RYZ012 and RA MCU

Firmware Update from BLE Radio OTA

Introduction

This document describes a sample application that updates the RYZ012 firmware via BLE Radio OTA.

The application example works in a configuration that uses the EK-RA4M2 board with RA4M2 as the host MCU and connects the PMOD™ Expansion Board for RYZ012 Bluetooth LE Module to the PMOD connector. The steps in this document show the user how to load an RYZ012 firmware file on a Mobile Device running the OTA application and connect over BLE to program the firmware into the RYZ012.

Target Devices

- RA4M2
- RYZ012

Related Documents

- Renesas Flexible Software Package (FSP) User's Manual (R11UM0155)
- e² studio 2023-04 User's Manual: Quick Start Guide MCU RA Family (R20UT4989)
- EK-RA4M2 Quick Start Guide (R20QS0018)
- RYZ012A1 PMOD Expansion Board Quick Start Guide (R21QS0002)
- RYZ012 Datasheet (R12DS0002)
- RYZ012 Bluetooth LE Sample Application (R01AN6116EJ)
- QE for BLE [RA,RE,RX] Release Note (R20UT5145EJ)

Required Resources

To build and run the RYZ012 firmware update application example, the following resources are needed.

Development tools and software

- e² studio IDE v2023-04
- Flexible Software Package (FSP) v4.4.0
- QE for BLE Tool [RA, RE, RX] V1.6.0 for e² studio IDE
- SEGGER J-Link RTT Viewer V7.60f
- TelinkBleOTA app v2.0.5 for iOS (Apple App Store)

Hardware

- EK-RA4M2 kit (RTK7EKA4M2S00001BE)
- PMOD Expansion Board for RYZ012x1 (RTKYZ012A1B00000BE)
  Must be programmed with v5.4 Firmware or higher to support OTA firmware updates
- PC running Windows® 10
- 1 x Micro USB cable

PMOD™ is registered to Digilent Inc.

Contents

1. Overview 4
1.1 Operating Environment 4
1.1.1 Hardware 4
1.1.2 Software 4
1.1.3 How to Assemble RYZ012 Module and EK-RA4M2 5
1.1.4 EK-RA4M2 USB Connection 6
1.1.5 Mobile Device OTA App Setup 6
1.2 RYZ012 Firmware Update Process 8
2. Firmware Update Application 9
2.1 Importing the Application Project 9
2.1.1 Specify e² studio Workspace 9
2.1.2 Import Project 9
2.1.3 Select Existing Project 10
2.1.4 Select Project 10
2.2 Project Build and Download 11
2.3 Application Project Operation 11
2.3.1 Launch J-Link RTT Viewer 11
2.3.2 Start the Application Execution 12
2.3.3 RTT Viewer Logs 14
2.4 RYZ012 Firmware Update Considerations 15
2.4.1 Firmware Update Issue #1 16
2.4.2 Firmware Update Issue #2 16
2.4.3 Initialize BLE Driver Failed 17
3. Application Software Architecture 18
3.1 Microcontroller Peripheral Functions 19
3.2 FSP Modules 20
4. Application Project Implementation 21
4.1 Entry Point 21
4.2 Main Loop 21
4.3 BLE Initialization Process 22
4.4 Register BLE Callback Functions 23
4.5 Get Firmware Version Function 24
4.6 SPP BLE OTA Notify Events 25
4.7 SPP BLE GAP Events 27
5. How to Make and Configure a New Project 27
5.1 Create a New Project 27
5.2 BSP Heap and Stack Configuration 27
5.3 Add the I/O Port Stack 28
5.4 Add and Configure the SPP BLE Module 28

5.4.1 SPP BLE Abstraction Driver Configuration 29
5.4.2 Configure Peripherals for SPP BLE Abstraction Driver 30
5.5 Add General PWM Timers 34
5.6 Add RYZ012 BLE OTA Service Profile 35
5.7 Build and Modify Application Skeleton Code 38
6. Next Steps 38
Appendix - Windows PC OTA App 39
Revision History 43

1. Overview

The RYZ012 BLE module is a highly integrated wireless communication module that provides a pre-certified solution for Bluetooth® 5.0 Low Energy (LE). The module is available in two configurations (RYZ012A1 and RYZ012B1) with or without a mounted antenna. Supported by the RA MCU family's Flexible Software Package (FSP) and the QE for BLE tool, customers can focus on application development without dealing with the details of Bluetooth LE.

Once an RYZ012 module is designed into an end customer application, designers will need to be able to update the firmware to adapt to changing conditions or requirements for the end systems. This update could be done via the host MCU or through a BLE radio OTA update. This application note covers an example of updating the RYZ012 module via BLE radio OTA including details of handling issues encountered when updating the BLE module.

The application project uses an EK-RA4M2 board connected with an RYZ012 module. Communication between the EK-RA4M2 and RYZ012 module is based on a command system called Serial Port Profile (here in after referred to as SPP).

Please check the *SPP Bluetooth Low Energy Abstraction with RYZ012 (rm_ble_abs_spp)* part of the *RA Flexible Software Package User's Manual* (R11UM0155) for more information on the APIs and callback event limitations.

1.1 Operating Environment

1.1.1 Hardware

The hardware requirements used in the sample application are shown in the following table.

Table 1. Hardware Requirements

Hardware	Description
EK-RA4M2	RTK7EKA4M2S00001BE
PMOD Expansion Board for RYZ012x1	RTKYZ012A1B00000BE
	Programmed with SPP SDK v5.4 Firmware to support MCU based firmware updates
Windows® 10 PC	---
1 x Micro USB Cable	EK-RA4M2 USB Debug J10 connecter (micro-B)

1.1.2 Software

The software requirements used in the sample application are shown in the following table.

Table 2. Software Requirements

Software	Version
e² studio IDE	2023-04
GCC Compiler	10.3.1
Renesas FSP	4.4.0
QE for BLE [RA,RE,RX]	V1.6.0 for e² studio IDE
SEGGER J-Link RTT Viewer	V7.60f
TelinkBleOTA app	v2.0.5 for iOS

1.1.3 How to Assemble RYZ012 Module and EK-RA4M2

This section describes how to assemble RYZ012 PMOD module and EK-RA4M2. The RYZ012 PMOD module and EK-RA4M2 are connected by one of 2 x 6 PMOD connectors. In this application note, the PMOD connector must be mounted as:

RYZ012 PMOD : CN1 > EK-RA4M2 : J26 PMOD1 (SPI / UART)
Connect CN1 of RYZ012 PMOD and J26 PMOD 1 Connector of EK-RA4M2.

