

RX Family

R01AN6290EJ0101

Rev.1.01

Mar.31.22

RYZ012 Bluetooth Low Energy Module Using Firmware Integration Technology

Introduction

This application note describes the usage of the RYZ012 Bluetooth Low Energy module control module software, which conforms to the Firmware Integration Technology (FIT) standard.

In the following pages, the RYZ012 Bluetooth Low Energy module control module is referred to collectively as “the RYZ012 Bluetooth Low Energy FIT module” or “the FIT module.”

The FIT module supports the following Bluetooth Low Energy module:

Renesas Electronics RYZ012 Bluetooth Low Energy Module (RYZ012x1).

In the following pages, the Renesas Electronics RYZ012 (RYZ012x1) is referred to as “the BLE module.”

The FIT module makes use of the following FIT modules:

- RX Family Board Support Package Module (R01AN1685)
- RX Family SCI Module (R01AN1815)
- RX Family BYTEQ, byte-based circular buffers, Module (R01AN1683)

Target Device

RX65N Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833)
- 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 Smart Configurator User's Guide: e² studio (R20AN0451)
- RX Family SCI Module Using Firmware Integration Technology (R01AN1815)
- RX Family BYTEQ Module Using Firmware Integration Technology (R01AN1683)

Contents

1. Overview.....	4
1.1 RYZ012 Bluetooth Low Energy FIT Module.....	4
1.2 Overview of RYZ012 Bluetooth Low Energy FIT Module	4
1.2.1 Software configuration.....	4
1.2.2 Structure of Files	5
2. Requirements	6
2.1 Hardware Requirements	6
2.2 Software Requirements.....	6
2.3 Supported Toolchain	6
2.4 Interrupt Vector.....	6
2.5 Configuration settings.....	7
2.6 Adding the FIT Module to Your Project.....	8
3. BLE Interface (Common).....	9
3.1 R_BLE_Open().....	9
3.2 R_BLE_Close().....	10
3.3 R_BLE_Execute().....	11
3.4 R_BLE_SetEvent().....	12
4. BLE Interface (GAP).....	13
4.1 R_BLE_GAP_Init()	13
4.2 R_BLE_GAP_SetAdvParam().....	14
4.3 R_BLE_GAP_SetAdvSresData()	16
4.4 R_BLE_GAP_StartAdv()	18
4.5 R_BLE_GAP_StopAdv()	20
5. BLE Interface (GATT Common)	21
5.1 R_BLE_GATT_GetMtu()	21
6. BLE Interface (GATT Server)	22
6.1 R_BLE_GATTS_Init().....	22
6.2 R_BLE_GATTS_SetDbInst().....	23
6.3 R_BLE_GATTS_RegisterCb().....	24
6.4 R_BLE_GATTS_Notification().....	25
6.5 R_BLE_GATTS_Indication()	26
6.6 R_BLE_GATTS_GetAttr()	27
6.7 R_BLE_GATTS_SetAttr().....	29
7. BLE Interface (GATT Client).....	31
7.1 R_BLE_GATTC_Init().....	31
7.2 R_BLE_GATTC_RegisterCb()	32

RX Family RYZ012 Bluetooth Low Energy Module Using Firmware Integration Technology

8. BLE Interface (Vendor Specific)	33
8.1 R_BLE_VS_SetTxPower()	33
8.2 R_BLE_VS_SetBdAddr()	35
9. Abstraction API for Renesas QE for BLE	36
9.1 RM_BLE_ABS_Open().....	36
9.2 RM_BLE_ABS_Close()	37
9.3 RM_BLE_ABS_StartLegacyAdvertising()	38
10. Sample Code Generation Using QE for BLE.....	39
11. Confirmed Operation Environment.....	53
12. Reference Documents	54
Revision History	55

1. Overview

1.1 RYZ012 Bluetooth Low Energy FIT Module

The FIT module is designed to be added to user projects as an API. For instructions on adding the FIT module, refer to “2.6, Adding the FIT Module to Your Project” and “10, Sample Code Generation Using QE for BLE”.

1.2 Overview of RYZ012 Bluetooth Low Energy FIT Module

1.2.1 Software configuration

Figure 1.1 shows the software configuration.

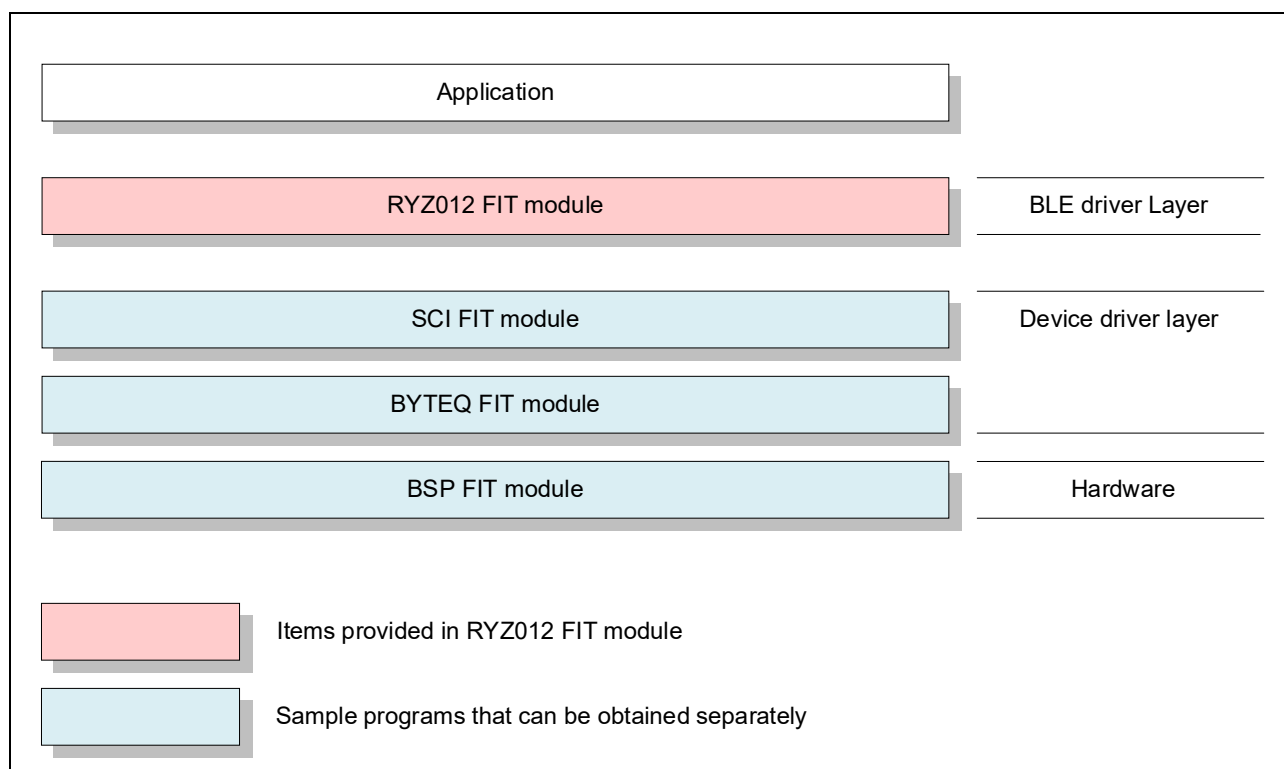


Figure 1.1 Software Configuration Diagram

- (1) RYZ012 FIT module
The FIT module. This software is used to control the BLE module.
- (2) SCI FIT module
Implements communication between the BLE module and the MCU. A sample program is available. Refer to “Related Documents” on page 1 and obtain the software.
- (3) BYTEQ FIT module
Implements circular buffers used by the SCI FIT module. A sample program is available. Refer to “Related Documents” on page 1 and obtain the software.
- (4) BSP FIT module
The Board Support Package module. A sample program is available. Refer to “Related Documents” on page 1 and obtain the software.

1.2.2 Structure of Files

This product includes the files listed in Table 1.1 below.

Table 1.1 Structure of Files

File/Directory (Bold) Name		Description
r_config		
	r_ryz012_rx_config.h	Config .h file for the FIT module
r_ryz012_rx		
	doc	
	en	Folder to store the FIT module Application Note (English)
	r01an6290ej0101-rx-ryz012-ble.pdf	The FIT module Application Note (English)
	Ja	Folder to store the FIT module Application Note (Japanese)
	r01an6290jj0101-rx-ryz012-ble.pdf	The FIT module Application Note (Japanese)
	src	
	hal_data.c	.c file which includes HAL function
	hal_data.h	.h file declarations about HAL function
	r_ble_spp.c	.c file which includes the FIT module functions of communication processing block
	r_ble_spp.h	.h file declarations about the FIT module functions of communication processing block
	r_ble_spp_api.c	.c file which includes the FIT module functions of interface block
	rm_ble_abs_api.h	.h file declarations about interface with functions generated by QE for BLE
	rm_ble_abs_spp.c	.c file which includes functions to interface with functions generated by QE for BLE
	wrap_sci.c	.c file which includes SCI driver wrapper function
	wrap_sci.h	.h file declarations about SCI driver wrapper function
	r_ble_api.h	Interface .h file for the FIT module
	rm_ble_abs.h	Interface .h file for functions generated by QE for BLE

2. Requirements

The FIT module has been confirmed to operate under the following conditions.

2.1 Hardware Requirements

The MCU used must support the following functions:

- Serial communication
- I/O ports

2.2 Software Requirements

The driver is dependent upon the following FIT modules:

- r_bsp
- r_sci_rx
- r_byteq_rx

2.3 Supported Toolchain

The FIT module has been confirmed to work with the toolchain listed in 11, Confirmed Operation Environment.

2.4 Interrupt Vector

None

2.5 Configuration settings

The configuration option settings of the FIT module are contained in `r_ryz012_rx_config.h` and `r_sci_rx_config.h`.

The names of the options for the FIT module and their setting values are listed in Table 2.1. The names of the options for SCI and their setting values are listed in Table 2.2. The settings listed in Table 2.1 and Table 2.2 are an example when using RSKRX65N-2MB as a Target Board.

Table 2.1 Configuration options(`r_ryz012_rx_config.h`)

Configuration options in <code>r_ryz012_rx_config.h</code>	
BLE_CFG_SCI_CHANNEL Note: The default is 6.	Set the SCI channel number assigned to PMOD-2 and 3.
BLE_CFG_RESET_PORT Note: The default is F.	Set GPIO for PMOD-8. e.g., PF5
BLE_CFG_RESET_PIN Note: The default is 5.	#define BLE_CFG_RESET_PORT F #define BLE_CFG_RESET_PIN 5
BLE_CFG_SCI_MODE_PORT Note: The default is G.	Set GPIO for PMOD-9. e.g., PG4
BLE_CFG_SCI_MODE_PIN Note: The default is 4.	#define BLE_CFG_SCI_MODE_PORT G #define BLE_CFG_SCI_MODE_PIN 4

Table 2.2 Configuration options(`r_sci_rx_config.h`)

Configuration options in <code>r_sci_rx_config.h</code>	
SCI_CFG_CHx_INCLUDED Notes: 1. CHx = CH0 to CH12 2. The default values are as follows: CH0 to CH5 and CH7 to CH12: 0, CH6: 1	Each channel has resources such as transmit and receive buffers, counters, interrupts, other programs, and RAM. Setting this option to 1 assigns related resources to the specified channel.
SCI_CFG_CHx_TX_BUFSIZ Notes: 1. CHx = CH0 to CH12 2. The default value is 80 for all channels.	Specifies the transmit buffer size of an individual channel. The buffer size of the channel specified by BLE_CFG_SCI_CHANNEL should be sufficiently larger than data size which should be transmitted.
SCI_CFG_CHx_RX_BUFSIZ Notes: 1. CHx = CH0 to CH12 2. The default value is 80 for all channels.	Specifies the receive buffer size of an individual channel. The buffer size of the channel specified by BLE_CFG_SCI_CHANNEL should be sufficiently larger than data size which should be received.
SCI_CFG_TEI_INCLUDED Note: The default is 0.	Enables the transmit end interrupt for serial transmissions. This option should be set to 1.

