

RL78 Family

US159-DA1453xEVZ BLE Control Module Using Software Integration System

Introduction

This application note describes the usage of the US159-DA1453xEVZ BLE control module, which conforms to the Software Integration System (SIS) standard.

In the following pages, the US159-DA1453xEVZ BLE control module software is referred to collectively as “the DA1453x BLE SIS module” or “the SIS module.”

The SIS module supports the following BLE module:

- DA14531MOD (US159-DA14531EVZ)
- DA14535MOD (US159-DA14535EVZ)

In the following pages, the DA14531MOD and DA14535MOD are referred to as “the BLE module”.

Target Devices

- RL78/G23 Group

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

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RL78 Family (CC-RL)

Related Documents

[1] RL78 Family Board Support Package Module Using Software Integration System (R01AN5522)

[2] RL78 Smart Configurator User's Guide: e² studio (R20AN0579)

[3] Smart Configurator User's Guide: RL78 API Reference (R20UT4852)

[4] RL78/G23 Serial Array Unit (UART Communication) (R01AN6645)

[5] RL78/G23-128p Fast Prototyping Board – User's Manual (R20UT4870)

[6] RL78 Family Renesas Flash Driver RL78 Type 01 User's Manual (R20UT4830EJ0121)

Contents

1. Overview	6
1.1. DA1453x SIS module	6
1.2. Overview of the DA1453x BLE SIS module	6
1.2.1. Connection with DA1453x BLE	6
1.2.2. Software configuration	7
1.3. Features	8
1.4. API Overview	10
1.5. Status Transitions	13
2. API Information	14
2.1. Hardware Requirements	14
2.2. Software Requirements	14
2.3. Support Toolchain	14
2.4. Interrupt Vector	14
2.5. Header Files	14
2.6. Integer Types	14
2.7. Compile Settings	15
2.8. Code Size	17
2.9. Return values	18
2.10. Parameter	21
2.11. Adding the SIS Module to Your Project	28
2.12. “for”, “while” and “do while” Statements	28
2.13. Usage Notes	29
2.13.1 Getting Started Guide	29
2.13.2 Addresses	29
2.13.3 Heap Requirements	29
2.13.4 Module Firmware Compatibility	29
2.13.5 SIS Module Limitations	29
3. API Functions	31
3.1. R_BLE_Open()	31
3.2. R_BLE_Close()	32
3.3. R_BLE_Execute()	33
3.4. R_BLE_IsTaskFree()	34
3.5. R_BLE_GetVersion()	35
3.6. R_BLE_GAP_Init()	36
3.7. R_BLE_GAP_Terminate()	37
3.8. R_BLE_GAP_UpdConn()	38
3.9. R_BLE_GAP_SetDataLen()	40
3.10. R_BLE_GAP_Disconnect()	41

3.11. R_BLE_GAP_GetVerInfo()	42
3.12. R_BLE_GAP_ReadRssi()	43
3.13. R_BLE_GAP_ReadChMap()	44
3.14. R_BLE_GAP_SetAdvParam()	45
3.15. R_BLE_GAP_SetAdvSresData()	47
3.16. R_BLE_GAP_StartAdv()	49
3.17. R_BLE_GAP_StopAdv()	50
3.18. R_BLE_GAP_GetRemainAdvBufSize()	51
3.19. R_BLE_GAP_GetRemDevInfo()	52
3.20. R_BLE_GAP_SetPairingParams()	53
3.21. R_BLE_GAP_StartPairing()	54
3.22. R_BLE_GAP_ReplyPairing()	55
3.23. R_BLE_GAP_ReplyPasskeyEntry()	56
3.24. R_BLE_GAP_ReplyNumComp()	57
3.25. R_BLE_GAP_ReplyExKeyInfoReq()	59
3.26. R_BLE_GAP_ReplyLtkReq()	60
3.27. R_BLE_GAP_StartScan()	62
3.28. R_BLE_GAP_StopScan()	64
3.29. R_BLE_GAP_CreateConn()	65
3.30. R_BLE_GAP_CancelCreateconn()	67
3.31. R_BLE_GAP_ConfWhiteList()	68
3.32. R_BLE_GAP_StartEnc()	70
3.33. R_BLE_GATT_GetMtu()	71
3.34. R_BLE_GATTS_SetDbInst()	72
3.35. R_BLE_GATTS_RegisterCb()	73
3.36. R_BLE_GATTS_DeregisterCb()	74
3.37. R_BLE_GATTS_Notification()	75
3.38. R_BLE_GATTS_Indication()	76
3.39. R_BLE_GATTS_GetAttr()	77
3.40. R_BLE_GATTS_SetAttr()	79
3.41. R_BLE_GATTC_RegisterCb()	81
3.42. R_BLE_GATTC_DeregisterCb()	82
3.43. R_BLE_GATTC_ReqExMtu()	83
3.44. R_BLE_GATTC_DiscAllPrimServ()	84
3.45. R_BLE_GATTC_DiscPrimServ()	85
3.46. R_BLE_GATTC_DiscIncServ()	87
3.47. R_BLE_GATTC_DiscAllChar()	88
3.48. R_BLE_GATTC_DiscCharByUuid()	89
3.49. R_BLE_GATTC_DiscAllCharDesc()	91
3.50. R_BLE_GATTC_ReadChar()	92
3.51. R_BLE_GATTC_ReadCharUsingUuid()	93

3.52.	R_BLE_GATTC_ReadLongChar()	95
3.53.	R_BLE_GATTC_ReadMultiChar()	96
3.54.	R_BLE_GATTC_WriteCharWithoutRsp()	97
3.55.	R_BLE_GATTC_SignedWriteChar()	98
3.56.	R_BLE_GATTC_WriteChar()	99
3.57.	R_BLE_GATTC_WriteLongChar()	100
3.58.	R_BLE_GATTC_ReliableWrites()	102
3.59.	R_BLE_GATTC_ExecWrite()	104
3.60.	R_BLE_VS_Init()	106
3.61.	R_BLE_VS_SetTxPower()	107
3.62.	R_BLE_VS_GetTxPower()	109
3.63.	R_BLE_VS_GetBdAddr()	110
3.64.	R_BLE_VS_SetBdAddr()	111
3.65.	R_BLE_VS_GetRand()	113
4.	Abstraction API for Renesas QE for BLE	114
4.1	RM_BLE_ABS_Open()	114
4.2	RM_BLE_ABS_Close()	115
4.3	RM_BLE_ABS_StartLegacyAdvertising()	116
4.4	RM_BLE_ABS_DeleteBondInformation()	117
4.5	RM_BLE_ABS_StartScanning()	118
4.6	RM_BLE_ABS_CreateConnection()	119
5.	Demo Project	120
5.1	BLE DA1453x Demo Projects	120
5.1.1	Prerequisites	120
5.1.2	Import the Demo Project	121
5.1.3	Hardware Setup	121
5.1.4	Software Setup	122
5.2	BLE DA1453x Peripheral Demo Projects	125
5.2.1	How to Run the Demo	125
5.3	BLE DA1453x Central Demo Projects	130
5.3.1	How to run the demo	130
5.4	Creating a New BLE DA1453x project	132
5.5	Adding a Demo to a Workspace	132
5.6	Downloading Demo Projects	132

6. Appendix 133

6.1 Confirmed Operation Environment..... 133

6.2 How to change UART module to work with BLE module 136

7. Reference Documents 137

Revision History 138

1. Overview

1.1. DA1453x SIS module

The SIS module is designed to be added to user projects as an API. For instruction on adding the SIS module, refer to 2.12 “for”, “while” and “do while” Statements .

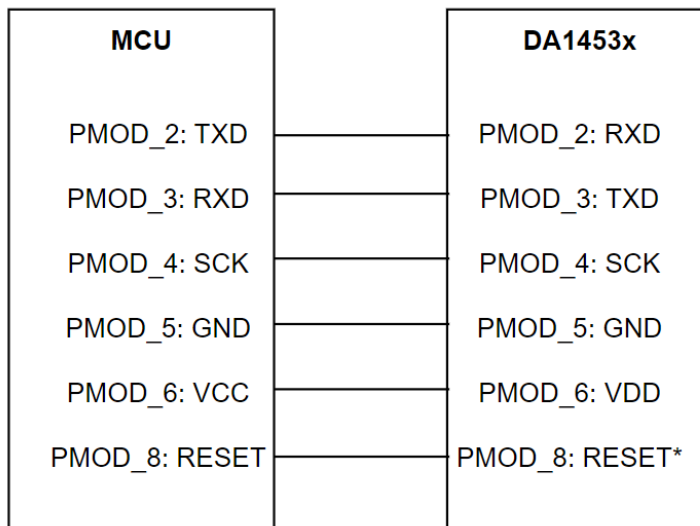
1.2. Overview of the DA1453x BLE SIS module

The DA14531 and DA14535 are ultra-low power SoC integrating a 2.4 GHz transceiver and an Arm® Cortex-M0+ microcontroller, with 4 kB of RAM and 32 kB of One-Time Programmable (OTP) memory for the DA14531, and 64 kB of RAM and 12 kB of OTP memory for the DA14535. Each can be used as a standalone application processor or as a data pump in hosted systems.

The Bluetooth® LE firmware includes the L2CAP service layer protocols, Security Manager (SM), Attribute Protocol (ATT), the Generic Attribute Profile (GATT), and the Generic Access Profile (GAP). All profiles published by the Bluetooth® SIG as well as custom profiles are supported.

1.2.1. Connection with DA1453x BLE

Examples of connection to the DA1453x BLE are shown below.



*Note: Active low with DA1453xMOD

Figure 1.1 Example Connection to the DA1453x Module

1.2.2. Software configuration

Figure 1.2 shows the software configuration.

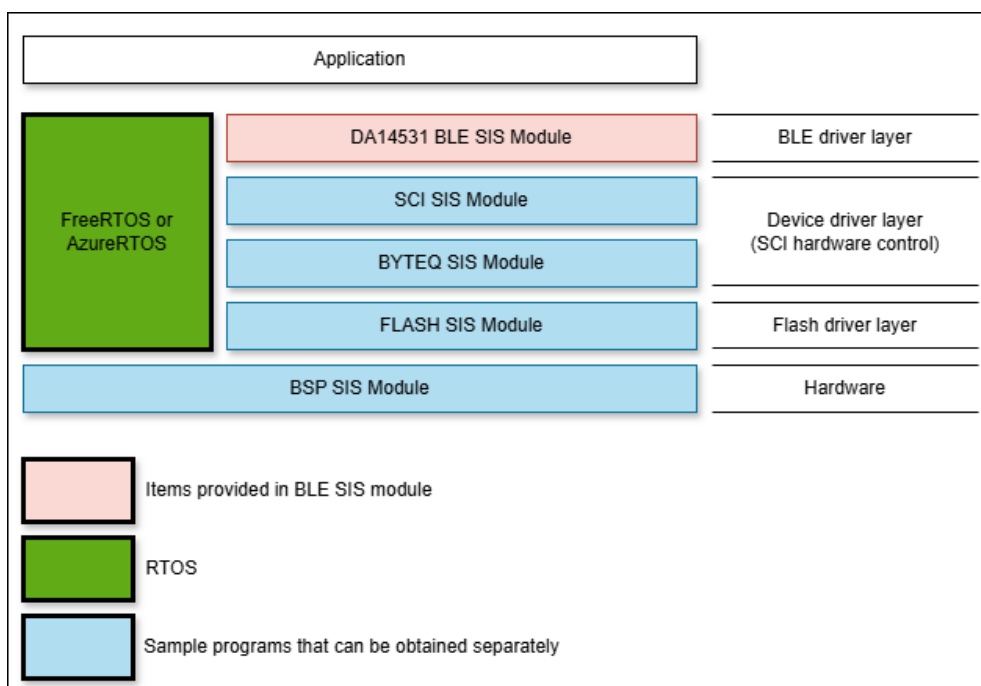


Figure 1.2 Software Configuration Diagram

- DA1453x BLE SIS module
This software is used to control the BLE module.
- SCI SIS 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.
- BYTEQ SIS module
Implements circular buffers used by the SCI SIS module. A sample programs is available. Refer to “Related Documents” on page 1 and obtain the software.
- FLASH SIS module
Allow users of supported devices to easily integrate reprogramming of internal flash memory into their applications using self-programming. A sample program is available. Refer to “Related Documents” on page 1 and obtain the software.
- BSP SIS module
The Board Support Package module. A sample programs is available. Refer to “Related Documents” on page 1 and obtain the software.
- RTOS
The RTOS manages the system overall. Operation of the SIS module has been verified using FreeRTOS or Bare metal by BSP_CFG_RTOS_USED.

1.3. Features

The Bluetooth Low Energy Abstraction module with GTL supports the following features:

- Common functionality for DA1453x
 - Boot from host or DA14531/DA14535 flash
 - Use the 1-wire (default) or the 2-wire UART for booting
 - **Note:** The 2-wire UART booting only supports DA14535.
 - Open/Close the BLE protocol stack
 - Enable sleep mode
- The following GAP Role support
 - Peripheral: The device that accepts a connection request from Central and establishes a connection
 - Central: The device that initiates a connection with a peripheral
 - All roles: The device may act as a central and peripheral the same time
- GAP functionality
 - Initialize the Host stack
 - Setting address
 - Start/Stop Advertising
 - Connect/Disconnect a link
 - Scan
- GATT Common functionality
 - Get MTU Size
- GATT Server functionality
 - Initialization of GATT Server
 - Loading of Profile definition
 - Notification of characteristics modification
 - Read/Write of GATT Profile from host
 - Optionally the application and not the device may handle the attributes (Read Indication)
- GATT Client functionality
 - Discover Peripheral Services/Characteristics
 - Read/Write Peripheral Characteristics
 - Receive Notifications and Indications

- Security functionality (DA14531/DA14535 module acting as Peripheral)
 - Legacy Pairing supporting Just works functionality
 - Legacy Pairing supporting Passkey functionality
 - Initiate security request procedure from Peripheral as well
 - Secure Connections with Numeric Comparison
 - Support security for both roles (Central – Peripheral)
 - Enable security configuration
 - Enable security from configurator.
 - Store bonding data to flash using data flash management generic module.
 - Delete all bonding data.
 - Delete bonding data with specific BD address.
 - Read and Write BD address to storage.
 - Add crypto libs to encode bonding data in flash.

1.4. API Overview

Table 1.1 lists the API functions included in the SIS module. The required memory sizes are lists in 2.8 Code Size.

Table 1.1 API Functions

Function	Function Description
BLE Common Interface	
R_BLE_Open()	Open the BLE protocol stack.
R_BLE_Close()	Close the BLE protocol stack.
R_BLE_Execute()	Execute the BLE task.
R_BLE_IsTaskFree()	Check if the BLE task queue is free or not.
R_BLE_GetVersion()	Get the BLE module version
BLE GAP Interface	
R_BLE_GAP_Init()	Initialize the Host Stack.
R_BLE_GAP_Terminate()	Terminate the Host Stack.
R_BLE_GAP_UpdConn()	Update the connection parameters.
R_BLE_GAP_SetDataLen()	Update the packet size and the packet transmit time.
R_BLE_GAP_Disconnect()	Disconnect the link.
R_BLE_GAP_GetVerInfo()	Get the version number of the Controller and the host stack.
R_BLE_GAP_ReadRssi()	Get RSSI.
R_BLE_GAP_ReadChMap()	Get the Channel Map.
R_BLE_GAP_SetAdvParam()	Set advertising parameters.
R_BLE_GAP_SetAdvSresData()	Set advertising data/scan response data/periodic advertising data.
R_BLE_GAP_StartAdv()	Start advertising.
R_BLE_GAP_StopAdv()	Stop advertising.
R_BLE_GAP_GetRemainAdvBufSize()	Get buffer size for advertising data/scan response data/periodic advertising data in the Controller.
R_BLE_GAP_GetRemDevInfo()	Get the information about remote device.
R_BLE_GAP_SetPairingParams()	Set the parameters using pairing.
R_BLE_GAP_StartPairing()	Start pairing.
R_BLE_GAP_ReplyPairing()	Reply the pairing request from a remote device.
R_BLE_GAP_ReplyPasskeyEntry()	Reply the passkey entry request.
R_BLE_GAP_ReplyNumComp()	Reply the numeric comparison request.
R_BLE_GAP_ReplyExKeyInfoReq()	Distribute the keys of local device.
R_BLE_GAP_ReplyLtkReq()	Reply the LTK request from a remote device.
R_BLE_GAP_StartScan()	Set scan parameter and start scan.
R_BLE_GAP_StopScan()	Stop scan.
R_BLE_GAP_CreateConn()	Request for a link establishment.

R_BLE_GAP_CancelCreateconn()	Cancel the request for a link establishment.
R_BLE_GAP_ConfWhiteList()	Set White List.
R_BLE_GAP_StartEnc()	Encryption the link
BLE GATT Common Interface	
R_BLE_GATT_GetMtu()	Gets the current MTU used in GATT communication.
BLE GATT Server Interface	
R_BLE_GATTS_SetDbInst()	Sets GATT Database to host stack.
R_BLE_GATTS_RegisterCb()	Registers a callback for GATT Server event.
R_BLE_GATTS_DeregisterCb()	Deregisters the callback function for GATT Server event.
R_BLE_GATTS_Notification()	Sends a notification of an attribute's value.
R_BLE_GATTS_Indication()	Sends an indication of an attribute's value.
R_BLE_GATTS_GetAttr()	Gets an attribute value from the GATT Database.
R_BLE_GATTS_SetAttr()	Sets an attribute value to the GATT Database.
BLE GATT Client Interface	
R_BLE_GATTC_RegisterCb()	Registers a callback function for GATT Client event.
R_BLE_GATTC_DeregisterCb()	Deregisters the callback function for GATT Client event.
R_BLE_GATTC_ReqExMtu()	Sends a MTU Exchange Request PDU to a GATT Server in order to change the current MTU.
R_BLE_GATTC_DiscAllPrimServ()	Discovers all Primary Services in a GATT Server.
R_BLE_GATTC_DiscPrimServ()	Discovers Primary Service specified by p_uuid in a GATT Server.
R_BLE_GATTC_DiscIncServ()	Discovers Included Services within the specified attribute handle range in a GATT Server.
R_BLE_GATTC_DiscAllChar()	Discovers Characteristic within the specified attribute handle range in a GATT Server.
R_BLE_GATTC_DiscCharByUuid()	Discovers Characteristic specified by uuid within the specified attribute handle range in a GATT Server.
R_BLE_GATTC_DiscAllCharDesc()	Discovers Characteristic Descriptor within the specified attribute handle range in a GATT Server.
R_BLE_GATTC_ReadChar()	Reads a Characteristic/Characteristic Descriptor in a GATT Server.
R_BLE_GATTC_ReadCharUsingUuid()	Reads a Characteristic in a GATT Server using a specified UUID.
R_BLE_GATTC_ReadLongChar()	Reads a Long Characteristic in a GATT Server.
R_BLE_GATTC_ReadMultiChar()	Reads multiple Characteristics in a GATT Server.
R_BLE_GATTC_WriteCharWithoutRsp()	Writes a Characteristic in a GATT Server without response.
R_BLE_GATTC_SignedWriteChar()	Writes Signed Data to a Characteristic in a GATT Server without response.

R_BLE_GATTC_WriteChar()	Writes a Characteristic in a GATT Server.
R_BLE_GATTC_WriteLongChar()	Writes a Long Characteristic in a GATT Server.
R_BLE_GATTC_ReliableWrites()	Performs the Reliable Writes procedure described in GATT Specification.
R_BLE_GATTC_ExecWrite()	Executes a write to Characteristic.
BLE Vendor Specific (VS) Interface	
R_BLE_VS_Init()	Initializes Vendor Specific API and registers a callback function for Vendor Specific Event.
R_BLE_VS_SetTxPower()	Configures transmit power.
R_BLE_VS_GetTxPower()	Gets transmit power.
R_BLE_VS_SetBdAddr()	Sets public/random address of local device to the area specified by the parameter.
R_BLE_VS_GetBdAddr()	Gets currently configured public/random address.
R_BLE_VS_GetRand()	Generates 4-16 bytes of random number used in creating keys.
Abstraction API for Renesas QE for BLE	
RM_BLE_ABS_Open()	Host stack is initialized with this function.
RM_BLE_ABS_Close()	Close the BLE channel.
RM_BLE_ABS_StartLegacyAdvertising()	Start Legacy Advertising after setting advertising parameters, advertising data and scan response data.
RM_BLE_ABS_DeleteBondInformation()	Delete bonding information from BLE stack and storage.
RM_BLE_ABS_StartScanning()	Start scanning after setting scan parameters.
RM_BLE_ABS_CreateConnection()	Request create connection.

1.5. Status Transitions

Figure 1.1 shows the status transitions of the SIS module up to communication status.

