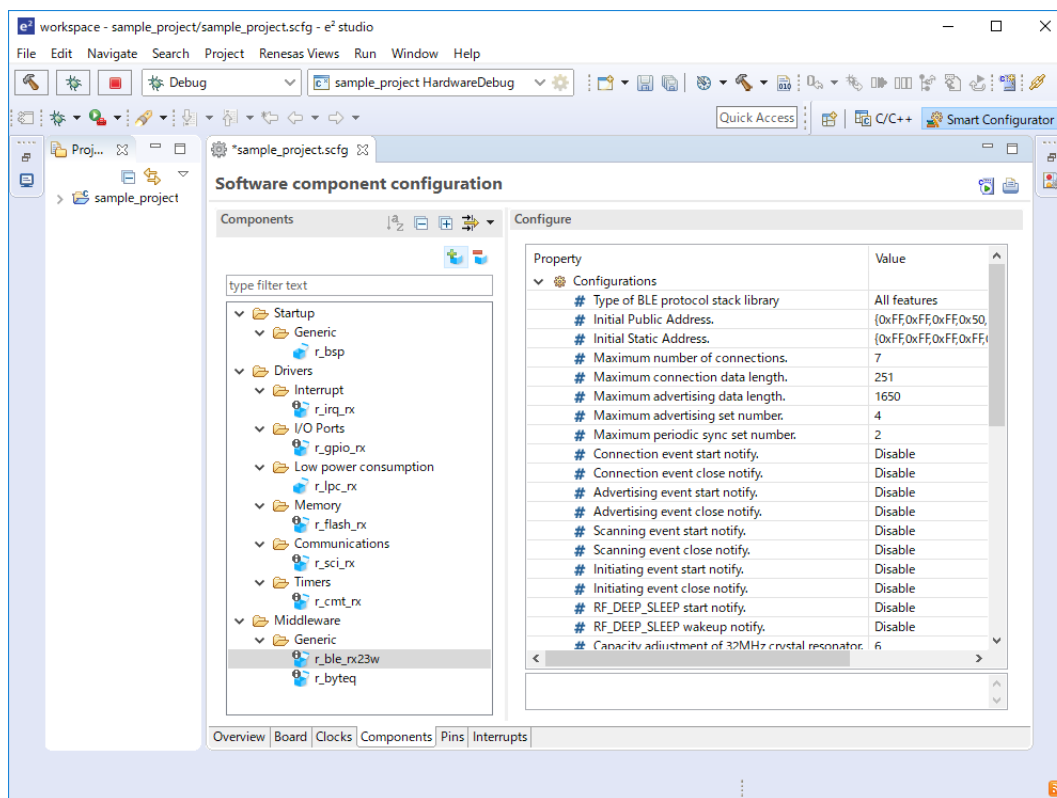


Figure 4.9 : Software component configuration

4.4 Configuration options

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

4.4.1 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”.

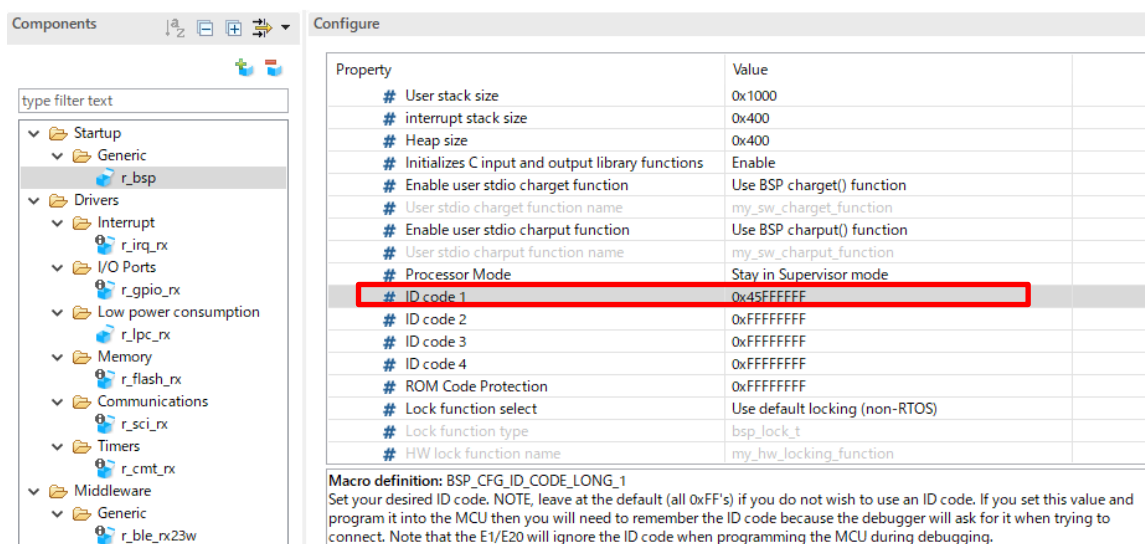


Figure 4.10 : BSP configuration options

4.4.2 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.

(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”.

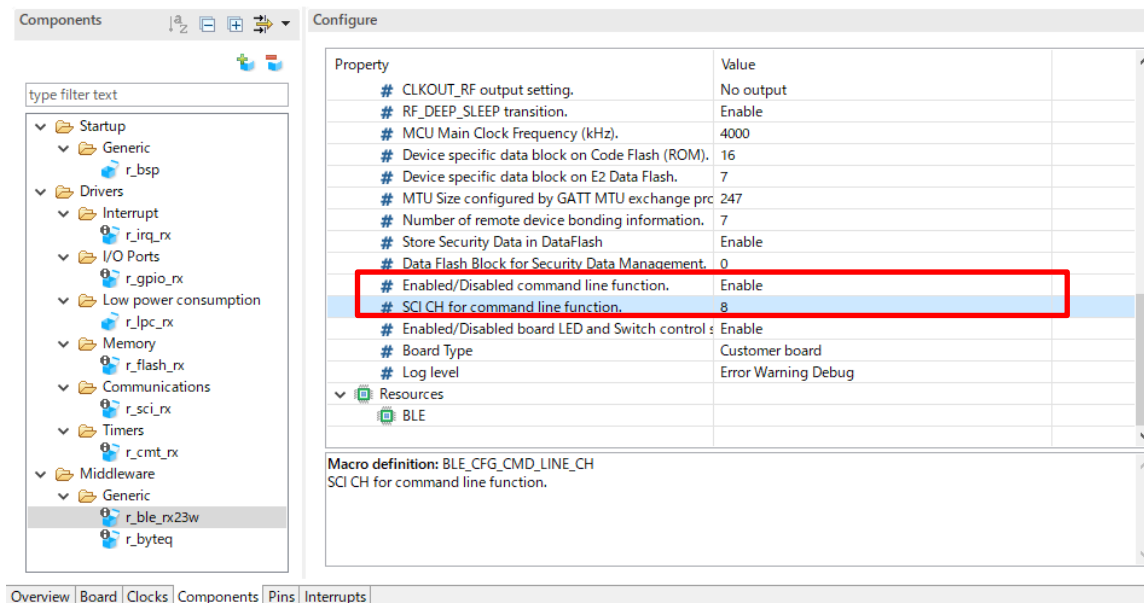


Figure 4.11 : BLE configuration options for Command Line Interface

(2) SCI(r_sci_rx)

Select “Components” tab→”r_sci_rx”→”Include software support for channel n” (n is the SCI channel number for command line) and set the option to “Include”. Check the SCI channel number, “RXDn/SMISOn” and “TXDn/SMOSIn” in “SCI” resources.

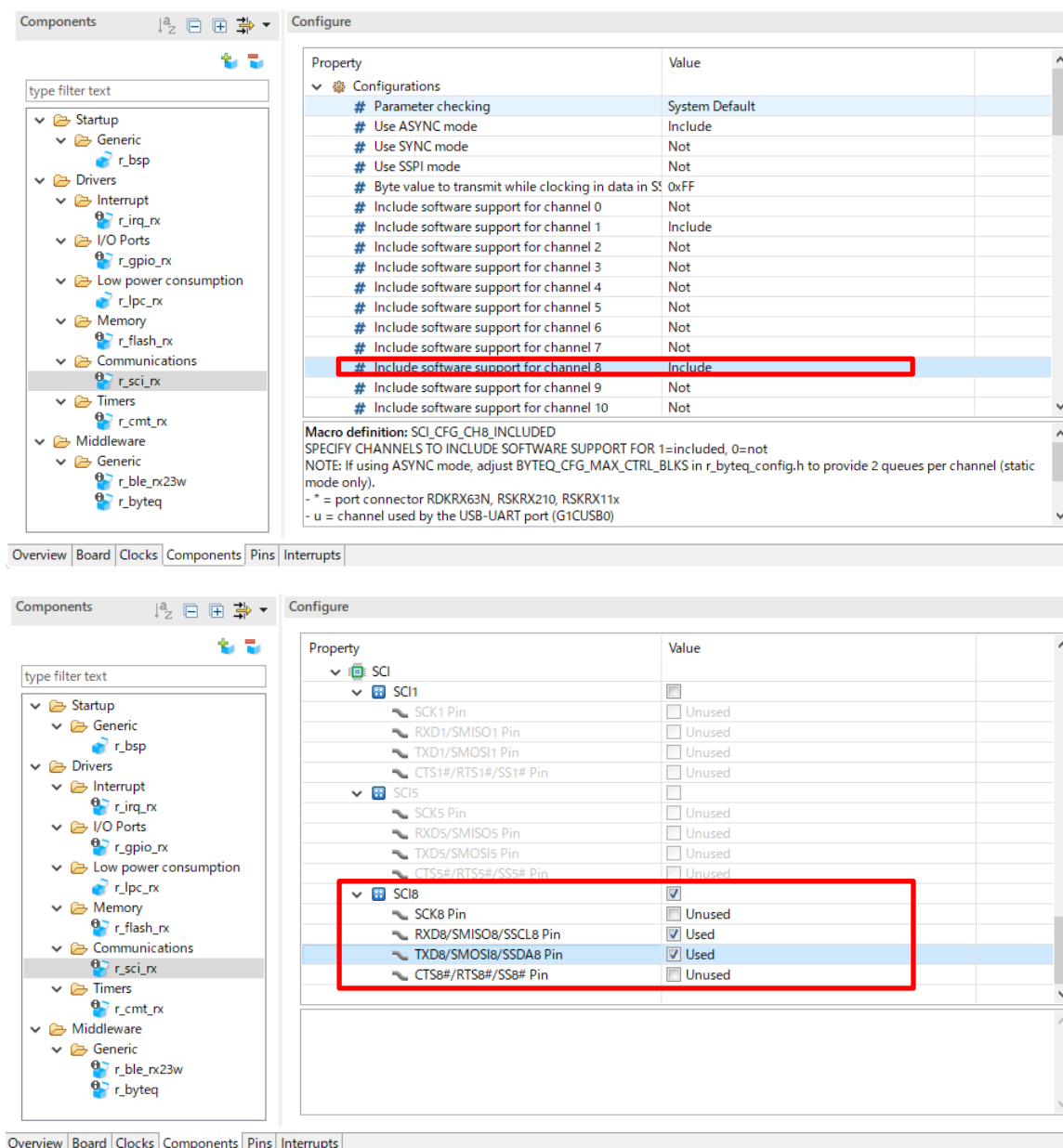


Figure 4.12 : SCI configuration options for Command Line Interface

4.4.3 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.