2.6 Adding the FIT Module to Your Project

The FIT module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

- (1) Adding the FIT module to your project using the Smart Configurator in e² studio
By using the Smart Configurator in e² studio, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: e² studio (R20AN0451)” for details.
- (2) Adding the FIT module to your project using the FIT Configurator in e² studio
By using the FIT Configurator in e² studio, the FIT module is automatically added to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.
- (3) Adding the FIT module to your project using the Smart Configurator in CS+
By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: CS+ (R20AN0470)” and “10, Sample Code Generation Using QE for BLE” 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 “RX Family Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

3. BLE Interface (Common)

3.1 R_BLE_Open()

This function opens the BLE protocol stack.

Format

```
ble_status_t R_BLE_Open (  
    void  
)
```

Parameters

None

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
----------------------------	----------------

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function should be called once before using the BLE protocol stack.

Reentrant

No

Examples

```
R_BLE_Open();
```

Special Notes

None

3.2 R_BLE_Close()

This function closes the BLE protocol stack.

Format

```
ble_status_t R_BLE_Close (  
    void  
)
```

Parameters

None

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
----------------------------	----------------

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function should be called once to close the BLE protocol stack.

Reentrant

No

Examples

```
R_BLE_Close();
```

Special Notes

None

3.3 R_BLE_Execute()

This function executes the BLE task.

Format

```
ble_status_t R_BLE_Execute (  
    void  
)
```

Parameters

None

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
----------------------------	----------------

Properties

Prototype declarations are contained in `r_ble_api.h`.

Description

This handles all the task queued in the BLE protocol stack internal task queue and return. This function should be called repeatedly in the main loop.

Reentrant

No

Examples

```
R_BLE_Open();  
  
while(1)  
{  
    R_BLE_Execute();  
}
```

Special Notes

None

3.4 R_BLE_SetEvent()

This function adds an event in the BLE protocol stack internal queue.

Format

```
ble_status_t R_BLE_SetEvent (  
    ble_event_cb_t cb  
)
```

Parameters

[in] *cb* *The callback for the event.*

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_ALREADY_IN_PROGRESS(0x000A)</i>	<i>The event already registered with the callback.</i>
<i>BLE_ERR_CONTEXT_FULL(0x000B)</i>	<i>No free slot for the event.</i>

Properties

Prototype declarations are contained in `r_ble_api.h`.

Description

The event is handled in `R_BLE_Execute` just like Bluetooth event. This function is intended to be called in hardware interrupt context. Even if calling this function with the same *cb* before the *cb* is invoked, only one event is registered. The maximum number of the events can be registered at a time is eight.

Reentrant

No

Examples

None

Special Notes

None

4. BLE Interface (GAP)

4.1 R_BLE_GAP_Init()

This function initializes the Host Stack.

Format

```
ble_status_t R_BLE_GAP_Init (
    ble_gap_app_cb_t gap_cb
)
```

Parameters

[in] *gap_cb* A callback function registered with this function.

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	<i>gap_cb is specified as NULL.</i>
<i>BLE_ERR_INVALID_STATE(0x0008)</i>	<i>The reason for this error is as follows:</i> <ul style="list-style-type: none"><i>Host Stack was already initialized.</i><i>The task for host stack is not running.</i>
<i>BLE_ERR_MEM_ALLOC_FAILED(0x000C)</i>	<i>Insufficient memory is needed to generate this function.</i>

Properties

Prototype declarations are contained in r_ble_api.h.

Description

Host stack is initialized with this function. Before using All the R_BLE APIs, it's necessary to call this function. A callback function is registered with this function. In order to receive the GAP event, it's necessary to register a callback function. The result of this API call is notified in BLE_GAP_EVENT_STACK_ON event.

Reentrant

No

Examples

None

Special Notes

None

4.2 R_BLE_GAP_SetAdvParam()

This function sets advertising parameters.

Format

```
ble_status_t R_BLE_GAP_SetAdvParam (
    st_ble_gap_adv_param_t * p_adv_param
)
```

Parameters

[in] *p_adv_param* Advertising parameters.

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	<i>p_adv_param is specified as NULL.</i>
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	<i>The below p_adv_param field value is out of range.</i> <ul style="list-style-type: none"> <i>adv_handle</i> <i>adv_intv_min/adv_intv_max</i> <i>adv_ch_map</i> <i>o_addr_type</i> <i>p_addr_type</i> <i>adv_phy</i> <i>sec_adv_phy</i> <i>scan_req_ntf_flag</i>
<i>BLE_ERR_INVALID_STATE(0x0008)</i>	<i>The task for host stack is not running.</i>
<i>BLE_ERR_MEM_ALLOC_FAILED(0x000C)</i>	<i>Insufficient memory is needed to generate this function.</i>

Properties

Prototype declarations are contained in *r_ble_api.h*.

Description

This function sets advertising parameters. It's possible to do advertising where the advertising parameters are different every each advertising set. The number of advertising set in the Controller is defined as *BLE_MAX_NO_OF_ADV_SETS_SUPPORTED*. Each advertising set is identified with advertising handle (0x00-0x03). Create an advertising set with this function before start advertising, setting periodic advertising parameters, start periodic advertising, setting advertising data/scan response data/periodic advertising data. The result of this API call is notified in *BLE_GAP_EVENT_ADV_PARAM_SET_COMP* event.

Reentrant

No

Examples

None

Special Notes

None

4.3 R_BLE_GAP_SetAdvSresData()

This function sets advertising data/scan response data/periodic advertising data.

Format

```
ble_status_t R_BLE_GAP_SetAdvSresData (
    st_ble_gap_adv_data_t * p_adv_srsp_data
)
```

Parameters

[in] *p_adv_srsp_data* Advertising data/scan response data/periodic advertising data.

Return Values

<i>BLE_SUCCESS(0x0000)</i>	Success
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	The reason for this error is as follows: <ul style="list-style-type: none"> <i>p_adv_srsp_data</i> is specified as NULL. <i>data_length</i> field in <i>p_adv_srsp_data</i> parameter is not 0 and <i>p_data</i> field is specified as NULL.
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	The following field in <i>p_adv_srsp_data</i> parameter is out of range. <ul style="list-style-type: none"> <i>adv_hdl</i> <i>data_type</i> <i>data_length</i> <i>zero_length_flag</i>
<i>BLE_ERR_INVALID_STATE(0x0008)</i>	The task for host stack is not running.
<i>BLE_ERR_MEM_ALLOC_FAILED(0x000C)</i>	Insufficient memory is needed to generate this function.

Properties

Prototype declarations are contained in *r_ble_api.h*.

Description

This function sets advertising data/scan response data/periodic advertising data to the advertising set. It is necessary to create an advertising set by *R_BLE_GAP_SetAdvParam()*, before calling this function. Set advertising data/scan response data/periodic advertising data, after allocating the memory for the data.

Reentrant

No

Examples

None

Special Notes

None

4.4 R_BLE_GAP_StartAdv()

This function starts advertising.

Format

```
ble_status_t R_BLE_GAP_StartAdv (
    uint8_t adv_hdl,
    uint16_t duration,
    uint8_t max_extd_adv_evts
)
```

Parameters

- | | | |
|-------------|--------------------------|---|
| <i>[in]</i> | <i>adv_hdl</i> | <i>The advertising handle pointing to the advertising set which starts advertising.
The valid range is 0x00 – 0x03.</i> |
| <i>[in]</i> | <i>duration</i> | <i>The duration for which the advertising set identified by adv_hdl is enabled.
Time = duration * 10ms. When the duration expires, BLE_GAP_EVENT_ADV_OFF event notifies that advertising is stopped. The valid range is 0x0000 – 0xFFFF. The duration parameter is ignored when the value is set to 0x0000.</i> |
| <i>[in]</i> | <i>max_extd_adv_evts</i> | <i>The maximum number of advertising events that be sent during advertising.
When all the advertising events(max_extd_adv_evts) have been sent, BLE_GAP_EVENT_ADV_OFF event notifies that advertising is stopped. The max_extd_adv_evts parameter is ignored when the value is set to 0x00.</i> |

Return Values

- | | |
|---|---|
| <i>BLE_SUCCESS(0x0000)</i> | <i>Success</i> |
| <i>BLE_ERR_INVALID_ARG(0x0003)</i> | <i>adv_hdl is out of range.</i> |
| <i>BLE_ERR_INVALID_STATE(0x0008)</i> | <i>The task for host stack is not running.</i> |
| <i>BLE_ERR_MEM_ALLOC_FAILED(0x000C)</i> | <i>Insufficient memory is needed to generate this function.</i> |

Properties

Prototype declarations are contained in r_ble_api.h.

Description

Create the advertising set specified with *adv_hdl* by R_BLE_GAP_SetAdvParam(), before calling this function. The result of this API call is notified in BLE_GAP_EVENT_ADV_ON event.

Reentrant

No

Examples

None

Special Notes

None

4.5 R_BLE_GAP_StopAdv()

This function stops advertising.

Format

```
ble_status_t R_BLE_GAP_StopAdv (  
    uint8_t adv_hdl  
)
```

Parameters

[in] *adv_hdl* The advertising handle pointing to the advertising set which stops advertising.
 The valid range is 0x00 – 0x03.

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	<i>adv_hdl is out of range.</i>
<i>BLE_ERR_INVALID_STATE(0x0008)</i>	<i>The task for host stack is not running.</i>
<i>BLE_ERR_MEM_ALLOC_FAILED(0x000C)</i>	<i>Insufficient memory is needed to generate this function.</i>

Properties

Prototype declarations are contained in `r_ble_api.h`.

Description

This function stops advertising. The result of this API call is notified in BLE_GAP_EVENT_ADV_OFF event.

Reentrant

No

Examples

None

Special Notes

None

5. BLE Interface (GATT Common)

5.1 R_BLE_GATT_GetMtu()

This function gets the current MTU used in GATT communication.

Format

```
ble_status_t R_BLE_GATT_GetMtu(  
    uint16_t conn_hdl,  
    uint16_t * p_mtu  
)
```

Parameters

<i>[in]</i>	<i>conn_hdl</i>	Connection handle identifying the GATT Server or the GATT Client.
<i>[in]</i>	<i>p_mtu</i>	The Current MTU. Before MTU exchange, this parameter is 23 bytes. After MTU exchange, this parameter is the negotiated MTU.

Return Values

<i>BLE_SUCCESS(0x0000)</i>	Success
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	The mtu parameter is NULL.
<i>BLE_ERR_INVALID_HDL(0x000E)</i>	The GATT Server or the GATT Client specified by <i>conn_hdl</i> was not found.

Properties

Prototype declarations are contained in *r_ble_api.h*.