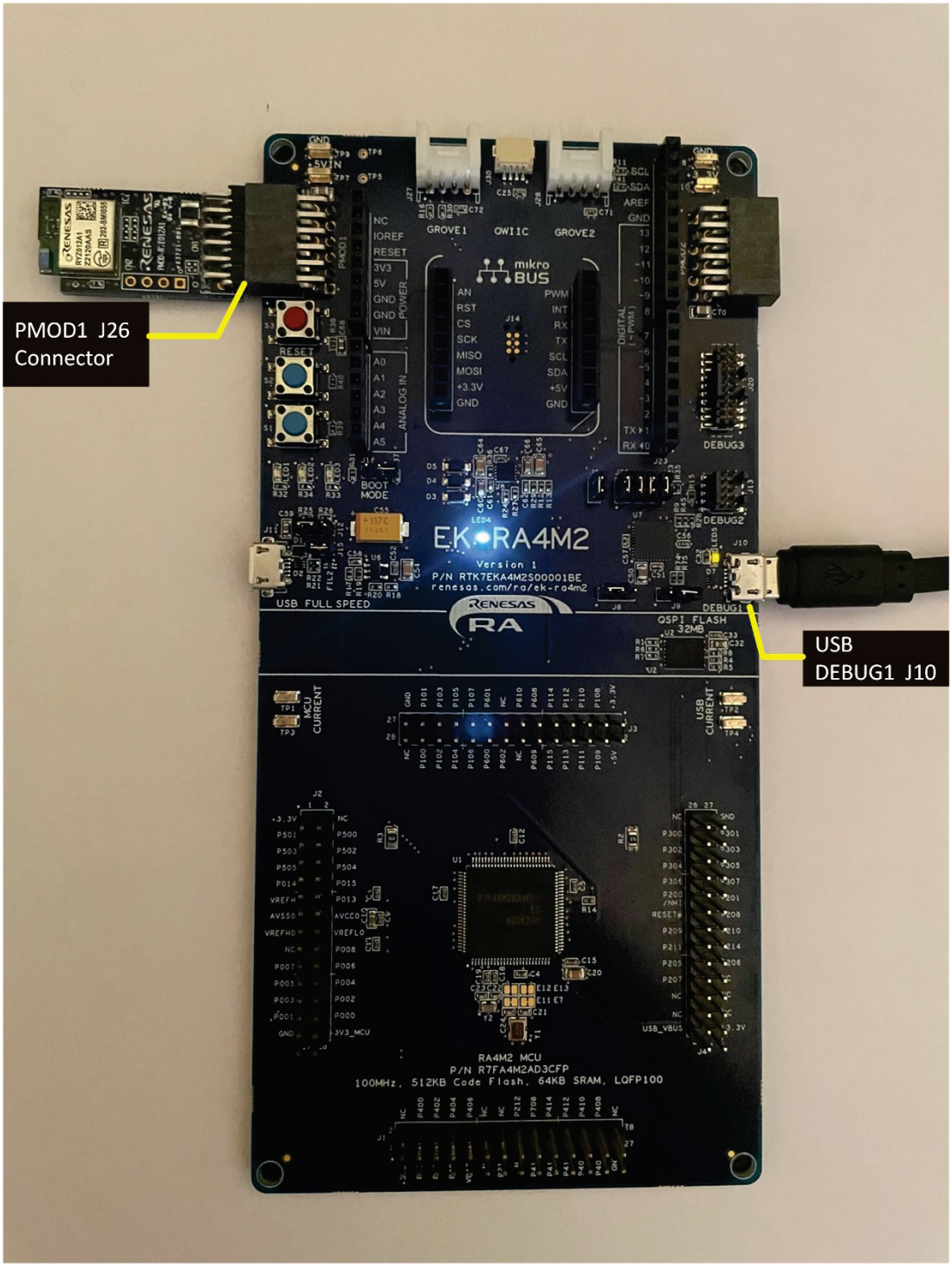


Figure 1. EK-RA4M2 and RYZ012 PMOD Assembly

1.1.4 EK-RA4M2 USB Connection

The USB Debug1 interface is used to communicate logging information with RTT Viewer.

Connect one micro-B USB cable to DEBUG1 (J10) with the other cable end connected to a PC port. See Figure 1.

1.1.5 Mobile Device OTA App Setup

In this app note, the firmware OTA Upgrade is demonstrated with an OTA App running on iOS devices. A basic overview of the mobile app install, and setup is provided here but is not supported by Renesas Support. Please contact Apple or the **TelinkBleOTA** app developer for support with mobile device or software installation issues. Make sure Bluetooth is enabled on the mobile device.

- Install the mobile app **TelinkBleOTA** from the iOS App Store. In the App Store, search for “Telink” to get a list of apps available and install the **TelinkBleOTA** app. See Figure 2.
- Locate the RYZ012 firmware bin files provided in the e^2 studio project workspace directory **bleapp_fwupdate_ryz012_ra4m2_baremetal/8258_moduleV5_4.bin** and
- **8258_moduleV5_5.bin**
- Transfer the RYZ012 firmware files by email or other method that can be accessed by the mobile device and store to app folder Locations->On My iPhone->TelinkBleOTA. The folder is created by the app during installation for newer versions (v16.3.1 or later) of iOS. If the folder is not created, then manually create the folder TelinkBleOTA on the mobile device. See Figure 3.

After the OTA App is installed on the iOS Device, skip to section 1.2 **RYZ012 Firmware Update Process** to learn about the OTA upgrade process and how to build and program the application project.