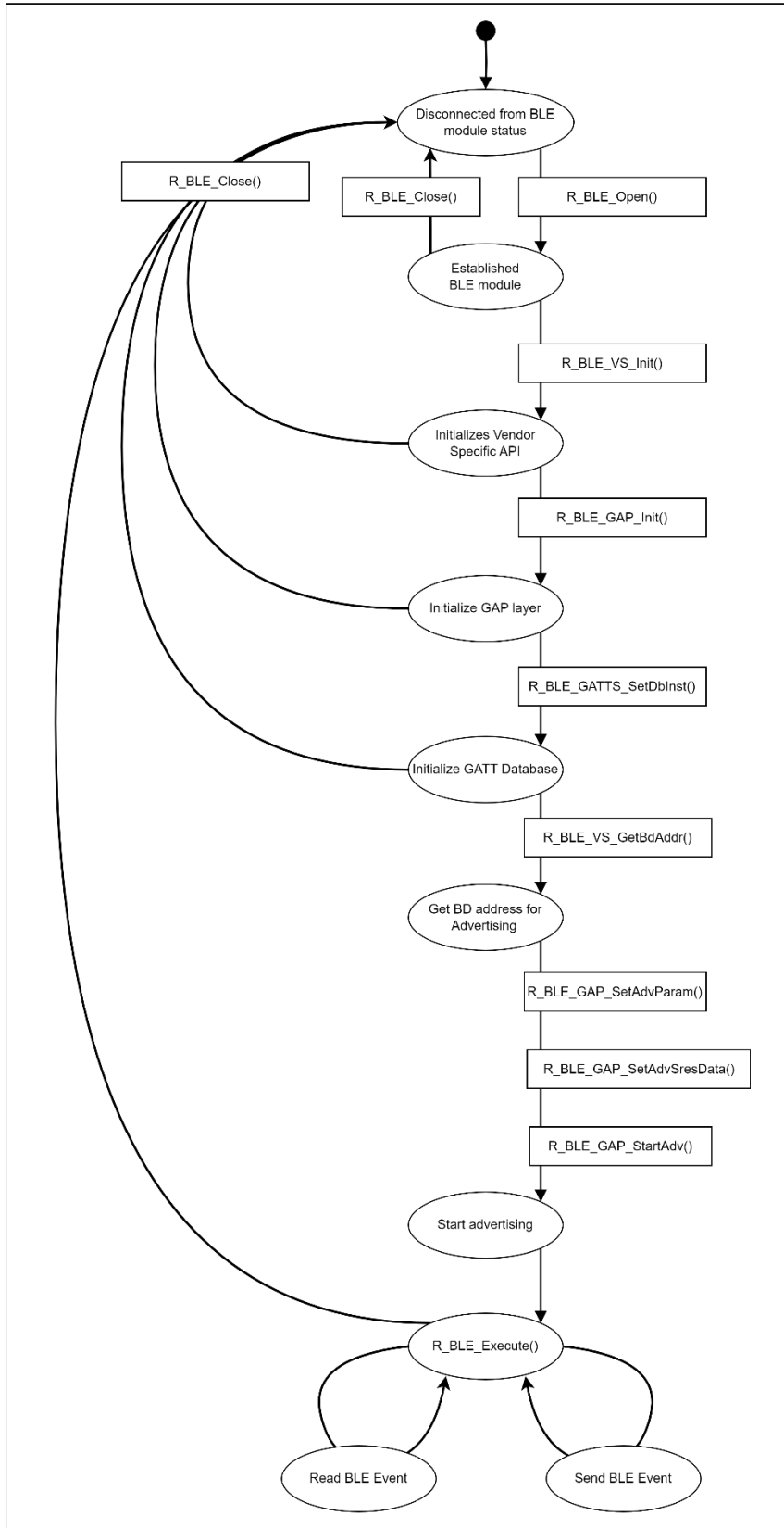


Figure 1.1 Status transitions

2. API Information

The SIS 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 SIS module:

- Board support package (r_bsp)
- UART module (Config_UART)
- PORT module (Config_PORT)
- FreeRTOS
- Flash driver [RL78 Type01 Data flash]
- Flash driver [RL78 Type01 Flash Common]

2.3. Support Toolchain

The SIS module has been confirmed to work with the toolchain listed in 6.1 Confirmed Operation Environment

2.4. Interrupt Vector

The BLE module has some interrupt vectors which overwrite default interrupt vectors of UART module using for communicating with MCU.

Check it in 6.2 How to change UART module to work with BLE module.

2.5. Header Files

All API calls and their supporting interface definitions are in r_ble_da1453x_if.h.

2.6. Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.7. Compile Settings

The configuration option settings of the SIS module are contained in `r_ble_da1453x_config.h`. The names of the options and their setting values are listed in the table below.

Table 2.1 Configuration Options (`r_ble_da1453x_config.h`)

Configuration Options in <code>r_ble_da1453x_config.h</code>	
BLE_CFG_PARAM_CHECKING_ENABLE Note: The default is System Default	Parameter checking.
BLE_CFG_SCI_CHANNEL Note: The default is 6	SCI channel for DA1453x GTL command communication.
BLE_CFG_SCI_INTERRUPT_LEVEL Note: The default is 3	Interrupt Level for BLE_CFG_SCI_CHANNEL.
BLE_CFG_RESET_PORT Note: The default is 5	General-purpose port PDR register connected to the DA1453x reset port.
BLE_CFG_RESET_PIN Note: The default is 5	General-purpose port PODR register connected to the DA1453x reset pin.
BLE_CFG_SCK_PORT Note: The default is 0	General-purpose port PDR register connected to the DA1453x SCK port.
BLE_CFG_SCK_PIN Note: The default is 2	General-purpose port PODR register connected to the DA1453x SCK pin.
BLE_CFG_RESET_POLARITY Note: The default is 0	Reset Polarity.
BLE_CFG_HOST_BOOT_MODE Note: The default is 0.	Boot SDK download from host MCU. When using this feature via 1-Wire UART or 2-Wire UART, please refer to <i>2.13.5 SIS Module Limitations</i>
BLE_CFG_IO_CAPABILITY Note: The default is BLE_GAP_IOCAP_NOINPUT_NOOUTPUT	IO capabilities of local device
BLE_CFG_SECURITY_KEY_DIST Note: The default is BLE_GAP_KEY_DIST_ENCKEY	Type of keys to be distributed from local device
BLE_CFG_SECURITY_KEY_DIST_PEER Note: The default is BLE_GAP_KEY_DIST_IDKEY	Types of keys which local device distributes to remote device
BLE_CFG_MITM_PROTECTION Note: The default is BLE_GAP_SEC_MITM_BEST Effort	MITM protection policy
BLE_CFG_SECURE_CONNECTIONS Note: The default is BLE_GAP_SC_BEST Effort	Determine whether to accept only Secure connections or not
BLE_CFG_MAX_LTK_SIZE Note: The default is 16	Maximum LTK size
BLE_CFG_DA1453X_DEVICE Note: The default is DA14531_DEVICE	Select PMOD device: Either DA14531PMOD or DA14535PMOD.
BLE_CFG_DEVICE_ROLE Note: The default is Peripheral	Select BLE role for the DA1453x.
BLE_ABS_CFG_ENABLE_SECURE_DATA Note: The default is 0.	Enables or disables the secure data storage feature for bonding information.
BLE_ABS_CFG_SECURE_DATA_DATAFLASH_BLOCK Note: The default is 0.	Specifies the data flash block number to be used.
BLE_ABS_CFG_NUMBER_BONDING Note: The default is 7.	Defines the maximum number of devices that can be bonded
BLE_DA1453x_SLEEP_FLOW_CONTROL Note: The default is 0.	Enables or disables the sleep flow control feature for DA1453x platform.
CRYPTO_ENABLED Note: The default is 0.	Enables or disables cryptographic operations for the DA1453X platform.

Table 2.2 Configuration Options (r_sci_rl_config.h)

Configuration Options in r_sci_rl_config.h	
#define SCI_CFG_CHx_INCLUDED Notes: 1. CHx = CH0 to CH3 2. The default values are as follows CH3: 1, CH0 to CH2: 0	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.
#define SCI_CFG_CHx_TX_BUFSIZ Notes: 1. CHx = CH0 to CH3 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 set to 2048.
#define SCI_CFG_CHx_RX_BUFSIZ Notes: 1. CHx = CH0 to CH3 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 set to 2048.
#define SCI_CFG_TEI_INCLUDED Note: The default is 0.	Enables the transmit end interrupt for serial transmissions. This option should be set to 1.

Table 2.3 Configuration Options (r_bsp_config.h)

Configuration Options in r_bsp_config.h	
#define BSP_CFG_RTOS_USED Note: The default is 0.	Specifies the type of real-time OS. When using this SIS module, set the following. FreeRTOS:1, Bare Metal: 0

2.8. Code Size

Typical code sizes associated with this module are listed below.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in 2.7 Compile Settings. The table lists reference values when the C compiler's compile options are set to their default values, as described in 2.3 Support Toolchain. The compile option default values are optimization level: Code Size Precedence (-Osize), and data endianness: little-endian. The code size varies depending on the C compiler version and compile options.

The values in the table below are confirmed under the following conditions.

Module Revision: r_ble_da1453x rev1.70.

Compiler Version: Renesas Electronics C Compiler Package for RL78 Family V1.15.01

Configuration Options: Default settings.

Table 2.1 Memory Sizes

Device	RTOS	Category	Memory usage
			Renesas Compiler
FPB RL78/G23 128p	FreeRTOS	ROM	57013 bytes
		RAM	8909 bytes
	Bare metal	ROM	56145 bytes
		RAM	9074 bytes

* **Note:** ROM usage included 15KB (15388 bytes) of DA14531 Boot image and qe_gen folder.

2.9. Return values

The error codes returned by API functions are listed below. The enumerated types of return values and API function declarations are contained in `r_ble_api.h`.

```
typedef uint16_t ble_status_t;

enum RBLE_STATUS_enum
{
    BLE_SUCCESS = 0x0000,

    /* common error code */
    BLE_ERR_INVALID_PTR          = 0x0001,
    BLE_ERR_INVALID_DATA        = 0x0002,
    BLE_ERR_INVALID_ARG         = 0x0003,
    BLE_ERR_INVALID_FUNC        = 0x0004,
    BLE_ERR_INVALID_CHAN        = 0x0005,
    BLE_ERR_INVALID_MODE        = 0x0006,
    BLE_ERR_UNSUPPORTED         = 0x0007,
    BLE_ERR_INVALID_STATE       = 0x0008,
    BLE_ERR_INVALID_OPERATION   = 0x0009,
    BLE_ERR_ALREADY_IN_PROGRESS = 0x000A,
    BLE_ERR_CONTEXT_FULL        = 0x000B,
    BLE_ERR_MEM_ALLOC_FAILED    = 0x000C,
    BLE_ERR_NOT_FOUND           = 0x000D,
    BLE_ERR_INVALID_HDL         = 0x000E,
    BLE_ERR_DISCONNECTED        = 0x000F,
    BLE_ERR_LIMIT_EXCEEDED     = 0x0010,
    BLE_ERR_RSP_TIMEOUT         = 0x0011,
    BLE_ERR_NOT_YET_READY       = 0x0012,
    BLE_ERR_UNSPECIFIED         = 0x0013,
    BLE_ERR_ALREADY_INITIALIZED = 0x0014,

    /* HCI Spec Error */
    BLE_ERR_HC_UNKNOWN_HCI_CMD  = 0x1001,
    BLE_ERR_HC_NO_CONN          = 0x1002,
    BLE_ERR_HC_HW_FAIL          = 0x1003,
    BLE_ERR_HC_PAGE_TO          = 0x1004,
    BLE_ERR_HC_AUTH_FAIL        = 0x1005,
    BLE_ERR_HC_KEY_MISSING      = 0x1006,
    BLE_ERR_HC_MEM_FULL         = 0x1007,
    BLE_ERR_HC_CONN_TO          = 0x1008,
    BLE_ERR_HC_MAX_NUM_OF_CONN  = 0x1009,
    BLE_ERR_HC_MAX_NUM_OF_SCO_CONN = 0x100A,
    BLE_ERR_HC_ACL_CONN_ALREADY_EXISTS = 0x100B,
    BLE_ERR_HC_CMD_DISALLOWED   = 0x100C,
    BLE_ERR_HC_HOST_REJ_LIMITED_RESRC = 0x100D,
    BLE_ERR_HC_HOST_REJ_SEC_REASONS = 0x100E,
    BLE_ERR_HC_HOST_REJ_PERSONAL_DEV = 0x100F,
    BLE_ERR_HC_HOST_TO          = 0x1010,
    BLE_ERR_HC_UNSPRT_FEAT_OR_PARAM = 0x1011,
    BLE_ERR_HC_INVALID_HCI_CMD_PARAM = 0x1012,
    BLE_ERR_HC_OTHER_END_TERM_USER = 0x1013,
    BLE_ERR_HC_OTHER_END_TERM_LOW_RESRC = 0x1014,
    BLE_ERR_HC_OTHER_END_TERM_PW_OFF = 0x1015,
    BLE_ERR_HC_CONN_TERM_BY_LOCAL_HOST = 0x1016,
    BLE_ERR_HC_REPEATED_ATTEMPTS = 0x1017,
    BLE_ERR_HC_PAIRING_NOT_ALLOWED = 0x1018,
    BLE_ERR_HC_UNKNOWN_LMP_PDU  = 0x1019,
    BLE_ERR_HC_UNSPRT_REM_FEAT   = 0x101A,
    BLE_ERR_HC_SCO_OFFSET_REJ    = 0x101B,
```

```

BLE_ERR_HC_SCO_INTERVAL_REJ           = 0x101C,
BLE_ERR_HC_SCO_AIR_MODE_REJ           = 0x101D,
BLE_ERR_HC_INVALID_LMP_PARAM          = 0x101E,
BLE_ERR_HC_UNSPECIFIED_ERR            = 0x101F,
BLE_ERR_HC_UNSPRT_LMP_PARAM_VAL       = 0x1020,
BLE_ERR_HC_ROLE_CHANGE_NOT_ALLOWED    = 0x1021,
BLE_ERR_HC_LMP_RSP_TO                  = 0x1022,
BLE_ERR_HC_LMP_ERR_TX_COLLISION       = 0x1023,
BLE_ERR_HC_LMP_PDU_NOT_ALLOWED        = 0x1024,
BLE_ERR_HC_ENC_MODE_NOT_ACCEPTABLE    = 0x1025,
BLE_ERR_HC_UNIT_KEY_USED               = 0x1026,
BLE_ERR_HC_QOS_IS_NOT_SPRT            = 0x1027,
BLE_ERR_HC_INSTANT_PASSED              = 0x1028,
BLE_ERR_HC_PAIRING_UNIT_KEY_NOT_SPRT  = 0x1029,
BLE_ERR_HC_DIFF_TRANSACTION_COLLISION = 0x102A,
BLE_ERR_HC_QOS_UNACCEPTABLE_PARAM     = 0x102C,
BLE_ERR_HC_QOS_REJ                     = 0x102D,
BLE_ERR_HC_CH_CLASSIFICATION_NOT_SPRT = 0x102E,
BLE_ERR_HC_INSUFFICIENT_SEC           = 0x102F,
BLE_ERR_HC_PARAM_OUT_OF_MANDATORY_RANGE = 0x1030,
BLE_ERR_HC_ROLE_SWITCH_PENDING        = 0x1032,
BLE_ERR_HC_RESERVED_SLOT_VIOLATION    = 0x1034,
BLE_ERR_HC_ROLE_SWITCH_FAIL           = 0x1035,
BLE_ERR_HC_EXT_INQUIRY_RSP_TOO_LARGE  = 0x1036,
BLE_ERR_HC_SSP_NOT_SPRT_BY_HOST       = 0x1037,
BLE_ERR_HC_HOST_BUSY_PAIRING          = 0x1038,
BLE_ERR_HC_CONN_REJ_NO_SUIT_CH_FOUND  = 0x1039,
BLE_ERR_HC_CTRL_BUSY                  = 0x103A,
BLE_ERR_HC_UNACCEPTABLE_CONN_INTERVAL = 0x103B,
BLE_ERR_HC_ADV_TO                      = 0x103C,
BLE_ERR_HC_CONN_TREM_DUE_TO_MIC_FAIL   = 0x103D,
BLE_ERR_HC_CONN_FAIL_TO_BE_EST        = 0x103E,
BLE_ERR_HC_MAC_CONN_FAIL               = 0x103F,
BLE_ERR_HC_COARSE_CLK_ADJUST_REJ      = 0x1040,
BLE_ERR_HC_TYPE0_SUBMAP_NOT_DEFINED    = 0x1041,
BLE_ERR_HC_UNKNOWN_ADV_ID              = 0x1042,
BLE_ERR_HC_LIMIT_REACHED               = 0x1043,
BLE_ERR_HC_OP_CANCELLED_BY_HOST        = 0x1044,

/* SMP Spec Error */
BLE_ERR_SMP_LE_PASSKEY_ENTRY_FAIL      = 0x2001,
BLE_ERR_SMP_LE_OOB_DATA_NOT_AVAILABLE = 0x2002,
BLE_ERR_SMP_LE_AUTH_REQ_NOT_MET        = 0x2003,
BLE_ERR_SMP_LE_CONFIRM_VAL_NOT_MATCH   = 0x2004,
BLE_ERR_SMP_LE_PAIRING_NOT_SPRT        = 0x2005,
BLE_ERR_SMP_LE_INSUFFICIENT_ENC_KEY_SIZE = 0x2006,
BLE_ERR_SMP_LE_CMD_NOT_SPRT            = 0x2007,
BLE_ERR_SMP_LE_UNSPECIFIED_REASON      = 0x2008,
BLE_ERR_SMP_LE_REPEATED_ATTEMPTS       = 0x2009,
BLE_ERR_SMP_LE_INVALID_PARAM           = 0x200A,
BLE_ERR_SMP_LE_DHKEY_CHECK_FAIL         = 0x200B,
BLE_ERR_SMP_LE_NUM_COMP_FAIL           = 0x200C,
BLE_ERR_SMP_LE_BREDR_PAIRING_IN_PROGRESS = 0x200D,
BLE_ERR_SMP_LE_CT_KEY_GEN_NOT_ALLOWED  = 0x200E,
BLE_ERR_SMP_LE_DISCONNECTED            = 0x200F,
BLE_ERR_SMP_LE_TO                       = 0x2011,
BLE_ERR_SMP_LE_LOC_KEY_MISSING          = 0x2014,

/* GATT Spec Error */
BLE_ERR_GATT_INVALID_HANDLE            = 0x3001,
BLE_ERR_GATT_READ_NOT_PERMITTED        = 0x3002,

```

```
BLE_ERR_GATT_WRITE_NOT_PERMITTED           = 0x3003,
BLE_ERR_GATT_INVALID_PDU                   = 0x3004,
BLE_ERR_GATT_INSUFFICIENT_AUTHENTICATION   = 0x3005,
BLE_ERR_GATT_REQUEST_NOT_SUPPORTED        = 0x3006,
BLE_ERR_GATT_INVALID_OFFSET               = 0x3007,
BLE_ERR_GATT_INSUFFICIENT_AUTHORIZATION    = 0x3008,
BLE_ERR_GATT_PREPARE_WRITE_QUEUE_FULL     = 0x3009,
BLE_ERR_GATT_ATTRIBUTE_NOT_FOUND          = 0x300A,
BLE_ERR_GATT_ATTRIBUTE_NOT_LONG           = 0x300B,
BLE_ERR_GATT_INSUFFICIENT_ENC_KEY_SIZE    = 0x300C,
BLE_ERR_GATT_INVALID_ATTRIBUTE_LEN        = 0x300D,
BLE_ERR_GATT_UNLIKELY_ERROR               = 0x300E,
BLE_ERR_GATT_INSUFFICIENT_ENCRYPTION      = 0x300F,
BLE_ERR_GATT_UNSUPPORTED_GROUP_TYPE       = 0x3010,
BLE_ERR_GATT_INSUFFICIENT_RESOURCES       = 0x3011,

/* defined in CSS */
BLE_ERR_GATT_WRITE_REQ_REJECTED           = 0x30FC,
BLE_ERR_GATT_CCCD_IMPROPERLY_CFG         = 0x30FD,
BLE_ERR_GATT_PROC_ALREADY_IN_PROGRESS    = 0x30FE,
BLE_ERR_GATT_OUT_OF_RANGE                 = 0x30FF,

/* L2CAP Spec Error */
BLE_ERR_L2CAP_PSM_NOT_SUPPORTED           = 0x4002,
BLE_ERR_L2CAP_NO_RESOURCE                 = 0x4004,
BLE_ERR_L2CAP_INSUF_AUTHEN               = 0x4005,
BLE_ERR_L2CAP_INSUF_AUTHOR                = 0x4006,
BLE_ERR_L2CAP_INSUF_ENC_KEY_SIZE         = 0x4007,
BLE_ERR_L2CAP_REFUSE_INSUF_ENC           = 0x4008,
BLE_ERR_L2CAP_REFUSE_INVALID_SCID        = 0x4009,
BLE_ERR_L2CAP_REFUSE_SCID_ALREADY_ALLOC  = 0x400A,
BLE_ERR_L2CAP_REFUSE_UNACCEPTABLE_PARAM  = 0x400B,
};
```

2.10. Parameter

```

#define BLE_CFG_DEVICE_ROLE                (0x0A)
#define BLE_EVENT_PATTERN                  0x0A0A

/* The reset pulse duration in milliseconds */
#define R_BLE_GTL_RESET_PULSE_MS          200
#define R_BLE_GTL_ADV_DATA_LEN            0x1F

/* Internal event bit definitions*/
#define R_BLE_GTL_ADV_TIMER_EXPIRED_EVENT  0x00000001UL

/* Maximum length of device name in bytes (as defined by Bluetooth Core v4.2 /
GAP) */
#define R_BLE_GTL_GAP_DEV_NAME_LEN_MAX    0xF8

#define R_BLE_GTL_MSG_LEN_MAX              528
#define R_BLE_GTL_MSG_HDR_LEN              8
#define R_BLE_GTL_MSG_INITIATOR            0x05

#define R_BLE_GTL_MSG_TX_TIMEOUT_MS        5000
#define R_BLE_GTL_MSG_TX_BLOCK_TIMEOUT_MS  5000
#define R_BLE_GTL_MSG_RSP_TIMEOUT_MS       5000
#define R_BLE_GTL_MSG_CMD_CMP_TIMEOUT_MS   5000

#define R_BLE_GTL_CB_EVT_QUEUE_LEN         4
#define R_BLE_GTL_MSG_QUEUE_LEN           4

/* Application callback event types */
#define R_BLE_GTL_CB_EVT_TYPE_MASK         0xF000U
#define R_BLE_GTL_CB_EVT_TYPE_GAP         0x1000U
#define R_BLE_GTL_CB_EVT_TYPE_GATTS       0x3000U
#define R_BLE_GTL_CB_EVT_TYPE_GATTC       0x4000U
#define R_BLE_GTL_CB_EVT_TYPE_L2CAP       0x5000U
#define R_BLE_GTL_CB_EVT_TYPE_VS         0x8000U

/* GTL Task ID's */
#define R_BLE_GTL_TASK_ID_GATTM           0x000B
#define R_BLE_GTL_TASK_ID_GATTC           0x000C
#define R_BLE_GTL_TASK_ID_GAPM            0x000D
#define R_BLE_GTL_TASK_ID_GAPC            0x000E
#define R_BLE_GTL_TASK_ID_GTL             0x0010
#define R_BLE_GTL_TASK_ID_AUX             0x00A0

/* GTL GATTM Command ID's */
#define R_BLE_GTL_GATTM_ADD_SVC_REQ        0x0B00
#define R_BLE_GTL_GATTM_ADD_SVC_RSP        0x0B01
#define R_BLE_GTL_GATTM_ATT_GET_VALUE_REQ  0x0B0A
#define R_BLE_GTL_GATTM_ATT_GET_VALUE_RSP  0x0B0B
#define R_BLE_GTL_GATTM_ATT_SET_VALUE_REQ  0x0B0C
#define R_BLE_GTL_GATTM_ATT_SET_VALUE_RSP  0x0B0D

/* GTL GATTC Command ID's */
#define R_BLE_GTL_GATTC_CMP_EVT            0x0C00
#define R_BLE_GTL_GATTC_EXC_MTU_CMD        0x0C01
#define R_BLE_GTL_GATTC_MTU_CHANGED_IND    0x0C02
#define R_BLE_GTL_GATTC_DISC_CMD           0x0C03
#define R_BLE_GTL_GATTC_DISC_SVC_IND       0x0C04
#define R_BLE_GTL_GATTC_DISC_CHAR_IND      0x0C06
#define R_BLE_GTL_GATTC_DISC_CHAR_DESC_IND 0x0C07
#define R_BLE_GTL_GATTC_READ_CMD           0x0C08