Description

Both GATT server and GATT Client can use this function. The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

6. BLE Interface (GATT Server)

6.1 R_BLE_GATTS_Init()

This function initializes the GATT Server and registers the number of the callbacks for GATT Server event.

Format

```
ble_status_t R_BLE_GATTS_Init(  
    uint8_t cb_num  
)
```

Parameters

[in] *cb_num* The number of callbacks to be registered.

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	<i>The cb_num parameter is out of range.</i>

Properties

Prototype declarations are contained in r_ble_api.h.

Description

Specify the *cb_num* parameter to a value between 1 and BLE_GATTS_MAX_CB.
R_BLE_GATTS_RegisterCb() registers the callback. The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

6.2 R_BLE_GATTS_SetDbInst()

This function sets GATT Database to host stack.

Format

```
ble_status_t R_BLE_GATTS_SetDbInst(  
    st_ble_gatts_db_cfg_t * p_db_inst  
)
```

Parameters

[in] *p_db_inst* GATT Database to be set.

Return Values

BLE_SUCCESS(0x0000) Success

BLE_ERR_INVALID_PTR(0x0001) The reason for this error is as follows.

- The *db_inst* parameter is specified as NULL.
- The array in the *db_inst* is specified as NULL.

Properties

Prototype declarations are contained in *r_ble_api.h*.

Description

The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

6.3 R_BLE_GATTS_RegisterCb()

This function registers a callback for GATT Server event.

Format

```
ble_status_t R_BLE_GATTS_RegisterCb(  
    ble_gatts_app_cb_t cb,  
    uint8_t priority  
)
```

Parameters

[in] <i>cb</i>	<i>Callback function for GATT Server event.</i>
[in] <i>priority</i>	<i>The priority of the callback function.</i> <i>Valid range is 1 <= priority <= BLE_GATTS_MAX_CB.</i> <i>A lower priority number means a higher priority level.</i>

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	<i>The cb parameter is specified as NULL.</i>
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	<i>The priority parameter is out of range.</i>
<i>BLE_ERR_CONTEXT_FULL(0x000B)</i>	<i>Host stack has already registered the maximum number of callbacks.</i>

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The number of the callback that may be registered by this function is the value specified by R_BLE_GATTS_Init(). The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

6.4 R_BLE_GATTS_Notification()

This function sends a notification of an attribute's value.

Format

```
ble_status_t R_BLE_GATTS_Notification(
    uint16_t conn_hdl,
    st_ble_gatt_hdl_value_pair_t * p_ntf_data
)
```

Parameters

[in] *conn_hdl* Connection handle identifying the remote device to be sent the notification.

[in] *p_ntf_data* The attribute value to send.

Return Values

<i>BLE_SUCCESS(0x0000)</i>	Success
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	The <i>p_ntf_data</i> parameter or the value field in the value field in the <i>p_ntf_data</i> parameter is NULL.
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	The <i>value_len</i> field in the value field in the <i>p_ntf_data</i> parameter is 0 or the <i>attr_hdl</i> field in the <i>p_ntf_data</i> parameter is 0.
<i>BLE_ERR_INVALID_OPERATION(0x0009)</i>	This function was called while processing another request.
<i>BLE_ERR_MEM_ALLOC_FAILED(0x000C)</i>	Insufficient memory is needed to generate this function.
<i>BLE_ERR_INVALID_HDL(0x000E)</i>	The remote device specified by <i>conn_hdl</i> was not found.

Properties

Prototype declarations are contained in *r_ble_api.h*.

Description

The maximum length of the attribute value that can be sent with notification is MTU-3. The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

6.5 R_BLE_GATTS_Indication()

This function sends an indication of an attribute's value.

Format

```
ble_status_t R_BLE_GATTS_Indication(
    uint16_t conn_hdl,
    st_ble_gatt_hdl_value_pair_t * p_ind_data
)
```

Parameters

[in] *conn_hdl* Connection handle identifying the remote device to be sent the indication.

[in] *p_ind_data* The attribute value to send.

Return Values

<i>BLE_SUCCESS(0x0000)</i>	Success
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	The <i>p_ind_data</i> parameter or the value field in the value field in the <i>p_ind_data</i> parameter is NULL.
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	The <i>value_len</i> field in the value field in the <i>p_ind_data</i> parameter is 0 or the <i>attr_hdl</i> field in the <i>p_ind_data</i> parameter is 0.
<i>BLE_ERR_INVALID_OPERATION(0x0009)</i>	This function was called while processing another request.
<i>BLE_ERR_MEM_ALLOC_FAILED(0x000C)</i>	Insufficient memory is needed to generate this function.
<i>BLE_ERR_INVALID_HDL(0x000E)</i>	The remote device specified by <i>conn_hdl</i> was not found.

Properties

Prototype declarations are contained in *r_ble_api.h*.

Description

The maximum length of the attribute value that can be sent with indication is MTU-3. The result of this API call is returned by a return value. The remote device that receives an indication sends a confirmation. BLE_GATTS_EVENT_HDL_VAL_CNF event notifies the application layer that the confirmation has been received.

Reentrant

No

Examples

None

Special Notes

None

6.6 R_BLE_GATTS_GetAttr()

This function gets an attribute value from the GATT Database.

Format

```
ble_status_t R_BLE_GATTS_GetAttr(
    uint16_t conn_hdl,
    uint16_t attr_hdl,
    st_ble_gatt_value_t * p_value
)
```

Parameters

<i>[in]</i>	<i>conn_hdl</i>	<i>If the attribute value that has information about the remote device is retrieved, specify the remote device with the conn_hdl parameter. When information about the remote device is not required, set the conn_hdl parameter to BLE_GAP_INVALID_CONN_HDL.</i>
<i>[in]</i>	<i>attr_hdl</i>	<i>The attribute handle of the attribute value to be retrieved.</i>
<i>[out]</i>	<i>p_value</i>	<i>The attribute value to be retrieved.</i>

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	<i>The p_value parameter is specified as NULL.</i>
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	<i>The attr_hdl parameter is 0 or larger than the last attribute handle of GATT Database.</i>
<i>BLE_ERR_INVALID_STATE(0x0008)</i>	<i>The attribute is not in a state to be read.</i>
<i>BLE_ERR_INVALID_OPERATION(0x0009)</i>	<i>The attribute cannot be read.</i>
<i>BLE_ERR_NOT_FOUND(0x000D)</i>	<i>The attribute specified by the attr_hdl parameter is not belonging to any services or characteristics.</i>
<i>BLE_ERR_INVALID_HDL(0x000E)</i>	<i>The remote device specified by the conn_hdl parameter was not found.</i>

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

6.7 R_BLE_GATTS_SetAttr()

This function sets an attribute value to the GATT Database.

Format

```
ble_status_t R_BLE_GATTS_SetAttr(
    uint16_t conn_hdl,
    uint16_t attr_hdl,
    st_ble_gatt_value_t * p_value
)
```

Parameters

[in] conn_hdl	If the attribute value that has information about the remote device is retrieved, specify the remote device with the conn_hdl parameter. When information about the remote device is not required, set the conn_hdl parameter to BLE_GAP_INVALID_CONN_HDL.
[in] attr_hdl	The attribute handle of the attribute value to be set.
[in] p_value	The attribute value to be set.

Return Values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_value parameter is specified as NULL.
BLE_ERR_INVALID_DATA(0x0002)	The write size is larger than the length of the attribute value.
BLE_ERR_INVALID_ARG(0x0003)	The attr_hdl parameter is 0 or larger than the last attribute handle of GATT Database.
BLE_ERR_INVALID_STATE(0x0008)	The attribute is not in a state to be written.
BLE_ERR_INVALID_OPERATION(0x0009)	The attribute cannot be written.
BLE_ERR_NOT_FOUND(0x000D)	The attribute specified by the attr_hdl parameter is not belonging to any services or characteristics.
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by the conn_hdl parameter was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

7. BLE Interface (GATT Client)

7.1 R_BLE_GATTC_Init()

This function initializes the GATT Client and registers the number of the callbacks for GATT Client event.

Format

```
ble_status_t R_BLE_GATTC_Init(  
    uint8_t cb_num  
)
```

Parameters

[in] *cb_num* *The number of callbacks to be registered.*

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	<i>The cb_num parameter is out of range.</i>

Properties

Prototype declarations are contained in *r_ble_api.h*.

Description

Specify the *cb_num* parameter to a value between 1 and *BLE_GATTC_MAX_CB*.
R_BLE_GATTC_RegisterCb() registers the callback. The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

7.2 R_BLE_GATTC_RegisterCb()

This function registers a callback function for GATT Client event.

Format

```
ble_status_t R_BLE_GATTC_RegisterCb(
    ble_gattc_app_cb_t cb,
    uint8_t priority
)
```

Parameters

<i>[in]</i>	<i>cb</i>	<i>Callback function for GATT Client event.</i>
<i>[in]</i>	<i>priority</i>	<i>The priority of the callback function.</i>
		<i>Valid range is 1 <= priority <= BLE_GATTC_MAX_CB.</i>
		<i>A lower priority number means a higher priority level.</i>

Return Values

<i>BLE_SUCCESS(0x0000)</i>	<i>Success</i>
<i>BLE_ERR_INVALID_PTR(0x0001)</i>	<i>The cb parameter is specified as NULL.</i>
<i>BLE_ERR_INVALID_ARG(0x0003)</i>	<i>The priority parameter is out of range.</i>
<i>BLE_ERR_CONTEXT_FULL(0x000B)</i>	<i>Host stack has already registered the maximum number of callbacks.</i>

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The number of the callback that may be registered by this function is the value specified by R_BLE_GATTC_Init(). The result of this API call is returned by a return value.

Reentrant

No

Examples

None

Special Notes

None

8. BLE Interface (Vendor Specific)

8.1 R_BLE_VS_SetTxPower()

This function configures transmit power.

Format

```
ble_status_t R_BLE_VS_SetTxPower (
    uint16_t conn_hdl,
    uint8_t tx_power
)
```

Parameters

- | | | |
|-------------|-----------------|--|
| <i>[in]</i> | <i>conn_hdl</i> | Connection handle identifying the link whose transmit power to be configured.
If non connected state, set BLE_GAP_INIT_CONN_HDL (0xFFFF). |
| <i>[in]</i> | <i>tx_power</i> | Transmission power. Select one of the following. <ul style="list-style-type: none"> • BLE_VS_TX_POWER_HIGH (0x00) • BLE_VS_TX_POWER_MID (0x01) • BLE_VS_TX_POWER_LOW (0x02) |

Return Values

- | | |
|--|--|
| <i>BLE_SUCCESS</i> (0x0000) | Success |
| <i>BLE_ERR_INVALID_STATE</i> (0x0008) | The task for host stack is not running. |
| <i>BLE_ERR_MEM_ALLOC_FAILED</i> (0x000C) | There are no memories for Vendor Specific Command. |

Properties

Prototype declarations are contained in *r_ble_api.h*.

Description

This function configures the following transmit power.

- The transmit power used in sending advertising PDU, scan request PDU, connection request PDU (in not connected state)
- The transmit power used in sending PDU in connected state. When configuring the transmit power used in not connected state, set the *conn_hdl* parameter to BLE_GAP_INIT_CONN_HDL(0xFFFF).

When the transmit power used in connected state is configured, set the *conn_hdl* parameter to the connection handle of the link. Select one of the following transmit power levels: High, Middle or Low.