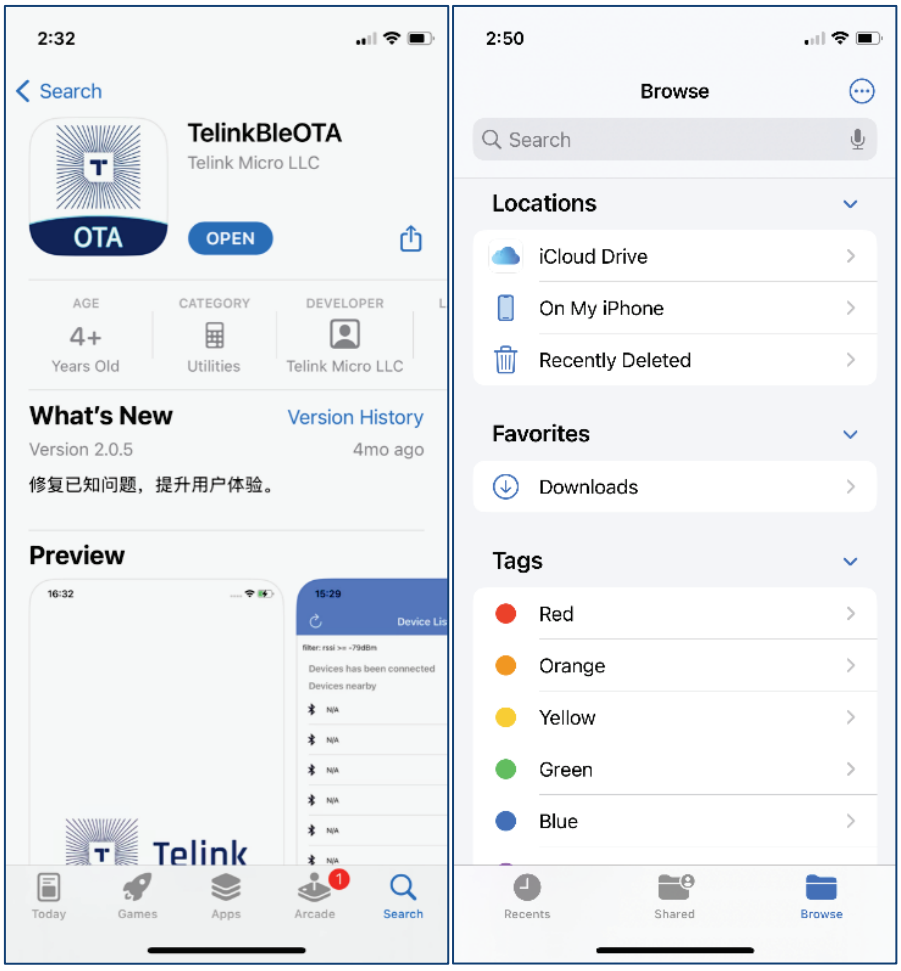


Figure 2. Install App and Load the RYZ012 Firmware Files

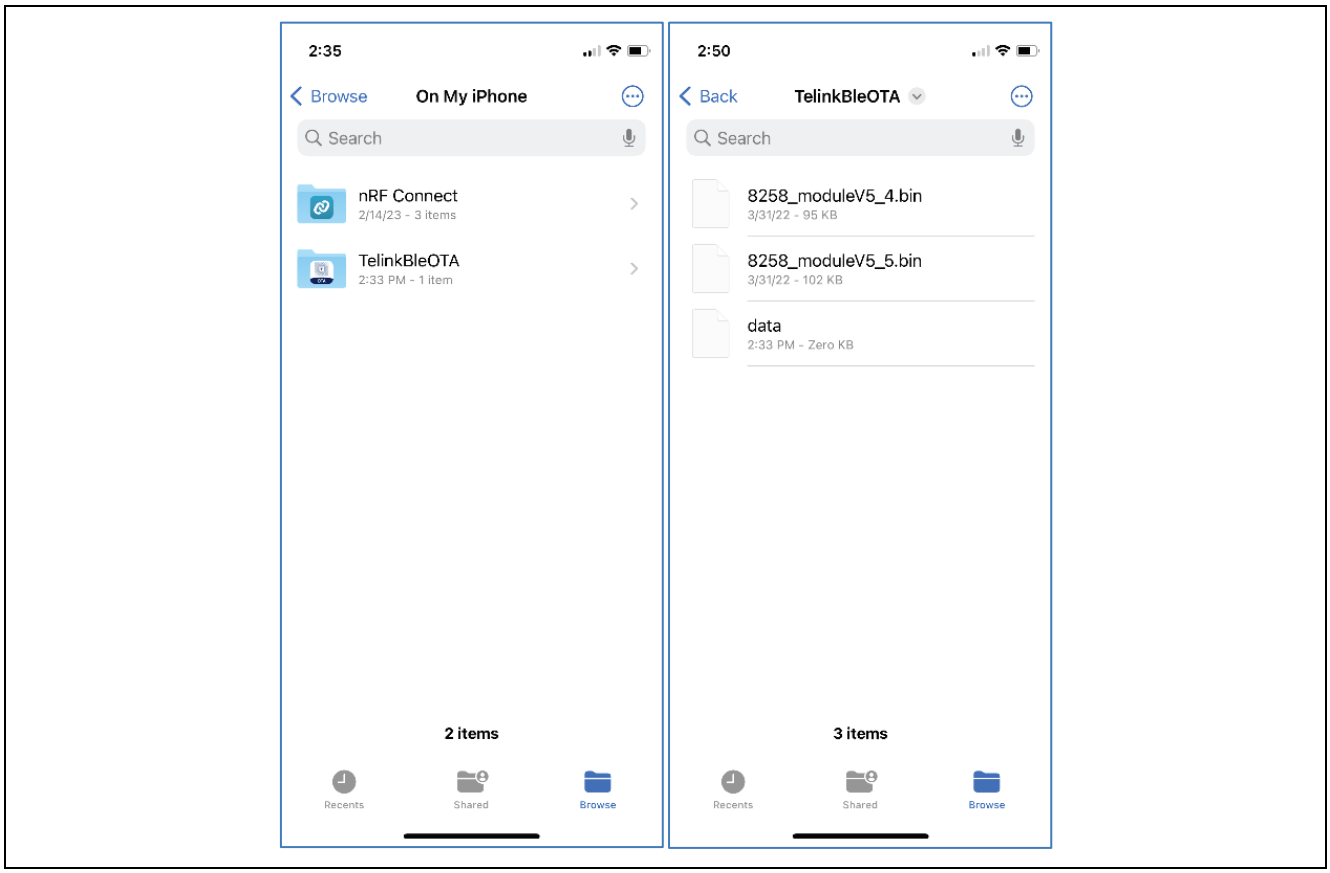


Figure 3. Store the RYZ012 Firmware Files

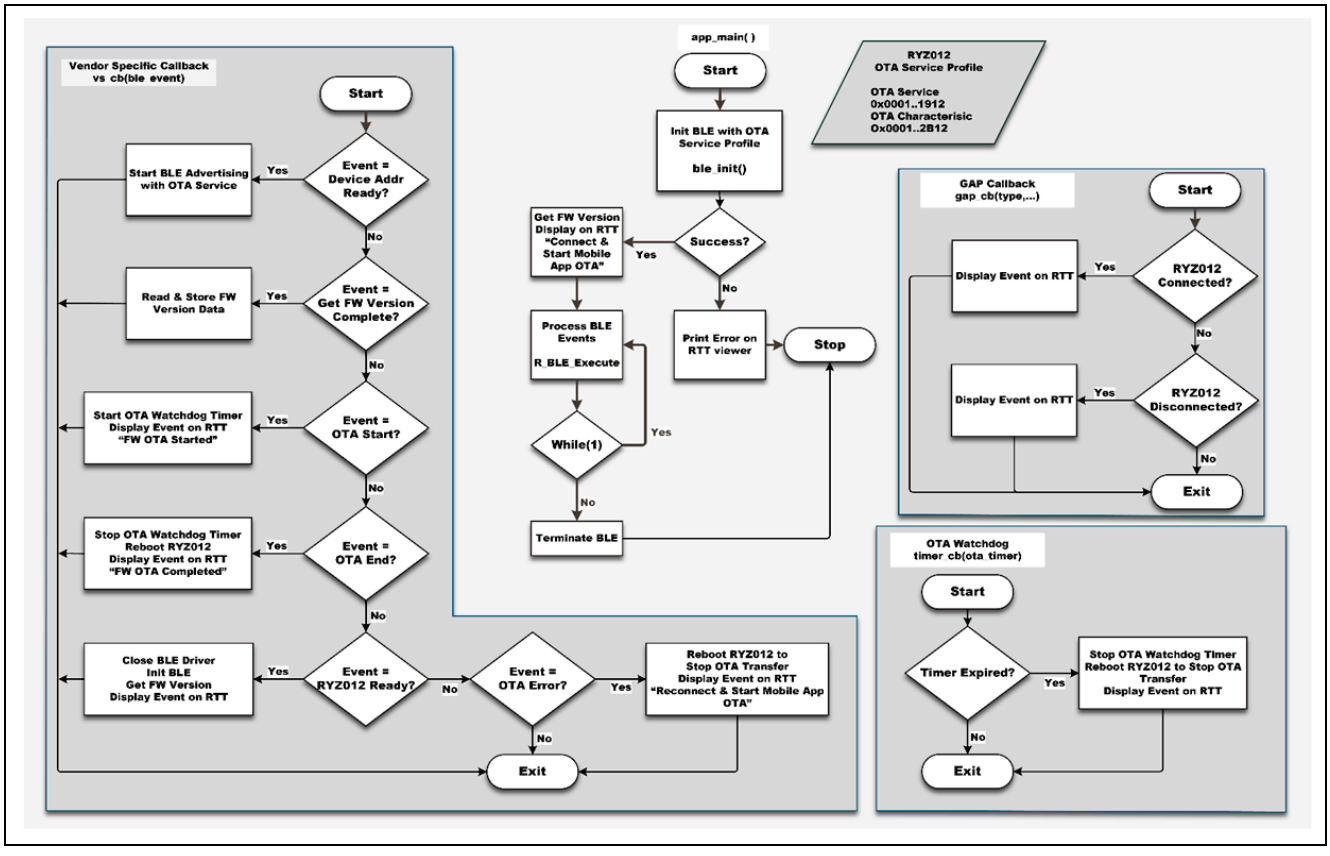


Figure 4. MCU User Application Flow Diagram

1.2 RYZ012 Firmware Update Process

In this application note, the RYZ012 module firmware is upgraded by BLE Radio OTA with a Mobile Device. The RYZ012 PMOD is connected to the EK-RA4M2 board. Only the RYZ012 firmware is upgraded and the host MCU is not upgraded.

The RYZ012 firmware files to use are supplied in the accompanying application project. See the root directory of the project that is imported into e^2 studio. See **Section 1.1.5 Mobile Device OTA App Setup** for more details. Instructions on how to import, build, and run the application project are provided in **Section 2, Firmware Update Application**.

The firmware image file that is used to upgrade the RYZ012 PMOD is transferred to the Mobile Device and opened by the OTA application. The OTA app scans for the custom OTA Service running on the RYZ012 and then connects to the RYZ012 module via BLE. The user selects the firmware file in the OTA app and then initiates the RYZ012 upgrade process. The upgrade process runs until the firmware file is fully transferred to the module via a BLE connection. The status of the upgrade process is monitored by the MCU and shown on the RTT Viewer.

After the RYZ012 is updated, the MCU restarts the RYZ012 which then runs the new image. Finally, the MCU reads the RYZ012 firmware version from the module and displays it in RTT Viewer.

The steps of the firmware upgrade process are shown in Figure 4 **MCU User Application Flow Diagram**.

Error! Reference source not found.4 shows the MCU user application (**project/qe_gen/ble/app_main.c**) flow diagram for the firmware update process. The FSP BLE SPP provided Vendor Specific Call back **vs_cb(ble_event)** simplifies the work required to perform the upgrade.

In this application note, the MCU is a monitor of the upgrade process and receives OTA notification events from the RYZ012 that are handled by the Vendor Specific Call back **vs_cb(ble_event)**.

Because the MCU is monitoring, it can intervene when it detects that an issue has occurred with the RYZ012 and Mobile App. We will cover the possible issues in section 2.4, RYZ012 Firmware Update Considerations.

- Note that the RYZ012 must receive the **SPP_CMD_REBOOT_BLE** command sent by `R_BLE_VS_RestartModule()`, else the firmware upgrade will not fully complete to run the new version firmware image. It will run the roll-back image, which is the version that ran prior to starting the upgrade.
- All firmware image frames must be sent by the Mobile Device OTA App for the transferred firmware image to pass the integrity check. If any of the data frames are missing or corrupted during the transfer process, the new image will fail integrity check and the RYZ012 will run the roll-back image instead after **SPP_CMD_REBOOT_BLE** is issued.
- If the RYZ012 is reset or loses BLE connection with the Mobile Device during the OTA process prior to completing all the update steps, then the RYZ012 will run the roll back image.
- If the RYZ012 loses power during the OTA process prior to completing all the update steps, then the RYZ012 will run the roll back image when power is restored.