```

```
#define R_BLE_GTL_GATTC_READ_IND          0x0C09
#define R_BLE_GTL_GATTC_SEND_EVT_CMD     0x0C10
#define R_BLE_GTL_GATTC_WRITE_CMD        0x0C0A
#define R_BLE_GTL_GATTC_WRITE_EXECUTE_CMD 0x0C0B
#define R_BLE_GTL_GATTC_EVENT_IND        0x0C0C
#define R_BLE_GTL_GATTC_READ_REQ_IND     0x0C13
#define R_BLE_GTL_GATTC_READ_CFM        0x0C14
#define R_BLE_GTL_GATTC_WRITE_REQ_IND    0x0C15
#define R_BLE_GTL_GATTC_WRITE_CFM       0x0C16
#define R_BLE_GTL_GATTC_ATT_INFO_REQ_IND 0x0C17
#define R_BLE_GTL_GATTC_ATT_INFO_CFM     0x0C18

/* GTL GAPM Command ID's */
#define R_BLE_GTL_GAPM_CMP_EVT           0x0D00
#define R_BLE_GTL_GAPM_DEVICE_READY_IND  0x0D01
#define R_BLE_GTL_GAPM_RESET_CMD         0x0D02
#define R_BLE_GTL_GAPM_CANCEL_CMD        0x0D03
#define R_BLE_GTL_GAPM_SET_DEV_CONFIG_CMD 0x0D04
#define R_BLE_GTL_GAPM_GET_DEV_INFO_CMD  0x0D06
#define R_BLE_GTL_GAPM_DEV_VERSION_IND   0x0D07
#define R_BLE_GTL_GAPM_DEV_BDADDR_IND    0x0D08
#define R_BLE_GTL_GAPM_WHITE_LIST_MGT_CMD 0x0D0B
#define R_BLE_GTL_GAPM_GEN_RAND_ADDR_CMD  0x0D16
#define R_BLE_GTL_GAPM_GEN_RAND_NB_CMD   0x0D19
#define R_BLE_GTL_GAPM_GEN_RAND_NB_IND   0x0D1A
#define R_BLE_GTL_GAPM_UNKNOWN_TASK_IND  0x0D1D
#define R_BLE_GTL_GAPM_START_ADVERTISE_CMD 0x0D0D
#define R_BLE_GTL_GAPM_START_SCAN_CMD    0x0D0F
#define R_BLE_GTL_GAPM_ADV_REPORT_IND     0x0D10
#define R_BLE_GTL_GAPM_START_CONNECTION_CMD 0x0D11
#define R_BLE_GTL_GAPM_RESOLV_ADDR_CMD   0x0D14
#define R_BLE_GTL_GAPM_ADDR_SOLVED_IND   0x0D15

/* GTL GAPC Command ID's */
#define R_BLE_GTL_GAPC_CMP_EVT           0x0E00
#define R_BLE_GTL_GAPC_CONNECTION_REQ_IND 0x0E01
#define R_BLE_GTL_GAPC_CONNECTION_CFM    0x0E02
#define R_BLE_GTL_GAPC_DISCONNECT_IND     0x0E03
#define R_BLE_GTL_GAPC_DISCONNECT_CMD     0x0E04
#define R_BLE_GTL_GAPC_GET_INFO_CMD       0x0E05
#define R_BLE_GTL_GAPC_PEER_VERSION_IND   0x0E07
#define R_BLE_GTL_GAPC_PEER_FEATURES_IND  0x0E08
#define R_BLE_GTL_GAPC_CON_RSSI_IND       0x0E09
#define R_BLE_GTL_GAPC_GET_DEV_INFO_REQ_IND 0x0E0A
#define R_BLE_GTL_GAPC_GET_DEV_INFO_CFM  0x0E0B
#define R_BLE_GTL_GAPC_PARAM_UPDATE_CMD   0x0E0E
#define R_BLE_GTL_GAPC_PARAM_UPDATE_REQ_IND 0x0E0F
#define R_BLE_GTL_GAPC_PARAM_UPDATE_CFM   0x0E10
#define R_BLE_GTL_GAPC_PARAM_UPDATED_IND  0x0E11

#define R_BLE_GTL_GAPC_BOND_CMD           0x0E12
#define R_BLE_GTL_GAPC_BOND_REQ_IND       0x0E13
#define R_BLE_GTL_GAPC_BOND_CFM          0x0E14
#define R_BLE_GTL_GAPC_BOND_IND          0x0E15

#define R_BLE_GTL_GAPC_ENCRYPT_CMD         0x0E16
#define R_BLE_GTL_GAPC_ENCRYPT_REQ_IND     0x0E17
#define R_BLE_GTL_GAPC_ENCRYPT_CFM        0x0E18
#define R_BLE_GTL_GAPC_ENCRYPT_IND        0x0E19

#define R_BLE_GTL_GAPC_SECURITY_CMD       0x0E1A
```

```
#define R_BLE_GTL_GAPC_SECURITY_IND 0x0E1B

#define R_BLE_GTL_GAPC_CON_CHANNEL_MAP_IND 0x0E1D
#define R_BLE_GTL_GAPC_LECB_CONNECT_CMD 0x0E20
#define R_BLE_GTL_GAPC_LECB_ADD_CMD 0x0E24
#define R_BLE_GTL_GAPC_LECB_SEND_CMD 0x0E25
#define R_BLE_GTL_GAPC_LECB_DISCONNECT_CMD 0x0E26
#define R_BLE_GTL_GAPC_SET_LE_PKT_SIZE_CMD 0x0E2B
#define R_BLE_GTL_GAPC_LE_PKT_SIZE_IND 0x0E2C

/* GTL Auxiliary Command ID's */
#define R_BLE_GTL_GAPC_APP_RAND_REQ 0xA001
#define R_BLE_GTL_APP_GEN_RAND_RSP 0xA002
#define R_BLE_GTL_AUX_SET_TX_POWER_CMD 0xA005
#define R_BLE_GTL_AUX_SET_TX_POWER_CMP_EVT 0xA006
#define R_BLE_GTL_AUX_GET_TX_POWER_CMD 0xA007
#define R_BLE_GTL_AUX_GET_TX_POWER_RSP 0xA008
#define R_BLE_GTL_AUX_SET_SLEEP_MODE_CMD 0xA009
#define R_BLE_GTL_AUX_GET_SLEEP_MODE_CMD 0xA00A
#define R_BLE_GTL_AUX_SET_SLEEP_MODE_CFM 0xA00B
#define R_BLE_GTL_AUX_SET_BD_ADDRESS_CMD 0xA00C
#define R_BLE_GTL_AUX_SET_BD_ADDRESS_CFM 0xA00D
#define R_BLE_GTL_AUX_GET_ENC_RAND_REQ 0xA00E
#define R_BLE_GTL_AUX_GET_ENC_RAND_RSP 0xA00F

#define R_BLE_GTL_PERIPHERAL_ROLE 0x0A
#define R_BLE_GTL_GAP_ROLE_CENTRAL 0x05
#define R_BLE_GTL_ADV_FLAG_FIELD_LEN 3
#define R_BLE_GTL_ADV_DATA_LEN_MAX 31
#define R_BLE_GTL_ADV_DATA_TYPE_FLAGS 0x01
#define R_BLE_GTL_SCAN_RSP_DATA_LEN_MAX 31
#define R_BLE_GTL_KEY_LEN 0x10
#define R_BLE_GTL_GET_RAND_SIZE_MAX 8
#define R_BLE_GTL_DATA_LEN_TX_OCTETS_MAX 251
#define R_BLE_GTL_DATA_LEN_TX_TIME_MAX 2120
#define R_BLE_GTL_GAP_NON_DISCOVERABLE 0x00
#define R_BLE_GTL_GAP_GEN_DISCOVERABLE 0x01
#define R_BLE_GTL_GAP_LIM_DISCOVERABLE 0x02
#define R_BLE_GTL_GAP_BROADCASTER_MODE 0x03

/* Attribute permissions defined in QE profile */
#define R_BLE_GTL_QE_ATT_PERM_READ 0x01
#define R_BLE_GTL_QE_ATT_PERM_WRITE 0x02

/* Attribute properties defined in QE profile */
#define R_BLE_GTL_QE_ATT_PROP_READ 0x02
#define R_BLE_GTL_QE_ATT_PROP_WRITE 0x08
#define R_BLE_GTL_QE_ATT_PROP_WRITE_NO_RESPONSE 0x04
#define R_BLE_GTL_QE_ATT_PROP_NOTIFY 0x10
#define R_BLE_GTL_QE_ATT_PROP_INDICATE 0x20

/* Attribute permissions defined in GTL message(s) */
#define R_BLE_GTL_ATT_PERM_READ_ENABLE 0x00000001UL
#define R_BLE_GTL_ATT_PERM_WRITE_ENABLE 0x00000008UL
#define R_BLE_GTL_ATT_PERM_INDICATE_ENABLE 0x00000040UL
#define R_BLE_GTL_ATT_PERM_NOIFY_ENABLE 0x00000200UL
#define R_BLE_GTL_ATT_PERM_WRITE_CMD_ACCEPTED 0x00008000UL
#define R_BLE_GTL_ATT_PERM_WRITE_REQ_ACCEPTED 0x00020000UL
#define R_BLE_GTL_ATT_PERM_UUID_LEN_128 0x00080000UL

#define R_BLE_GTL_SVC_GAP_UUID 0x1800
```

```
#define R_BLE_GTL_SVC_GATT_UUID 0x1801
#define R_BLE_GTL_ATT_PRIMARY_SVC_DECL 0x2800
#define R_BLE_GTL_ATT_SECONDARY_SVC_DECL 0x2801
#define R_BLE_GTL_CHAR_DECLARATION 0x2803
#define R_BLE_GTL_CHAR_USER_DESC 0x2901
#define R_BLE_GTL_CLIENT_CHAR_CONF 0x2902
#define R_BLE_GTL_SERVER_CHAR_CONF 0x2903
#define R_BLE_GTL_CHAR_DEVICE_NAME 0x2A00
#define R_BLE_GTL_CHAR_APPEARANCE 0x2A01

/* Service permissions defined in GTL messages(s), can be or'd together */
#define R_BLE_GTL_SVC_PERM_ENABLE 0x04
#define R_BLE_GTL_SVC_PERM_UUID_LEN_128 0x40
#define R_BLE_GTL_SVC_PERM_PRIMARY 0x80

#define R_BLE_GTL_RI_FLAG 0x8000

/* The first two bits of a non-public (random) address must be binary ones */
#define R_BLE_GTL_PUBLIC_BD_ADDR_MASK 0xC0

#define R_BLE_GTL_MS_PER_SECOND 1000UL
#define R_BLE_GTL_ADV_TIMER_TICKS_PER_SECOND 100UL

/* "RBLE" in ASCII. Used to determine if the control block is open. */
#define R_BLE_GTL_OPEN 0x52424C45U
#define R_BLE_GTL_CLOSED 0x00000000U

/* Mutex give/take defines */
#define R_BLE_GTL_MUTEX_TX (1UL << 0)
#define R_BLE_GTL_MUTEX_RX (1UL << 1)
#define R_BLE_GTL_MUTEX_TEI (1UL << 2)

/* UART boot protocol message types */
#define R_BLE_GTL_BOOT_STX 0x02
#define R_BLE_GTL_BOOT_SOH 0x01
#define R_BLE_GTL_BOOT_ACK 0x06
#define R_BLE_GTL_BOOT_NACK 0x15

/* Defines for host DB */
#define DB_NON_RI_INDEX 0xFFFF
#define DB_INVALID_PARAM 0xFFFE
#define DB_VALID_RI_INDEX 0x0000

/* Defines for scan parameters */
#define R_BLE_GTL_GAPM_STATIC_ADDR 0x00
#define R_BLE_GTL_GAP_GEN_DISCOVERY 0x00
#define R_BLE_GTL_GAP_LIM_DISCOVERY 0x01
#define R_BLE_GTL_GAP_OBSERVER_MODE 0x02

#define R_BLE_GTL_SCAN_ALLOW_ADV_ALL 0x00
#define R_BLE_GTL_SCAN_ALLOW_ADV_WLST 0x01

/* Defines for connection parameters */
#define R_BLE_GTL_MIN_CONNECTION_CE_LENGTH 0x0000
#define R_BLE_GTL_MAX_CONNECTION_CE_LENGTH 0xFFFF

/* Defines for whitelisting */
#define R_BLE_GTL_GAPM_GET_WLIST_SIZE 0x08
#define R_BLE_GTL_GAPM_ADD_DEV_IN_WLIST 0x09
#define R_BLE_GTL_GAPM_RMV_DEV_FRM_WLIST 0x0A
#define R_BLE_GTL_GAPM_CLEAR_WLIST 0x0B
```

```
typedef enum e_r_ble_gtl_rx_msg_parser_state
{
    R_BLE_GTL_RX_MSG_PARSER_STATE_IDLE = 0,
    R_BLE_GTL_RX_MSG_PARSER_STATE_RX_HEADER,
    R_BLE_GTL_RX_MSG_PARSER_STATE_RX_PARAM
} r_ble_gtl_rx_msg_parser_state_t;

typedef enum e_r_ble_gtl_gapm_operation
{
    R_BLE_GTL_GAPM_OP_NONE = 0x00,
    R_BLE_GTL_GAPM_OP_RESET,
    R_BLE_GTL_GAPM_OP_CANCEL,
    R_BLE_GTL_GAPM_OP_SET_DEV_CONFIG,
    R_BLE_GTL_GAPM_OP_SET_CHANNEL_MAP,
    R_BLE_GTL_GAPM_OP_GET_DEV_VERSION,
    R_BLE_GTL_GAPM_OP_GET_DEV_BDADDR,
    R_BLE_GTL_GAPM_OP_GET_DEV_ADV_TX_POWER,
    R_BLE_GTL_GAPM_OP_GET_WLIST_SIZE,
    R_BLE_GTL_GAPM_OP_ADD_DEV_IN_WLIST,
    R_BLE_GTL_GAPM_OP_RMV_DEV_FRM_WLIST,
    R_BLE_GTL_GAPM_OP_CLEAR_WLIST,
    R_BLE_GTL_GAPM_OP_ADV_NON_CONN,
    R_BLE_GTL_GAPM_OP_ADV_UNDIRECT,
    R_BLE_GTL_GAPM_OP_ADV_DIRECT,
    R_BLE_GTL_GAPM_OP_ADV_DIRECT_LDC,
    R_BLE_GTL_GAPM_OP_UPDATE_ADVERTISE_DATA,
    R_BLE_GTL_GAPM_OP_SCAN_ACTIVE,
    R_BLE_GTL_GAPM_OP_SCAN_PASSIVE,
    R_BLE_GTL_GAPM_OP_CONNECTION_DIRECT,
    R_BLE_GTL_GAPM_OP_CONNECTION_AUTO,
    R_BLE_GTL_GAPM_OP_CONNECTION_SELECTIVE,
    R_BLE_GTL_GAPM_OP_CONNECTION_NAME_REQUEST,
    R_BLE_GTL_GAPM_OP_RESOLV_ADDR,
    R_BLE_GTL_GAPM_OP_GEN_RAND_ADDR,
    R_BLE_GTL_GAPM_OP_USE_ENC_BLOCK,
    R_BLE_GTL_GAPM_OP_GEN_RAND_NB,
    R_BLE_GTL_GAPM_OP_PROFILE_TASK_ADD,
    R_BLE_GTL_GAPM_OP_DBG_GET_MEM_INFO,
    R_BLE_GTL_GAPM_OP_PLF_RESET,
    R_BLE_GTL_GAPM_OP_SET_SUGGESTED_DFLT_LE_DATA_LEN,
    R_BLE_GTL_GAPM_OP_GET_SUGGESTED_DFLT_LE_DATA_LEN,
    R_BLE_GTL_GAPM_OP_GET_MAX_LE_DATA_LEN,
    R_BLE_GTL_GAPM_OP_GET_RAL_SIZE,
    R_BLE_GTL_GAPM_OP_GET_RAL_LOC_ADDR,
    R_BLE_GTL_GAPM_OP_GET_RAL_PEER_ADDR,
    R_BLE_GTL_GAPM_OP_ADD_DEV_IN_RAL,
    R_BLE_GTL_GAPM_OP_RMV_DEV_FRM_RAL,
    R_BLE_GTL_GAPM_OP_CLEAR_RAL,
    R_BLE_GTL_GAPM_OP_USE_P256_BLOCK,
    R_BLE_GTL_GAPM_OP_NETWORK_MODE_RAL,
    R_BLE_GTL_GAPM_OP_DEVICE_MODE_RAL,
    R_BLE_GTL_GAPM_OP_KEY_RENEW,
    R_BLE_GTL_GAPM_OP_GEN_P256_KEY = R_BLE_GTL_GAPM_OP_KEY_RENEW,
    R_BLE_GTL_GAPM_OP_LAST
} r_ble_gtl_gapm_operation_t;

typedef enum e_r_ble_gtl_gapc_operation
{
    R_BLE_GTL_GAPC_OP_NONE = 0x00,
    R_BLE_GTL_GAPC_OP_DISCONNECT,
```

```

R_BLE_GTL_GAPC_OP_GET_PEER_NAME,
R_BLE_GTL_GAPC_OP_GET_PEER_VERSION,
R_BLE_GTL_GAPC_OP_GET_PEER_FEATURES,
R_BLE_GTL_GAPC_OP_GET_PEER_APPEARANCE,
R_BLE_GTL_GAPC_OP_GET_PEER_SLV_PREF_PARAMS,
R_BLE_GTL_GAPC_OP_GET_CON_RSSI,
R_BLE_GTL_GAPC_OP_GET_CON_CHANNEL_MAP,
R_BLE_GTL_GAPC_OP_UPDATE_PARAMS,
R_BLE_GTL_GAPC_OP_BOND,
R_BLE_GTL_GAPC_OP_ENCRYPT,
R_BLE_GTL_GAPC_OP_SECURITY_REQ,
R_BLE_GTL_GAPC_OP_LE_CB_CREATE,
R_BLE_GTL_GAPC_OP_LE_CB_DESTROY,
R_BLE_GTL_GAPC_OP_LE_CB_CONNECTION,
R_BLE_GTL_GAPC_OP_LE_CB_DISCONNECTION,
R_BLE_GTL_GAPC_OP_LE_CB_ADDITION,
R_BLE_GTL_GAPC_OP_GET_LE_PING_TO,
R_BLE_GTL_GAPC_OP_SET_LE_PING_TO,
R_BLE_GTL_GAPC_OP_SET_LE_PKT_SIZE,
R_BLE_GTL_GAPC_OP_GET_PEER_CENTRAL_RPA,
R_BLE_GTL_GAPC_OP_GET_PEER_RPA_ONLY,
R_BLE_GTL_GAPC_OP_LE_CB_SEND,
} r_ble_gtl_gapc_operation_t;

typedef enum e_r_ble_gtl_gattc_operation
{
    R_BLE_GTL_GATTC_OP_NONE = 0x00,
    R_BLE_GTL_GATTC_OP_MTU_EXCH,
    R_BLE_GTL_GATTC_OP_DISC_ALL_SVC,
    R_BLE_GTL_GATTC_OP_DISC_BY_UUID_SVC,
    R_BLE_GTL_GATTC_OP_DISC_INCLUDED_SVC,
    R_BLE_GTL_GATTC_OP_DISC_ALL_CHAR,
    R_BLE_GTL_GATTC_OP_DISC_BY_UUID_CHAR,
    R_BLE_GTL_GATTC_OP_DISC_DESC_CHAR,
    R_BLE_GTL_GATTC_OP_READ,
    R_BLE_GTL_GATTC_OP_READ_LONG,
    R_BLE_GTL_GATTC_OP_READ_BY_UUID,
    R_BLE_GTL_GATTC_OP_READ_MULTIPLE,
    R_BLE_GTL_GATTC_OP_WRITE,
    R_BLE_GTL_GATTC_OP_WRITE_NO_RESPONSE,
    R_BLE_GTL_GATTC_OP_WRITE_SIGNED,
    R_BLE_GTL_GATTC_OP_EXEC_WRITE,
    R_BLE_GTL_GATTC_OP_REGISTER,
    R_BLE_GTL_GATTC_OP_UNREGISTER,
    R_BLE_GTL_GATTC_OP_NOTIFY,
    R_BLE_GTL_GATTC_OP_INDICATE,
} r_ble_gtl_gattc_operation_t;

typedef enum e_r_ble_gtl_aux_operation
{
    R_BLE_GTL_AUX_OP_NONE = 0x00,
    R_BLE_GTL_AUX_SET_TX_POWER = 0x06,
} r_ble_gtl_aux_operation_t;

typedef enum e_r_ble_gtl_host_error_code
{
    R_BLE_GTL_GAP_ERR_NO_ERROR = 0x00,
    R_BLE_GTL_ATT_ERR_INVALID_HANDLE,
    R_BLE_GTL_ATT_ERR_READ_NOT_PERMITTED,
    R_BLE_GTL_ATT_ERR_REQUEST_NOT_SUPPORTED = 0x06,
    R_BLE_GTL_GAP_ERR_CANCELED = 0x44,
}

```

```
R_BLE_GTL_GAP_ERR_TIMEOUT          = 0x45,  
R_BLE_GTL_GAP_ERR_REJECTED        = 0x48,  
R_BLE_GTL_GAP_ERR_ANY             = 0x7F  
} r_ble_gtl_host_error_code_t;  
  
typedef enum e_r_ble_gtl_gapc_device_info  
{  
    R_BLE_GTL_GAPC_DEV_NAME = 0x00,  
    R_BLE_GTL_GAPC_DEV_APPEARANCE,  
    R_BLE_GTL_GAPC_DEV_SLV_PREF_PARAMS,  
    R_BLE_GTL_GAPC_DEV_CENTRAL_RPA,  
    R_BLE_GTL_GAPC_DEV_RPA_ONLY,  
} r_ble_gtl_gapc_device_info_t;  
  
typedef enum e_r_ble_gtl_device_state  
{  
    R_BLE_GTL_DEV_STATE_IDLE = 0x00,  
    R_BLE_GTL_DEV_STATE_CONNECTED,  
    R_BLE_GTL_DEV_STATE_CONNECTING,  
    R_BLE_GTL_DEV_STATE_ADVERTISING = 0x10  
} r_ble_gtl_device_state_t;
```

2.11. Adding the SIS Module to Your Project

The SIS module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in below:

Adding the SIS module to your project using the Smart Configurator in e² studio . By using the Smart Configurator in e² studio , the SIS module is automatically added to your project. Refer to “RL78 Smart Configurator User’s Guide: e² studio (R20AN0579)” for details.