Reentrant

No

Examples

None

Special Notes

None

8.2 R_BLE_VS_SetBdAddr()

This function sets public/random address of local device to the area specified by the parameter.

Format

```
ble_status_t R_BLE_VS_SetBdAddr(
    uint8_t area,
    st_ble_dev_addr_t * p_addr
)
```

Parameters

[in] area	The area that the address is to be written in.
[in] p_addr	The address to be set to the area. Set BLE_GAP_ADDR_PUBLIC(0x00) or BLE_GAP_ADDR_RAND(0x01) to the type field in the p_addr parameter.

Return Values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_addr parameter is specified as NULL.
BLE_ERR_INVALID_STATE(0x0008)	The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	There are no memories for Vendor Specific Command.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

If the address is written in non-volatile area, the address is used as default address on the next MCU reset. For more information on the random address, refer to Core Specification Vol 6, PartB, "1.3.2 Random Device Address". The result of this API call is notified in BLE_VS_EVENT_SET_ADDR_COMP event.

Reentrant

No

Examples

None

Special Notes

None

9. Abstraction API for Renesas QE for BLE

9.1 RM_BLE_ABS_Open()

Host stack is initialized with this function.

Format

```
fsp_err_t RM_BLE_ABS_Open (
    ble_abs_ctrl_t *const    p_ctrl,
    ble_abs_cfg_t const *const p_cfg
)
```

Parameters

[in] *p_ctrl* *Pointer to control structure.*
[in] *p_cfg* *Pointer to the configuration structure for this instance.*

Return Values

<i>FSP_SUCCESS</i>	<i>Channel opened successfully.</i>
<i>FSP_ERR_ASSERTION</i>	<i>Null pointer presented.</i>
<i>FSP_ERR_INVALID_CHANNEL</i>	<i>The channel number is invalid.</i>
<i>FSP_ERR_ALREADY_OPEN</i>	<i>Requested channel is already open in a different configuration.</i>
<i>FSP_ERR_INVALID_ARGUMENT</i>	<i>Invalid input parameter.</i>

Properties

Prototype declarations are contained in `rm_ble_abs.h`.

Description

Before using all the R_BLE APIs, it's necessary to call this function. A callback functions are registered with this function. In order to receive the GAP, GATT, Vendor specific event, it's necessary to register a callback function. The result of this API call is notified in BLE_GAP_EVENT_STACK_ON event. Implements `ble_abs_api_t::open`.

Reentrant

No

Examples

```
/* Open the module. */
err = RM_BLE_ABS_Open(&g_ble_abs0_ctrl, &g_ble_abs0_cfg);
```

Special Notes

None

9.2 RM_BLE_ABS_Close()

This function closes the BLE channel.

Format

```
fsp_err_t RM_BLE_ABS_Close (  
    ble_abs_ctrl_t * const p_ctrl  
)
```

Parameters

[in] *p_ctrl* *Pointer to control structure.*

Return Values

FSP_SUCCESS *Channel closed successfully.*

FSP_ERR_ASSERTION *Null pointer presented.*

FSP_ERR_NOT_OPEN *Control block not open.*

Properties

Prototype declarations are contained in *rm_ble_abs.h*.

Description

Implements *ble_abs_api_t::close*.

Reentrant

No

Examples

```
/* Close BLE driver */  
err = RM_BLE_ABS_Close(&g_ble_abs0_ctrl);
```

Special Notes

None

9.3 RM_BLE_ABS_StartLegacyAdvertising()

This function starts Legacy Advertising after setting advertising parameters, advertising data and scan response data.

Format

```
fsp_err_t RM_BLE_ABS_StartLegacyAdvertising (
    ble_abs_ctrl_t * const      p_ctrl,
    ble_abs_legacy_advertising_parameter_t const * const p_advertising_parameter
)
```

Parameters

[in] <i>p_ctrl</i>	<i>Pointer to control structure.</i>
[in] <i>p_advertising_parameter</i>	<i>Pointer to Advertising parameters for Legacy Advertising.</i>

Return Values

<i>FSP_SUCCESS</i>	<i>Operation succeeded</i>
<i>FSP_ERR_ASSERTION</i>	<i>p_instance_ctrl is specified as NULL.</i>
<i>FSP_ERR_NOT_OPEN</i>	<i>Control block not open.</i>
<i>FSP_ERR_INVALID_STATE</i>	<i>Host stack hasn't been initialized.</i>
<i>FSP_ERR_INVALID_POINTER</i>	<i>p_advertising_parameter is specified as NULL.</i>
<i>FSP_ERR_INVALID_ARGUMENT</i>	<i>The advertising parameter is out of range.</i>

Properties

Prototype declarations are contained in `rm_ble_abs.h`.

Description

The legacy advertising uses the advertising set whose advertising handle is 0. The advertising type is connectable and scannable (ADV_IND). The address type of local device is Public Identity Address or RPA (If the resolving list contains no matching entry, use the public address.). Scan request event (BLE_GAP_EVENT_SCAN_REQ_RECV) is not notified. Implements `ble_abs_api_t::startLegacyAdvertising`

Reentrant

No

Examples

```
/* Start advertising. */
err = RM_BLE_ABS_StartLegacyAdvertising(&g_ble_abs0_ctrl,
&legacy_advertising_parameter);
```

Special Notes

None

10. Sample Code Generation Using QE for BLE

This section describes how to create a new e² studio project with the FIT module added using Smart Configurator and how to generate sample code using QE for BLE in that project. The settings in this section are an example when using RSKRX65N-2MB as a Target Board.

1. Copy the FIT module to the folder listed below to be available that with Smart Configurator.

Files to be copied:

r_ryz012_rx_v1.00.xml

r_ryz012_rx_v1.00.zip

r_ryz012_rx_v1.00_extend.mdf

Copy to:

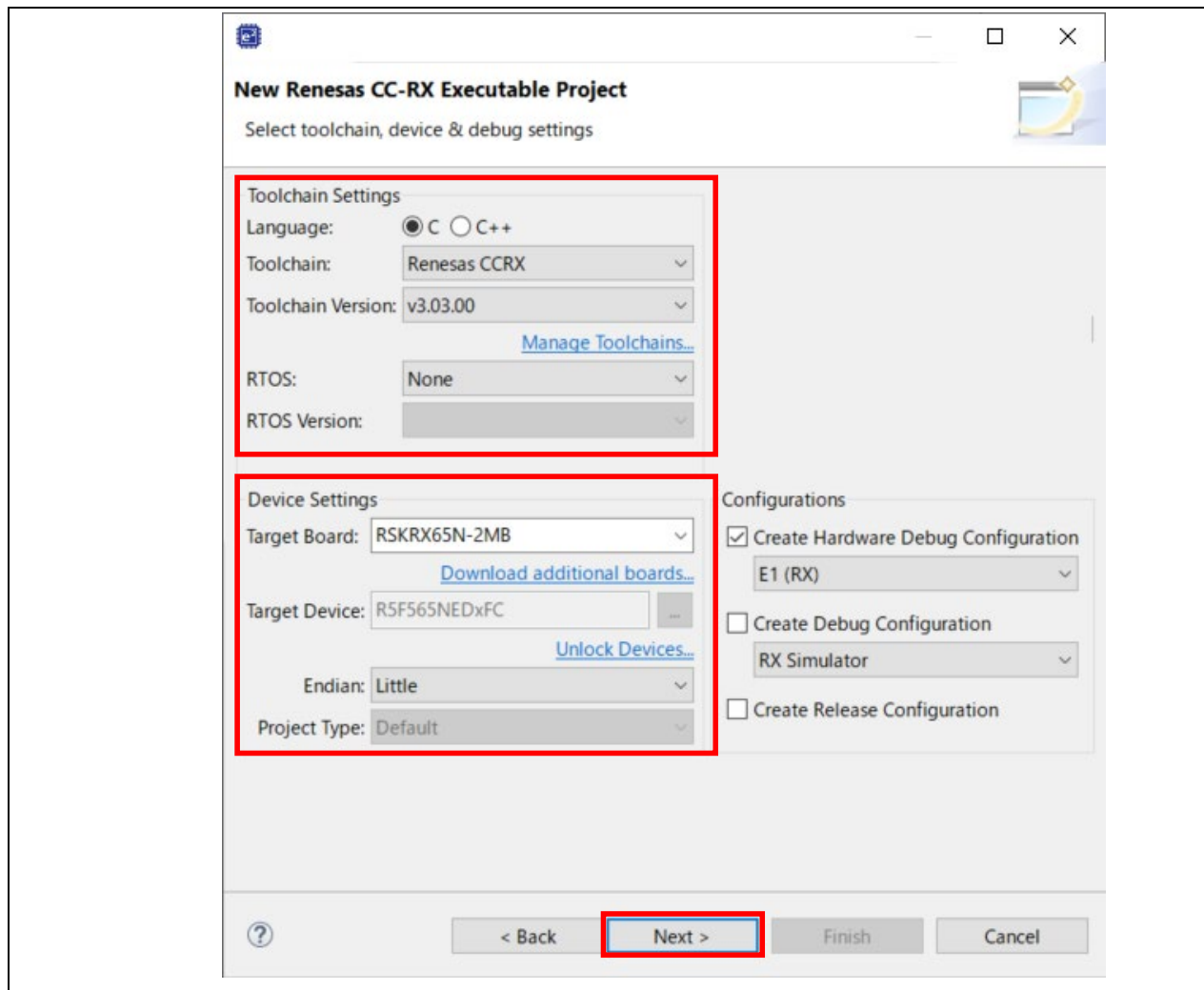
C:\Users\[User name]\.eclipse\com.renesas.platform_download\FITModules\

RX Family RYZ012 Bluetooth Low Energy Module Using Firmware Integration Technology