To start using the application project immediately, see section 2, Firmware Update Application.

The design details of the application software architecture are covered in section 3, Application Software Architecture.

The implementation details of the application software are covered in section 4, Application Project Implementation.

2. Firmware Update Application

2.1 Importing the Application Project

The steps to import the application project into e² studio are shown in the following sections.

2.1.1 Specify e² studio Workspace

Launch e² studio, specify the workspace directory, and click the **Launch** button.

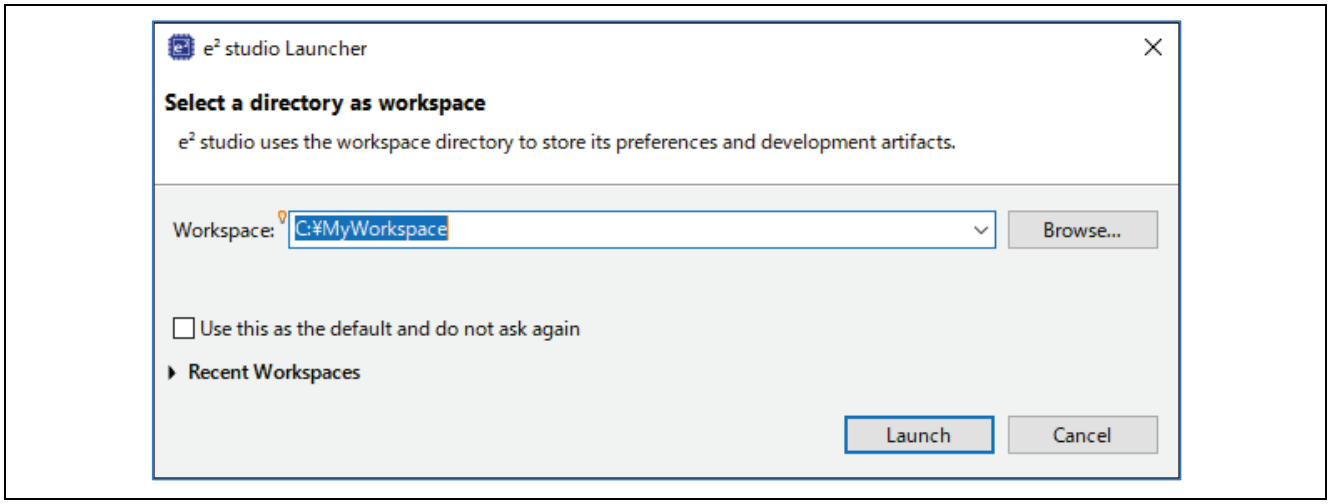


Figure 5. e² studio Workspace

2.1.2 Import Project

Select **File > Import** from the menu bar.

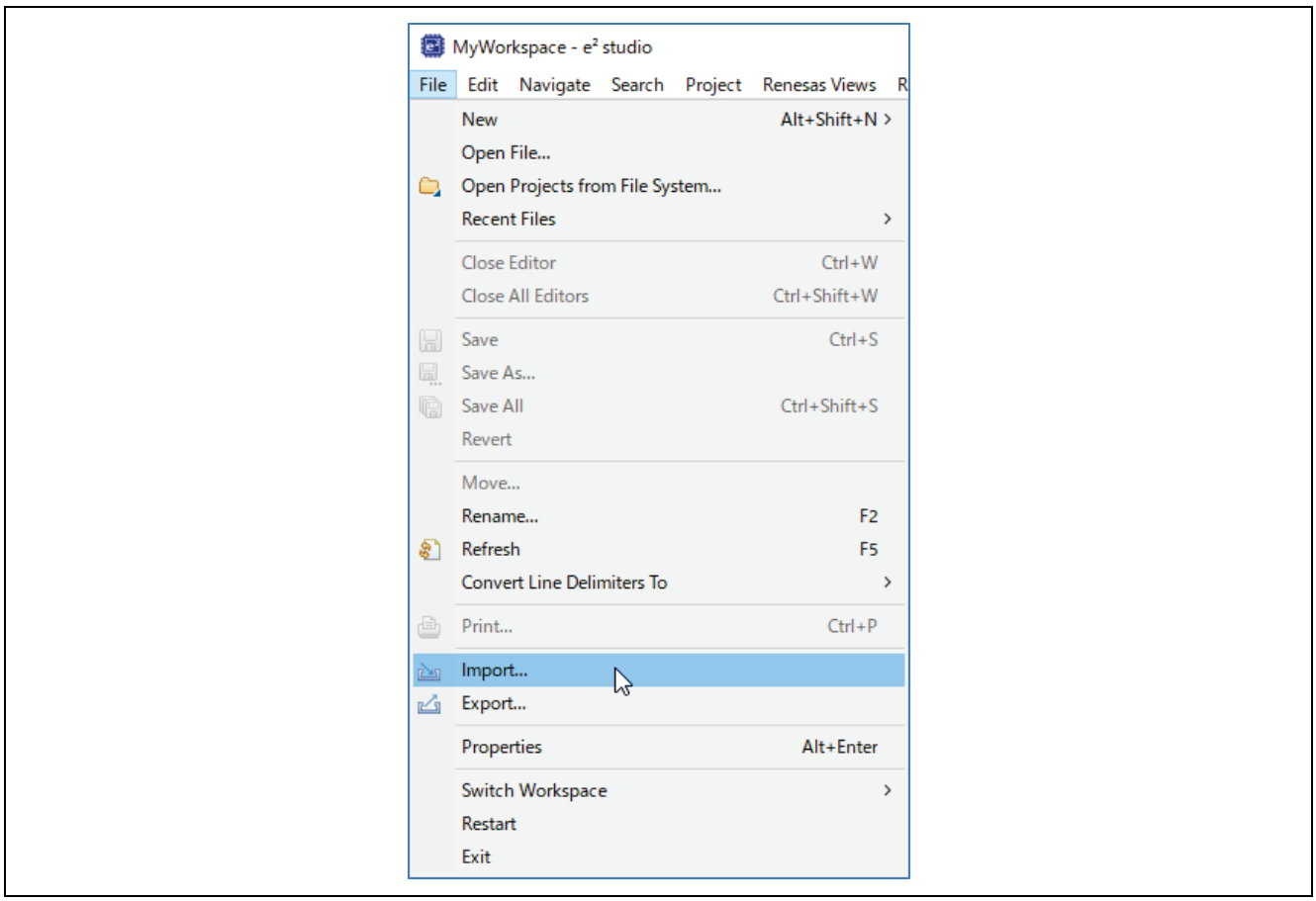


Figure 6. Importing Project

2.1.3 Select Existing Project

Select **Existing Projects into Workspace** and click **Next.**

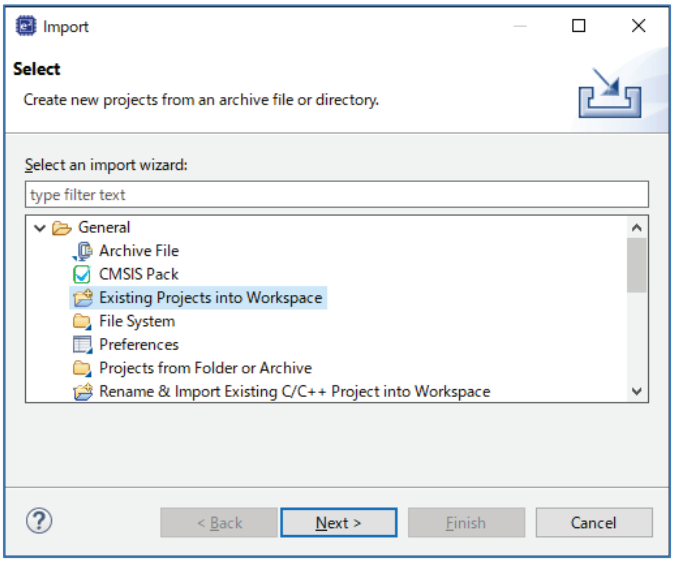


Figure 7. Selecting Existing Projects

2.1.4 Select Project

Choose **Select root directory**, click **Browse** and select the directory for the project to import. Check the box in **Projects**: window and click **Finish** to import the project. If importing from the zip file, then choose **Select archive file** instead and navigate to zip file to import.