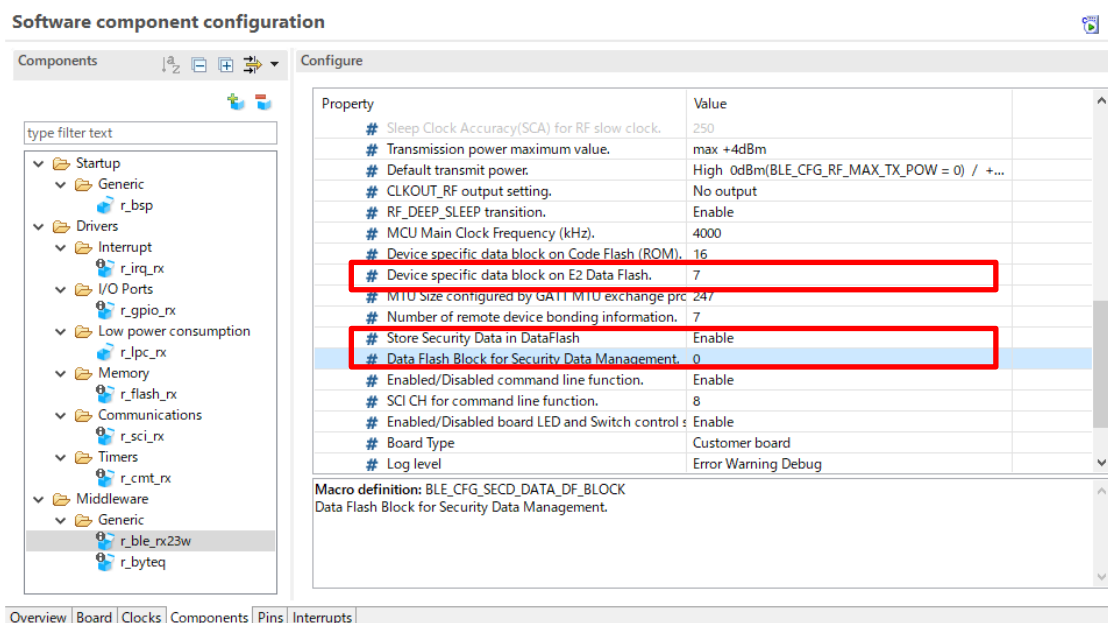


Figure 4.13 : BLE configuration options for Security Data Management

4.4.4 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.

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

(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 “4.5.2 LED and Switch control for customer board”.

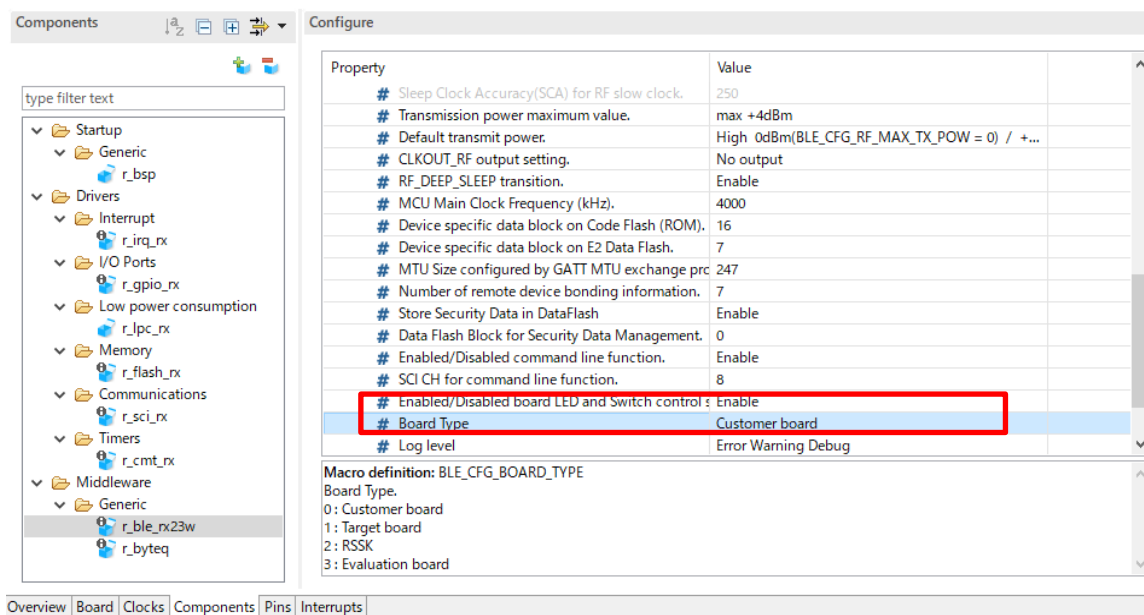


Figure 4.14 : BLE configuration options for LED and Switch control

(2) IRQ(`r_irq_rx`)

Select “Components” tab → “`r_irq_rx`” and check the IRQ pin that connects the switch in “ICU” resources.

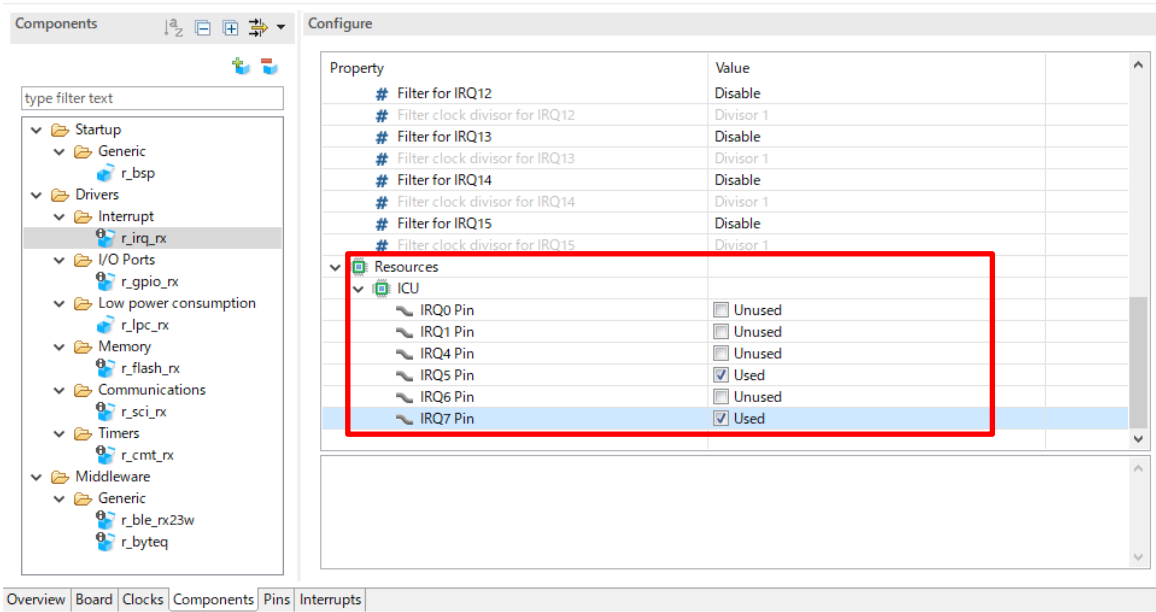


Figure 4.15 : IRQ configuration options for LED and Switch control

4.5 Configuration after code generation

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

4.5.1 Changing CMT for BLE

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 4.16)

```
#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 4.16 : Changing the r_cmt_rx.c for BLE

4.5.2 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 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 */
```

Figure 4.17 : 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 4.17.