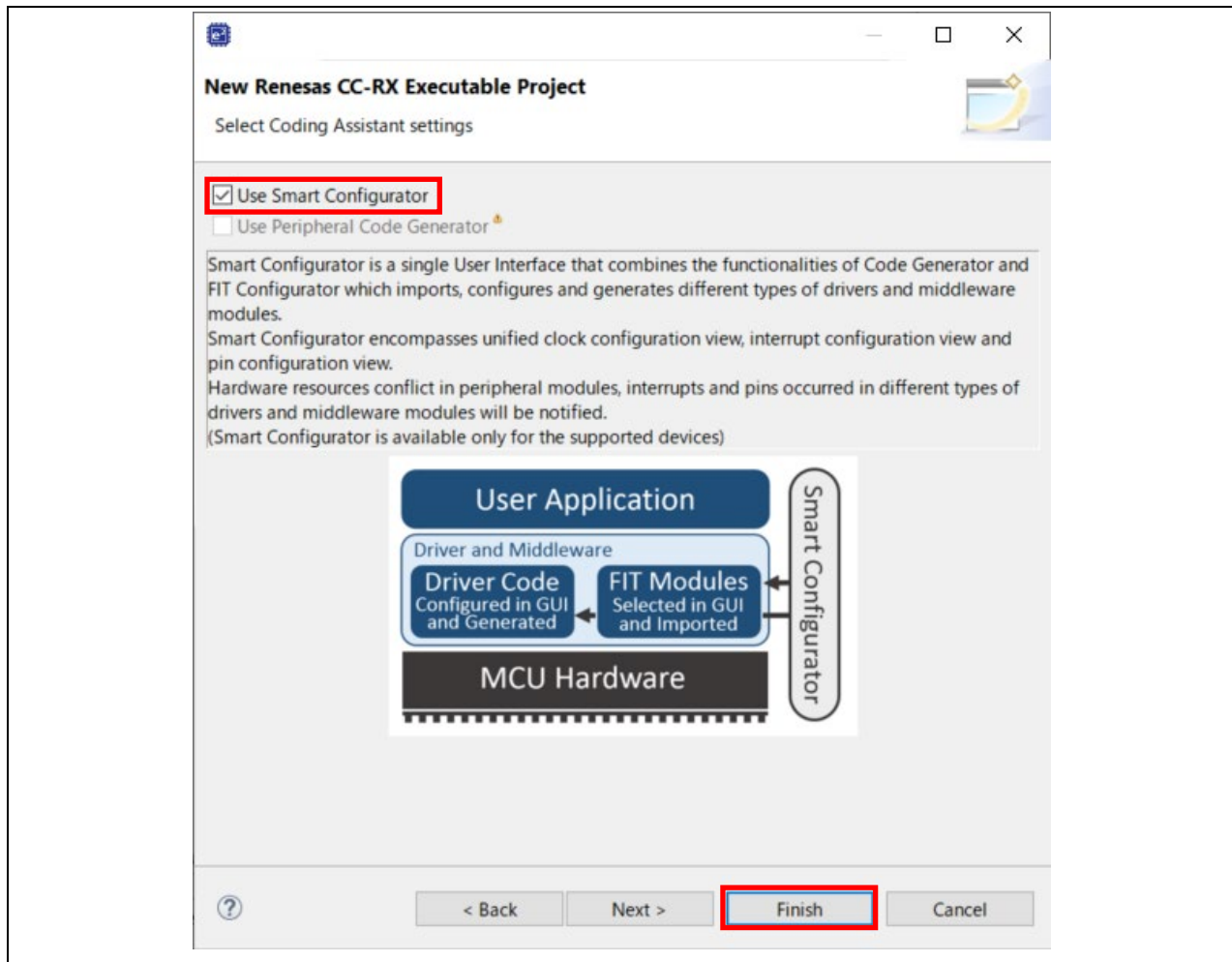
2. Create a new e² studio project with the following settings:

- Renesas CC-RX C/C++ Executable Project
- Project Name: (Arbitrary)
- RTOS: None
- Target Board: RSKRX65N-2MB

※The Target Board should be set to RSKRX65N-2MB when using TSIP-equipped RX MCU, R5F565NEHxFC, similarly.

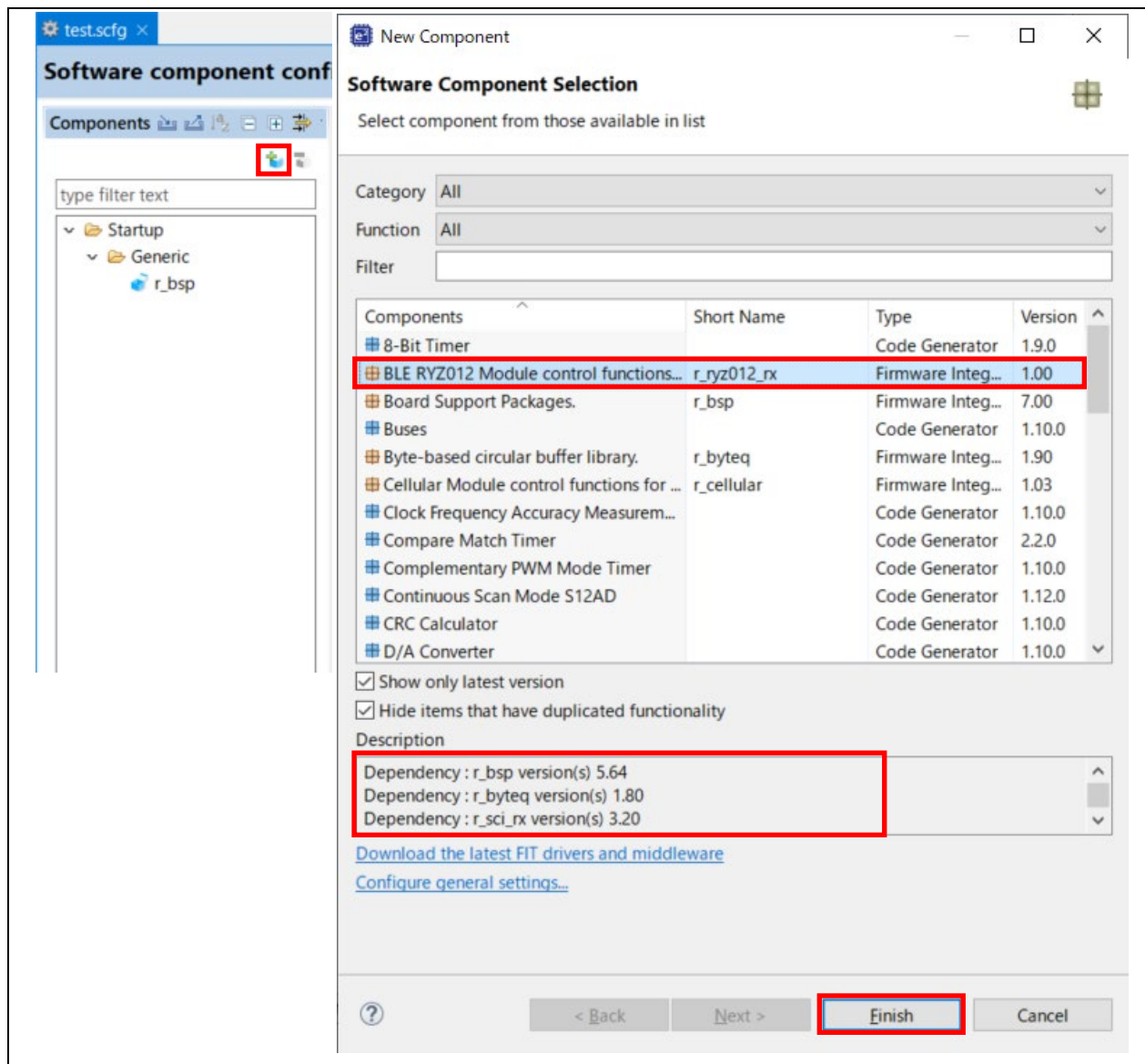


3. Check **Use Smart Configurator** and click the **Finish** button.

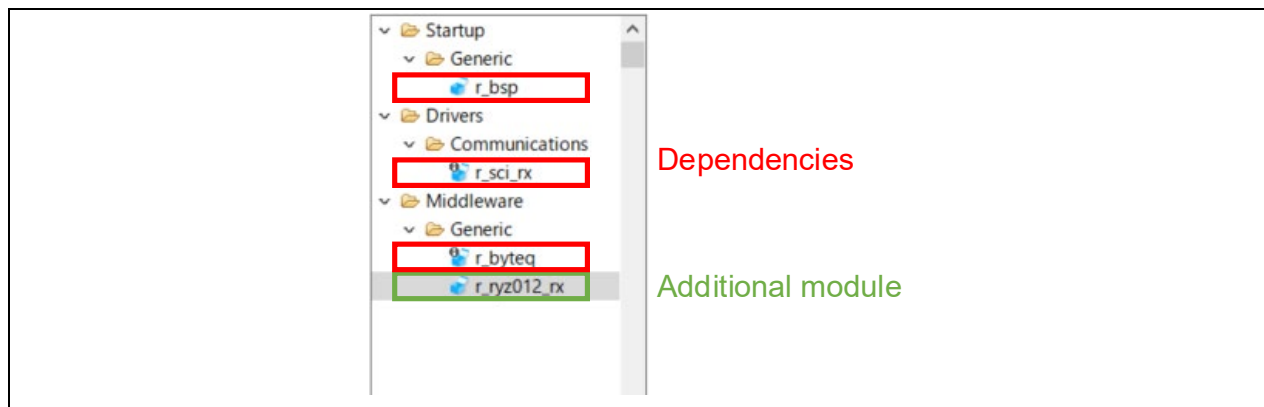


RX Family RYZ012 Bluetooth Low Energy Module Using Firmware Integration Technology

- Click **Add component** button on **Components** tab in the Smart Configurator perspective to add the FIT module to the list of components in **New Component** window. Select the FIT module added and click **Finish** button on the window.

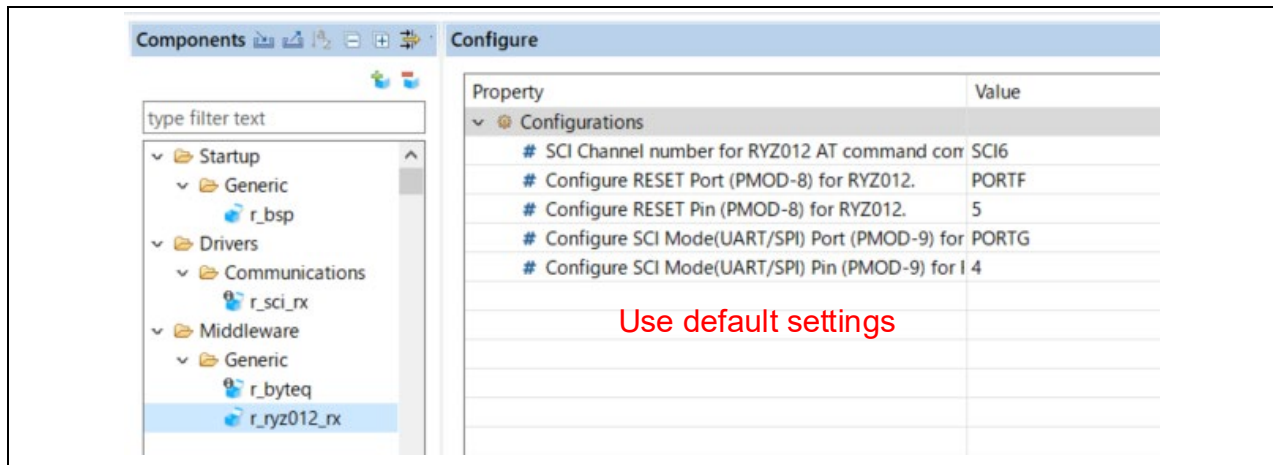


Make sure the FIT module is added.