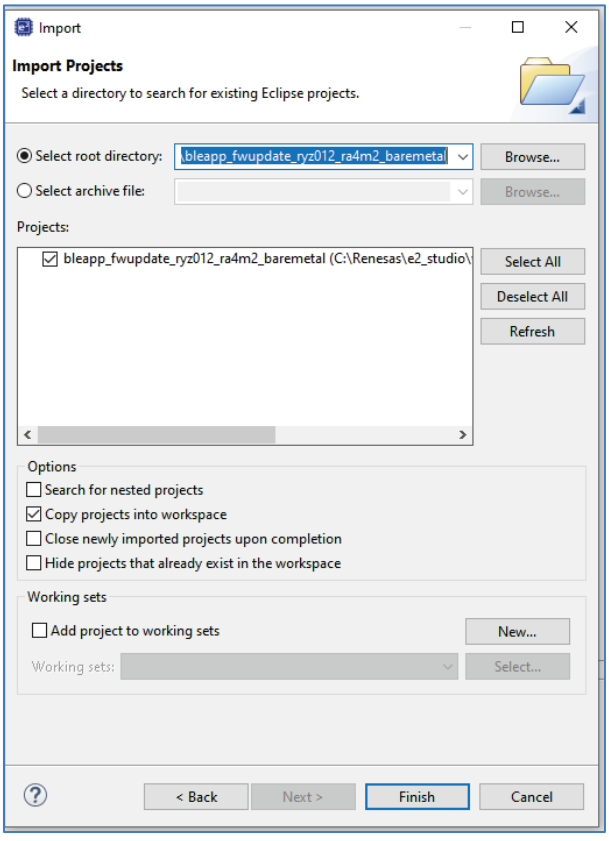


Figure 8. Selecting Existing Projects

2.2 Project Build and Download

1. Select **Project > Build Project** from the menu bar or click the Build icon to build the project.
2. Make sure that the hardware is connected according to section 1.1, Operating Environment. Click the debug icon to launch the project. When the project starts, the application will be downloaded to the EK-RA4M2.

2.3 Application Project Operation

2.3.1 Launch J-Link RTT Viewer

Launch J-Link RTT Viewer, set as follows, and click the **OK** button.

- **Connection to J-Link** : USB
- **Specify Target Device** : R7FA4M2AD
- **Target Interface & Speed** : SWD, 4000 kHz
- **RTT Control Block** : Address, 0x2000050c

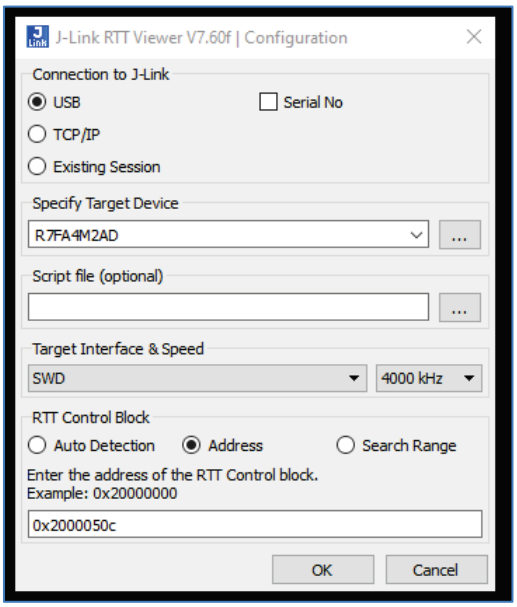


Figure 9. J-Link RTT Viewer Configuration

Note: The RTT Control Block Address can be found in the `.bss._SEGGER_RTT` section address of the map file generated in the Debug directory. If the application source code is modified with your custom changes, this address will change.

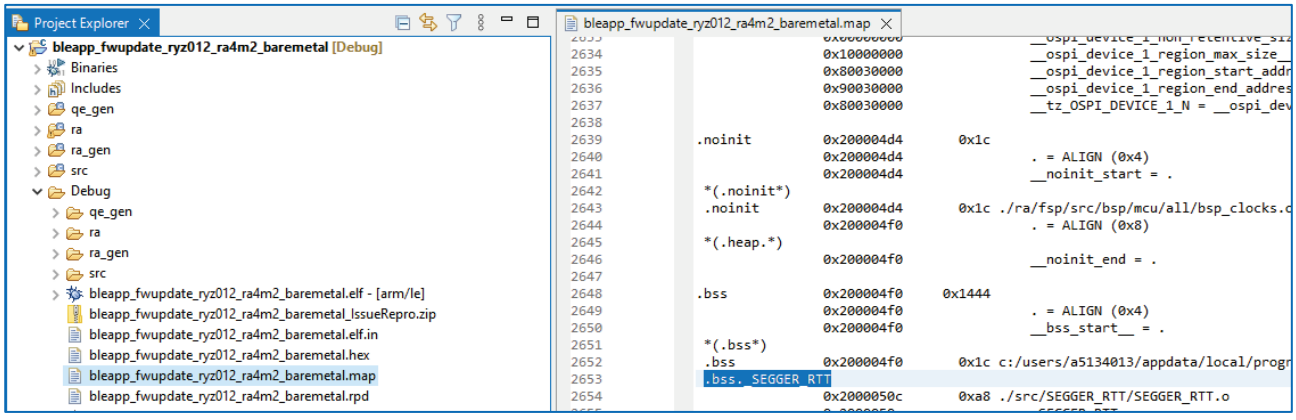


Figure 10. RTT Control Block Address

2.3.2 Start the Application Execution

Click the resume icon in the e² studio Debug Perspective to run the application.

RTT Viewer logging will show that the application is running, and 3 board LEDs (blue, green, red) will blink on and off to indicate that the program is running.

If the LEDs fail to blink and the white LED 4 is not lighted, then verify a USB cable is plugged into connector J10 (with the other end connected to PC). See section 1.1.4, EK-RA4M2 for more details.

In Figure 111, observe the instructions to start the firmware upgrade by connecting the Mobile Device OTA App to the RYZ012 and starting the Firmware OTA transfer.