2.12. “for”, “while” and “do while” Statements

In SIS module, “for”, “while” and “do while” statements (loop processing) are used in processing to wait for register to be reflected and so on. For these loop processing, comments with “WAIT_LOOP” as a keyword are described. Therefore, if user incorporates fail-safe processing into loop processing, user can search the corresponding processing with “WAIT_LOOP”.

- This SIS module does not have any WAIT_LOOP. But others might have. Please take care for this WAIT_LOOP.

2.13. Usage Notes

2.13.1 Getting Started Guide

The below guide walks users through building a fully working solution in order to run a BLE application from the RL78 MCU using the GTL interface.

https://lpccs-docs.renesas.com/DA1453x-RL78_BLE_Framework/index.html

2.13.2 Addresses

When using a public BD address the address pre-programmed into the DA14531 will be used and can't be overridden. A random address can be set by calling the R_BLE_VS_SetBdAddr function before the R_BLE_GAP_Init function is called.

2.13.3 Heap Requirements

Ensure the BSP heap size is set to at least 2K bytes.

When using FreeRTOS ensure the heap 4 size is set to a minimum of 2K bytes.

2.13.4 Module Firmware Compatibility

This middleware module is compatible with GTL binary version 6.0.22 and later. You must ensure that the DA14531/DA14535 Module (or PMOD) you are using contains this version (or later) firmware or that you use the boot from host feature and have the host MCU load the binary into the DA14531/DA14535. Note that DA14531 and DA14535 are not firmware compatible even though the GTL API is the same.

Instructions detailing how to upgrade the firmware in a DA14531 Module can be found here:

https://lpccs-docs.renesas.com/US159-DA14531EVZ_Firmware_Upgrade/index.html

The GTL binary file can be downloaded using the tool described in the above instructions, or by using the following link:

<https://www.renesas.com/document/swo/fsp-gtl-binary-us159-da14531evz-pmod-programming>

2.13.5 SIS Module Limitations

Developers should be aware of the following limitations when using the BLE_ABS:

- Following a power on reset, the R_BLE_VS_GetRand function always returns the same number. Subsequent calls to this function produce random numbers.
- Service and characteristic write callback functions, created when using the QE Tool are not supported.
- The RI feature of attributes (values are not stored in DA1453x database) is controlled from the flag Enable Characteristic Value Read Callback of QE Tool.
 - In case of a write request the local data base is automatically updated. The app will be notified by an BLE_GATTS_OP_PEER_XXX_WRITE_REQ.
 - In case of a read request and Value Read Callback is enabled (read indication) the local data base is not automatically accessed to get the value.
 - The user is responsible to provide the reply value from gatts_cb by using R_BLE_GTL_GATTS_ReadCfm after a BLE_GATTS_EVENT_DB_ACCESS_IND with BLE_GATTS_OP_PEER_XXX_READ_REQ. The Client Characteristic Configuration (CCC) values are always treated as RI values.
 - If the attribute is CCC and it is not marked as read_notify, then r_ble_gtl automatically handles it by using a cached value.
- The boot from host feature currently support 1-wire UART & 2-wire UART:
 - When using a 1-wire boot from host with DA14531/DA14535, the UART RX and TX pins on the host RL78 MCU must be connected together using a 1K ohm resistor to boot which resistor can remain in place after the boot operation is completed.
 - When using a 2-wire boot from host with DA14535MOD, the 1K ohm resistor is not required to initiate the process, as it has already been written with a second bootloader supported in its memory.
 - Boot from host using 2-wire UART is not supported when using a DA14531MOD module because not all the required pins are exposed.

- Some code-generated setting with the custom profile generation feature do not work in combination with SIS for the DA1453x module. Also, be sure to perform sufficient test on the generated code.
 - Workaround: Please refer to SIS documents about details of functional restriction.
- Notes on arguments for R_BLE_GATTS_GetAttr functions **(1)**: In the case of DA1453x modules, add code to allocate memory for the members of the structure to be passed to the third argument at the call of the R_BLE_GATTS_GetAttr function in the code generated by QE for BLEAPI Functions.
 - Please note that if you use QE for BLE to generate code again, the changes will be removed.
- Notes on Notification and Client Characteristic Configuration Descriptor **(2)**: In the case of DA1453x modules, the value of the Client Characteristic Configuration Descriptor cannot be obtained from the R_BLE_SERVS_GetDesc function. As a result, calling R_BLE_<Service>_Notify<Characteristic> function generated by QE for BLE does not issue a Notify.
 - To issue a Notify, comment out the part where getting the value of the Client Characteristic Configuration Descriptor and set the value of cccd appropriately.
 - Also, please note that if you use QE for BLE to generate code again, the changes will be removed.
- Up to 3 concurrent connections are supported.
- R_BLE_GTL_VS_SetSleepMode() is not supported, R_BLE_GTL_VS_SetSleepModeExt should be called directly instead.
- Maximum bonded devices are limited to seven (7). Application must delete an entry before bonding with a new device.
- When the device operates as Central role in an environment with multiple surrounding Bluetooth devices, the application may experience instability due to insufficient memory resources on the board.
- Role switching between Central and Peripheral is not supported after the devices have been bonded.
- Out Of Band (OOB) pairing is not supported.
- GAP operations cannot overlap, e.g. you cannot initiate two connections concurrently.

Example Notes **(1)**, **(2)** above can be found here: <https://www.renesas.com/en/document/rln/25574473>

3. API Functions

3.1. R_BLE_Open()

Open the BLE protocol stack.

Format

```
ble_status_t R_BLE_Open (  
    void  
)
```

Parameters

None

Return values

BLE_SUCCESS	Success
-------------	---------

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

Example

```
R_BLE_Open ();
```

Special Notes:

None.

3.2. R_BLE_Close()

Close the BLE protocol stack.

Format

```
ble_status_t R_BLE_Close (  
    void  
)
```

Parameters

None

Return values

BLE_SUCCESS	Success
-------------	---------

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

Example

```
R_BLE_Close();
```

Special Notes:

None

3.4. R_BLE_IsTaskFree()

Check if the BLE task queue is free or not.

Format

```
uint32_t R_BLE_IsTaskFree(  
    void  
)
```

Parameters

None

Return values

0x0	BLE task queue is not free.
0x1	BLE task queue is free.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function returns the BLE task queue free status.

When this function returns 0x0, call R_BLE_Execute() to execute the BLE task.

Example

```
R_BLE_Open();  
while (1)  
{  
    R_BLE_Execute();  
    if(0 != R_BLE_IsTaskFree())  
    {  
        xEventGroupWaitBits();  
    }  
}
```

Special Notes:

None

3.5. R_BLE_GetVersion()

Get the BLE module version.

Format

```
uint32_t R_BLE_GetVersion(  
    void  
)
```

Parameters

None

Return values

Version number

Properties

Prototype declarations are contained in `r_ble_api.h`.

Description

This function returns the BLE SIS module version.

The major version(`BLE_VERSION_MAJOR`) is contained in the two most significant bytes, and the minor version(`BLE_VERSION_MINOR`) occupies the remaining two bytes.

Example

```
uint32_t version;  
  
version = R_BLE_GetVersion();
```

Special Notes:

None

3.6. R_BLE_GAP_Init()

Initialize the Host Stack.

Format

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

Parameters

gap_cb A callback function registered with this function.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	gap_cb is specified as NULL.
BLE_ERR_INVALID_STATE(0x0008)	The reason for this error is as follows: - Host Stack was already initialized. - The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.

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

Example

None

Special Notes:

None

3.7. R_BLE_GAP_Terminate()

Terminate the Host Stack.

Format

```
ble_status_t R_BLE_GAP_Terminate(  
    void  
)
```

Parameters

None

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_STATE(0x0008)	Host stack hasn't been initialized.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The host stack is terminated with this function.

In order to reset all the Bluetooth functions, it's necessary to call this function.

The result of this API call is notified in BLE_GAP_EVENT_STACK_OFF event.

Reentrant

No

Example

None

Special Notes:

None

3.8. R_BLE_GAP_UpdConn()

Update the connection parameters.

Format

```
ble_status_t R_BLE_GAP_UpdConn(
    uint16_t          conn_hdl,
    uint8_t           mode,
    uint16_t          accept,
    st_ble_gap_conn_param_t * p_conn_updt_param
)
```

Parameters

conn_hdl Connection handle identifying the link to be updated.

mode Connection parameter update request or response.

macro	description
BLE_GAP_CONN_UPD_MODE_REQ (0x01)	Request for updating the connection parameters.
BLE_GAP_CONN_UPD_MODE_RSP (0x02)	Reply a connection parameter update request.

accept When mode is BLE_GAP_CONN_UPD_MODE_RSP, accept or reject the connection parameters update request. If mode is BLE_GAP_CONN_UPD_MODE_REQ, accept is ignored.

macro	description
BLE_GAP_CONN_UPD_ACCEPT (0x0000)	Accept the update request.
BLE_GAP_CONN_UPD_REJECT (0x0001)	Reject the update request.

p_conn_updt_param Connection parameters to be updated. When mode is BLE_GAP_CONN_UPD_MODE_RSP and accept is BLE_GAP_CONN_UPD_REJECT, p_conn_updt_param is ignored.

Return values BLE_SUCCESS(0x0000) Success

BLE_ERR_INVALID_PTR(0x0001) When accept is BLE_GAP_CONN_UPD_ACCEPT, p_conn_updt_param is specified as NULL.

BLE_ERR_INVALID_ARG(0x0003) The following is out of range.

- mode
- accept
- conn_intv_min field in p_conn_updt_param
- conn_intv_max field in p_conn_updt_param
- conn_latency in p_conn_updt_param
- sup_to in p_conn_updt_param
- conn_hdl

BLE_ERR_INVALID_STATE(0x0008) Not connected with the remote device.

BLE_ERR_CONTEXT_FULL(0x000B) Sending a L2CAP command, an error occurred.

BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by conn_hdl is not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function updates the connection parameters or replies to a request for updating connection parameters notified by BLE_GAP_EVENT_CONN_PARAM_UPD_REQ event. When the connection parameters have been updated, BLE_GAP_EVENT_CONN_PARAM_UPD_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.9. R_BLE_GAP_SetDataLen()

Update the packet size and the packet transmit time.

Format

```
ble_status_t R_BLE_GAP_SetDataLen(  
    uint16_t    conn_hdl,  
    uint16_t    tx_octets,  
    uint16_t    tx_time  
)
```

Parameters

conn_hdl Connection handle identifying the link whose the transmission packet size or the transmission time to be changed.

tx_octets Maximum transmission packet size. Valid range is 0x001B - 0x00FB.

tx_time Maximum transmission time(us). Valid range is 0x0148 - 0x4290.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_STATE(0x0008)	The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function requests for changing the maximum transmission packet size and the maximum packet transmission time. When Controller has received the request from host stack, BLE_GAP_EVENT_SET_DATA_LEN_COMP event is notified to the application layer. When the transmission packet size or the transmission time has been changed, BLE_GAP_EVENT_DATA_LEN_CHG event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.10. R_BLE_GAP_Disconnect()

Disconnect the link.

Format

```
ble_status_t R_BLE_GAP_Disconnect (
    uint16_t      conn_hdl,
    uint8_t       reason
)
```

Parameters

conn_hdl Connection handle identifying the link to be disconnected.

Reason The reason for disconnection. Usually, set 0x13 which indicates that a user disconnects the link. If setting other than 0x13, refer the error code described in Core Specification Vol.2 Part D , "2 Error Code Descriptions"

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_ARG(0x0003)	conn_hdl is out of range.
BLE_ERR_INVALID_STATE(0x0008)	The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by conn_hdl is not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function disconnects a link. When the link has disconnected, BLE_GAP_EVENT_DISCONN_IND event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.11. R_BLE_GAP_GetVerInfo()

Get the version number of the Controller and the host stack.

Format

```
ble_status_t R_BLE_GAP_GetVerInfo (
    void
)
```

Parameters

None

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_STATE(0x0008)	The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function retrieves the version information of local device. The result of this API call is notified in BLE_GAP_EVENT_LOC_VER_INFO event.

Reentrant

No

Example

None

Special Notes:

None

3.12. R_BLE_GAP_ReadRssi()

Get RSSI.

Format

```
ble_status_t R_BLE_GAP_ReadRssi (
    uint16_t conn_hdl
)
```

Parameters

conn_hdl Connection handle identifying the link whose RSSI to be retrieved.

Return values

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

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function retrieves RSSI. The result of this API call is notified in BLE_GAP_EVENT_RSSI_RD_COMP event.

Reentrant

No

Example

None

Special Notes:

None

3.13. R_BLE_GAP_ReadChMap()

Get the Channel Map.

Format

```
ble_status_t R_BLE_GAP_ReadChMap (
    uint16_t conn_hdl
)
```

Parameters

conn_hdl Connection handle identifying the link whose channel map to be retrieved.

Return values

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

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function retrieves the channel map. The result of this API call is notified in BLE_GAP_EVENT_CH_MAP_RD_COMP event.

Reentrant

No

Example

None

Special Notes:

None

3.14. R_BLE_GAP_SetAdvParam()

Set advertising parameters.

Format

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

Parameters

p_adv_param Advertising parameters.

Return values

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

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

Example

None

Special Notes:

None

3.15. R_BLE_GAP_SetAdvSresData()

Set 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

p_adv_srsp_data Advertising data/scan response data/periodic advertising data.

Return values

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

The following shall be applied regarding the adv_prop_type field and the data_type field in st_ble_gap_adv_param_t parameter specified in R_BLE_GAP_SetAdvParam().

Reentrant

No

Example

None

Special Notes:

None

3.16. R_BLE_GAP_StartAdv()

Start advertising.

Format

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

Parameters

adv_hdl	The advertising handle pointing to the advertising set which starts advertising. The valid range is 0x00 - 0x03.
duration	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.
max_extd_adv_evts	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.

Return values

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

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function starts advertising. 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

Example

None

Special Notes:

None

3.17. R_BLE_GAP_StopAdv()

Stop advertising.

Format

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

Parameters

adv_hdl The advertising handle pointing to the advertising set which stops advertising. The valid range is 0x00 - 0x03.

Return values

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

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

Example

None

Special Notes:

None

3.18. R_BLE_GAP_GetRemainAdvBufSize()

Get buffer size for advertising data/scan response data/periodic advertising data in the Controller.

Format

```
ble_status_t R_BLE_GAP_GetRemainAdvBufSize      (  
    uint16_t * p_remain_adv_data_size,  
    uint16_t * p_remain_perd_adv_data_size  
)
```

Parameters

p_remain_adv_data_size	The free buffer size of Controller to which advertising data/scan response data can be currently set.
p_remain_perd_adv_data_size	The free buffer size of Controller to which periodic advertising data can be currently set.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	p_remain_adv_data_size or p_remain_perd_adv_data_size is specified as NULL.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function gets the total size of advertising data/scan response data/periodic advertising data which can be currently set to Controller(all of the advertising sets). The application layer gets the data sizes via the parameters. By this API function call, no events occur.

Reentrant

No

Example

None

Special Notes:

None

3.19. R_BLE_GAP_GetRemDevInfo()

Get the information about remote device.

Format

```
ble_status_t R_BLE_GAP_GetRemDevInfo      (  
    uint16_t    conn_hdl  
)
```

Parameters

conn_hdl Connection handle identifying the remote device whose information to be retrieved.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_STATE(0x0008)	The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function retrieves information about the remote device. The information includes BD_ADDR, the version number and LE features. The result of this API call is notified in BLE_GAP_EVENT_GET_REM_DEV_INFO event.

Reentrant

No

Example

None

Special Notes:

None

3.20. R_BLE_GAP_SetPairingParams()

Set the parameters using pairing.

Format

```
ble_status_t R_BLE_GAP_SetPairingParams(  
    st_ble_gap_pairing_param_t * p_pair_param  
)
```

Parameters

p_pair_param Pairing parameters.

Return values

BLE_SUCCESS(0x0000)

Success

BLE_ERR_INVALID_ARG(0x0003)

The following field in p_pair_param is out of range.

- iocap
- max_key_size
- mitm
- bonding
- key_notf
- sec_conn_only

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function sets the parameters used in pairing.

Reentrant

No

Example

None

Special Notes:

None

3.21. R_BLE_GAP_StartPairing()

Start pairing.

Format

```
ble_status_t R_BLE_GAP_StartPairing(  
    uint16_t conn_hdl  
)
```

Parameters

conn_hdl Connection handle identifying the remote device which local device starts pairing with.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_STATE(0x0008)	While generating OOB data, this function was called.
BLE_ERR_CONTEXT_FULL(0x000B)	While pairing, this function was called.
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by conn_hdl is not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function starts pairing with a remote device. The result of this API call is returned by a return value. The result of pairing is notified in BLE_GAP_EVENT_PAIRING_COMP event.

Reentrant

No

Example

None

Special Notes:

None

3.22. R_BLE_GAP_ReplyPairing()

Reply the pairing request from a remote device.

Format

```
ble_status_t R_BLE_GAP_ReplyPairing(
    uint16_t conn_hdl,
    uint8_t response
)
```

Parameters

conn_hdl Connection handle identifying the remote device which local device starts pairing with.

response Accept or reject the pairing request from the remote device.

macro	description
BLE_GAP_PAIRING_ACCEPT(0x00)	Accept the pairing request
BLE_GAP_PAIRING_REJECT(0x01)	Reject the pairing request

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_ARG(0x0003)	Response is out of range.
BLE_ERR_INVALID_STATE(0x0008)	While generating OOB data, this function was called.
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by conn_hdl is not found.
BLE_ERR_NOT_YET_READY(0x0012)	When this function was called, host stack has not yet received BLE_GAP_EVENT_PAIRING_REQ event.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function replies to the pairing request from the remote device. The pairing request from the remote device is notified in BLE_GAP_EVENT_PAIRING_REQ event. The result of this API call is returned by a return value. The result of pairing is notified in BLE_GAP_EVENT_PAIRING_COMP event.

Reentrant

No

Example

None

Special Notes:

None

3.23. R_BLE_GAP_ReplyPasskeyEntry()

Reply the passkey entry request.

Format

```
ble_status_t R_BLE_GAP_ReplyPasskeyEntry(
    uint16_t conn_hdl,
    uint32_t passkey,
    uint8_t response
)
```

Parameters

conn_hdl Connection handle identifying the remote device which the reply to passkey entry is sent.

passkey Passkey. The valid range is 000000 - 999999 in decimal.

response Active or negative reply to passkey entry.

macro	description
BLE_GAP_PAIRING_ACCEPT(0x00)	Accept the passkey entry pairing
BLE_GAP_PAIRING_REJECT(0x01)	Reject the passkey entry pairing

Return values

BLE_SUCCESS(0x0000) Success

BLE_ERR_INVALID_ARG(0x0003) Passkey or response is out of range.

BLE_ERR_INVALID_HDL(0x000E) The remote device specified by conn_hdl is not found.

BLE_ERR_NOT_YET_READY(0x0012) When this function was called, pairing has not yet started.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When BLE_GAP_EVENT_PASSKEY_ENTRY_REQ event is notified, the response to passkey entry is sent by this function. The result of this API call is returned by a return value.

Reentrant

No

Example

None

Special Notes:

None
Reply the numeric comparison request.

3.24. R_BLE_GAP_ReplyNumComp()

Reply the numeric comparison request.

Format

```
ble_status_t R_BLE_GAP_ReplyNumComp(
    uint16_t conn_hdl,
    uint8_t response
)
```

Parameters

conn_hdl Connection handle identifying the remote device which the reply to Numeric Comparison is sent.

response Active or negative reply in Numeric Comparison.

macro	description
BLE_GAP_PAIRING_ACCEPT(0x00)	The number displayed in the local is the same as the one of the remote.
BLE_GAP_PAIRING_REJECT(0x01)	The number displayed in the local is differs from the one of the remote.