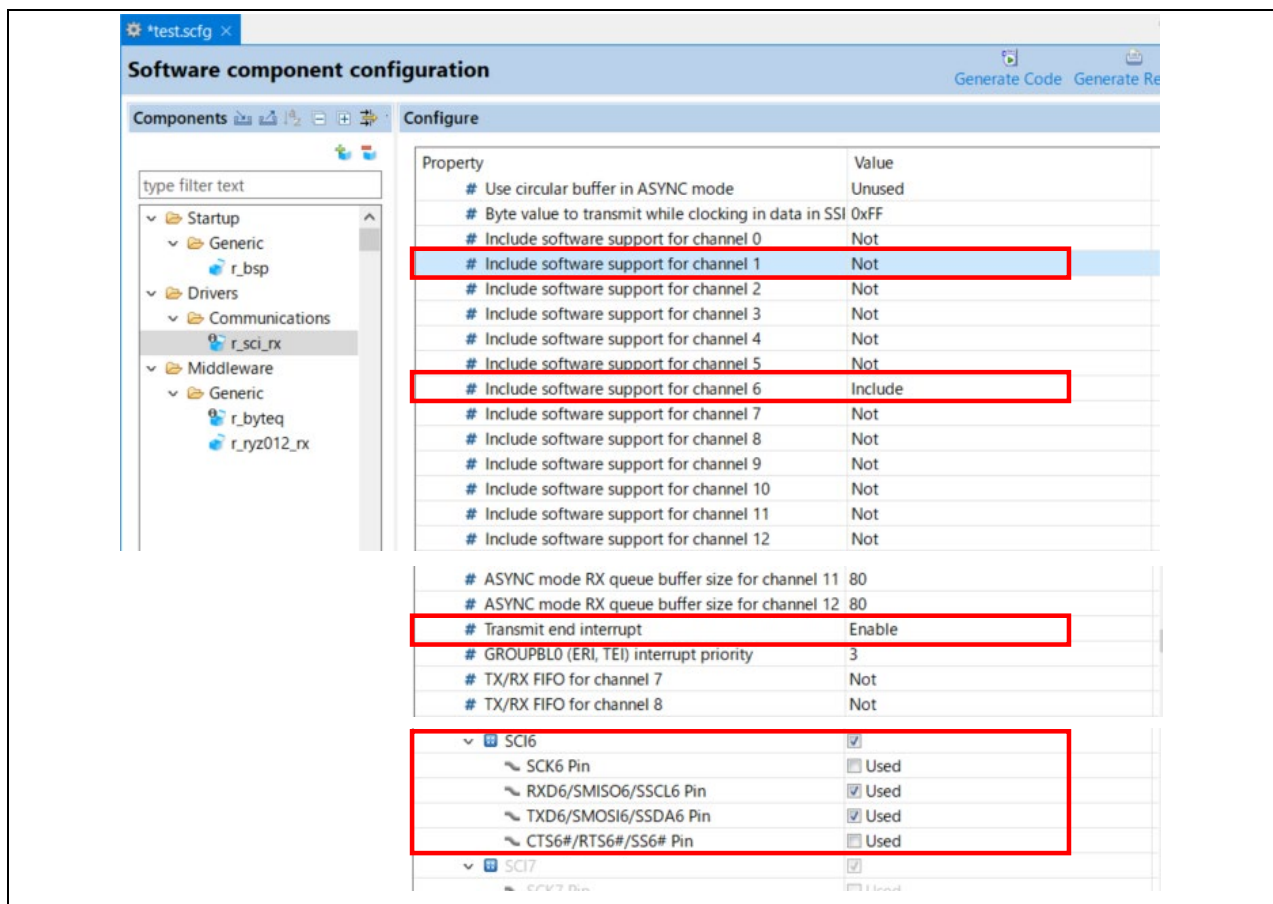


5. Set configures.

r_ryz012_rx



r_sci_rx (When using RSKRX65N-2MB)



Pin settings

Enabl...	Function	Assignment	Pin Number	Direction	Remarks
<input type="checkbox"/>	CTS6#	/ Not assigned	/ Not assigned	None	
<input type="checkbox"/>	RTS6#	/ Not assigned	/ Not assigned	None	
<input type="checkbox"/>	RXD6	/ Not assigned	/ Not assigned	None	
<input type="checkbox"/>	SCK6	/ Not assigned	/ Not assigned	None	
<input checked="" type="checkbox"/>	SMISO6	/ P01/TMCI0/RXD6/SMISO6/SSCL6/IRQ9/AN119	/ 7	IO	
<input checked="" type="checkbox"/>	SMOSI6	/ P00/TMRI0/TXD6/SMOSI6/SSDA6/IRQ8/AN118	/ 8	IO	
<input type="checkbox"/>	SS6#	/ Not assigned	/ Not assigned	None	
<input type="checkbox"/>	SSCL6	/ Not assigned	/ Not assigned	None	
<input type="checkbox"/>	SSDA6	/ Not assigned	/ Not assigned	None	
<input type="checkbox"/>	TXD6	/ Not assigned	/ Not assigned	None	

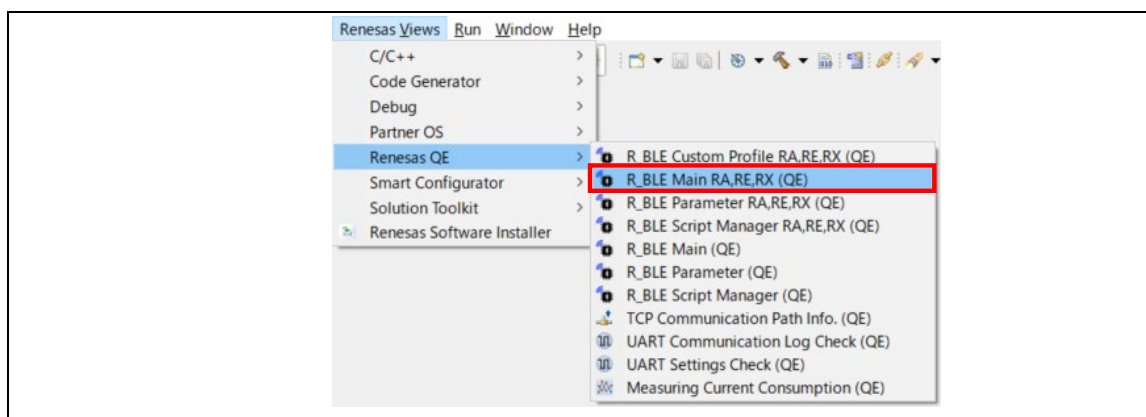
- Click **Generate Code** button to add the FIT module and its dependencies in smc_gen folder.

The 'Generate Code' button is highlighted in the top toolbar. The 'Project Explorer' on the right shows the project structure with the following folders highlighted in the 'src' directory:

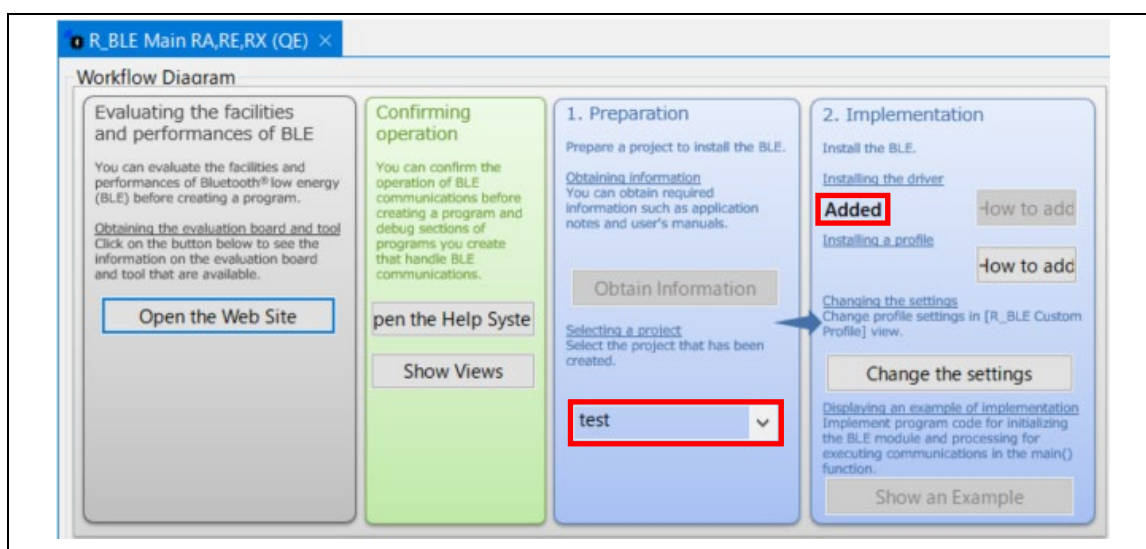
- test
 - Includes
 - src
 - smc_gen
 - general
 - r_bsp
 - r_byteq**
 - r_config
 - r_pincfg
 - r_ryz012_rx**
 - r_sci_rx**
 - test.c
 - trash
 - test.scfg
 - test HardwareDebug.launch

7. Open QE for BLE window to generate sample code.

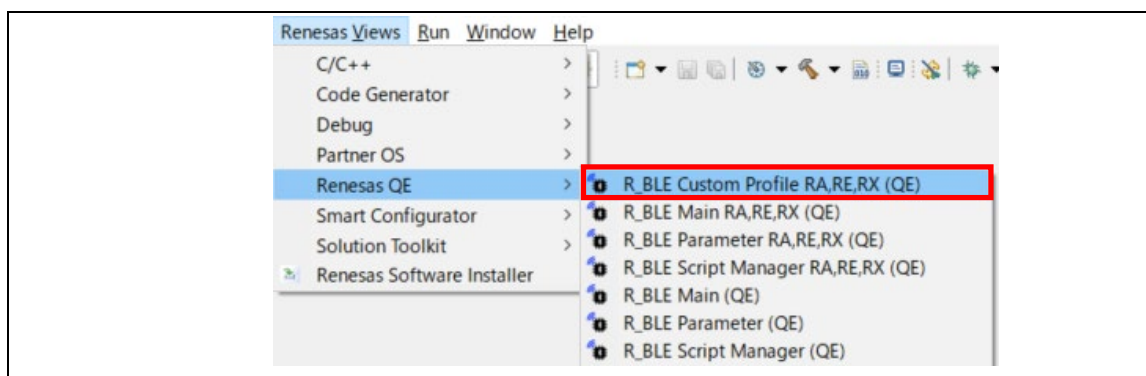
(1) Open R_BLE Main tab.



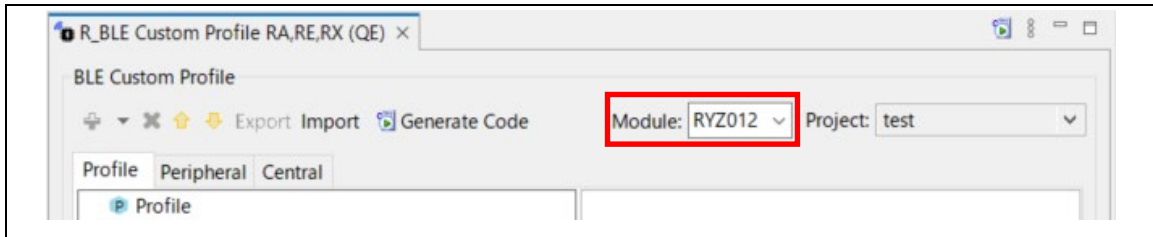
(2) Select the project created in Step 2 and 3. Confirm **Added** is displayed.



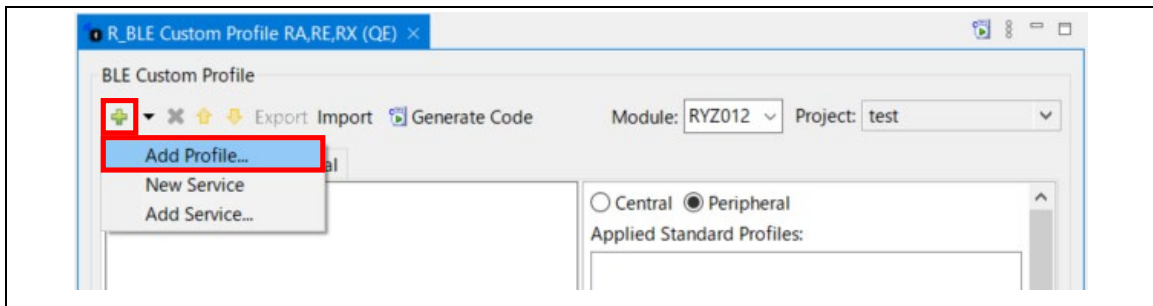
(3) Open R_BLE Custom Profile tab.