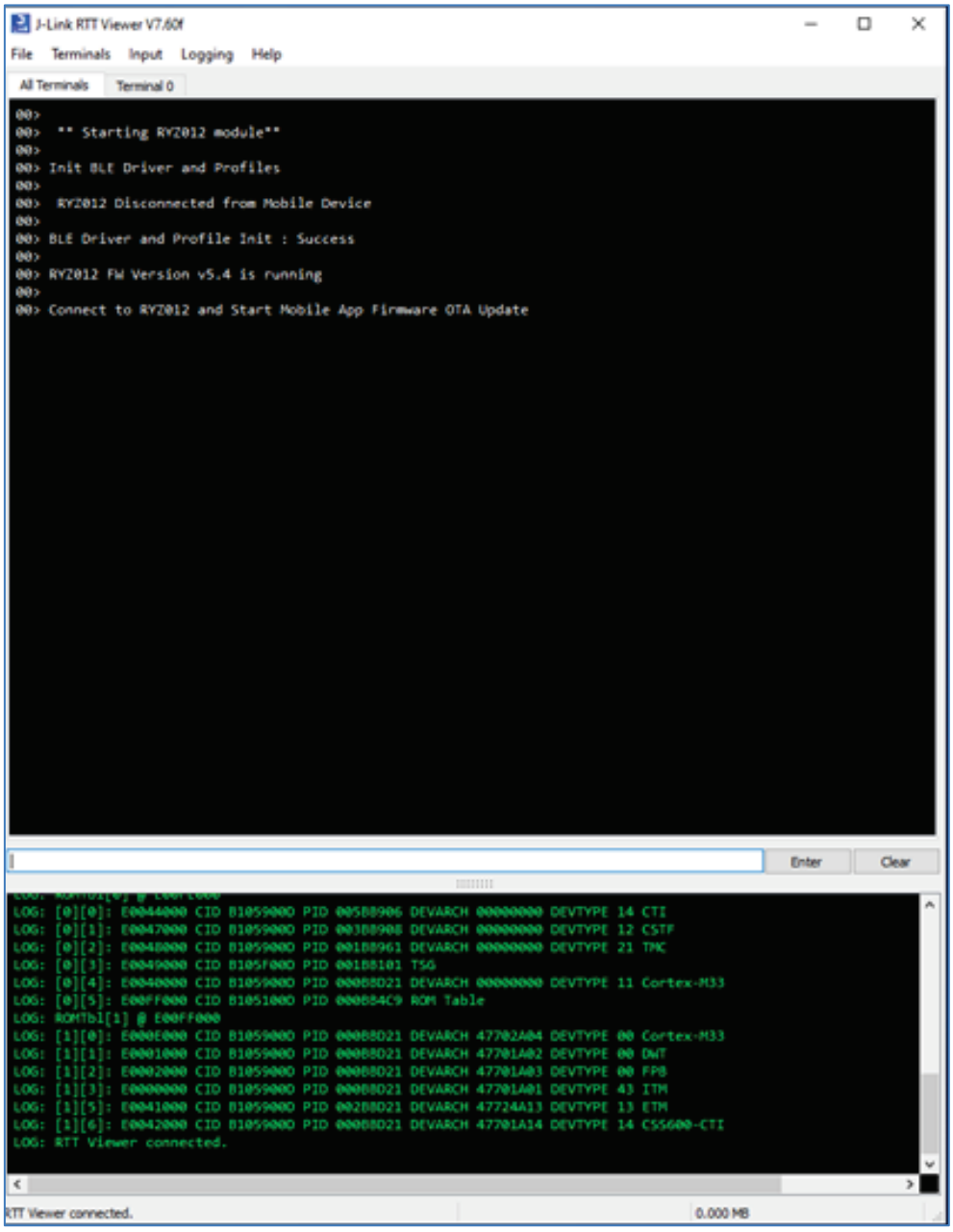


Figure 11. RTT Viewer- MCU App Start

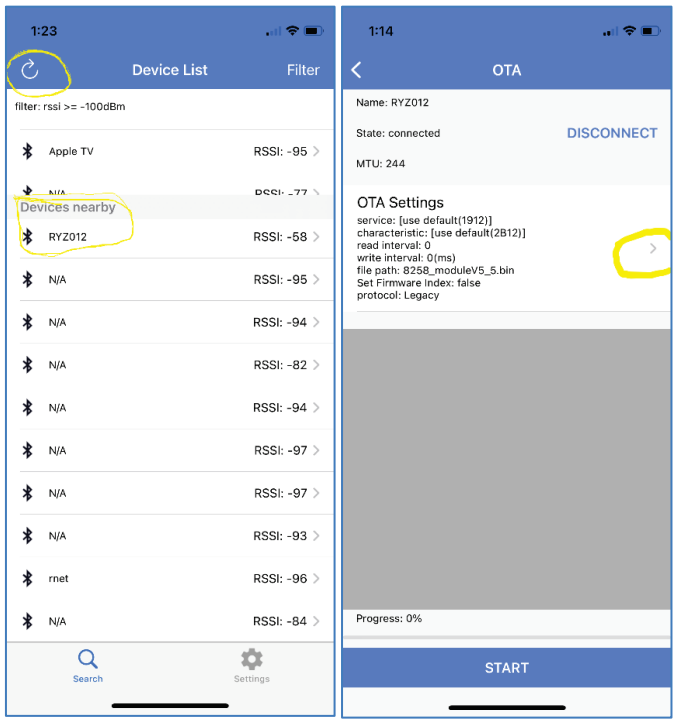


Figure 12. Mobile App – Connect to RYZ012

Using the Mobile Device OTA App, scan for “RYZ012” module that is advertising and connect to the RYZ012. In the OTA App, enter the OTA Settings menu to select the RYZ012 firmware file that was previously loaded to the Mobile Device. See Figure 12.

See section 1.1.5, EK-RA4M2 Setup for more information on loading firmware files on the Mobile Device.

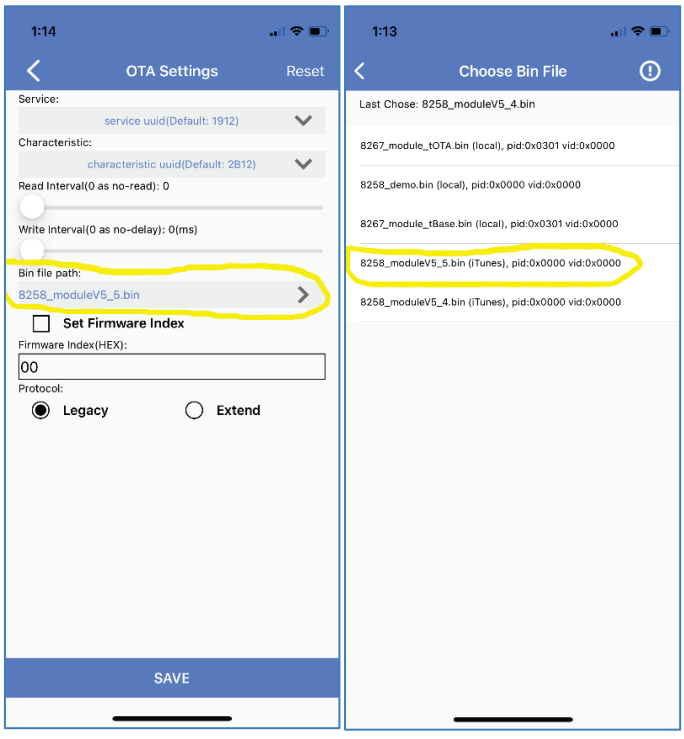


Figure 13. Mobile App – Choose OTA Settings to Select Firmware File

Once the firmware file is selected, press Save to store the OTA Settings. See Figure 13.

Press the Start button on the OTA App. The firmware file will start transferring to the RYZ012. See Figure 14.

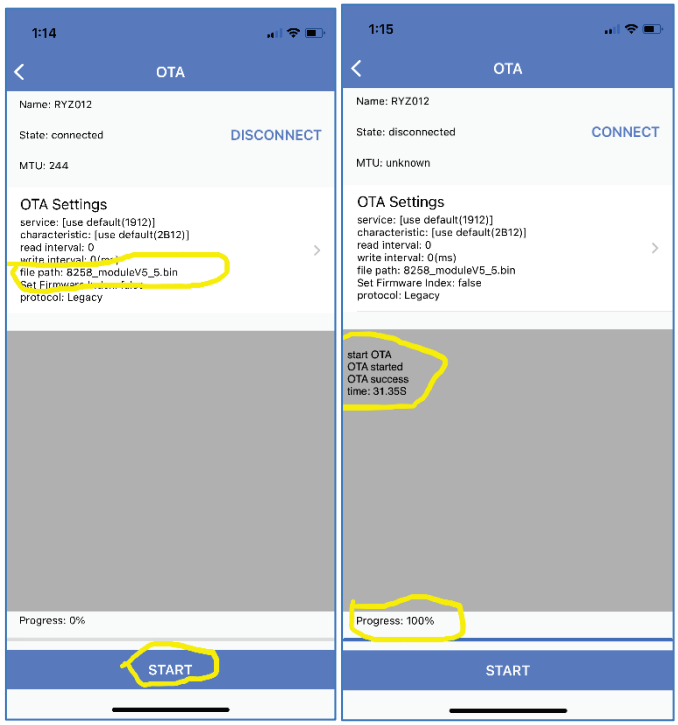


Figure 14. Mobile App – Start OTA Transfer & OTA Success

2.3.3 RTT Viewer Logs

At this point, move your attention to RTT Viewer to monitor the firmware upgrade status. You should observe logging similar to Figure 155.

RTT Viewer will show the results of the RYZ012 firmware OTA upgrade process.