(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)
    /*Set IRQ5 pin */
    PORT1.PMR.BIT.B5 = 0U;
    PORT1.PDR.BIT.B5 = 0U;
    MPC.P15PFS.BYTE = 0x40U;
#elif (BLE_CFG_BOARD_TYPE == 2)
    /*Set IRQ0 pin */
    PORT3.PMR.BIT.B0 = 0U;
    PORT3.PDR.BIT.B0 = 0U;
    MPC.P30PFS.BYTE = 0x40U;
    /*Set IRQ1 pin */
    PORT3.PMR.BIT.B1 = 0U;
    PORT3.PDR.BIT.B1 = 0U;
    MPC.P31PFS.BYTE = 0x40U;
#else /* (BLE_CFG_BOARD_TYPE == x) */
    /*Set IRQ5 pin */
    PORT1.PMR.BIT.B5 = 0U;
    PORT1.PDR.BIT.B5 = 0U;
    MPC.P15PFS.BYTE = 0x40U;
    /*Set IRQ7 pin */
    PORT1.PMR.BIT.B7 = 0U;
    PORT1.PDR.BIT.B7 = 0U;
    MPC.P17PFS.BYTE = 0x40U;
#endif /* (BLE_CFG_BOARD_TYPE == x) */
```

Figure 4.18 : The changes in irq_pin_set()

4.5.3 Add application code

When a project is created, the application code("{project name}.c") is generated. But it is a blank template. Therefore, delete this file and add the application base code in the following methods. For the application implementation details, see "5 Application Guide".

(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)".

(2) FITDemos

Copy the application codes(app_main.c, gatt_db.c, gatt_db.h) and the profile codes in "services" directory in the FITDemos and paste the "src" directory.

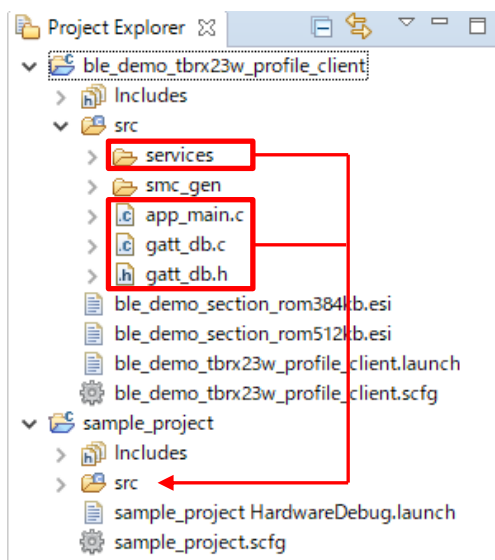


Figure 4.19 : Project Explorer

Select "Project" menu → "Properties" → "C/C++ build" → "Settings" → "Tool Settings" tab → "Compiler" → "Source" → "Include file directories" and add the application and the profile code directories.

4.6 Linker configurations

4.6.1 BLE Protocol Stack program section separation

Configure the section for the BLE.

Table 4.2 shows the section name of the BLE Protocol Stack.

Table 4.2 : 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 4.2 to the BLE Application project in the following procedure.

4.6.1.1 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

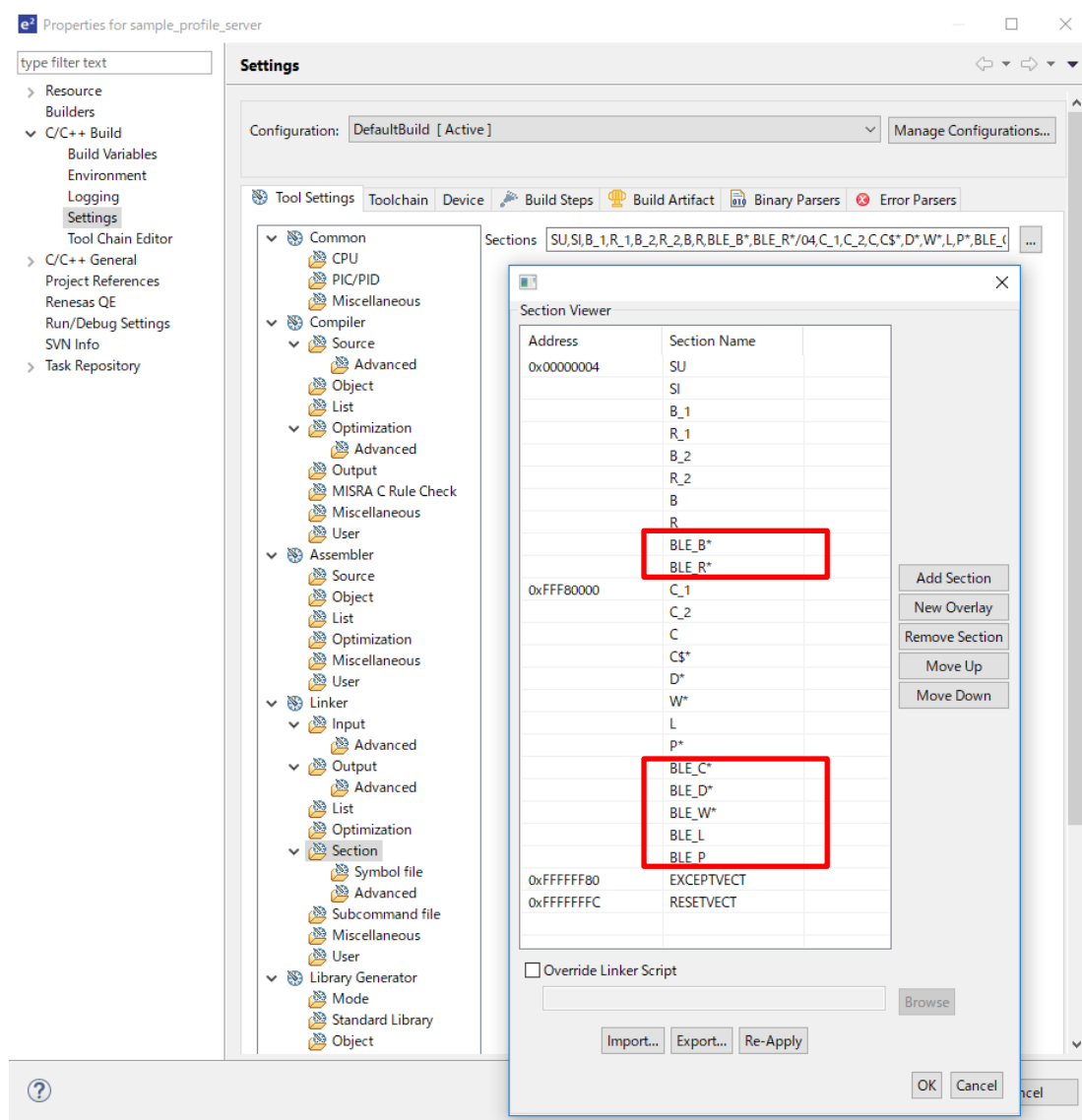


Figure 4.20 : Section Configuration

4.6.1.2 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 4.6)

BLE_D=BLE_R

BLE_D_1=BLE_R_1

BLE_D_2=BLE_R_2

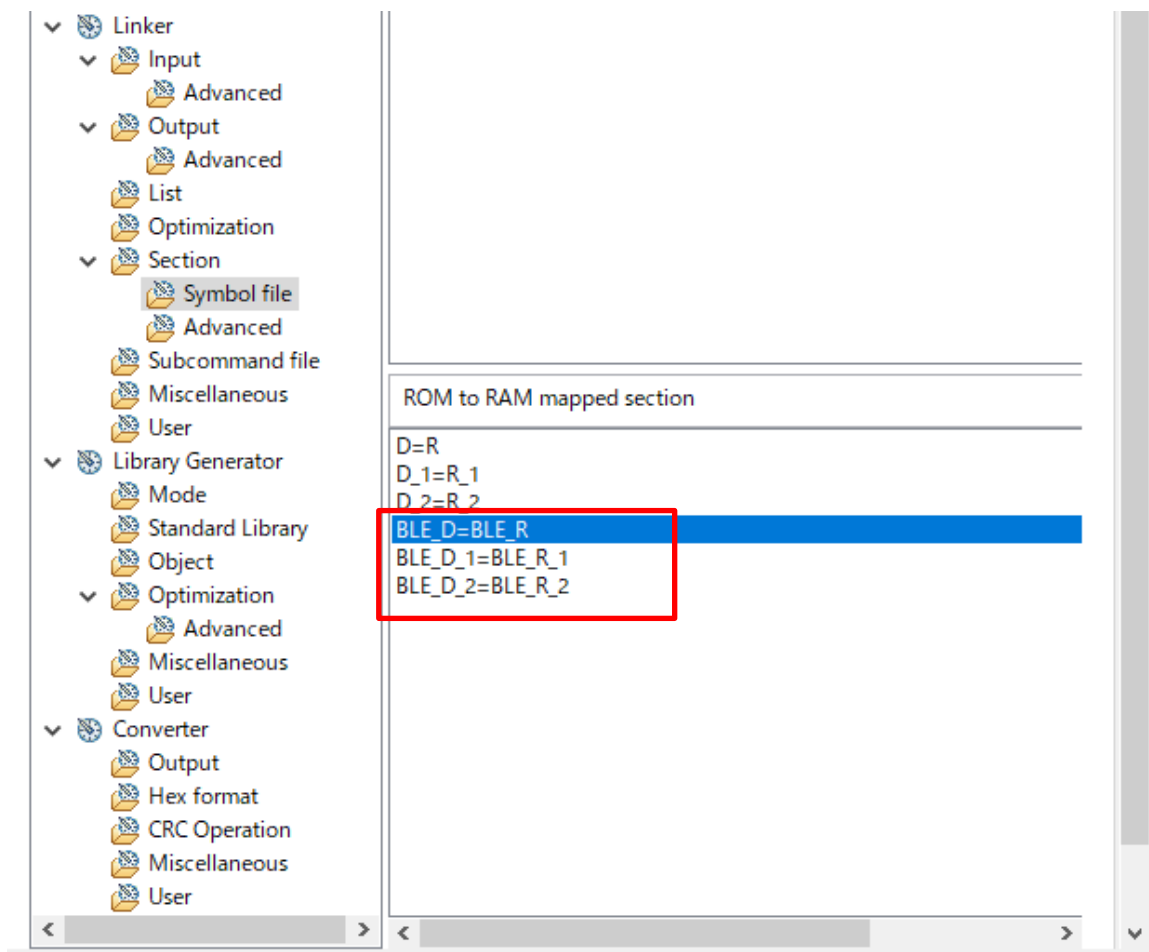


Figure 4.21 : Rom to RAM mapped section configuration

[Note]

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