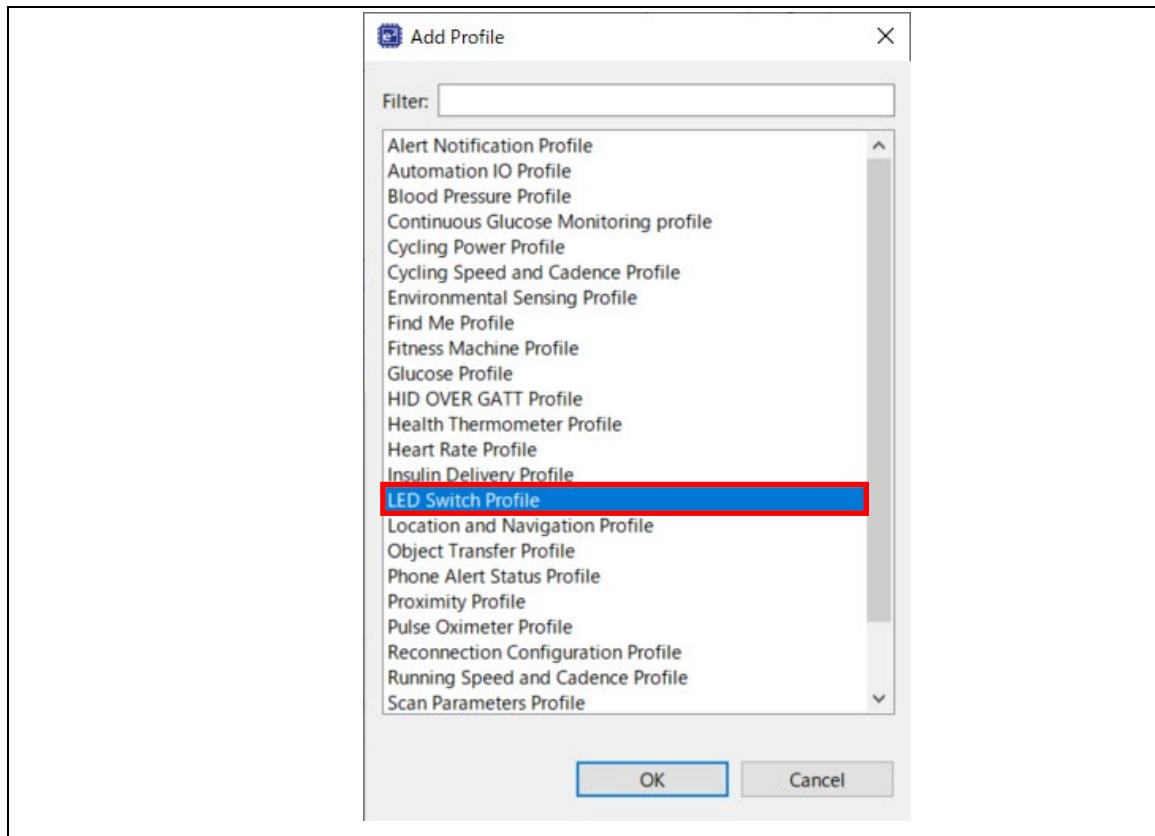
- (4) Select “RYZ012” from the **Module** drop-down list.



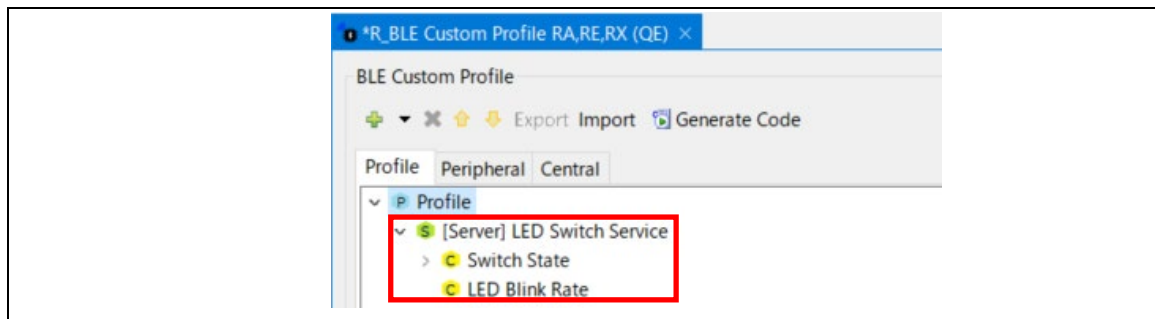
- (5) Open **Add Profile** window.



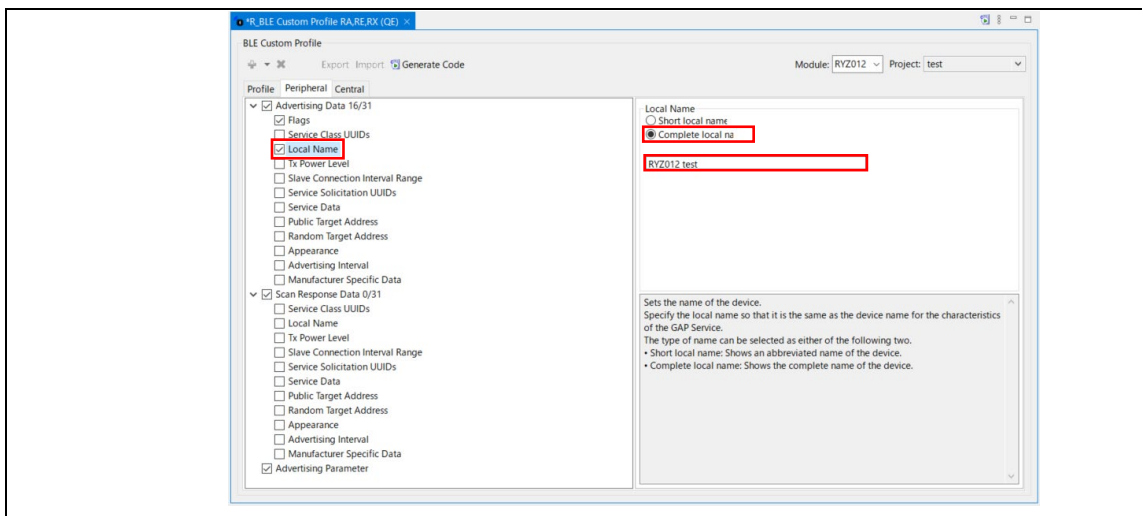
(6) Select “LED Switch Profile”.



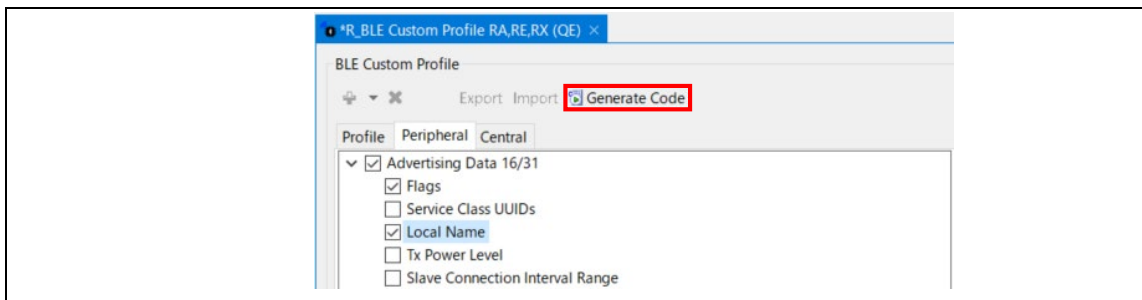
Then the profile is added.



- (7) Check “Local Name” checkbox in **Peripheral** tab.
In **Local Name** group box, select “Complete local name” and enter the device name string in the text box. (In the screen image below, the device name string is “RYZ012 test”.)



- (8) Click **Generate Code** button to make QE for BLE generate sample code.



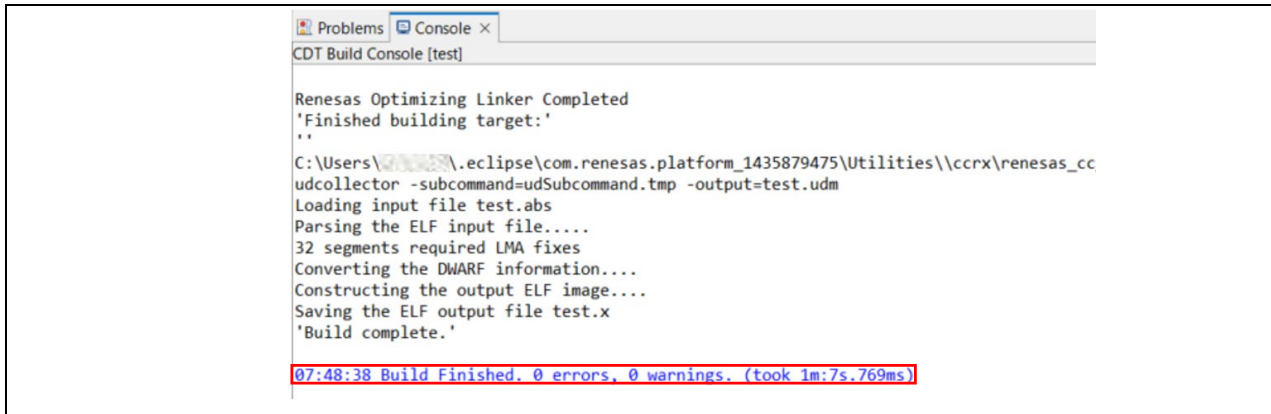
8. In the e² studio project explorer, open the file src\[Project name].c including the main function and add the yellow highlighted code, resulting in the code shown below:

```
#include "r_smc_entry.h"

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

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


9. Build the project and confirm no build error occurs.



The screenshot shows the CDT Build Console window with the following output:

```
Problems Console x
CDT Build Console [test]