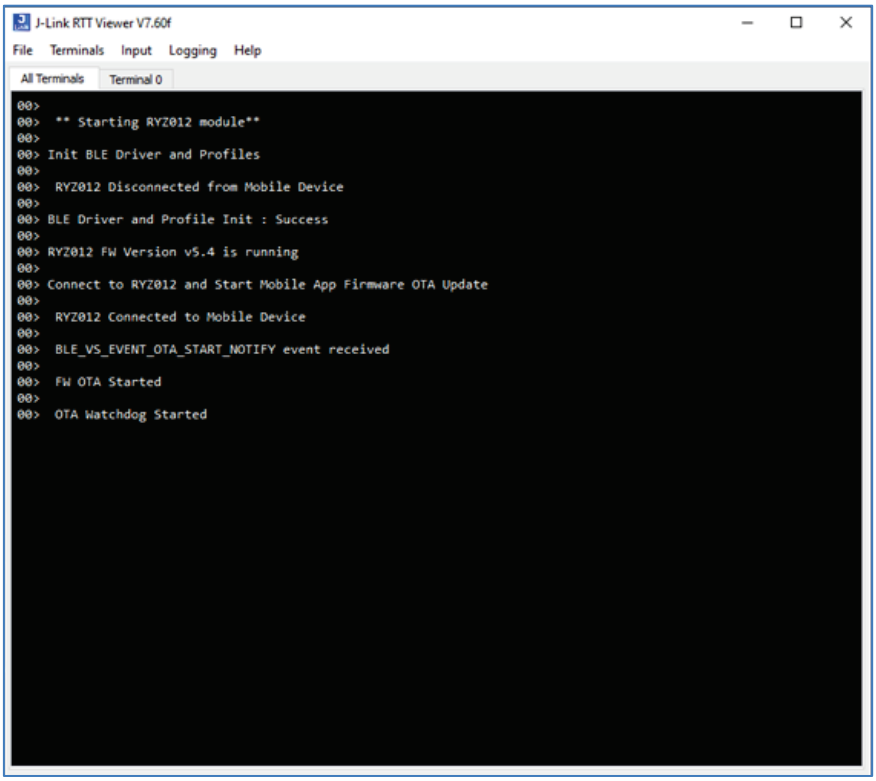


Figure 15. RTT Viewer – RYZ012 OTA Started

When the RYZ012 firmware upgrade completes with success, RTT Viewer will show logging as shown in Figure 166.

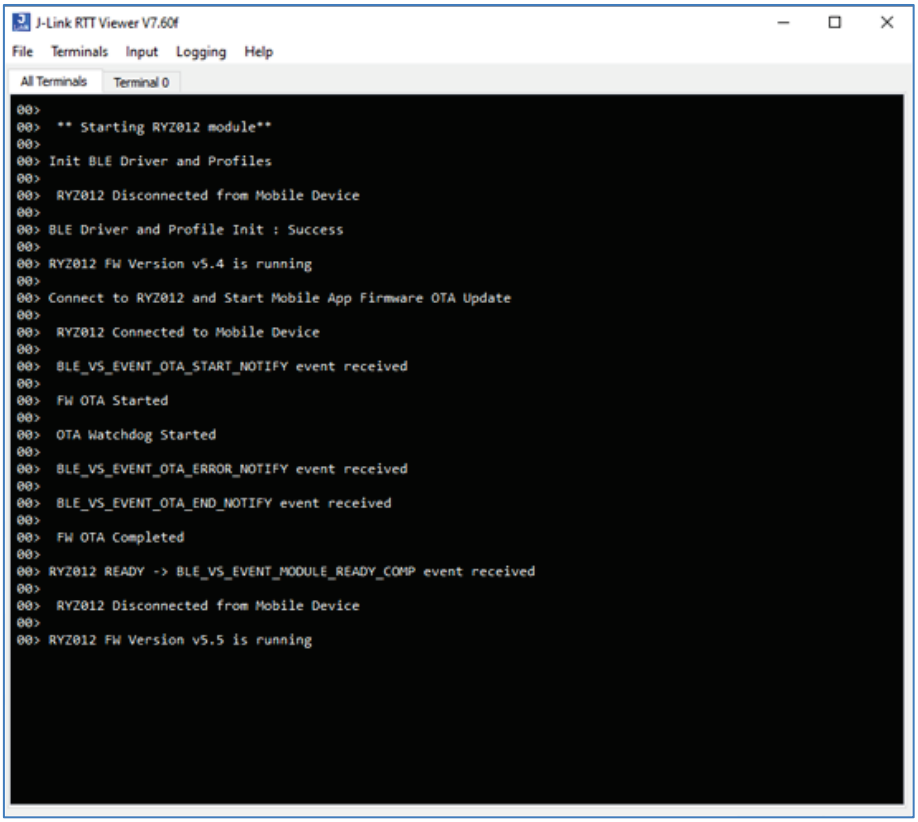


Figure 16. RTT Viewer – RYZ012 Firmware OTA Update Success

Notice that the RYZ012 firmware version was originally v5.4 before the OTA upgrade, and then after upgrade completes, is running at v5.5.

To verify that the RYZ012 is functioning use the OTA app on the mobile device and scan for BLE devices. The RYZ012 module will resume advertising again as “RYZ012” when the firmware upgrade has completed.

Currently, there is a known issue at 30 seconds after starting any OTA upgrade session where one **BLE_VS_EVENT_OTA_ERROR_NOTIFY** event is sent to the MCU by the RYZ012. This occurs even when the OTA is successful. If more than one **BLE_VS_EVENT_OTA_ERROR_NOTIFY** event is sent, then it is assumed that too many packet errors have occurred in the transfer between the Mobile Device OTA app and RYZ012. The transfer will be stopped by the MCU by restarting the RYZ012. The upgrade can be restarted by connecting the OTA app to the RYZ012 and starting the firmware upgrade.

In the event that the firmware upgrade fails with error messages shown in RTT Viewer, the next section 2.4, RYZ012 Firmware Update Considerations will help you to troubleshoot and resolve the issues.

2.4 RYZ012 Firmware Update Considerations

In the event that the upgrade process fails, see the following information to help troubleshoot your issue.

- First verify that the EK-RA4M2 and RYZ012 PMOD have been setup and connected correctly. Refer to section 1.1, Operating Environment**.**
- Next, verify the RYZ012 firmware image transferred is the one of two available images included in the app project.
  The firmware images are located in the e^2 studio project workspace at the project directory **bleapp_fwupdate_ryz012_ra4m2_baremetal/8258_moduleV5_4.bin** or **bleapp_fwupdate_ryz012_ra4m2_baremetal/8258_moduleV5_5.bin:**
  - The RYZ012 has built-in verification steps to recover from an incorrect or failing firmware update.
  - The RYZ012 attempts to revert to the previous image if the update fails or halts.

- Once a firmware upgrade succeeds, the user must re-run the entire update process to downgrade to a previous firmware version.
- Review the troubleshooting section below for solutions to the most common issues. For all other issues that cannot be resolved in this section, please check the FAQs, search the Knowledge Base, or submit a support ticket at Renesas Support