Return values

BLE_SUCCESS(0x0000) Success

BLE_ERR_INVALID_ARG(0x0003) Response is out of range.

BLE_ERR_INVALID_STATE(0x0008) When this function was called, host stack has not yet received BLE_GAP_EVENT_NUM_COMP_REQ event.

BLE_ERR_INVALID_HDL(0x000E) The remote device specified by conn_hdl is not found.

BLE_ERR_NOT_YET_READY(0x0012) When this function was called, pairing has not yet started.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When BLE_GAP_EVENT_NUM_COMP_REQ event is notified, the response to Numeric Comparison is sent by this function. The result of this API call is returned by a return value.

Reentrant

No

Example

None

Special Notes:

None

3.25. R_BLE_GAP_ReplyExKeyInfoReq()

Distribute the keys of local device.

Format

```
ble_status_t R_BLE_GAP_SetPairingParams (  
    st_ble_gap_pairing_param_t * p_pair_param  
)
```

Parameters

conn_hdl Connection handle identifying the remote device to which the key is distributed.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by conn_hdl is not found.
BLE_ERR_NOT_YET_READY(0x0012)	When this function was called, pairing has not yet started.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When key exchange request is notified by BLE_GAP_EVENT_EX_KEY_REQ event at pairing, keys of the local device are distributed. The result is returned from this API.

Reentrant

No

Example

None

Special Notes:

None

3.26. R_BLE_GAP_ReplyLtkReq()

Reply the LTK request from a remote device.

Format

```
ble_status_t R_BLE_GAP_ReplyLtkReq(
    uint16_t conn_hdl,
    uint16_t ediv,
    uint8_t *p_peer_rand,
    uint8_t response
)
```

Parameters

conn_hdl Connection handle identifying the remote device which sent the LTK request.

ediv Ediv notified in BLE_GAP_EVENT_LTK_REQ event.

p_peer_rand Rand notified in BLE_GAP_EVENT_LTK_REQ event.

response Response to the LTK request. If "BLE_GAP_LTK_REQ_ACCEPT" is specified, when no LTK has been exchanged in pairing, reject the LTK request.

macro	description
BLE_GAP_LTK_REQ_ACCEPT(0x00)	Reply for the LTK request
BLE_GAP_LTK_REQ_DENY(0x01)	Reject the LTK request

Return values

BLE_SUCCESS(0x0000) Success

BLE_ERR_INVALID_PTR(0x0001) p_peer_rand is specified as NULL in case of legacy pairing.

BLE_ERR_INVALID_ARG(0x0003) response is out of range.

BLE_ERR_INVALID_STATE(0x0008) The task for host stack is not running.

BLE_ERR_MEM_ALLOC_FAILED(0x000C) Insufficient memory is needed to generate this function.

BLE_ERR_INVALID_HDL(0x000E) The remote device specified by conn_hdl is not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function replies to the LTK request in BLE_GAP_EVENT_LTK_REQ event from a remote device. The result of the LTK reply is returned in BLE_GAP_EVENT_LTK_RSP_COMP event. When the link encryption has completed, BLE_GAP_EVENT_ENC_CHG event is notified.

Reentrant

No

Example

None

Special Notes:

None

3.27. R_BLE_GAP_StartScan()

This function starts scanning.

Format

```
ble_status_t R_BLE_GAP_StartScan (
    st_ble_gap_scan_param_t * p_scan_param,
    st_ble_gap_scan_on_t * p_scan_enable
)
```

Parameters

p_scan_param Scan parameter. When p_scan_param is specified as NULL, host stack doesn't set scan parameters and start scanning with the previous parameters.

p_scan_enable Scan period, scan duration, duplicate filter and procedure type.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The reason for this error is as follows: <ul style="list-style-type: none"> • p_scan_enable is specified as NULL. • p_phy_param_1M field and p_phy_param_coded field in • p_scan_param are specified as NULL.
BLE_ERR_INVALID_ARG(0x0003)	The reason for this error is as follows: <ul style="list-style-type: none"> • proc_type field in p_scan_enable is out of range. • filter_dups in p_scan_enable is out of range. • o_addr_type in p_scan_param is out of range. • filter_policy in p_scan_param is out of range. • scan_type of p_scan_param's p_phy_param_1M or p_phy_param_coded is out of range.
BLE_ERR_UNSUPPORTED(0x0007)	Not supported.
BLE_ERR_INVALID_STATE(0x0008)	The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When scanning for the first time, set the p_scan_param. Setting scan parameters can be omitted by specifying p_scan_param as NULL after next time. The result of this API call is notified in BLE_GAP_EVENT_SCAN_ON event. Advertising report is notified in BLE_GAP_EVENT_ADV_REPT_IND event.

Reentrant

No

Example

None

Special Notes:

None

3.28. R_BLE_GAP_StopScan()

This function stops scanning

Format

```
ble_status_t R_BLE_GAP_StopScan(void)
```

Parameters**Return values**

BLE_SUCCESS(0x0000)	Success
BLE_ERR_UNSUPPORTED(0x0007)	Not supported.
BLE_ERR_INVALID_STATE(0x0008)	The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_SUCCESS(0x0000)	Success
BLE_ERR_UNSUPPORTED(0x0007)	Not supported.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The result of this API call is notified in BLE_GAP_EVENT_SCAN_OFF event.

Reentrant

No

Example

None

Special Notes:

None

3.29. R_BLE_GAP_CreateConn()

Request for a link establishment.

Format

```
ble_status_t R_BLE_GAP_CreateConn (st_ble_gap_create_conn_param * p_param)
```

Parameters

p_param Connection parameters.

Return values

BLE_SUCCESS(0x0000)

Success

BLE_ERR_INVALID_PTR(0x0001)

The reason for this error is as follows:

- p_param is specified as NULL.
- p_conn_param_1M field and p_conn_param_2M and p_conn_param_coded field in p_param are specified as NULL.
- When creating a link with 1M PHY, p_conn_param in p_conn_param_1M field in p_param is specified as NULL.
- When creating a link with 2M PHY, p_conn_param in p_conn_param_2M field in p_param is specified as NULL.
- When creating a link with coded MPHY, p_conn_param in p_conn_param_coded field in p_param is specified as NULL.

BLE_ERR_INVALID_ARG(0x0003)

The reason for this error is as follows:

- init_filter_policy in p_param is out of range.
- remote_bd_addr_type field or own_addr_type address field in p_param is out of range.

BLE_ERR_UNSUPPORTED(0x0007)

Not supported.

BLE_ERR_INVALID_STATE(0x0008)

The task for host stack is not running.

BLE_ERR_MEM_ALLOC_FAILED(0x000C)

Insufficient memory is needed to generate this function.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function sends a connection request to a remote device to create a link. When Controller has received a request for establishment of a link from host stack, BLE_GAP_EVENT_CREATE_CONN_COMP event is notified to the application layer. When the link is established, BLE_GAP_EVENT_CONN_IND event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.30. R_BLE_GAP_CancelCreateConn()

Cancel the request for a link establishment

Format

ble_status_t R_BLE_GAP_CancelCreateConn(void)

Parameters**Return values**

BLE_SUCCESS(0x0000)	Success
BLE_ERR_UNSUPPORTED(0x0007)	Not supported.
BLE_ERR_INVALID_STATE(0x0008)	The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_SUCCESS(0x0000)	Success
BLE_ERR_UNSUPPORTED(0x0007)	Not supported.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function cancels a request for establishing a link. When Controller has received the cancel request from host stack, BLE_GAP_EVENT_CONN_CANCEL_COMP event is notified to the application layer. When the cancel procedure has completed, BLE_GAP_EVENT_CONN_IND event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.31. R_BLE_GAP_ConfWhiteList()

Set white list.

Format

```
ble_status_t R_BLE_GAP_ConfWhiteList (
    uint8_t          op_code,
    st_ble_dev_addr_t * p_addr,
    uint8_t          device_num)
```

Parameters

op_code The operation for White List

macro	description
BLE_GAP_LIST_ADD_DEV(0x01)	Add the device to the list
BLE_GAP_LIST_REM_DEV(0x02)	Delete the device from the list
BLE_GAP_LIST_CLR(0x03)	Clear the list

p_addr An array of device address to add/delete to the list. The number of elements is specified by device_num. If op_code is BLE_GAP_LIST_CLR, p_addr is ignored.

device_num The number of devices add/delete to the list. Valid range is 1- BLE_GAP_WHITE_LIST_MAX_ENTRY. If op_code is BLE_GAP_LIST_CLR, device_num is ignored.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	When op_code is BLE_GAP_LIST_ADD_DEV or BLE_GAP_LIST_REM_DEV, p_addr is specified as NULL.
BLE_ERR_INVALID_ARG(0x0003)	op_code or address type field in p_addr is out of range.
BLE_ERR_INVALID_STATE(0x0008)	The reason for this error is as follows: <ul style="list-style-type: none"> • While operating White List, this function was called. • The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	There are no memories for operating the White List.
BLE_SUCCESS(0x0000)	Success

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function supports the following operations regarding White List.

- Add the device to White List.
- Delete the device from White List.
- Clear White List.

The total number of White List entries is defined as BLE_GAP_WHITE_LIST_MAX_ENTRY. The result of this API call is notified in BLE_GAP_EVENT_WHITE_LIST_CONF_COMP event.

Reentrant

No

Example

None

Special Notes:

None

3.32. R_BLE_GAP_StartEnc()

Encryption the link.

Format

ble_status_t R_BLE_GAP_StartEnc (uint16_t conn_hdl)

Parameters

conn_hdl: Connection handle identifying the link which is encrypted.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_STATE(0x0008)	The reason for this error is as follows: <ul style="list-style-type: none">• Pairing has not been completed.• The task for host stack is not running.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by conn_hdl is not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

This function starts encryption of the link. In case of master device, the local device requests for the encryption to a remote device. In case of slave device, the local device sends a Security Request to a remote device. After receiving the Security Request, the remote device requests for the encryption to the local device. The result of the encryption is returned in BLE_GAP_EVENT_ENC_CHG event.

Reentrant

No

Example

None

Special Notes:

None

3.33. 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

conn_hdl Connection handle identifying the GATT Server or the GATT Client.
p_mtu The Current MTU. Before MTU exchange, this parameter is 23 bytes.
 After MTU exchange, this parameter is the negotiated MTU.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The mtu parameter is NULL.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server or the GATT Client specified by conn_hdl 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

Example

None

Special Notes:

None

3.34. 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

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

Example

None

Special Notes:

None

3.35. 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

cb Callback function for GATT Server event.

priority The priority of the callback function.
Valid range is 1 <= priority <= BLE_GATTS_MAX_CB.
A lower priority number means a higher priority level.

Return values

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

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

Example

None

Special Notes:

None

3.36. R_BLE_GATTS_DeregisterCb()

This function deregisters the callback function for GATT Server event.

Format

```
ble_status_t R_BLE_GATTS_DeregisterCb (
    ble_gatts_app_cb_t cb
)
```

Parameters

cb Callback function for GATT Server event.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The cb parameter is specified as NULL.
BLE_ERR_NOT_FOUND(0x000D)	The callback has not been registered.

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

Example

None

Special Notes:

None

3.37. 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

conn_hdl Connection handle identifying the remote device to be sent the notification.
p_ntf_data The attribute value to send.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_ntf_data parameter or the value field in the value field in the p_ntf_data parameter is NULL.
BLE_ERR_INVALID_ARG(0x0003)	The value_len field in the value field in the p_ntf_data parameter is 0 or the attr_hdl field in the p_ntf_data parameters is 0.
BLE_ERR_INVALID_OPERATION(0x0009)	This function was called while processing other request.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by conn_hdl 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

Example

None

Special Notes:

None

3.38. 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

conn_hdl Connection handle identifying the remote device to be sent the indication.
p_ind_data The attribute value to send.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_ind_data parameter or the value field in the value field in the p_ind_data parameter is NULL.
BLE_ERR_INVALID_ARG(0x0003)	The value_len field in the value field in the p_ind_data parameter is 0 or the attr_hdl field in the p_ind_data parameters is 0.
BLE_ERR_INVALID_OPERATION(0x0009)	This function was called while processing other request.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The remote device specified by conn_hdl 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 a indication sends a confirmation.

BLE_GATTS_EVENT_HDL_VAL_CNF event notifies the application layer that the confirmation has been received.

Reentrant

No

Example

None

Special Notes:

None

3.39. 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

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`.

attr_hdl The attribute handle of the attribute value to be retrieved.

p_value The attribute value to be retrieved.

Return values

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

Example

None

Special Notes:

None

3.40. R_BLE_GATTS_SetAttr()

This function sets an attribute value to the GATT Database event.

Format

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

Parameters

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`.

attr_hdl The attribute handle of the attribute value to be set.

p_value The attribute value to be set.

Return values

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

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

Example

None

Special Notes:

None

3.41. 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

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

Return values

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

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

Example

None

Special Notes:

None

3.42. R_BLE_GATTC_DeregisterCb()

This function deregisters the callback function for GATT Client event.

Format

```
ble_status_t R_BLE_GATTC_DeregisterCb (
    ble_gattc_app_cb_t cb
)
```

Parameters

cb The callback function to be deregistered.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The cb parameter is specified as NULL.
BLE_ERR_NOT_FOUND(0x000D)	The callback has not been registered.

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

Example

None

Special Notes:

None

3.43. R_BLE_GATTC_ReqExMtu()

This function sends a MTU Exchange Request PDU to a GATT Server in order to change the current MTU.

Format

```
ble_status_t R_BLE_GATTC_ReqExMtu (
    uint16_t    conn_hdl,
    uint16_t    mtu
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be sent.

mtu The maximum size(in bytes) of the GATT PDU that GATT Client can receive.
Valid range is 23 <= mtu <= 247.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_ARG(0x0003)	The mtu parameter is out of range.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

MTU Exchange Response is notified by BLE_GATTC_EVENT_EX_MTU_RSP event.

The new MTU is the minimum value of the mtu parameter specified by this function and the mtu field in BLE_GATTC_EVENT_EX_MTU_RSP event. Default MTU size is 23 bytes.

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

Reentrant

No

Example

None

Special Notes:

None

3.44. R_BLE_GATTC_DiscAllPrimServ()

This function discovers all Primary Services in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_DiscAllPrimServ (
    uint16_t conn_hdl
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be discovered.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_OPERATION(0x0009)	This function was called while processing other requests.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When 16-bit UUID Primary Service has been discovered, BLE_GATTC_EVENT_PRIM_SERV_16_DISC_IND event is notified to the application layer.

When 128-bit UUID Primary Service has been discovered, BLE_GATTC_EVENT_PRIM_SERV_128_DISC_IND event is notified to the application layer.

When the Primary Service discovery has been completed, BLE_GATTC_EVENT_ALL_PRIM_SERV_DISC_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.45. R_BLE_GATTC_DiscPrimServ()

This function discovers Primary Service specified by p_uuid in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_DiscPrimServ (
    uint16_t    conn_hdl,
    uint8_t *   p_uuid,
    uint8_t     uuid_type
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be discovered.
p_uuid UUID of Primary Service to be discovered.
uuid_type UUID type(16-bit or 128-bit).

macro	description
BLE_GATT_16_BIT_UUID_FORMAT(0x01)	16-bit UUID
BLE_GATT_128_BIT_UUID_FORMAT(0x02)	128-bit UUID

Return values

BLE_SUCCESS(0x0000) Success
BLE_ERR_INVALID_PTR(0x0001) The p_uuid parameter is specified as NULL.
BLE_ERR_INVALID_ARG(0x0003) The uuid_type parameter is out of range.
BLE_ERR_INVALID_OPERATION(0x0009) While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C) Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E) The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When Primary Service whose uuid is the same as the specified uuid has been discovered, BLE_GATTC_EVENT_PRIM_SERV_16_DISC_IND event or BLE_GATTC_EVENT_PRIM_SERV_128_DISC_IND event is notified to the application layer.

When the Primary Service discovery has been completed, BLE_GATTC_EVENT_PRIM_SERV_DISC_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.46. R_BLE_GATTC_DiscIncServ()

This function discovers Included Services within the specified attribute handle range in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_DiscIncServ      (  
    uint16_t    conn_hdl,  
    st_ble_gatt_hdl_range_t *    p_range  
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be discovered.
p_range Retrieval range of Included Service.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_range parameter is specified as NULL.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When Included Service that includes 16-bit UUID Service has been discovered, BLE_GATTC_EVENT_INC_SERV_16_DISC_IND event is notified to the application layer.

When Included Service that includes 128-bit UUID Service has been discovered, BLE_GATTC_EVENT_INC_SERV_128_DISC_IND event is notified to the application layer.

When the Included Service discovery has been completed, BLE_GATTC_EVENT_INC_SERV_DISC_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.47. R_BLE_GATTC_DiscAllChar()

This function discovers Characteristic within the specified attribute handle range in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_DiscAllChar      (  
    uint16_t                               conn_hdl,  
    st_ble_gatt_hdl_range_t *             p_range  
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be discovered.
p_range Retrieval range of Characteristic.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_range parameter is specified as NULL.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When 16-bit UUID Characteristic has been discovered, BLE_GATTC_EVENT_CHAR_16_DISC_IND event is notified to the application layer.

When 128-bit UUID Characteristic has been discovered, BLE_GATTC_EVENT_CHAR_128_DISC_IND event is notified to the application layer.

When the Characteristic discovery has been completed, BLE_GATTC_EVENT_ALL_CHAR_DISC_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.48. R_BLE_GATTC_DiscCharByUuid()

This function discovers Characteristic specified by uuid within the specified attribute handle range in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_DiscCharByUuid (
    uint16_t          conn_hdl,
    uint8_t *        p_uuid,
    uint8_t          uuid_type,
    st_ble_gatt_hdl_range_t * p_range
)
```

Parameters

- conn_hdl Connection handle identifying the GATT Server to be discovered.
- p_uuid UUID of Characteristic to be discovered.
- uuid_type UUID type of Characteristic to be discovered.

macro	description
BLE_GATT_16_BIT_UUID_FORMAT(0x01)	The p_uuid parameter is 16-bit UUID.
BLE_GATT_128_BIT_UUID_FORMAT(0x02)	The p_uuid parameter is 128-bit UUID.

- p_range Retrieval range of Characteristic.

Return values

- BLE_SUCCESS(0x0000) Success
- BLE_ERR_INVALID_PTR(0x0001) The p_uuid parameter or the p_range parameter is specified as NULL.
- BLE_ERR_INVALID_ARG(0x0003) The uuid_type parameter is out of range.
- BLE_ERR_INVALID_OPERATION(0x0009) While processing other request, this function was called.
- BLE_ERR_MEM_ALLOC_FAILED(0x000C) Insufficient memory is needed to generate this function.
- BLE_ERR_INVALID_HDL(0x000E) The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When 16-bit UUID Characteristic has been discovered, BLE_GATTC_EVENT_CHAR_16_DISC_IND event is notified to the application layer.

When 128-bit UUID Characteristic has been discovered, BLE_GATTC_EVENT_CHAR_128_DISC_IND event is notified to the application layer.

When the Characteristic discovery has been completed, BLE_GATTC_EVENT_CHAR_DISC_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.49. R_BLE_GATTC_DiscAllCharDesc()

This function discovers Characteristic Descriptor within the specified attribute handle range in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_DiscAllChar      (  
    uint16_t                               conn_hdl,  
    st_ble_gatt_hdl_range_t *             p_range  
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be discovered.
p_range Retrieval range of Characteristic Descriptor.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_range parameter is specified as NULL.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When 16-bit UUID Characteristic Descriptor has been discovered, BLE_GATTC_EVENT_CHAR_DESC_16_DISC_IND event is notified to the application layer.

When 128-bit UUID Characteristic Descriptor has been discovered, BLE_GATTC_EVENT_CHAR_DESC_128_DISC_IND event is notified to the application layer.

When the Characteristic Descriptor discovery has been completed, BLE_GATTC_EVENT_ALL_CHAR_DESC_DISC_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.50. R_BLE_GATTC_ReadChar()

This function reads a Characteristic/Characteristic Descriptor in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_ReadChar (
    uint16_t    conn_hdl,
    uint16_t    value_hdl
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be read.
value_hdl Value handle of the Characteristic/Characteristic Descriptor to be read.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_ARG(0x0003)	0 is specified in the value_hdl parameter.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The result of the read is notified in BLE_GATTC_EVENT_CHAR_READ_RSP event.

Reentrant

No

Example

None

Special Notes:

None

3.51. R_BLE_GATTC_ReadCharUsingUuid()

This function reads a Characteristic in a GATT Server using a specified UUID.

Format

```
ble_status_t R_BLE_GATTC_ReadCharUsingUuid      (
    uint16_t          conn_hdl,
    uint8_t *        p_uuid,
    uint8_t          uuid_type,
    st_ble_gatt_hdl_range_t *  p_range
)
```

Parameters

conn_hdl Connection handle that identifies Characteristic to be read to GATT Server.

p_uuid UUID of the Characteristic to be read.

uuid_type UUID type of the Characteristic to be read.

macro	description
BLE_GATT_16_BIT_UUID_FORMAT(0x01)	The p_uuid parameter is 16-bit UUID.
BLE_GATT_128_BIT_UUID_FORMAT(0x02)	The p_uuid parameter is 128-bit UUID.

p_range Retrieval range of Characteristic.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_uuid parameter or the p_range parameter is specified as NULL.
BLE_ERR_INVALID_ARG(0x0003)	The uuid_type parameter is out of range.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The result of the read is notified in BLE_GATTC_EVENT_CHAR_READ_BY_UUID_RSP event.

Reentrant

No

Example

None

Special Notes:

None

3.52. R_BLE_GATTC_ReadLongChar()

This function reads a Long Characteristic in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_ReadLongChar    (  
    uint16_t    conn_hdl,  
    uint16_t    value_hdl,  
    uint16_t    offset  
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be read.
value_hdl Value handle of the Long Characteristic to be read.
offset Offset that indicates the location to be read.
Normally, set 0 to this parameter.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_ARG(0x0003)	0 is specified in the value_hdl parameter.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The contents of the Long Characteristic that has been read is notified every MTU-1 bytes to the application layer by BLE_GATTC_EVENT_CHAR_READ_RSP event.