4.6.2 BLE Protocol Stack Library Configuration

The BLE Protocol Stack is provided as static library. Three types of libraries shown in Table 4.3 are included in the lib directory of the BLE FIT module.

Table 4.3 : 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 4.22). If not, add the library.

"\${workspace_loc:\${ProjName}}/src/smc_gen/r_ble_rx23w/lib/lib_ble_ps_ccrx.lib)"

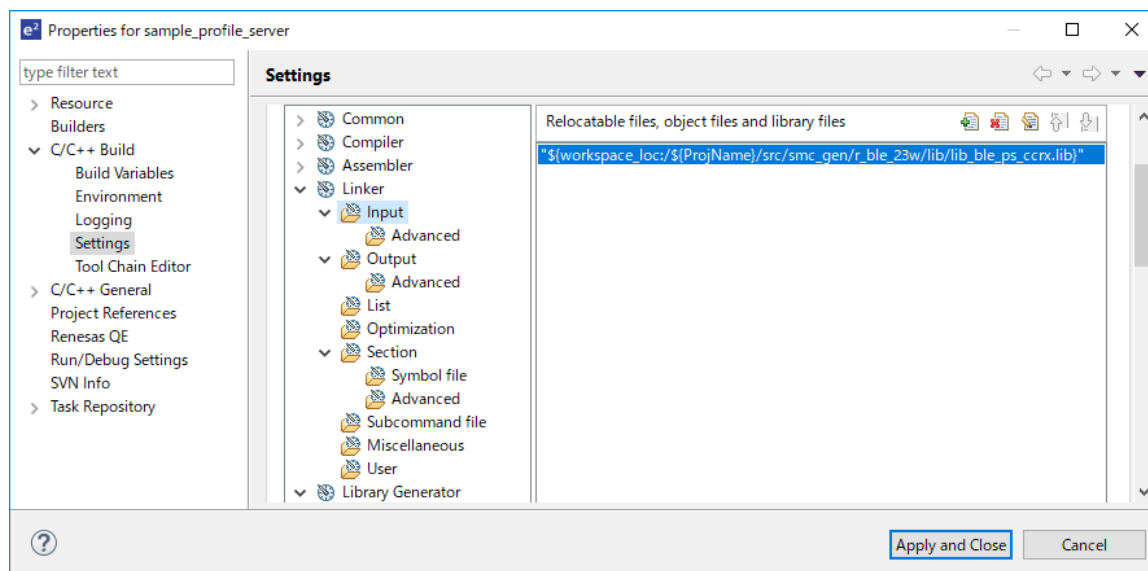


Figure 4.22 : Linker configuration

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 e² studio.

Select “C/C++ Build” → “Settings” → “Build Steps”.

Input the following to the “Command(s)” in the “Pre-build steps”. (see Figure 4.23)

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

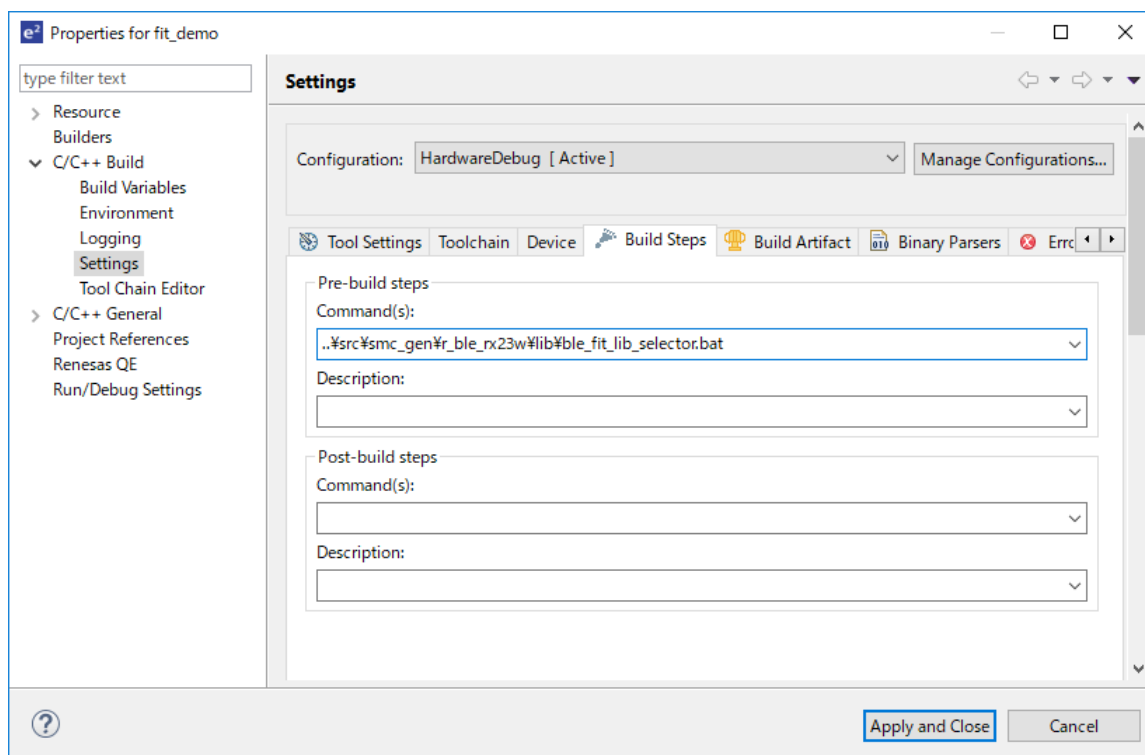


Figure 4.23 : Build Steps configuration

4.7 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 are successfully finished, write the firmware to the board.

4.7.1 Flash ID Code

Click [RUN]→[Debug Configuration] on the e²studio.

Select the BLE Application project from the “Renesas GDB Hardware Debugging” and input “45FFFFFFFFFFFFFFFFFFFFFFFF” 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”.

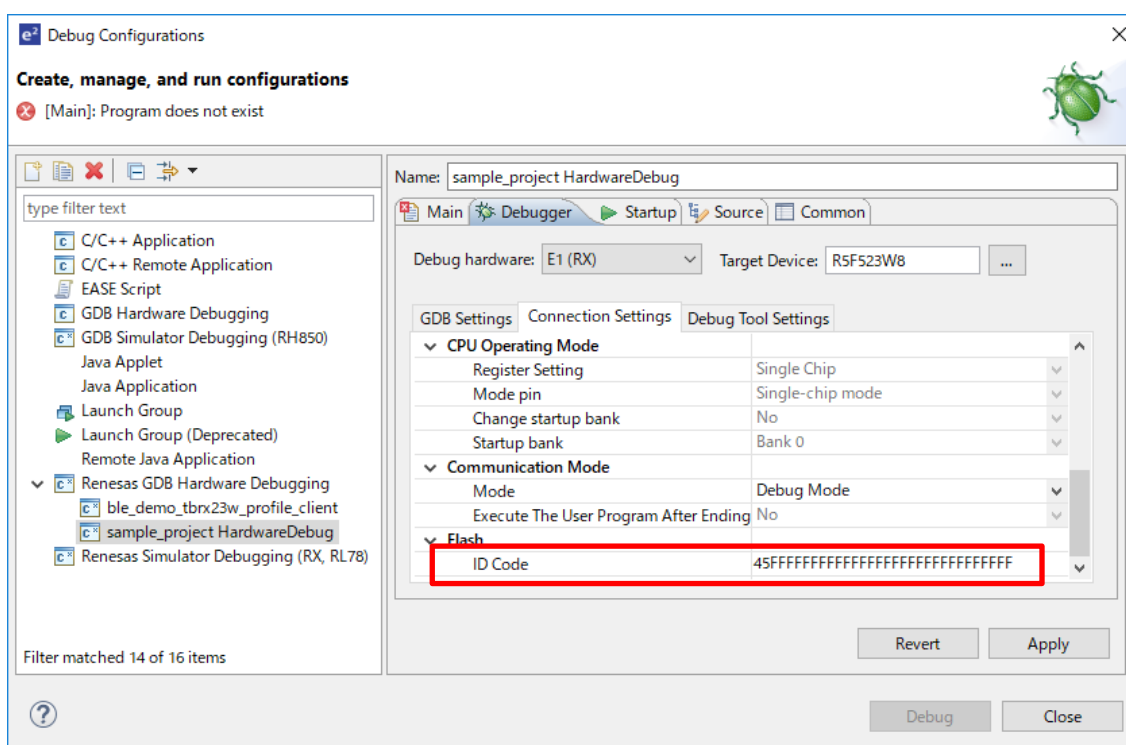


Figure 4.24 : ID Code protection configuration

4.7.2 Power

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

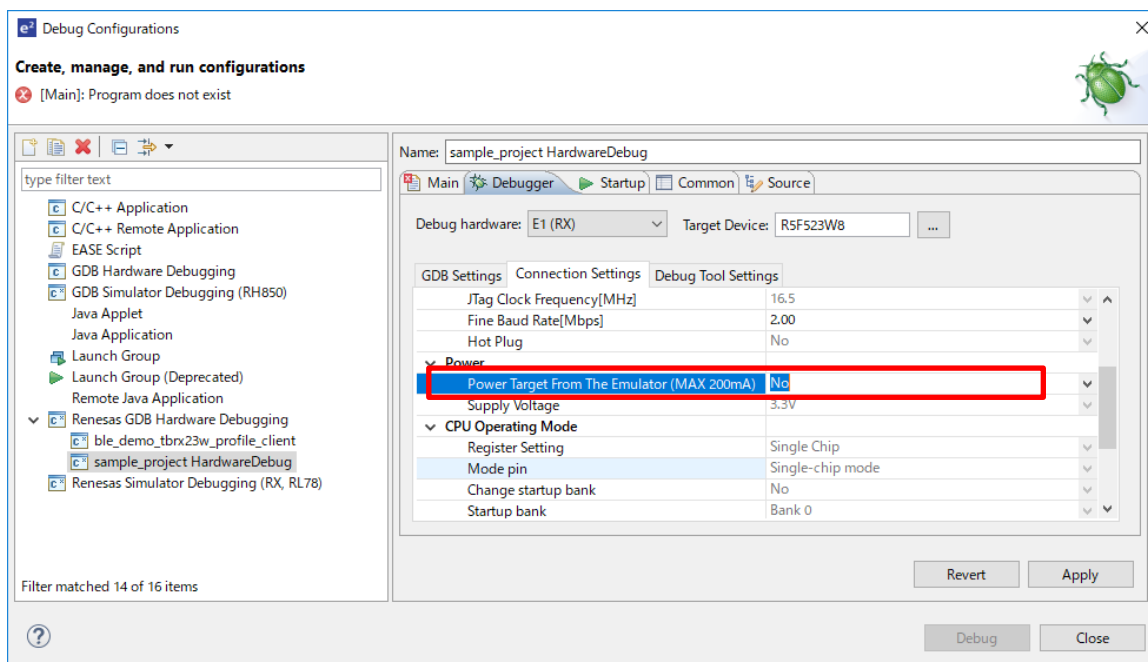
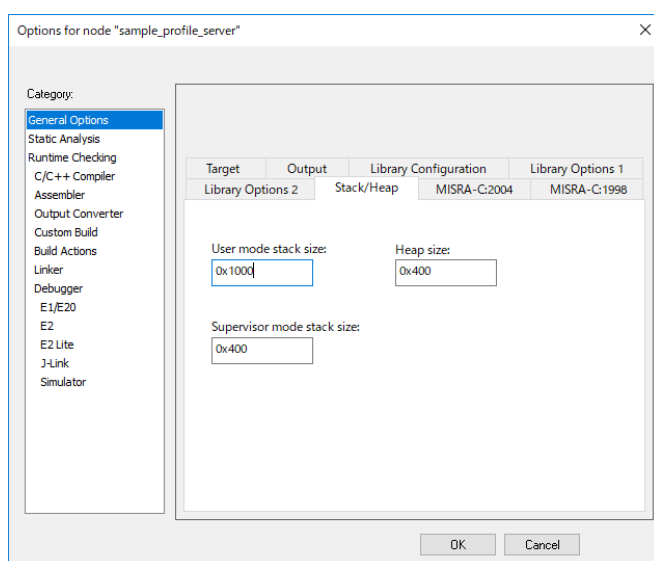


Figure 4.25 : Power supply from the emulator

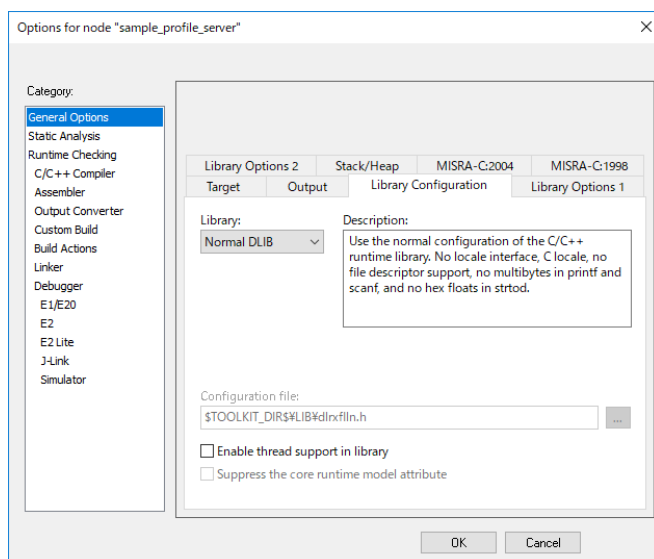
4.8 Create a project on the IAR development environments

This section describes the project creation on the IAR Embedded Workbench for Renesas RX.

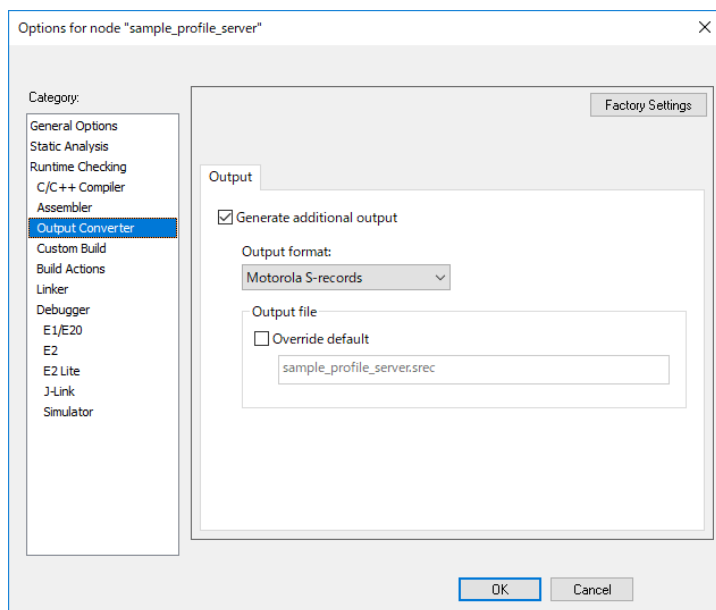
1. Create a new project on e² studio according to Section 4.1 to 4.6.
 2. Convert the new project to the IAR project according to Section 7.2 “Adding FIT Modules to the IAR Project “ in “RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)”.
 3. Start “IAR Embedded Workbench for Renesas RX” and open the eww file in the IAR project and click “Project >> Options...” and change the following option.
- **Stack/Heap**
Click “General Options >> Stack/Heap”. Set “User mode stack size” and “Heap size” and “Supervisor mode stack size” same as the sizes on e2studio project according to “Stacks Area and Heap Area” and “Table 3.2 Stack & Heap Defines” in R01AN1685.