Renesas Optimizing Linker Completed
'Finished building target:'
''
C:\Users\...\.eclipse\com.renesas.platform_1435879475\Utilities\ccrx\renesas_cc
udcollector -subcommand=udSubcommand.tmp -output=test.udm
Loading input file test.abs
Parsing the ELF input file.....
32 segments required LMA fixes
Converting the DWARF information....
Constructing the output ELF image....
Saving the ELF output file test.x
'Build complete.'
```

The final line of the output is highlighted in red: **07:48:38 Build Finished. 0 errors, 0 warnings. (took 1m:7s.769ms)**

10. Set debug settings as shown below and click **OK** button.

Edit Configuration

Edit Renesas GDB Hardware Debugging configuration test HardwareDebug for Debug

Launch Configuration Name: test HardwareDebug

Main Debugger Startup Common Source

Debug hardware: E1 (RX) Target Device: R5F565NE_DUAL

GDB Settings Connection Settings Debug Tool Settings

▼ Clock

Main Clock Source	EXTAL
Extal Frequency[MHz]	27.0
Operating Frequency [MHz]	120.000
Permit Clock Source Change On Writing Internal Flash Memory	Yes

▼ Connection with Target Board

Emulator	(Auto)
Connection Type	JTag
JTag Clock Frequency[MHz]	16.5
Fine Baud Rate[Mbps]	2.00
Hot Plug	No

▼ Power

Power Target From The Emulator (MAX 200mA)	No
Supply Voltage	3.3V

▼ CPU Operating Mode

Register Setting	Single Chip
Mode pin	Single-chip mode
Change startup bank	No
Startup bank	Bank 0

▼ Communication Mode

Mode	Debug Mode
Execute The User Program After Ending The Debugger	No

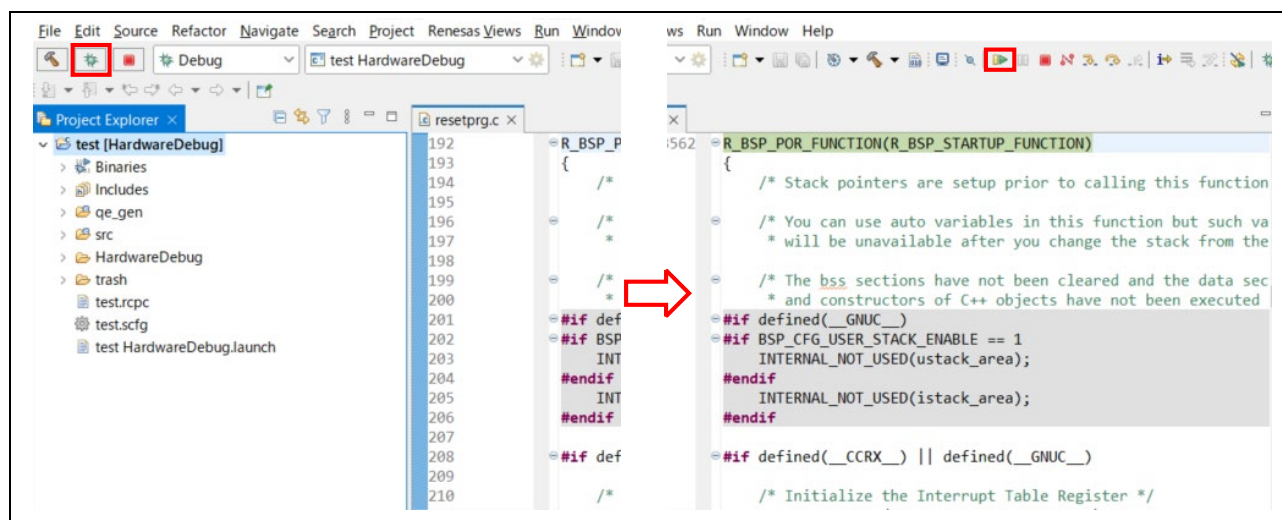
▼ Flash

ID Code	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
---------	----------------------------------

?

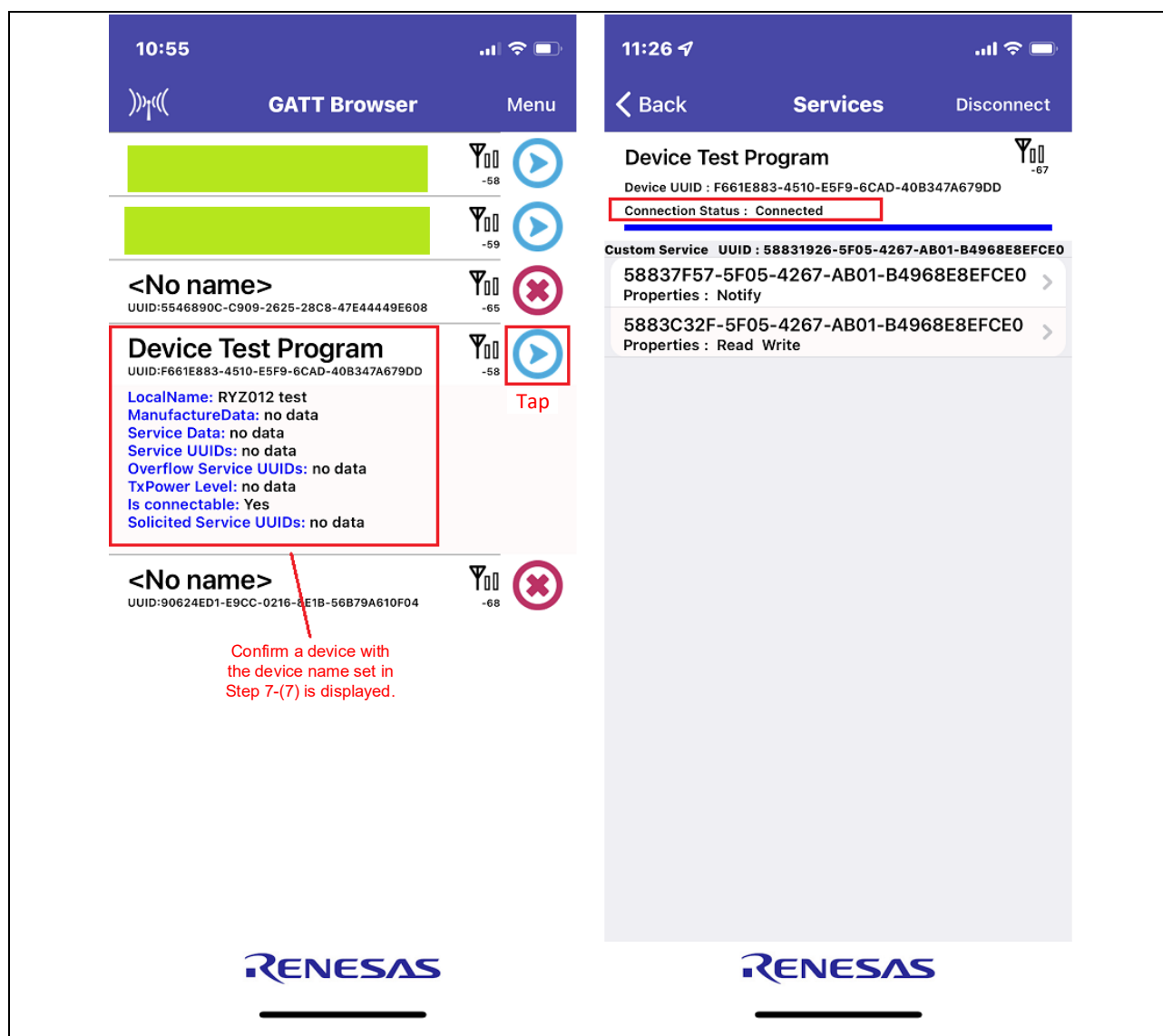
Duplicate Delete **OK** Cancel

11. Click the **Launch in Debug Mode** button to write the application to the target board and execute it.

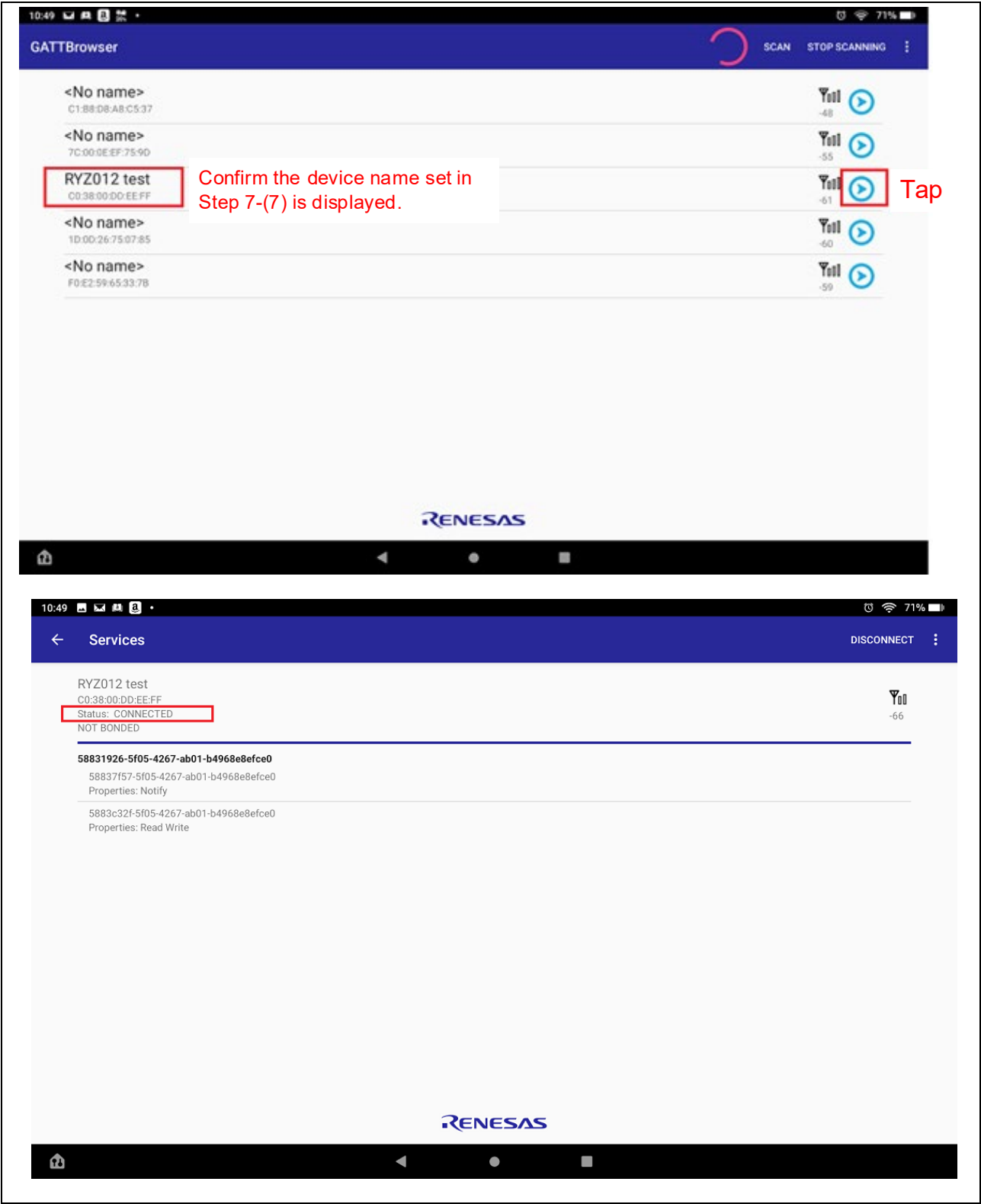


12. Connect to the application from Renesas GATT Browser.

iOS



Android



11. Confirmed Operation Environment

This section describes confirmed operation environment for the FIT module.

Table 11.1 Confirmed Operation Environment

Item	Contents
Integrated development environment	Renesas Electronics e ² studio 2021.10
C compiler	Renesas Electronics C/C++ Compiler for RX Family V3.03.00 Compiler option: The following option should be added to the default settings of the integrated development environment. -lang = c99
Endian order	Little endian
Revision of the module	Rev.1.00
Board used	Renesas Starter Kit+ for RX65N-2MB (Product No.: RTK50565N2CxxxxxBR)

12. Reference Documents

User's Manual: Hardware

(The latest versions can be downloaded from the Renesas Electronics website.)

Technical Update/Technical News

(The latest information can be downloaded from the Renesas Electronics website.)

User's Manual: Development Tools

RX Family CC-RX Compiler User's Manual (R20UT3248)

(The latest versions can be downloaded from the Renesas Electronics website.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.01	2022/03/31	-	First edition issued

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/.