When all of the contents has been received in GATT Client, BLE_GATTC_EVENT_LONG_CHAR_READ_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.53. R_BLE_GATTC_ReadMultiChar()

This function reads multiple Characteristics in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_ReadMultiChar (
    uint16_t                               conn_hdl,
    st_ble_gattc_rd_multi_req_param_t *    p_list
)
```

Parameters

conn_hdl Connection handle that identifies Characteristic to be read to GATT Server.

p_list List of Value Handles that point the Characteristics to be read.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_list parameter or the p_hdl_list field in the p_list parameter is specified as NULL.
BLE_ERR_INVALID_ARG(0x0003)	0 is specified in the value_hdl parameter.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The contents of the multiple Characteristics that has been read is notified to the application layer by BLE_GATTC_EVENT_MULTI_CHAR_READ_RSP event.

Reentrant

No

Example

None

Special Notes:

None

3.54. R_BLE_GATTC_WriteCharWithoutRsp()

This function writes a Characteristic in a GATT Server without response.

Format

```
ble_status_t R_BLE_GATTC_WriteCharWithoutRsp (
    uint16_t conn_hdl,
    st_ble_gatt_hdl_value_pair_t * p_write_data
)
```

Parameters

conn_hdl Connection handle that identifies Characteristic to be read to GATT Server.
p_write_data Value to be written to the Characteristic.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_write_data parameter or the p_value field in the value field in the p_write_data parameter is specified as NULL.
BLE_ERR_INVALID_ARG(0x0003)	The reason for this error is as follows: <ul style="list-style-type: none"> 0 is specified in the value_len field in the p_value field in the p_write_data parameter. 0 is specified in the attr_hdl field in the p_write_data parameter.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The result is returned from the API.

Reentrant

No

Example

None

Special Notes:

None

3.55. R_BLE_GATTC_SignedWriteChar()

This function writes Signed Data to a Characteristic in a GATT Server without response.

Format

```
ble_status_t R_BLE_GATTC_SignedWriteChar (
    uint16_t                conn_hdl,
    st_ble_gatt_hdl_value_pair_t * p_write_data
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be written.
p_write_data Signed Data to be written to the Characteristic.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_write_data parameter or the p_value field in the value field in the p_write_data parameter is specified as NULL.
BLE_ERR_INVALID_ARG(0x0003)	The reason for this error is as follows: <ul style="list-style-type: none"> 0 is specified in the value_len field in the p_value field in the p_write_data parameter. 0 is specified in the attr_hdl field in the p_write_data parameter.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl 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

Example

None

Special Notes:

None

3.56. R_BLE_GATTC_WriteChar()

This function writes a Characteristic in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_WriteChar (
    uint16_t                conn_hdl,
    st_ble_gatt_hdl_value_pair_t * p_write_data
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be written.

p_write_data Signed Data to be written to the Characteristic.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_write_data parameter or the p_value field in the value field in the p_write_data parameter is specified as NULL.
BLE_ERR_INVALID_ARG(0x0003)	The reason for this error is as follows: <ul style="list-style-type: none"> • 0 is specified in the value_len field in the value field in the p_write_data parameter. • 0 is specified in the attr_hdl field in the p_write_data parameter.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The result of the write is notified in BLE_GATTC_EVENT_CHAR_WRITE_RSP event.

Reentrant

No

Example

None

Special Notes:

None

3.57. R_BLE_GATTC_WriteLongChar()

This function writes a Long Characteristic in a GATT Server.

Format

```
ble_status_t R_BLE_GATTC_WriteLongChar (
    uint16_t                conn_hdl,
    st_ble_gatt_hdl_value_pair_t * p_write_data,
    uint16_t                offset
)
```

Parameters

conn_hdl Connection handle identifying the GATT Server to be written.

p_write_data Value to be written to the Long Characteristic.

Offset Offset that indicates the location to be written. Normally, set 0 to this parameter.

If this parameter sets to a value other than 0, adjust the offset parameter and the length of the value to be written not to exceed the length of the Long Characteristic.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The p_write_data parameter or the p_value field in the value field in the p_write_data parameter is specified as NULL.
BLE_ERR_INVALID_ARG(0x0003)	The reason for this error is as follows: <ul style="list-style-type: none"> • The value_len field in the value field in the p_write_data parameter is 0. • The sum of the value_len field in the value field in the p_write_data parameter and the offset parameter larger than 512. • The attr_hdl field in the p_write_data parameter is 0.
BLE_ERR_INVALID_OPERATION(0x0009)	While processing other request, this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

The result of a write that has been done every segmentation is notified to the application layer in BLE_GATTC_EVENT_CHAR_PART_WRITE_RSP event.

The maximum writable size to a Long Characteristic with this function is 512 bytes.

When all of the contents has been written to the Long Characteristic, BLE_GATTC_EVENT_LONG_CHAR_WRITE_COMP event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.58. R_BLE_GATTC_ReliableWrites()

This function performs the Reliable Writes procedure described in GATT Specification.

Format

```
ble_status_t R_BLE_GATTC_ReliableWrites (
    uint16_t                conn_hdl,
    st_ble_gattc_reliable_writes_char_pair_t * p_char_pair,
    uint8_t                 pair_num,
    uint8_t                 auto_flag
)
```

Parameters

- conn_hdl** Connection handle identifying the GATT Server to be written.
- p_char_pair** Pair of Characteristic Value and Characteristic Value Handle identifying the Characteristic to be written by Reliable Writes.
- pair_num** The number of the pairs specified by the p_char_pair parameter.
Valid range is 0 < pair_num <= BLE_GATTC_RELIABLE_WRITES_MAX_CHAR_PAIR.
- auto_flag** The flag that indicates whether auto execution or not.

macro	description
BLE_GATTC_EXEC_AUTO(0x01)	Auto execution.
BLE_GATTC_EXEC_NOT_AUTO (0x02)	Not auto execution.

Return values

- BLE_SUCCESS(0x0000)** Success
- BLE_ERR_INVALID_PTR(0x0001)** The reason for this error is as follows:
- The p_char_pair parameter is specified as NULL.
 - The p_value field in the value field in the write_data field in the p_char_pair parameter is specified as NULL.
- BLE_ERR_INVALID_ARG(0x0003)** The reason for this error is as follows:
- The pair_num parameter or the auto_flag parameter is out of range.
 - The value_len field in the value field in the write_data field in the p_char_pair parameter is 0.
- BLE_ERR_INVALID_OPERATION(0x0009)** While processing other request, this function was called.
- BLE_ERR_MEM_ALLOC_FAILED(0x000C)** Insufficient memory is needed to generate this function or to store the temporary write data.
- BLE_ERR_INVALID_HDL(0x000E)** The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in `r_ble_api.h`.

Description

When the data written to the Characteristic has been transmitted, `BLE_GATTC_EVENT_CHAR_PART_WRITE_RSP` event is notified to the application layer.

If the data included in the event is different from the data that GATT Client has sent, host stack automatically cancels the Reliable Writes.

After all of the contents has been sent to the GATT Server, if the `auto_flag` parameter has been set to `BLE_GATTC_EXEC_AUTO`, the GATT Server automatically writes the data to the Characteristic.

If the `auto_flag` parameter has been set to `BLE_GATTC_EXEC_NOT_AUTO`, `BLE_GATTC_EVENT_RELIABLE_WRITES_TX_COMP` event notifies the application layer in GATT Client that all of the contents has been sent to the GATT Server. Then GATT Client requests for writing the data to the Characteristic to the GATT Server with `R_BLE_GATTC_ExecWrite()`.

When the write has been done, `BLE_GATTC_EVENT_RELIABLE_WRITES_COMP` event is notified to the application layer.

Reentrant

No

Example

None

Special Notes:

None

3.59. R_BLE_GATTC_ExecWrite()

This function is used to execute a write to Characteristic.

Format

```
ble_status_t R_BLE_GATTC_ExecWrite (
    uint16_t    conn_hdl,
    uint8_t     exe_flag
)
```

Parameters

conn_hdl Connection handle identifying the target GATT Server.

exe_flag The flag that indicates whether execution or cancellation.

macro	description
BLE_GATTC_EXECUTE_WRITE_CANCEL_FLAG(0x00)	Execute the write.
BLE_GATTC_EXECUTE_WRITE_EXEC_FLAG(0x01)	Cancel the write.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_ARG(0x0003)	The exe_flag parameter is out of range.
BLE_ERR_INVALID_OPERATION(0x0009)	The reason for this error is as follows: <ul style="list-style-type: none"> GATT Client has not requested for Reliable Writes by R_BLE_GATTC_ReliableWrites(). Although auto execution has been specified by R_BLE_GATTC_ReliableWrites(), this function was called.
BLE_ERR_MEM_ALLOC_FAILED(0x000C)	Insufficient memory is needed to generate this function.
BLE_ERR_INVALID_HDL(0x000E)	The GATT Server specified by conn_hdl was not found.

Properties

Prototype declarations are contained in r_ble_api.h.

Description

When all of the contents has been sent to the GATT Server, BLE_GATTC_EVENT_RELIABLE_WRITES_TX_COMP event notifies the application layer.

After this event has been received, execute the write by this function.

The result of the write is notified by BLE_GATTC_EVENT_RELIABLE_WRITES_COMP event.

Reentrant

No

Example

None

Special Notes:

None

3.60. R_BLE_VS_Init()

This function initializes Vendor Specific API and registers a callback function for Vendor Specific Event.

Format

```
ble_status_t R_BLE_VS_Init    (  
    ble_vs_app_cb_t    vs_cb  
)
```

Parameters

vs_cb Callback function to be registered.

Return values

BLE_SUCCESS(0x0000)	Success
BLE_ERR_INVALID_PTR(0x0001)	The vs_cb parameter is specified as NULL.
BLE_ERR_CONTEXT_FULL(0x000B)	Callback function has already been registered.

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

Example

None

Special Notes:

None

3.61. 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

conn_hdl Connection handle identifying the link whose transmit power to be configured.

tx_power Transmission power. Select one of the following.

macro	description
BLE_VS_TX_POWER_HIGH	High power level with address 0x00
BLE_VS_TX_POWER_MID	Middle power level with address 0x01
BLE_VS_TX_POWER_LOW	Low power level with address 0x02

Return values

BLE_SUCCESS(0x0000) Success

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

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
- Low

Max transmit power of "High" is dependent on the configuration of the firmware.

The result of this API call is notified in BLE_VS_EVENT_SET_TX_POWER event.

Reentrant

No

Example

None

Special Notes:

None

3.62. R_BLE_VS_GetTxPower()

This function gets transmit power.

Format

```
ble_status_t R_BLE_VS_GetTxPower    (  
    uint16_t conn_hdl  
)
```

Parameters

conn_hdl Connection handle identifying the link whose transmit power to be retrieved.

Return values

BLE_SUCCESS(0x0000)	Success
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

This function gets 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 getting 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 retrieved, set the conn_hdl parameter to the connection handle of the link.

The result of this API call is notified in BLE_VS_EVENT_GET_TX_POWER event.

Reentrant

No

Example

None

Special Notes:

None

3.63. R_BLE_VS_GetBdAddr()

This function gets currently configured public/random address.

Format

```
ble_status_t R_BLE_VS_GetBdAddr (
    uint8_t    area,
    uint8_t    addr_type
)
```

Parameters

area The area that the address is to be retrieved.

Select one of the following.

macro	description
BLE_VS_ADDR_AREA_REG(0x00)	Retrieve the address in register.
BLE_VS_ADDR_AREA_DATA_FLASH(0x01)	Retrieve the address in DataFlash area.

addr_type The address type that is type of the address to be retrieved.

macro	description
BLE_GAP_ADDR_PUBLIC(0x00)	Public address.
BLE_GAP_ADDR_RAND(0x01)	Random address.

Return values

BLE_SUCCESS(0x0000)	Success
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

The area parameter specifies the place where this function retrieves public/random address. The result of this API call is notified in BLE_VS_EVENT_GET_ADDR_COMP event.

Reentrant

No

Example

None

Special Notes:

None

3.64. 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

area The area that the address is to be written in.

Select one of the following.

macro	description
BLE_VS_ADDR_AREA_REG(0x00)	Address writing to non-volatile area is not performed. Only the address in register is written.
BLE_VS_ADDR_AREA_DATA_FLASH(0x01)	Address wiring to DataFlash area is performed.

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

Example

None

Special Notes:

None

3.65. R_BLE_VS_GetRand()

This function generates 4-16 bytes of random number used in creating keys.

Format

```
ble_status_t R_BLE_VS_GetRand (  
    uint8_t    rand_size  
)
```

Parameters

rand_size Length of the random number (byte).
 The valid range is $4 \leq \text{rand_size} \leq 16$.

Return values=

BLE_SUCCESS(0x0000) Success
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

The result of this API call is notified in BLE_VS_EVENT_GET_RAND event.

Reentrant

No

Example

None

Special Notes:

None

4. Abstraction API for Renesas QE for BLE

4.1 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 * p_cfg
)
```

Parameters

p_ctrl Pointer to control structure.
p_cfg Pointer to the configuration structure for this instance.

Return values=

FSP_SUCCESS	Channel opened successfully.
FSP_ERR_ASSERTION	Null pointer presented.
FSP_ERR_ALREADY_OPEN	Requested channel is already open in a different configuration.
FSP_ERR_INVALID_ARGUMENT	Invalid input parameter.
FSP_ERR_INVALID_MODE	Invalid mode during open call.

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

Example

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

Special Notes:

None

4.2 RM_BLE_ABS_Close()

Close the BLE channel.

Format

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

Parameters

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

Example

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

Special Notes:

None

4.3 RM_BLE_ABS_StartLegacyAdvertising()

Start 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

p_ctrl	Pointer to control structure.
p_advertising_parameter	Pointer to Advertising parameters for Legacy Advertising.

Return values=

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

Properties

Prototype declarations are contained in rm_ble_abs.h.

Description

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

Example

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

Special Notes:

None

4.4 RM_BLE_ABS_DeleteBondInformation()

Delete bonding information from BLE stack and storage.

Format

```
fsp_err_t RM_BLE_ABS_DeleteBondInformation      (  
ble_abs_ctrl_t * const                          p_ctrl,  
ble_abs_bond_information_parameter_t const * const p_bond_information_parameter  
)
```

Parameters

p_ctrl	Pointer to control structure.
p_bond_information_parameter	Pointer to Bonding information parameters.

Return values

FSP_SUCCESS	Operation succeeded.
FSP_ERR_ASSERTION	p_instance_ctrl is specified as NULL.
FSP_ERR_NOT_OPEN	Control block not open.
FSP_ERR_INVALID_POINTER	p_advertising_parameter is specified as NULL.

Properties

Prototype declarations are contained in rm_ble_abs.h.

Description

Delete bonding information from BLE stack and storage. Implements ble_abs_api_t::deleteBondInformation.

Reentrant

No

Example

```
/* Delete bonding information. */  
err = RM_BLE_ABS_DeleteBondInformation(&g_ble_abs0_ctrl,  
&g_bond_information_parameter);
```

Special Notes:

None

4.5 RM_BLE_ABS_StartScanning()

Start scanning after setting scan parameters.

Format

```
fsp_err_t RM_BLE_ABS_StartScanning (
ble_abs_ctrl_t * const                p_ctrl,
ble_abs_scan_parameter_t const * const p_scan_parameter
)
```

Parameters

p_ctrl	Pointer to control structure.
p_scan_parameter	Pointer to Scan information parameters.

Return values

FSP_ERR_ASSERTION	p_instance_ctrl is specified as NULL.
FSP_ERR_NOT_OPEN	Control block not open.
FSP_ERR_INVALID_POINTER	p_scan_parameter is specified as NULL.
FSP_ERR_UNSUPPORTED	Function is not supported

Properties

Prototype declarations are contained in rm_ble_abs.h.

Description

Start scanning after setting scan parameters. The scanner address type is Public Identity Address. Fast scan is followed by slow scan. The end of fast scan or slow scan is notified with BLE_GAP_EVENT_SCAN_TO event. If fast_period is 0, only slow scan is carried out. If scan_period is 0, slow scan continues. Implements ble_abs_api_t::startScanning

Reentrant

No

Example

```
/* Start scanning.*/
err = RM_BLE_ABS_StartScanning(&g_ble_abs0_ctrl, &scan_parameter);
```

Special Notes:

None

4.6 RM_BLE_ABS_CreateConnection()

Request create connection.

Format

```
fsp_err_t RM_BLE_ABS_CreateConnection      (  
ble_abs_ctrl_t * const                    p_ctrl,  
ble_abs_connection_parameter_t const * const p_connection_parameter  
)
```

Parameters

p_ctrl	Pointer to control structure.
p_connection_parameter	Pointer to connection information parameters.

Return values

FSP_ERR_ASSERTION	p_instance_ctrl is specified as NULL.
FSP_ERR_NOT_OPEN	Control block not open.
FSP_ERR_INVALID_POINTER	p_connection_parameter is specified as NULL.
FSP_ERR_INVALID_ARGUMENT	The create connection parameter is out of range.

Properties

Prototype declarations are contained in rm_ble_abs.h.

Description

The initiator address type is Public Identity Address. The scan interval is 60ms and the scan window is 30ms in case of 1M PHY or 2M PHY. The scan interval is 180ms and the scan window is 90ms in case of coded PHY.

The Minimum CE Length and the Maximum CE Length are 0xFFFF. When the request for a connection has been received by the Controller, BLE_GAP_EVENT_CREATE_CONN_COMP event is notified.

When a link has been established, BLE_GAP_EVENT_CONN_IND event is notified. Implements ble_abs_api_t::createConnection.

Reentrant

No

Example