- **Library Configuration**
Click “General Options >> Library Configuration”. Select “Normal DLIB” on “Library”.



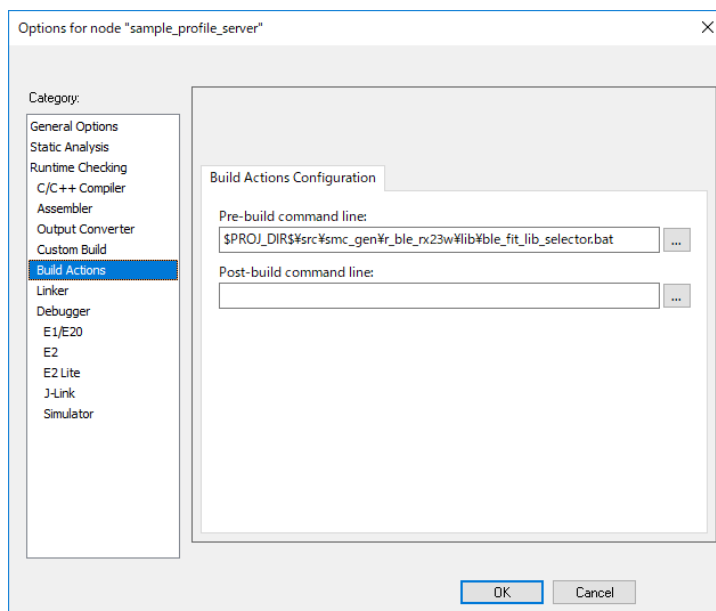
- **Output Converter**
Click “Output Converter”. Check “Generate additional output” and select “Motorola S-records” on “Output format”.



- **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:

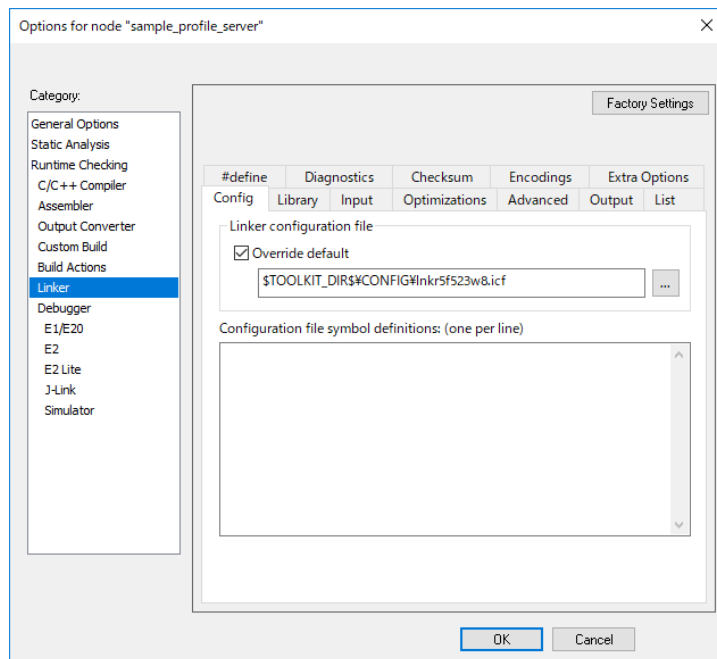
The bat file path is \$PROJ_DIR\$\src\smc_gen\r_ble_rx23w\lib\ble_fit_lib_selector.bat.



- **Linker configuration file**
Click “Linker >> Config”. Check “Override default” and enter the file path to Inkr5f523w8.icf or Inkr5f523w7.icf.

Example :

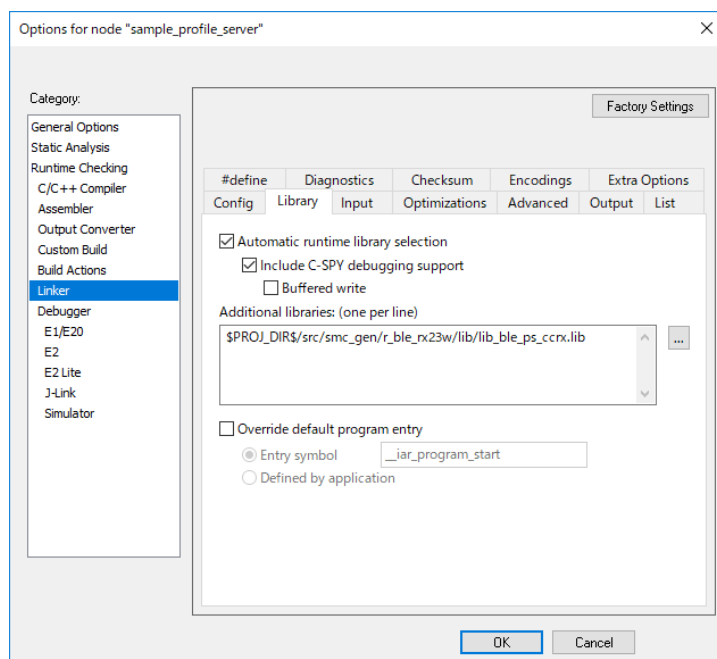
The Inkr5f523w8.icf file path is \$TOOLKIT_DIR\$\CONFIG\Inkr5f523w8.icf.



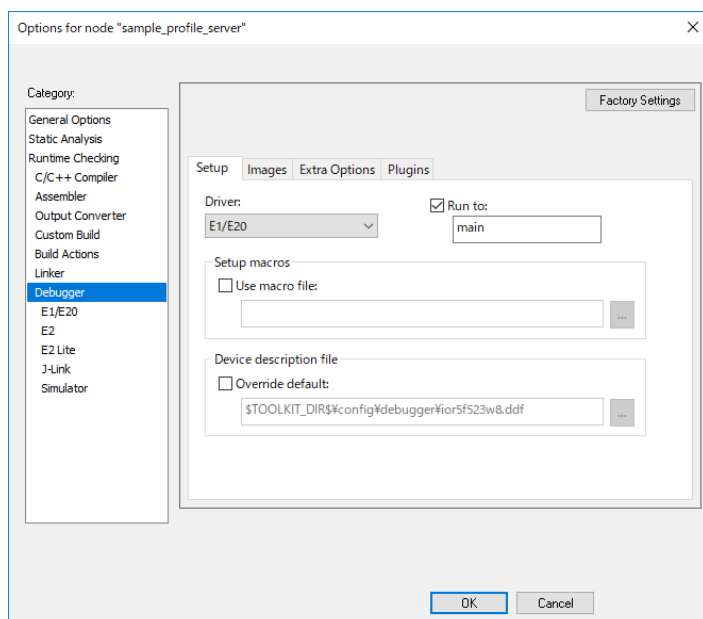
- **Additional libraries**
Click “Linker >> Library”. Enter the library file path on “Additional libraries”.

Example :

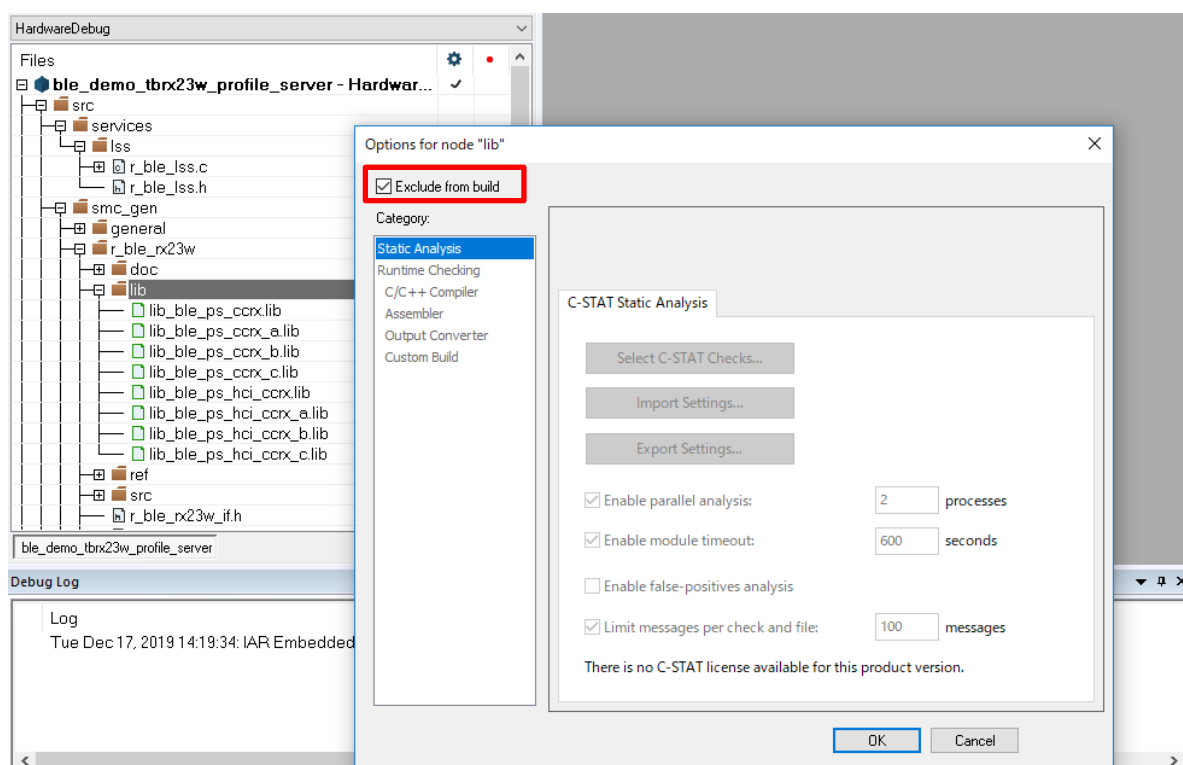
The library file path is \$PROJ_DIR\$/src/smc_gen/r_ble_rx23w/lib/lib_ble_ps_ccrx.lib.



- **Debugger**
Click “Debugger >> Setup”. Select the emulator on “Driver”.



4. Exclude the BLE FIT module lib directory from build target on GUI.
Right-click on src/smc_gen/r_ble_rx23w/lib directory and select "Options...".
On the Option window, tick the "Exclude from build" check box.



5. Build the project after changing the above options and download the generated firmware.

5. Application Guide

5.1 main function

The main function is implemented in the BLE Application.

The main function includes the initialization and the main loop. If the BLE functions are used, implement the followings.

- The BLE initialization by the R_BLE_Open(). Call first the R_BLE_Open() in main function.
- The BLE Host Stack and Profile initialization by the ble_app_init().
- The main loop

If timer, board configuration and command line are used, initialize those in the main function.

As an example, Figure 5.1 shows the main function constituted of the followings.

- The initialization.
 - BLE
 - Board
 - Low power consumption control
 - Timer for the BLE Application
 - BLE Host stack and Profile
 - Command Line
- Main loop
- Transition to the Low power consumption state

```
/* *****  
 * Function Name: main  
 * Description   : The main loop  
 * Arguments     : none  
 * Return Value  : none  
 * ***** */  
void main(void)  
{  
    /* Initialize BLE */  
    R_BLE_Open();  
  
    /* Configure the board */  
    R_BLE_BOARD_Init();  
    R_BLE_BOARD_RegisterSwitchCb(BLE_BOARD_SW2, sw_cb);  
  
    /* Initialize the Low Power Control function */  
    R_BLE_LPC_Init();  
  
    /* Initialize timer for LED blink */  
    R_BLE_TIMER_Init();  
  
    /* Configure CommandLine */  
    R_BLE_CLI_Init();  
    R_BLE_CLI_RegisterCmds(gsp_cmds, ARRAY_SIZE(gsp_cmds));  
  
    /* Initialize BLE host stack and profiles */  
    ble_app_init();  
  
    /* main loop */  
    while (1)  
    {  
        /* Process Command Line */  
        R_BLE_CLI_Process();  
        /* Process Event */  
        R_BLE_Execute();  
        /* Enter the Lower Power Mode */  
        R_BLE_LPC_EnterLowPowerMode();  
    }  
  
    /* Terminate timer for LED blink */  
    R_BLE_TIMER_Terminate();  
  
    /* Terminate CommandLine */  
    R_BLE_CLI_Terminate();  
  
    /* Terminate BLE */  
    R_BLE_Close();  
}
```

Figure 5.1 An example of main function

5.1.1 main loop

The BLE software uses a scheduler for event handling. Running the scheduler, call the `R_BLE_Execute()` in the main loop. (Figure 5.2)

```

while (1)
{
    R_BLE_Execute();
}

```

Figure 5.2 : An example of a main loop

The callback function processes an occurred event. Refer to “5.2 Callback function” for details.

5.1.2 Initialization of the BLE Host Stack and Profile

The ble_app_init() implements the initialization of the BLE Host Stack and Profile.

The initialization of the BLE Host stack is performed by the followings. Refer to “R_BLE API document (r_ble_spec.chm)” for the details.

- R_BLE_GAP_Init()
- R_BLE_ABS_Init()

The Profile initialization needs the call of functions shown in Table 5.1 according to the role of the profile. Using the QE for BLE, the profile source codes which implements the initialization shown in Table 5.1 are output.

Table 5.1 : GATT initialization

Type of Application	The function called in ble_app_init()	Description
GATT Server	R_BLE_GATTS_SetDBInst()	GATT Database initialization
	R_BLE_SERVS_Init(), R_BLE_XXX_Init() (XXX : GATT Server Name)	GATT Server initialization
GATT Client	R_BLE_SERVC_Init(), R_BLE_XXX_Init() (XXX : GATT Client Name)	GATT Client initialization
	R_BLE_DISC_Init()	Service Discovery initialization

Figure 5.3 shows an example of ble_app_init().


```

/*****
 * Function Name: ble_app_init
 * Description   : Initialize host stack and profiles.
 * Arguments     : none
 * Return Value  : BLR_SUCCESS - SUCCESS
 *                BLE_ERR_INVALID_OPERATION -
 *                Failed to initialize host stack or profiles.
 *****/
static ble_status_t ble_app_init(void)
{
    ble_status_t status;

    gs_conn_hdl  = BLE_GAP_INVALID_CONN_HDL;
    gs_timer_hdl = BLE_TIMER_INVALID_HDL;

    /* Initialize host stack */
    status = R_BLE_ABS_Init(&gs_abs_init_param);
    if (BLE_SUCCESS != status)
    {
        return BLE_ERR_INVALID_OPERATION;
    }

    /* Initialize GATT Database */
    status = R_BLE_GATTS_SetDbInst(&g_gatt_db_table);
    if (BLE_SUCCESS != status)
    {
        return BLE_ERR_INVALID_OPERATION;
    }

    /* Initialize GATT Server */
    status = R_BLE_SERVS_Init();
    if (BLE_SUCCESS != status)
    {
        return BLE_ERR_INVALID_OPERATION;
    }

    /* Initialize LED and Switch Service */
    status = R_BLE_LSS_Init(lss_cb);
    if (BLE_SUCCESS != status)
    {
        return BLE_ERR_INVALID_OPERATION;
    }

    /* Create timer for LED blink */
    status = R_BLE_TIMER_Create(&gs_timer_hdl, 1, BLE_TIMER_PERIODIC, timer_cb);
    if (BLE_SUCCESS != status)
    {
        return BLE_ERR_INVALID_OPERATION;
    }

    return status;
}

```

Figure 5.3 : An example of ble_app_init()

5.2 Callback function

Registering a callback function, processing can be performed in the BLE application in receiving each event.

Table 5.2 shows the callback registration functions.

Table 5.2 : Callback registration functions

Function Block	Registration Function	Callback timing
GAP	R_BLE_GAP_Init() or R_BLE_ABS_Init()	Advertising, Scan, Connection establishment event
GATT	Server : R_BLE_GATTS_RegisterCb()	Server : Access from a Client.
	Client : R_BLE_GATTC_RegisterCb() or R_BLE_ABS_Init()	Client : Receive a response from a Server.
Vendor Specific	R_BLE_VS_Init() or R_BLE_ABS_Init()	Receive a vendor specific event
L2CAP	R_BLE_L2CAP_RegisterCfPsm()	Receive a L2CAP Credit-Based Flow Control response. Receive a L2CAP Credit-Based Flow Control event.
board	R_BLE_BOARD_RegisterSwitchCb()	Push the Switch on the board.
Timer	R_BLE_TIMER_Create()	Timer expired.
Profile(Service, Client)	R_BLE_XXX_Init() (XXX : Service, Client name)	Service : Access from a Client Client : Receive a response from a Server.
Service Discovery	R_BLE_DISC_Start()	Complete the service discovery.

5.3 Event notification

The event notification sets the scheduler in the BLE software to an event.

The BLE Protocol Stack confirms the event status at the next call of the R_BLE_Execute(). If an event is set, the callback function registered by the R_BLE_SetEvent() is called.

This function uses the following cases.

- A time-consuming processing in an interrupt context runs outside of it.
- The function which can't be executed in an interrupt context runs outside of it.

Figure 5.4 shows an example of the event notification which is executed in the callback to put on/off the LED on the board.