```
/* Create connection with remote device.*/  
err = RM_BLE_ABS_CreateConnection(&g_ble_abs0_ctrl, &g_connection_parameter);
```

Special Notes:

None

5. Demo Project

5.1 BLE DA1453x Demo Projects

5.1.1 Prerequisites

- Hardware requirements:
 - FPB-RL78G23-128p (RTK7RLG230CSN000BJ).
 - PC running Windows® 11.
 - Micro-USB cable for Power supply (included as part of the kit. See RL78/G23-128p Fast Prototyping Board – User’s Manual at “Related Documents” on page 1)
 - [US159-DA14531EVZ BLE Pmod](#)
- Software requirements for Windows 11 PC:
 - e² studio 2025-10.0 (25.10.0) or later.
 - Compiler: Renesas Electronics C Compiler for RL78 Family V1.15.01.
 - [QE for BLE Tool](#) version 1.8.0 or later.



Figure 5.1 iOS Renesas GATT Browser



Figure 5.2 Android Renesas GATT Browser

5.1.2 Import the Demo Project

Users can import the demo project by adding the demo to their e² studio workspace (see section 5.4 Creating a New BLE DA1453x project) or by downloading the demo project (5.5 Adding a Demo to a Workspace).

Peripheral mode

- Import “rl78_da1453x_ble_baremetal” for Bare metal application.
- Import “rl78_da1453x_ble_freertos” for FreeRTOS application.

Central mode

It is recommended to create a separate workspace in e² studio (e.g., central_workspace) to avoid project conflicts.

- Import “rl78_da1453x_ble_baremetal” for Bare metal application.
- Import “rl78_da1453x_ble_freertos” for FreeRTOS application.

5.1.3 Hardware Setup

- Connect the DA14535 PMOD module to the FPB RL78/G23-128p PMOD1 connector.
- Connect the micro-USB cable from PC to the FPB RL78/G23-128p micro-USB connector (J12).
- Connect the USB to TTL cable from PC to the FPB RL78/G23-128p micro-USB connector (UART2)
 - Establish the UART2 connection on the FPB RL78/G23-128p board using the following pin mapping as table below:

Table 5.1 Pin mapping of UART2 connection with USB to TTL

USB-to-TTL	FPB RL78/G23-128p
TxD	Pin 19
RxD	Pin 18
GND	GND

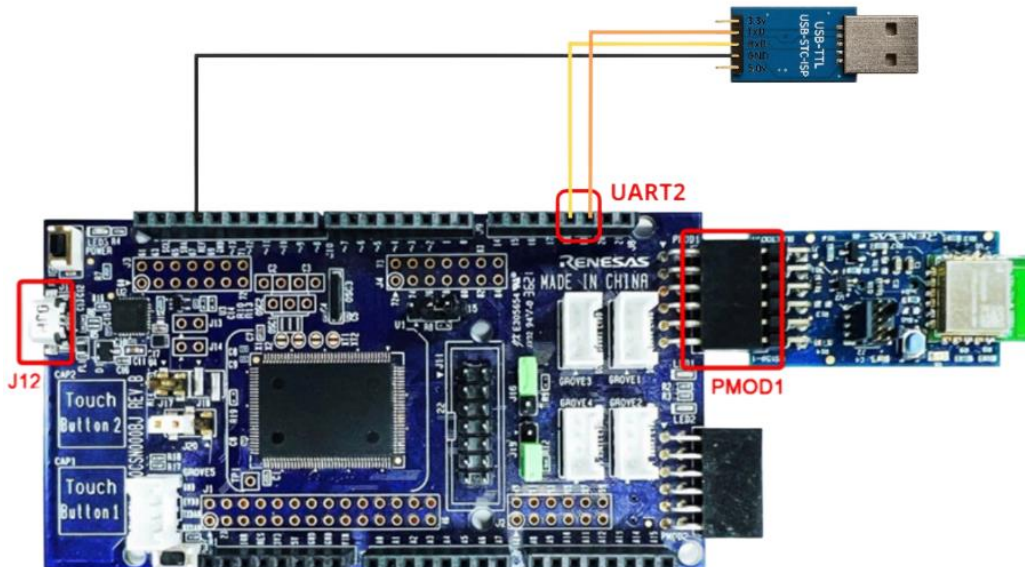


Figure 5.3 Operating Environment

5.1.4 Software Setup

a) Folder Structure

The following table lists the file structure of the Bare metal sample program.

Table 5.2 File Structure of the Bare Metal Sample Program

Folder name, file name	Explanation
rl78_da1453x_ble_baremetal	Project folder
intel	Library of cryptographic algorithms.
-qe_gen	Generated by QE tool
-src	Program storage folder
-ble_log	Handle print log UART to serial port with TTL device
-bsp_wrapper	BSP wrapper functions storage folder
-r_byteq	BYTEQ module storage folder
-r_config	BYTEQ, SCI configuration storage folder
-r_sci	SCI module storage folder
-RL78_G23	Essential configurations of Flash module
-smc_gen	Smart Configurator generator folder
-Config_PORT	
-Config_UART3	
-Config_UART2	
-general	
-r_ble_da1453x	
-r_bsp	
-r_config	
-r_pincfg	
-rl78_da1453x_ble_baremetal.c	Main processing source file

The following table lists the file structure of the FreeRTOS sample program.

Table 5.3 File Structure of the FreeRTOS Sample Program

Folder name, file name	Explanation
rl78_da1453x_ble_freertos	Project folder
intel	Library of cryptographic algorithms.
-qe_gen	Generated by QE tool
-src	Program storage folder
-ble_log	Handle print log UART to serial port with TTL device
-freertos_config	FreeRTOS packages
-FreeRTOS-Kernel	
-freertos_startup	
-fertos_skeleton	
-task_function.h	
-ble_main.c	BLE main thread
-bsp_wrapper	BSP wrapper functions storage folder
-r_byteq	BYTEQ module storage folder
-r_config	BYTEQ, SCI configuration storage folder
-r_sci	SCI module storage folder
-RL78_G23	Essential configurations of Flash module
-smc_gen	Smart Configurator generator folder
-Config_PORT	
-Config_UART3	
-Config_UART2	
-general	
-r_ble_da1453x	
-r_bsp	
-r_config	
-r_pincfg	

b) Project Settings

Open the Project Settings, go to Tool Settings -> Compiler -> Source, and make sure that all folders and directories have been added before building project.

c) Serial port settings

In section 5.1.3 Hardware Setup after connect the-USB to TTL from PC to FPB RL78/G23-128p UART2 then serial port COM number is enable and available to occurred in **Tera Term***

- Ensure the correct COM port is selected, and the baud rate is set to 115200.

Note(^{*)}: Install the Tera Term tool first if it is not already available on PC

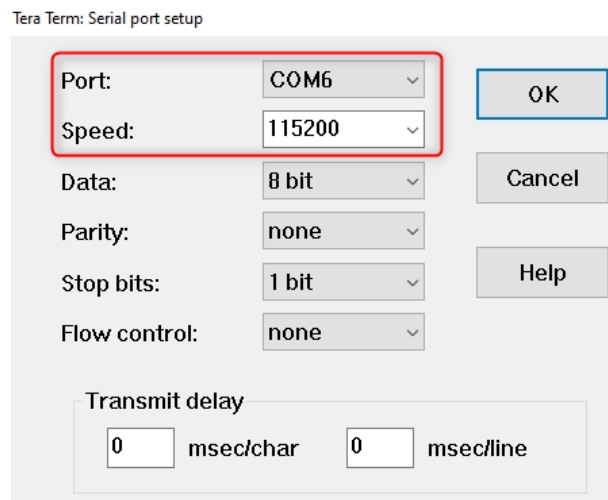


Figure 5.4 Tera Term Settings

d) Third-party Library Configurations

The sample application provided with this application note already includes the **intel** third-party library. If you create a new project from scratch and wish to use this library, follow the steps below:

1. Copy library files

From the sample application (*), copy the entire **intel/tinycrypt** folder into your project (recommended path: project_name/intel/).

After copying, open the **Properties** of the intel folder and **disable** “Exclude resource from build” to ensure all files are included in the build process.

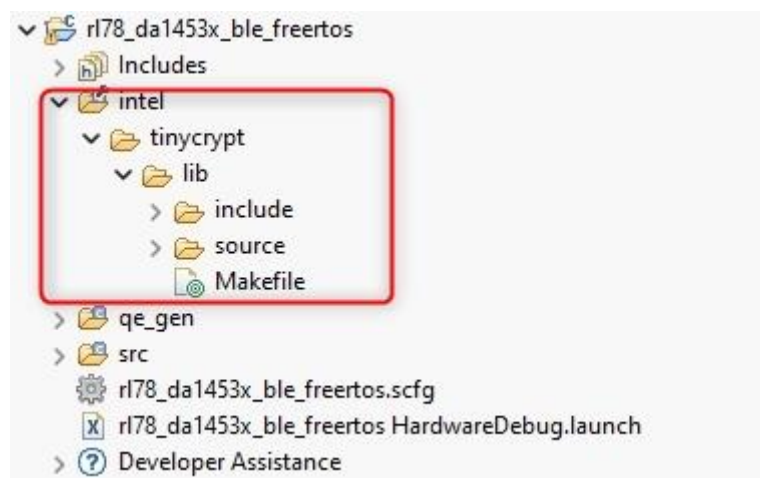


Figure 5.5 Copy intel folder into workspace

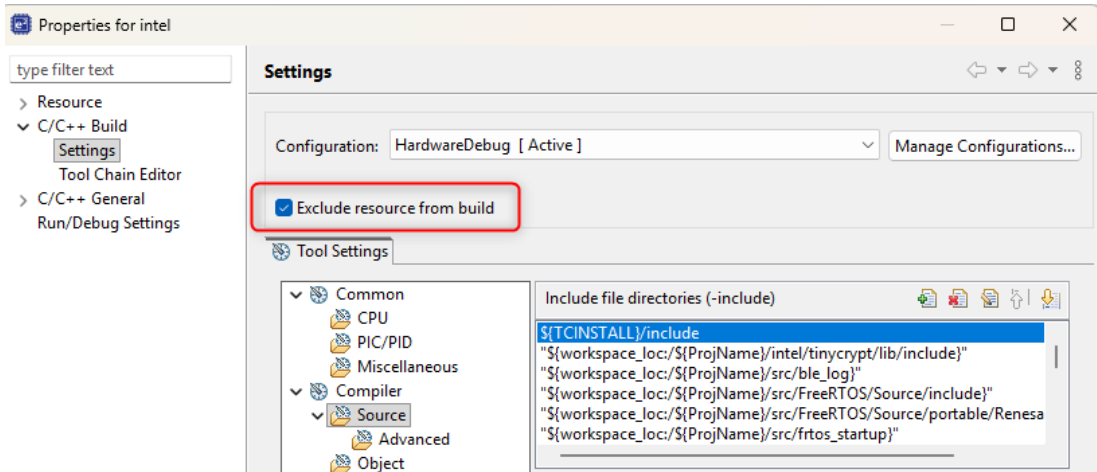


Figure 5.6 Uncheck the option to include the *intel* folder in the build

2. Add include paths

Add the copied folder to the project's compiler include paths.

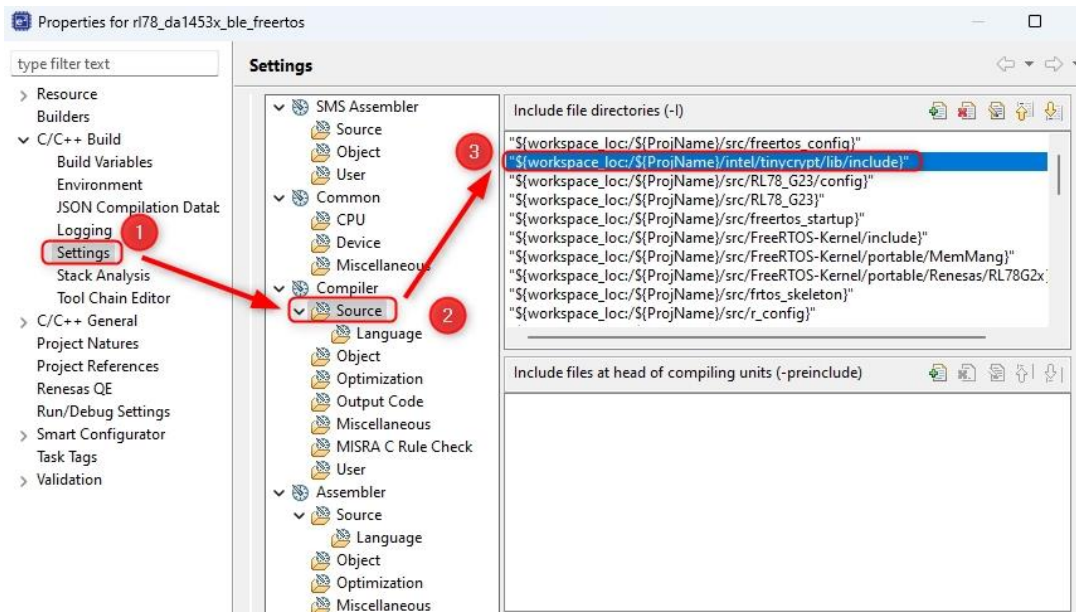


Figure 5.7 Add intel folder path into include paths

No additional configuration is required when using the provided sample application, as all settings have been pre-applied.

5.2 BLE DA1453x Peripheral Demo Projects

5.2.1 How to Run the Demo

- a) Select Device and PMOD Setting
 Use the Smart Configurator to configure
 Open the Smart Configurator as shown in the image below, select the appropriate device and PMOD

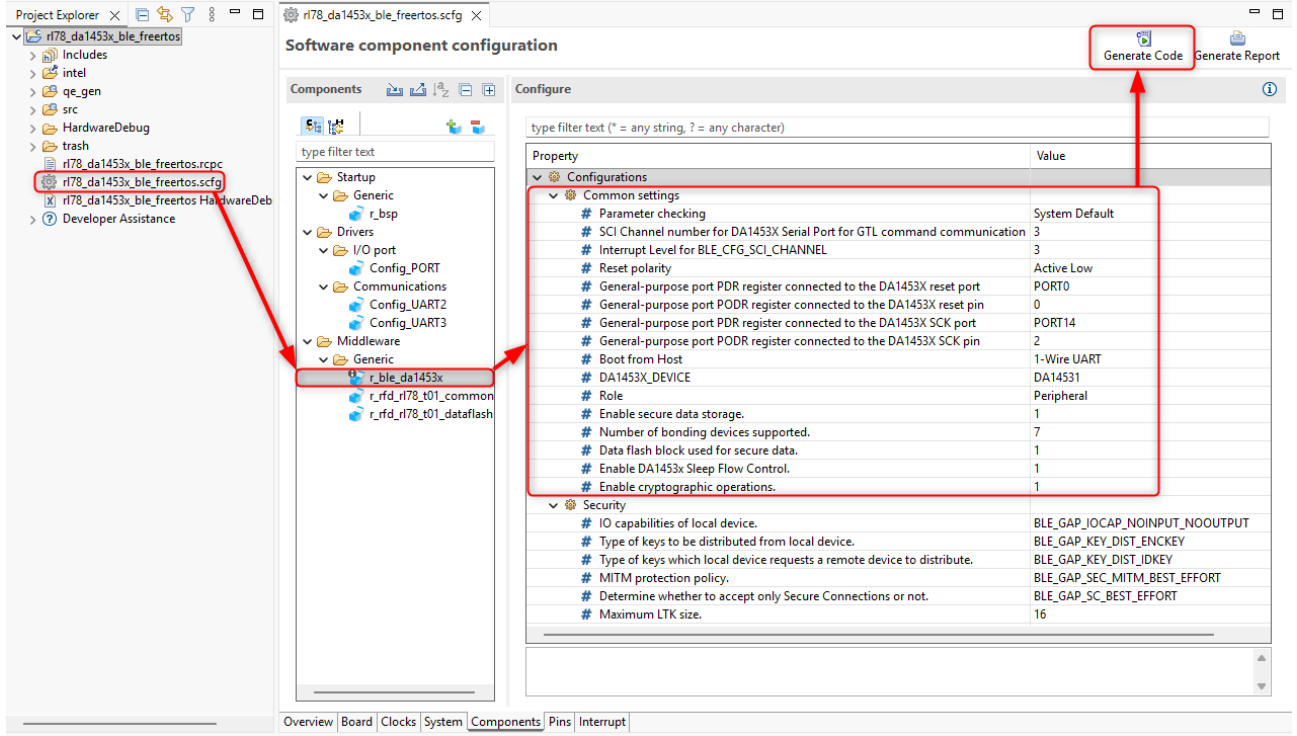


Figure 5.5 Device and PMOD Setting

- “DA1453X_DEVICE”: Allows to choose between two devices, DA14531 and DA14535.
- “BLE_CFG_HOST_BOOT_MODE”: The default for this macro is currently disabled. Please select “**1-wire UART**” if you want to run the demo with the DA14531/DA14535 device. In case you use “**2-wire UART**”, make sure that “**DA1453X_DEVICE**” is selected with the DA14535 device. Other cases are not supported at the moment.
- The PMOD pins are configured as shown in the table below:

Table 5.4 Configuration PMOD

	PMOD1	PMOD2
Reset port	0	-
Reset pin	0	-
SCK port	14	-
SCK pin	2	-

Note: PMOD2 is not use for BLE of RL78 board.

b) Pairing Settings

With the Pairing feature, it supports three connection methods as below:

- Just works functionality
- Passkey functionality
- Secure Connections

Pairing mode is configured through macros in Smart Configurator.

Common parameters

The following parameters are commonly applied across all pairing modes:

- Encryption Key Distribution – Configure the local device to distribute the encryption key (BLE_GAP_KEY_DIST_ENCKEY).
- Identity Key Request – Request the identity key from the remote device (BLE_GAP_KEY_DIST_IDKEY).
- MITM Protection – Enable Man-In-The-Middle (MITM) protection (BLE_GAP_SEC_MITM_STRICT).
- Maximum LTK Size – Set the maximum Long Term Key (LTK) size to 16.

Table 5.2 Common parameters

Parameters	Value
Local device key distribution	BLE_GAP_KEY_DIST_ENCKEY
Remote device key request	BLE_GAP_KEY_DIST_IDKEY
MITM protection policy	BLE_GAP_SEC_MITM_STRICT
Maximum Long Term Key size	16

Pairing mode configuration

Depending on the selected pairing method, the following parameters must be configured in Smart Configurator

Table 5.6 Pairing setting mode

Pairing Mode	IO capabilities of local device	Determine whether to accept only Secure Connections or not
Just Works	BLE_GAP_IOCAP_NOINPUT_NOOUTPUT	BLE_GAP_SC_BEST Effort
Passkey Entry	BLE_GAP_IOCAP_DISPLAY_YESNO	BLE_GAP_SC_BEST Effort
Secure Connections	BLE_GAP_IOCAP_DISPLAY_YESNO	BLE_GAP_SC_STRICT

After selecting the desired pairing mode, ensure that the macro is correctly referenced and linked in `smc_gen` → `r_ble_da1453x` → `src` → `hal_data.c` for proper functionality.

c) Building & Debugging the Demo Project

Refer to the 2.13.1 *Getting Started Guide* or following: https://lpcss-docs.renesas.com/DA1453x-RL78_BLE_Framework/application/application.html#building-and-running-the-application.

d) Connect to the application from Renesas GATT Browser

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 device name configured in “Peripheral > Local Name” through the QE tool introduced in guide 2.13.1 *Getting Started Guide*.

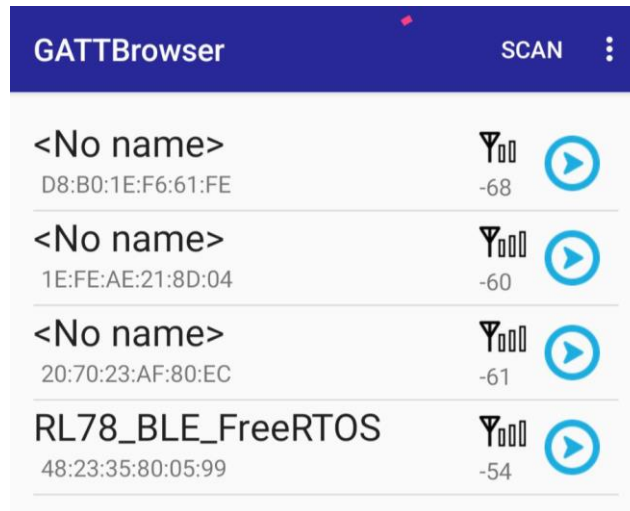


Figure 5.6 Determine the Device Name

- When connected, it stops advertising.

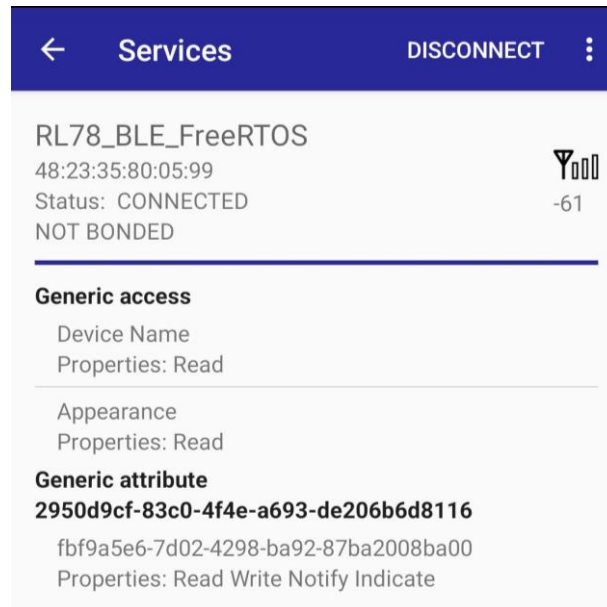


Figure 5.7 Connect Device

- Tera Term will print logs by writing a number (0x00 to 0xFF) to the characteristic
- When the notification button is enabled the value will run from 0 to 10 to simulate sending data from the MCU and will be displayed on the app interface . Furthermore, the Read button allows users to easily check the current value status.
- When disconnected, it restarts advertising.

The GAP Service for Legacy Pairing works as below.

- After the remote device successfully connects to GATT, click on the three dots in the top-left corner of the GATT browser app and select “**Create bond**” to proceed with pairing.

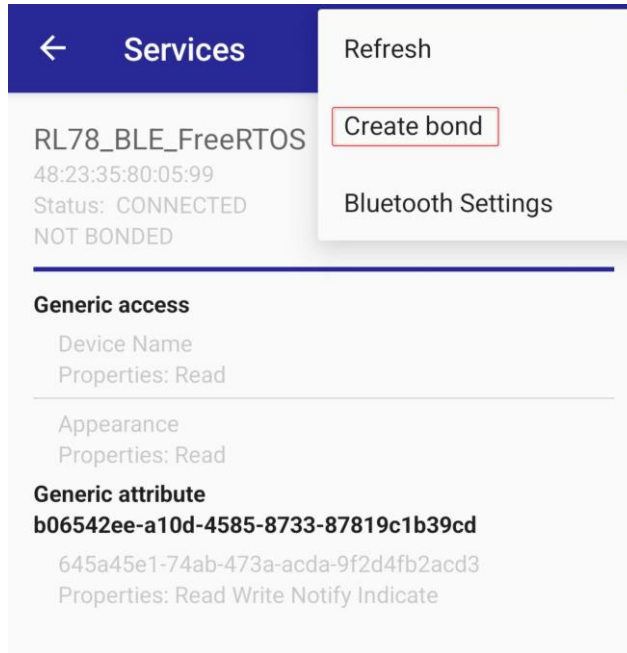


Figure 5.8 Start Pairing

- When clicking on “Create Bond”, a notification appears to pair with the device.

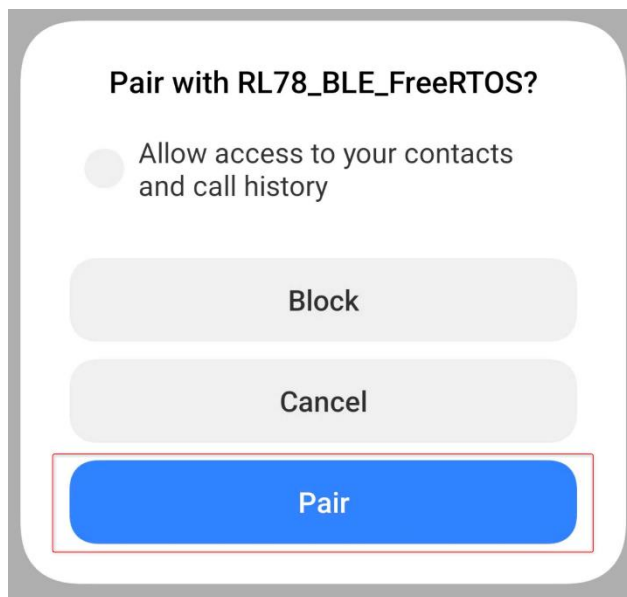


Figure 5.9 Legacy Pairing with Just Works mode

In Passkey mode, the default password is "123456".

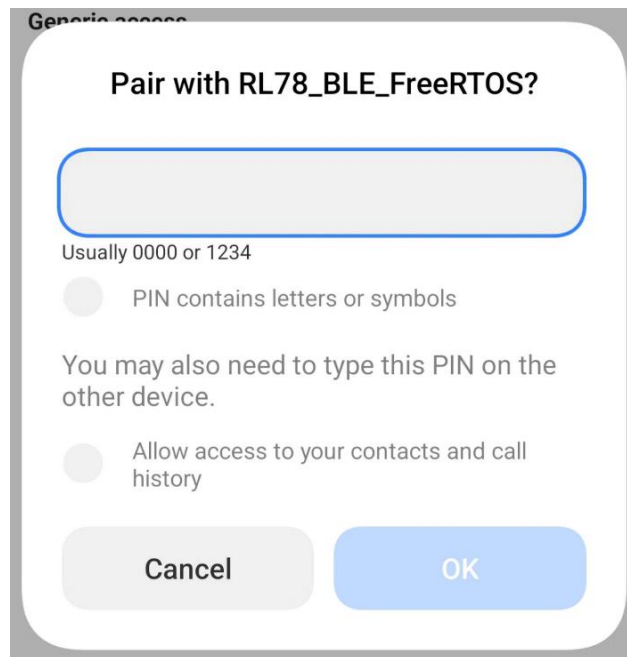


Figure 5.10 Legacy Pairing with Paskey mode

- In Secure Connections mode, the password is automatically displayed on the screen. It's stored in the MCU and compared using the LTK key. If the password matches, pairing is successful.

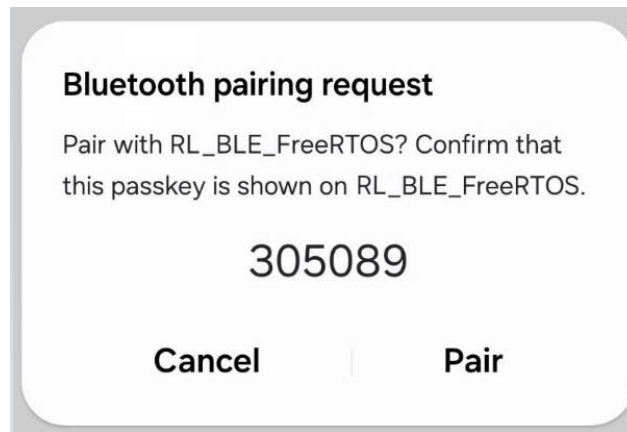


Figure 5.11 Pairing with Secure Connections mode

- After bonding is successfully completed, Security Establishment will be automatically triggered when the remote device disconnects from GATT and reconnects.
- The main role of Security Establishment is to ensure that the encrypted link between previously paired devices is securely re-established without the need for pairing again.

5.3 BLE DA1453x Central Demo Projects

5.3.1 How to run the demo

- a) Select device and PMOD setting

Use the Smart Configurator to configure

Open the Smart Configurator as shown in the image below, select the appropriate device and PMOD.

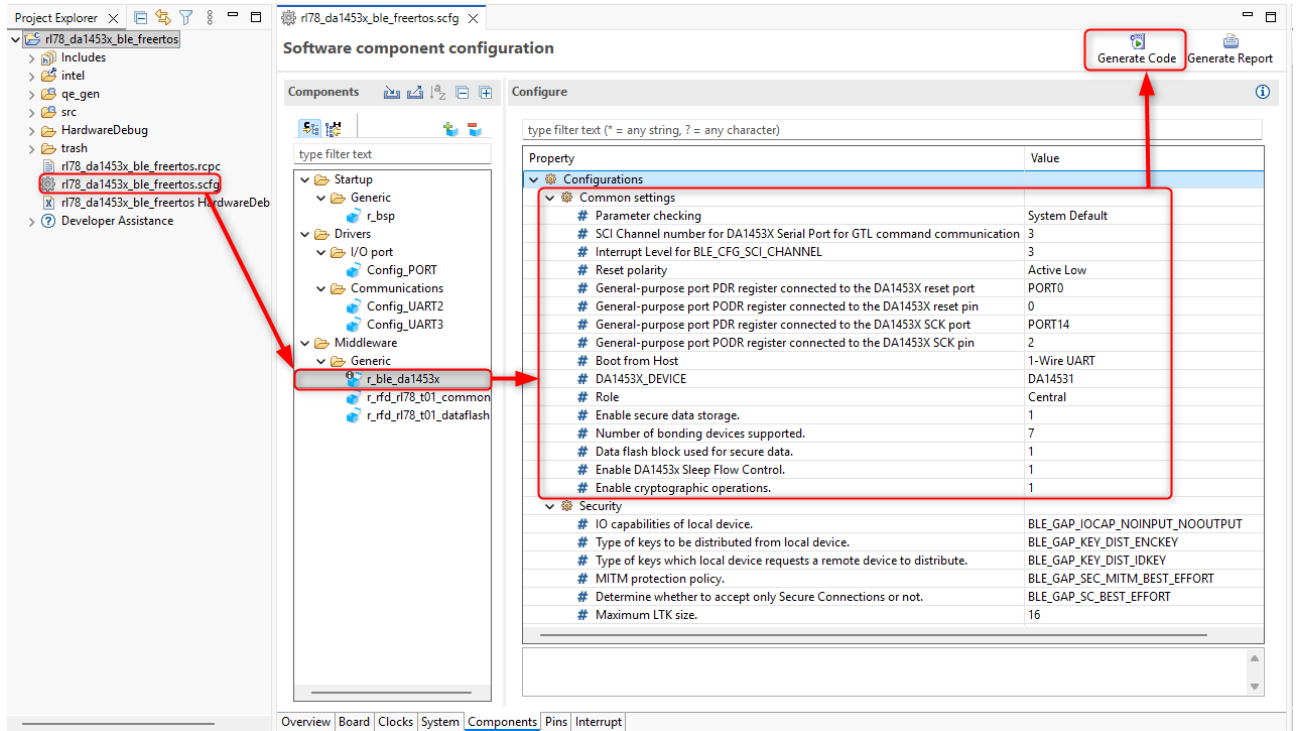


Figure 5.13 Device and PMOD settings

The configuration is the same as for the Peripheral role, except that the **Role** must be set to **Central**.

- b) Identify the peripheral device MAC address

Before running the Central application, the MAC address of the target Peripheral device must be identified.

Users can obtain this address using any standard BLE scanning tool. For example, the **Renesas GATT Browser** can be used to discover nearby devices and display their MAC addresses.

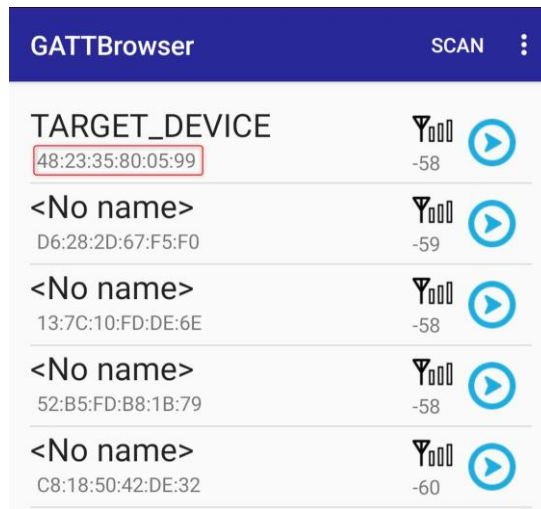


Figure 5.14 Identification MAC address

Convert the MAC address to decimal format and manually add it to *qe_gen* → *ble* → *app_main.c* following the little-endian rule.

c) Building & Debugging the Demo Project

Refer to the section “4.5. Building and running the application” at https://lpccsdocs.renesas.com/DA1453x-RL78_BLE_Framework/index.html

d) Scan and connect target device.

Before starting the Central application, ensure that the Peripheral device is actively advertising and that its MAC address has been correctly configured in *app_main.c*.

When the Central application is executed, it will perform a scan of all nearby Peripheral devices. During this process, the MAC addresses and advertising names of detected devices will be displayed in Tera Term.

Once the specified target device is identified, the Central will automatically stop scanning and establish a connection.

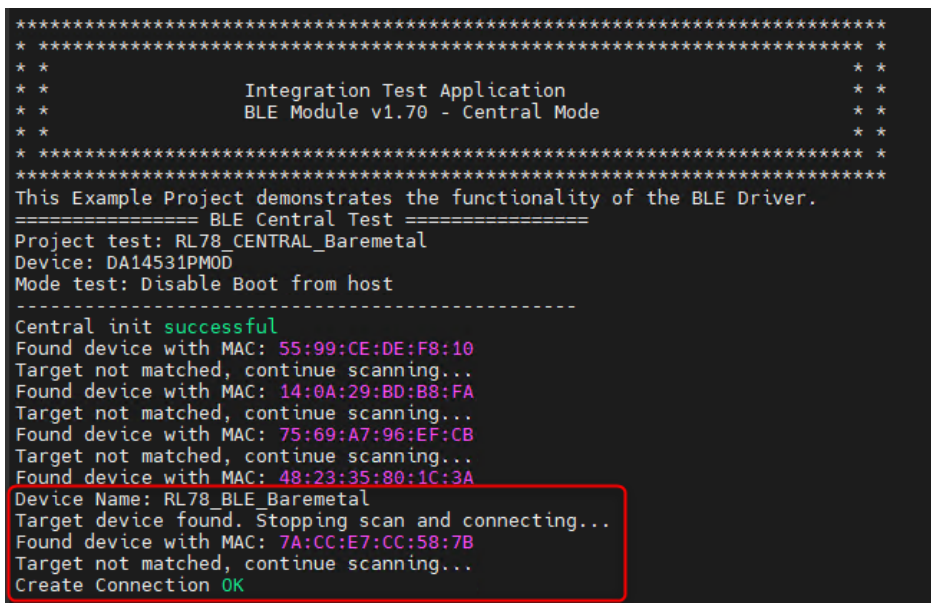


Figure 5.15 Target device found and established connection

5.4 Creating a New BLE DA1453x project

Refer to “*Getting Started Guide*” from section 2.13.1 Getting Started Guide

5.5 Adding a Demo to a Workspace

Demo projects are found in the sample_code subdirectory of the distribution file for this application note. To add a demo project to a workspace, select File >> Import >> General >> Existing Projects into Workspace, then click “Next”. From the Import Projects dialog, choose the “Select archive file” radio button. “Browse” to the sample_code subdirectory, select the desired demo zip file, then click “Finish”.

5.6 Downloading Demo Projects

When using the demo project, the SIS module needs to be downloaded. To download the SIS module, right click on this application note and select “Sample Code (download)” from the context menu in the Smart Browser >> Application Notes tab.

6. Appendix

6.1 Confirmed Operation Environment

This section describes confirmed operation environment for the SIS module.

Table 6.1 Confirmed Operation Environment (Ver. 1.00)

Item	Contents
Integrated development environment	Renesas Electronics e ² studio 2023.01
C compiler	Renesas Electronics C/C++ Compiler for RL78 Family V1.08.00
	Compiler option: The following option is 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	FPB-RL78G23-128p (RTK7RLG230CSN000BJ)

Table 6.2 Confirmed Operation Environment (Ver. 1.20)

Item	Contents
Integrated development environment	Renesas Electronics e ² studio 2023.07
C compiler	Renesas Electronics C/C++ Compiler for RL78 Family V1.12.01
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Little endian
Revision of the module	Rev.1.20
Board used	FPB-RL78G23-128p (RTK7RLG230CSN000BJ)

Table 6.3 Confirmed Operation Environment (Ver. 1.30)

Item	Contents
Integrated development environment	Renesas Electronics e ² studio 2024.04
C compiler	Renesas Electronics C Compiler for RL78 Family V1.13.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Little endian
Revision of the module	Rev.1.30
Board used	FPB-RL78G23-128p (RTK7RLG230CSN000BJ)

Table 6.4 Confirmed Operation Environment (Ver. 1.40)

Item	Contents
Integrated development environment	Renesas Electronics e ² studio 2024.10
C compiler	Renesas Electronics C Compiler for RL78 Family V1.13.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Little endian
Revision of the module	Rev.1.40
Board used	FPB-RL78G23-128p (RTK7RLG230CSN000BJ)

Table 6.5 Confirmed Operation Environment (Ver. 1.50)

Item	Contents
Integrated development environment	Renesas Electronics e ² studio 2025.01
C compiler	Renesas Electronics C Compiler for RL78 Family V1.15.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Little endian
Revision of the module	Rev.1.50
Renesas QE for BLE (R_BLE Custom profile)	Rev.1.7.0
Board used	FPB-RL78G23-128p (RTK7RLG230CSN000BJ)

Table 6.5 Confirmed Operation Environment (Ver. 1.60)

Item	Contents
Integrated development environment	Renesas Electronics e ² studio 2025.07
C compiler	Renesas Electronics C Compiler for RL78 Family V1.15.01
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Little endian
Revision of the module	Rev.1.60
Renesas QE for BLE (R_BLE Custom profile)	Rev.1.8.0
Board used	FPB-RL78G23-128p (RTK7RLG230CSN000BJ)

Table 6.5 Confirmed Operation Environment (Ver. 1.70)

Item	Contents
Integrated development environment	Renesas Electronics e ² studio 2025.10
C compiler	Renesas Electronics C Compiler for RL78 Family V1.15.01
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Little endian
Revision of the module	Rev.1.70
Renesas QE for BLE (R_BLE Custom profile)	Rev.1.8.0
Board used	FPB-RL78G23-128p (RTK7RLG230CSN000BJ)

6.2 How to change UART module to work with BLE module

This section describes how to change the UART module to work with BLE module in a demo project.

- a) Adding new UART module for communication between MCU and BLE module.
After creating new UART module, the structure is as below (UART2 and UART3 are used in this example, same for others):

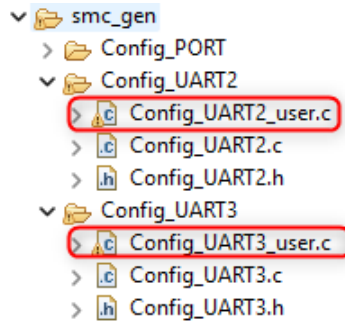


Figure 6.1 Config UART3, UART2

To prevent the error “E0562142: Interrupt table address of “.vect” has multiple definition”. Wrap UART2 and UART3 interrupt vector definitions with conditional macros (e.g., #ifdef) to avoid multiple vector assignments:

```

  /* Start user code for include. Do not edit comment generated here */
  #ifdef DELETE
  /* End user code. Do not edit comment generated here */

  Pragma directive
  #pragma interrupt r_Config_UART3_interrupt_send(vect=INTST3)
  #pragma interrupt r_Config_UART3_interrupt_receive(vect=INTSR3)
  #pragma interrupt r_Config_UART3_interrupt_error(vect=INTSRE3)
  /* Start user code for pragma. Do not edit comment generated here */
  #endif
  /* End user code. Do not edit comment generated here */

  /* Start user code for include. Do not edit comment generated here */
  #ifdef DELETE
  /* End user code. Do not edit comment generated here */

  Pragma directive
  #pragma interrupt r_Config_UART2_interrupt_send(vect=INTST2)
  #pragma interrupt r_Config_UART2_interrupt_receive(vect=INTSR2)
  #pragma interrupt r_Config_UART2_interrupt_error(vect=INTSRE2)
  /* Start user code for pragma. Do not edit comment generated here */
  #endif
  /* End user code. Do not edit comment generated here */
  
```

- b) Rebuild the project.

7. 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

RL78 Family's C Compiler CC-RL User's Manual (R20UT3123)

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

Revision History

Rev.	Date	Revision History	
		Page	Summary
1.00	June 30, 2023	-	First edition issued
1.10	Sep 18, 2023	5	Add AzureRTOS
		11	Table 1.1 API functions
		11	Update Table 2.1
		19-81	Update description of API functions
		85-93	Add Sample Code Generation using QE for BLE
		94	Update Revision of Table 5.1
1.20	Feb 23, 2024	-	Update document format
		1	Update document information
		5	Update Figure 1-1 to update the connection with BLE DA14531 module
		6	Remove AzureRTOS
		6	Update description of RTOS in Section 1.2.2
		7	Add 1.3 Features
		7, 26	Add R_BLE_GetVersion()
		10	Add 1.5 Status Transitions
		11	Add 1.6 Usage Notes
		12	Remove AzureRTOS in 2.2 Software Requirements
		13	Update Table 2.1
		14	Update descriptions in Table 2.3
		15	Update Table Memory Usage
		19-20	Add new parameters about UART boot protocol message types
		21	Rename 2.11 Adding the SIS Module to Your Project
		87	Update the target board in 5. Sample Code Generation Using QE for BLE
		96-100	Update source code in Sample app
		101	Add 6.1 Limitations
		102	Add 6.2 How to change UART module to work with BLE module
		103	Update Table 6.1: <ul style="list-style-type: none"> • Change name of the Board used • Update Endian order Add Table 6.2
104	Updated User's Manual: Development Tools		
1.30	Sep 30, 2024	-	Update document format
		1	Top page: Update related documents with RL board manual.
		5	Section 1.2.1 Update diagram.
		6	Section 1.2.2 Update Image & add description.
		7	Section 1.3 Update new feature for DA14535.
		10	Section 1.4 Add new function & description for R_BLE_VS_SetTxPower() & R_BLE_VS_GetTxPower()
		15	Section 2.8 Update description for note
		20	Section 2.10 Add new macro of GTL Auxiliary Command ID's
		24	Add section 2.12 "for", "while" and "do while"
		25	Section 2.13 Update Usage Notes
		25-26	Section 2 Add new section 2.13.1, 2.13.2, 2.13.3, 2.13.4, 2.13.5
		86-87	Section 3.54 Add new function & description for R_BLE_VS_SetTxPower()
		88	Section 3.55 Add new function & description for R_BLE_VS_GetTxPower()
		95-102	Update section 5 Demo Project
103	Section 6.1 Add new table for latest version (v1.30)		

1.40	Nov 21, 2024	-	Update document format
		7	Section 1.3 Update new features for Legacy Pairing
		8	Section 1.4 Update new function support Legacy Pairing
		13	Section 2.7 Update Table 2.1 Configuration Options (r_ble_da14531_config.h)
		14	Section 2.8 Update Module revision & memory usage
		20, 22	Section 2.10 Update new macro Mutex give/take defines, Defines for host DB
		46-52	Section 3. API function: Add new function 3.20. R_BLE_GAP_SetPairingParams() 3.21. R_BLE_GAP_StartPairing() 3.22. R_BLE_GAP_ReplyPairing() 3.23. R_BLE_GAP_ReplyPasskeyEntry() 3.24. R_BLE_GAP_ReplyExKeyInfoReq() 3.25. R_BLE_GAP_ReplyLtkReq()
		103	Section 5.1.4 Update format structure software setup
		106 108-109	Section 5.1.5 Update Legacy Pairing Settings
		112	Section 6.1 Add table 6.4 Confirmed Operation Environment (Ver. 1.40)
1.50	July 07, 2025	-	Features update: <ul style="list-style-type: none"> Secure Connections with Numeric Comparison Update Read Indication for GATT Server
		-	Update document format
		1	Renamed "DA14531 BLE SIS Module" to "DA1453x BLE SIS Module" and added support for DA14535MOD (US159-DA14535EVZ).
		5-6	Section 1.2 Update pictures and overview for SIS module
		7	Section 1.3 Update features
		8-10	Section 1.4 Add new APIs <ul style="list-style-type: none"> R_BLE_GAP_ReplyNumComp() RM_BLE_ABS_DeleteBondInformation()
		13	Section 2.7 Compile settings. Update configuration options
		14	Section 2.8 Update memory sizes
		23	Section 2.10 Parameter Add parameter typedef enum e_r_ble_gtl_aux_operation
		27-93	Section 3 Update format and add APIs 3.24. R_BLE_GAP_ReplyNumComp()
		97	Section 4 Add new API 4.4. RM_BLE_ABS_DeleteBondInformation()
		99	Section 5.1.3 Update hardware setup
		100	Section 5.1.4 Update software setup
		102-105	Section 5.1.5 How to run Demo: Update guide to run demo a) Update device and PMOD settings b) Delete QE Custom profile setting. d) Update new interface of "Renesas GATT Browser".
		109	Section 6.2 Update interrupt vector and add UART 2
1.60	Sep. 29, 2025	-	Update document format
		-	Delete L2CAP related APIs
		-	Add Central demo projects
		7	Features update: <ul style="list-style-type: none"> Central All roles Scan Support security for both roles (Central – Peripheral) Enable security configuration
		8-10	Section 1.4 API Overview – Added - R_BLE_GAP_StartScan()

			<ul style="list-style-type: none"> - R_BLE_GAP_StopScan() - R_BLE_GAP_CreateConn() - R_BLE_GAP_CancelCreateConn() - RM_BLE_ABS_StartScanning() - RM_BLE_ABS_CreateConnection()
		14	Section 2.8 Update codesize
		18-24	Section 2.10 Update parameter
		26	Section 2.13.4 Module Firmware Compability Update latest link
		26-27	Section 2.13.5 SIS Module Limitations Update limitations
		59-66	Section 3. API Function – Added new function <ul style="list-style-type: none"> - R_BLE_GAP_StartScan() - R_BLE_GAP_StopScan() - R_BLE_GAP_CreateConn() - R_BLE_GAP_CancelCreateConn() - R_BLE_GAP_ConfWhiteList
		114-115	Section 4. Abstraction API – Added new funtion <ul style="list-style-type: none"> - RM_BLE_ABS_StartScanning() - RM_BLE_ABS_CreateConnection()
		120-124	Section 5.2 BLE DA1453x Peripheral Demo Projects Update details for peripheral project
		125-126	Section 5.3 BLE DA1453x Central Demo Projects Add central demo project
		129	Section 6.1 Confirmed operation environments Add confirmed operation environment for module version 1.60
1.70	Jan. 21,2026	-	Update document format
		1	Added related documents
		7	Section 1.2.2 Software configurations <ul style="list-style-type: none"> - Update software diagram - Add driver informations
		8-9	Section 1.3 Features Update new security functionality
		10	Section 1.4 API Overview – Add new API R_BLE_GAP_StartEnc()
		14	Section 2.2 Software requirements Add drivers dependency
		15	Section 2.7 Compile settings Add Configuration options
		17	Section 2.8 Code size Update code size
		23	Section 2.10 Parameter Add parameters
		29-30	Section 2.13.5 SIS Module Limitations Update module limitations
		70	Section 3. API Function – Add new API R_BLE_GAP_StartEnc()
		120	Section 5.1.1 Prerequisites <ul style="list-style-type: none"> - Update e² studio version Update board information
		122-123	Section 5.1.4 Software Setup – Add new section

		125-129	Section 5.2 BLE DA1453x Peripheral Demo Projects - Update details for peripheral project
		130-131	Section 5.3 BLE DA1453x Central Demo Projects - Update details for central project
		135	Section 6.1 Confirmed operation environments - Add new table for version 1.70

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/.