```
static void sw_ntf_rcv_event(void)
{
    R_BLE_BOARD_ToggleLEDState(BLE_BOARD_LED1);
}

static void lsc_cb(uint16_t type, ble_status_t result, st_ble_sercv_evt_data_t *p_data)
{
    ...

    switch (type)
    {
        case BLE_LSC_EVENT_SWITCH_STATE_HDL_VAL_NTF:
        {
            pf("lsc: Receive Switch State Ntf.\n");
            R_BLE_SetEvent(sw_ntf_rcv_event);
        } break;

        default:
            break;
    }
}
```

Figure 5.4 : An example of the event notification

5.4 GATT Database

If the BLE Application includes the GATT Service, the GATT Database is implemented in the followings.

- gatt_db.c
- gatt_db.h

The QE for BLE generates the above files according to the GATT Service.

The generated GATT Database is registered with the R_BLE_GATTS_SetDbInst().

5.5 Low power consumption status

The transition to the low power consumption status of the MCU can perform even in using the BLE functions.

The policy of the transition to the low power consumption status is as the follows.

- The BLE Protocol Stack doesn't prevent the transition to the low power consumption status during the period before the next the R_BLE_Execute() is called after the processing by R_BLE_Execute() has been completed.
- Confirming whether all the components including BLE have a problem to enter the low power consumption status, the application puts the MCU into the low power consumption status.

The BLE FIT module provides the sample program (r_ble_pf_lowpower.c) having the following functions.

- The transition to the low power consumption status uses the LPC FIT module.
- Types of the low power consumption status are Sleep mode, Deep Sleep mode and Software Standby mode.
- The R_BLE_LPC_Init() initialize the low power consumption control.
- The MCU enters the low power consumption status by the R_BLE_LPC_EnterLowPowerMode().
- The function is called after the processing by the R_BLE_Execute() has been completed and executes the followings.
 - Disable the interrupt
 - Confirm whether each component has a problem to enter the low power consumption status.
 - Each component enters the low power consumption status.
 - The MCU enters the low power consumption status.
 - Each component comes back from the low power consumption status after the MCU has come back from the status.
- Come back from the low power consumption status by the interrupt from the RF, if the BLE communication occurs.

The "sys stby" command included in the command line is used for the transition to the low power consumption status.

5.6 Data Flash Block

If your application holds data in Data Flash, use the block except the following.

- BLE_CFG_DEV_DATA_DF_BLOCK
- BLE_CFG_SECD_DATA_DF_BLOCK (only if use security data management)

6. Tools

The following tools are provided for HCI mode firmware.

6.1 BDAAddrWriter

BDAAddrWriter configures BD_ADDR in the board for HCI mode.

6.2 CLVALTune

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

7. Demo Projects

The applications shown in Table 7.1 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.

Table 7.1 : 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.
ble_demo_tbrx23w_profile_server	GATT Server demo application for Target Board.
ble_demo_tbrx23w_uart_hci	HCI mode demo application for Target Board.

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.

7.1 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” or “RBLE” device name.

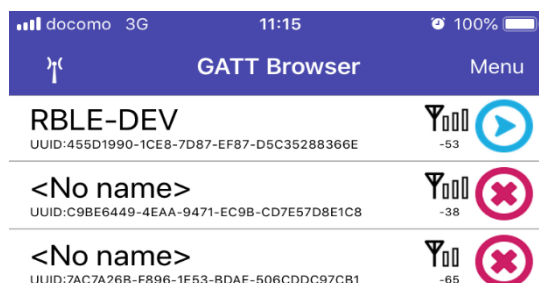


Figure 7.1 : Scan result example

- 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)

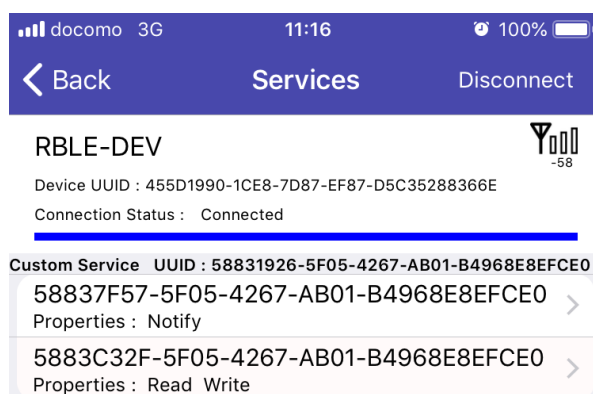


Figure 7.2 : GATT Services

- 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**, 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[] =
{
    ...
    /* LED Switch */
    {
        /* Num of Services */
        {
            1,
        },
        /* Description */
        BLE_GATT_DB_SER_SECURITY_UNAUTH,
        /* Service Start Handle */
        0x0010,
        /* Service End Handle */
        0x0015,
        /* Characteristic Start Index */
        6,
        /* Characteristic End Index */
        7,
    },
};
```

Pairing is necessary.

```
static const st_ble_gatts_db_serv_cfg_t gs_gatt_service[] =
{
    ...
    /* LED Switch */
    {
        /* Num of Services */
        {
            1,
        },
        /* Description */
        0,
        /* Service Start Handle */
        0x0010,
        /* Service End Handle */
        0x0015,
        /* Characteristic Start Index */
        6,
        /* Characteristic End Index */
        7,
    },
};
```

Pairing is not necessary.

Figure 7.3 : 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 7.4 shows usage example for the GATT Server demo project and a remote device.

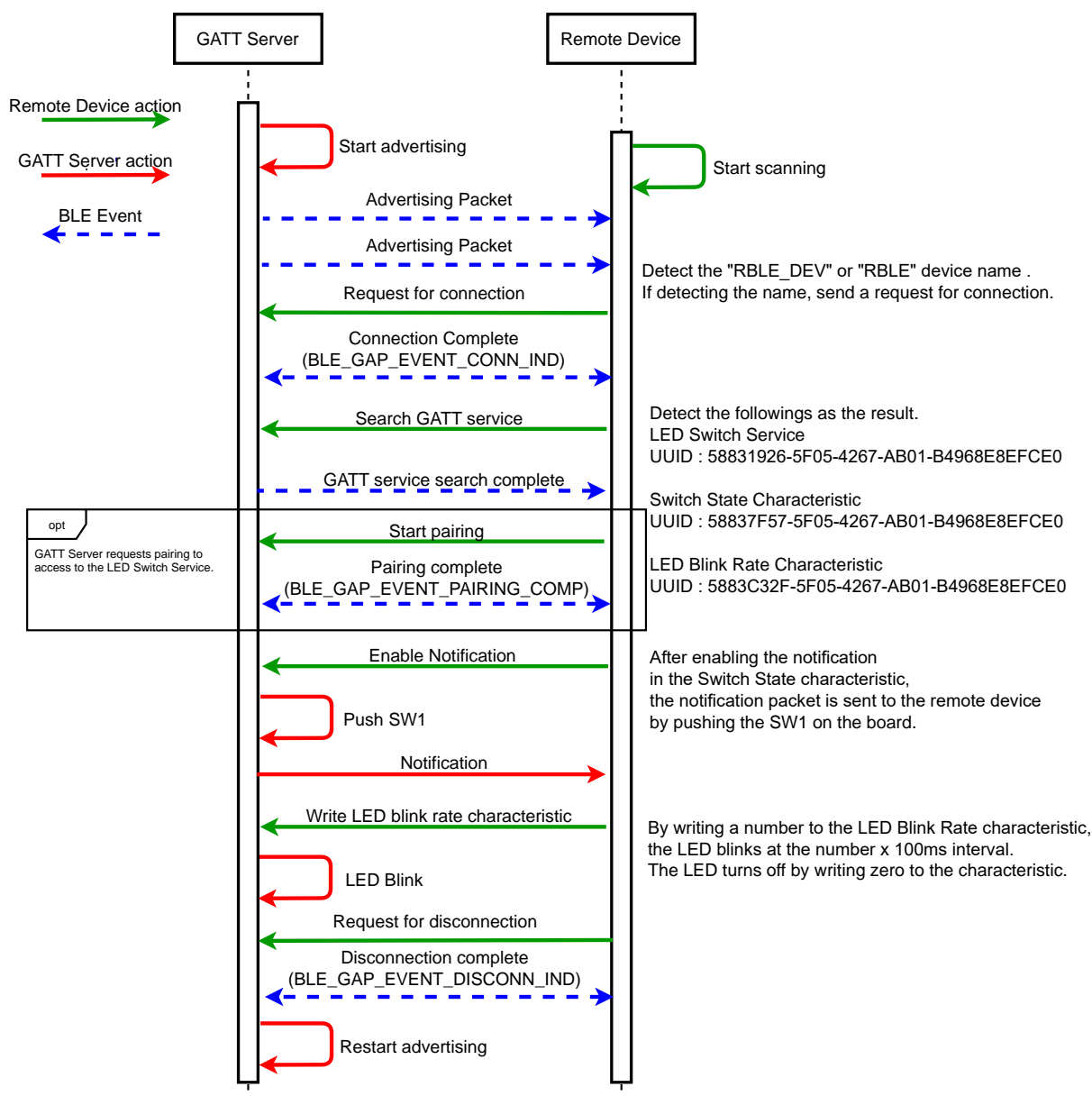


Figure 7.4 : Usage example for the GATT Server demo project and a remote device

7.2 GATT Client demo project

The GATT Client demo works as below.

- After starting, it waits for a 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.
- After connecting to the GATT Server demo project, it allows to use the following LED Switch client commands. If the GATT Server demo project requests pairing to access to the LED Switch Service, before executing the LED Switch client command, the GATT Client has to start pairing by the “gap auth” command.
 - Enable / Disable Notification
lsc cmd_lsc_set_switch_state_ntf [connection handle] [0(disabled) or 1(enabled)]
 - Write the LED blink rate
lsc cmd_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.

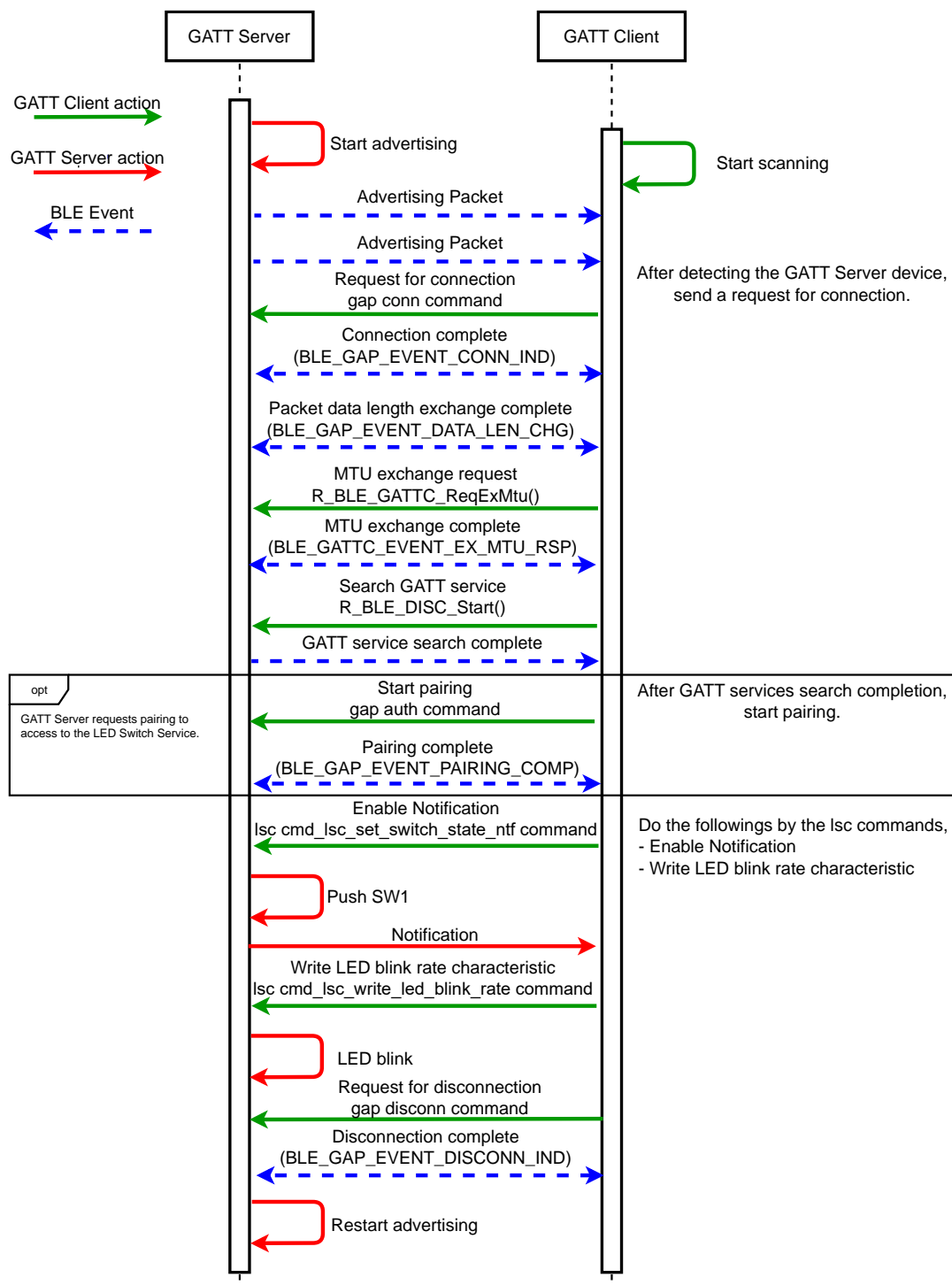


Figure 7.5 : Usage example for the GATT Client demo project and the GATT Server demo project

7.3 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 Trial Tool Suite : R01AN4554).

7.4 Adding the demo project

(1) Select “File” → “Import”.

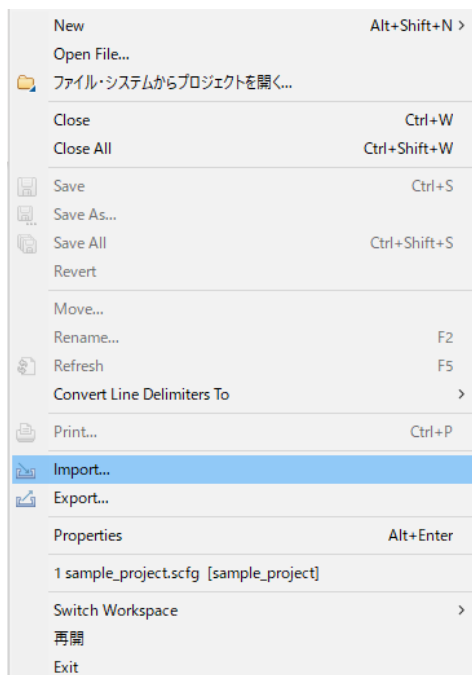


Figure 7.6 : File menu

(2) Select “Existing Projects into Workspace” and click “Next” button.

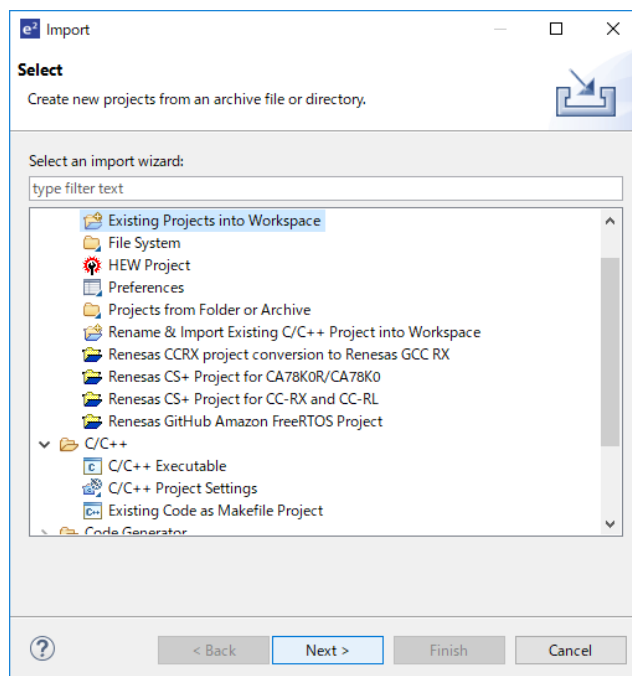


Figure 7.7 : Select an import wizard

(3) Select “Select archive file”, click “Browse...” button and select the demo project archive files. Click “Finish” button and the demo project is imported.

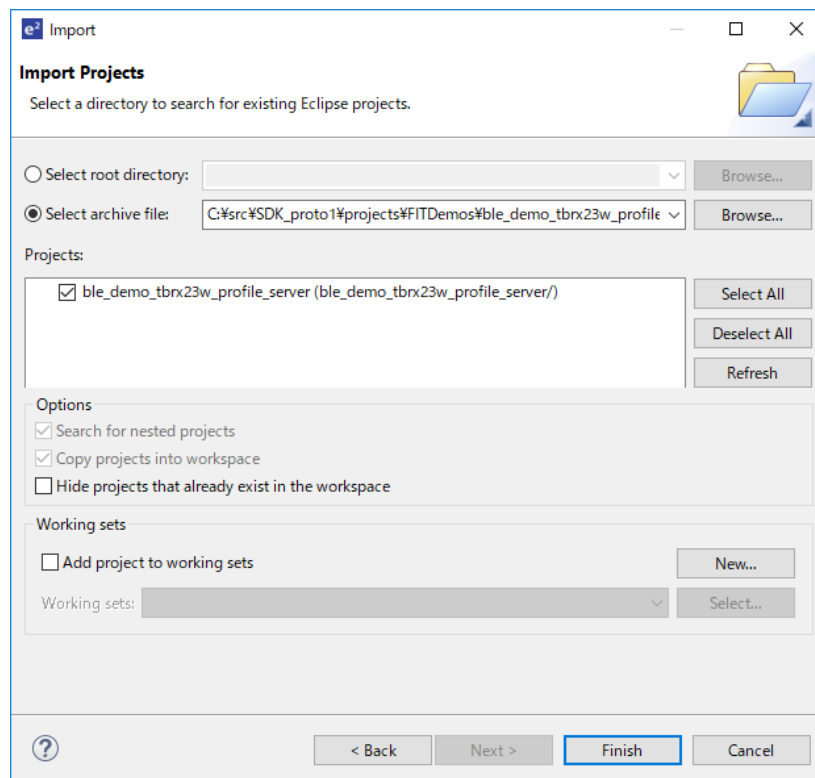


Figure 7.8 : Import Projects

8. Appendices

8.1 Confirmed Operation Environment

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

Table 8.1 : Confirmed Operation Environment (Rev.1.10)

Item	Contents
Integrated development environment	Renesas Electronics e2 studio Version 7.6.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.08.00 IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
Endian	little endian
Revision of the module	Rev 1.10
Board used	Target Board for RX23W (RTK5RX23W0C00000BJ) RSSK RX23W(RTK5523W8AC00001BJ)

8.2 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. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:
Application note "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)"
- Using e2 studio:
Application note "Adding Firmware Integration Technology Modules to Projects (R01AN1723)"

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 R_BLE_VS_ADDR_AREA_REG(0x00).
1.10	Dec.26.2019	—	Supported the following compilers. - IAR C/C++ Compiler for Renesas RX
		Library	<ul style="list-style-type: none"> - 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 advertizing.

		Program	<ul style="list-style-type: none"> - 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)
--	--	---------	---

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 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.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

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

6. 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.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. 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.
8. 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.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(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.4.0-1 November 2017)

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/.