2.4.1 Firmware Update Issue #1

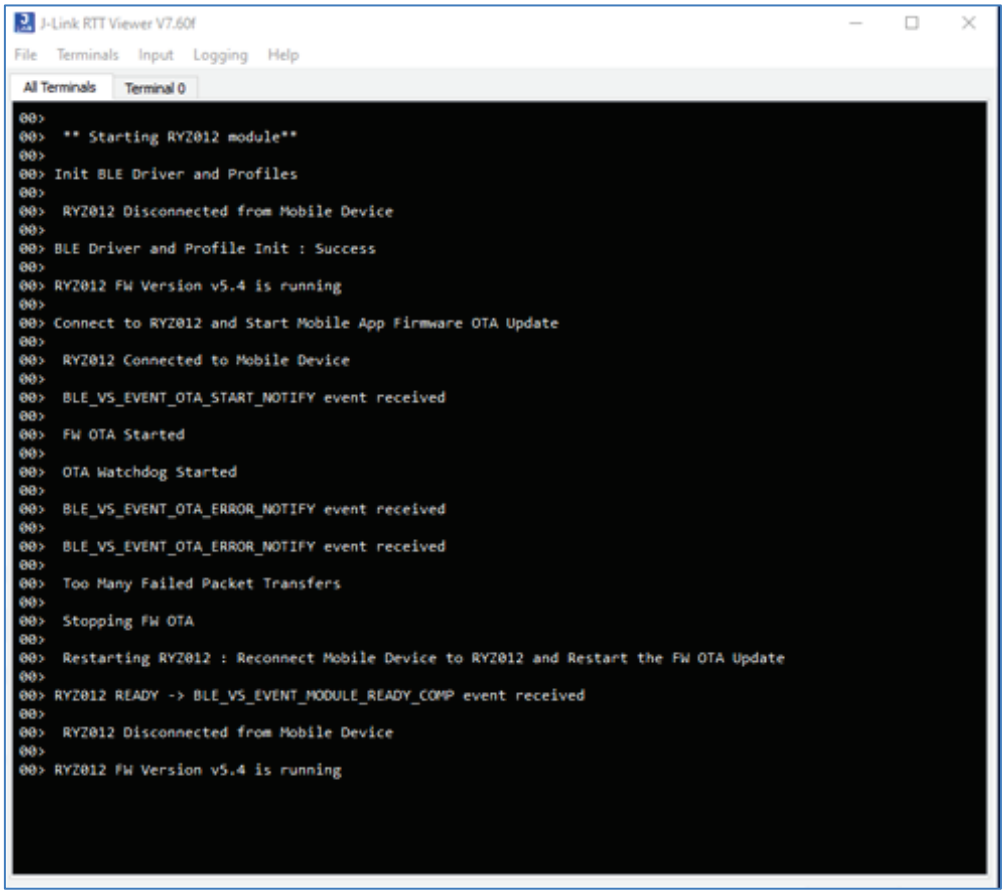


Figure 17. RTT Viewer – RYZ012 Firmware Packet Transfers Failed

In Figure 177, notice that the RYZ012 firmware version is v5.4 before the upgrade starts and is at v5.4 after the upgrade fails. This indicates that not all firmware packets were received during the transfer to the RYZ012. In this case, the integrity check failed on the RYZ012 before programming the memory in the RYZ012. Since the image failed integrity check, the old image v5.4 (roll back image) was retained and selected by the RYZ012 to execute.

Make sure that the EK-RA4M2 board remains powered during the file OTA transfer and write to RYZ012. If the EK-RA4M2 board is reset during the firmware OTA upgrade then the file integrity check will fail on the OTA transfer and the RYZ012 will run the last known good image version (roll back image).

2.4.2 Firmware Update Issue #2

For the case that the OTA App disconnects from the RYZ012 and does not complete sending the entire firmware file, the MCU based OTA Watchdog will expire and restart the RYZ012 module so that the Mobile Device can re-connect.

The OTA Watchdog expires 45 seconds after the start of the OTA transfer indicated by event BLE_VS_EVENT_OTA_START_NOTIFY. The OTA Watchdog time was chosen based on the OTA app fastest transfer rate and the current size of the firmware image. If the RYZ012 file size increases or a slower transfer rate is used then consider increasing the OTA Watchdog expiration time for you application. See `#define BLE_APP_OTA_TRANSFER_FAST in /src/common_init.h`

If the OTA App cannot re-connect to the RYZ012 after the OTA is started and disconnect occurs, wait for the OTA Watchdog to reboot and reset the RYZ012.

Figure 18 shows the issue where the Mobile Device disconnected after OTA start. See the two events BLE_VS_EVENT_OTA_ERROR_NOTIFY that were received by the MCU after the OTA Watchdog Started. Seconds later the OTA Timed-Out because the OTA Transfer did not complete which should have been indicated by event BLE_VS_EVENT_OTA_END_NOTIFY.

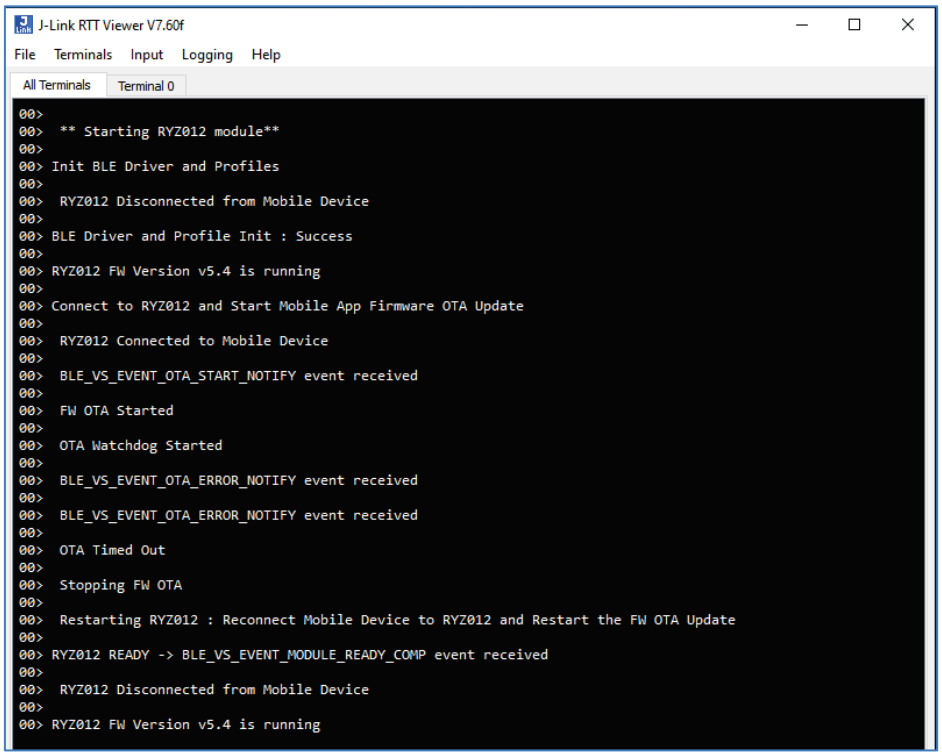


Figure 18. RTT Viewer – Mobile Device Disconnected

2.4.3 Initialize BLE Driver Failed

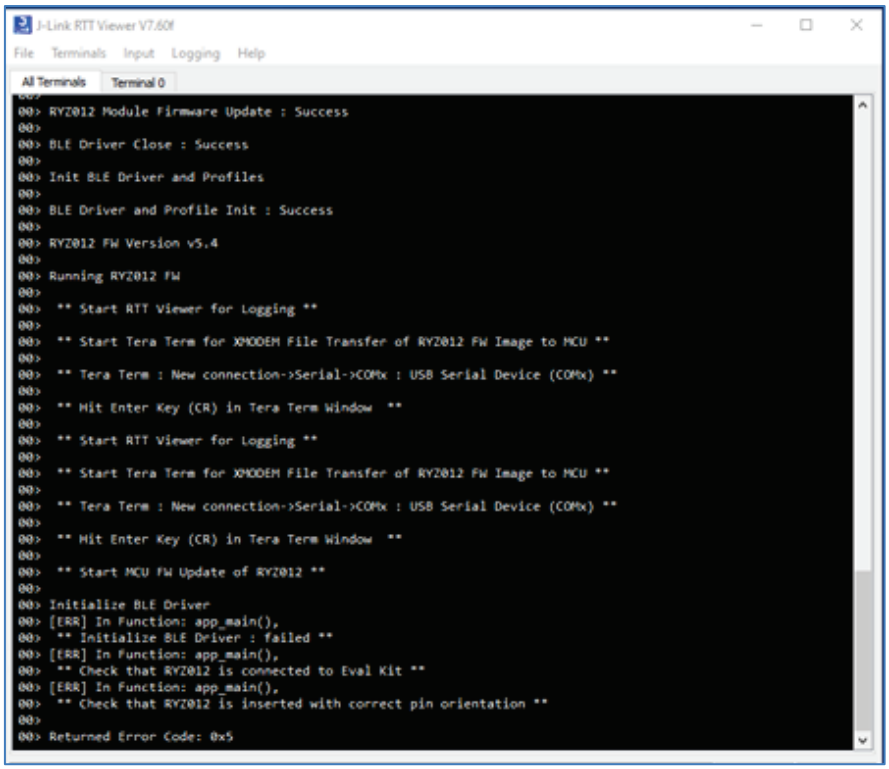


Figure 19. Init BLE Driver Failed

If you receive the **Initialize BLE Driver: failed** message then check that the RYZ012 PMOD is installed correctly in PMOD1 connector (J26) on the EK-RA4M2. Make sure the PMOD Mode Select GPIO pin is set for the correct initial logic state for the SPI or UART mode that you are using. See section 6, Next Steps, step 1 to review the SPP BLE Module and PMODx configuration settings. Make certain the RYZ012 PMOD has a factory firmware image programmed before attempting module upgrade.

3. Application Software Architecture

The application software architecture for the BLE Radio OTA firmware upgrade project is covered in this section.

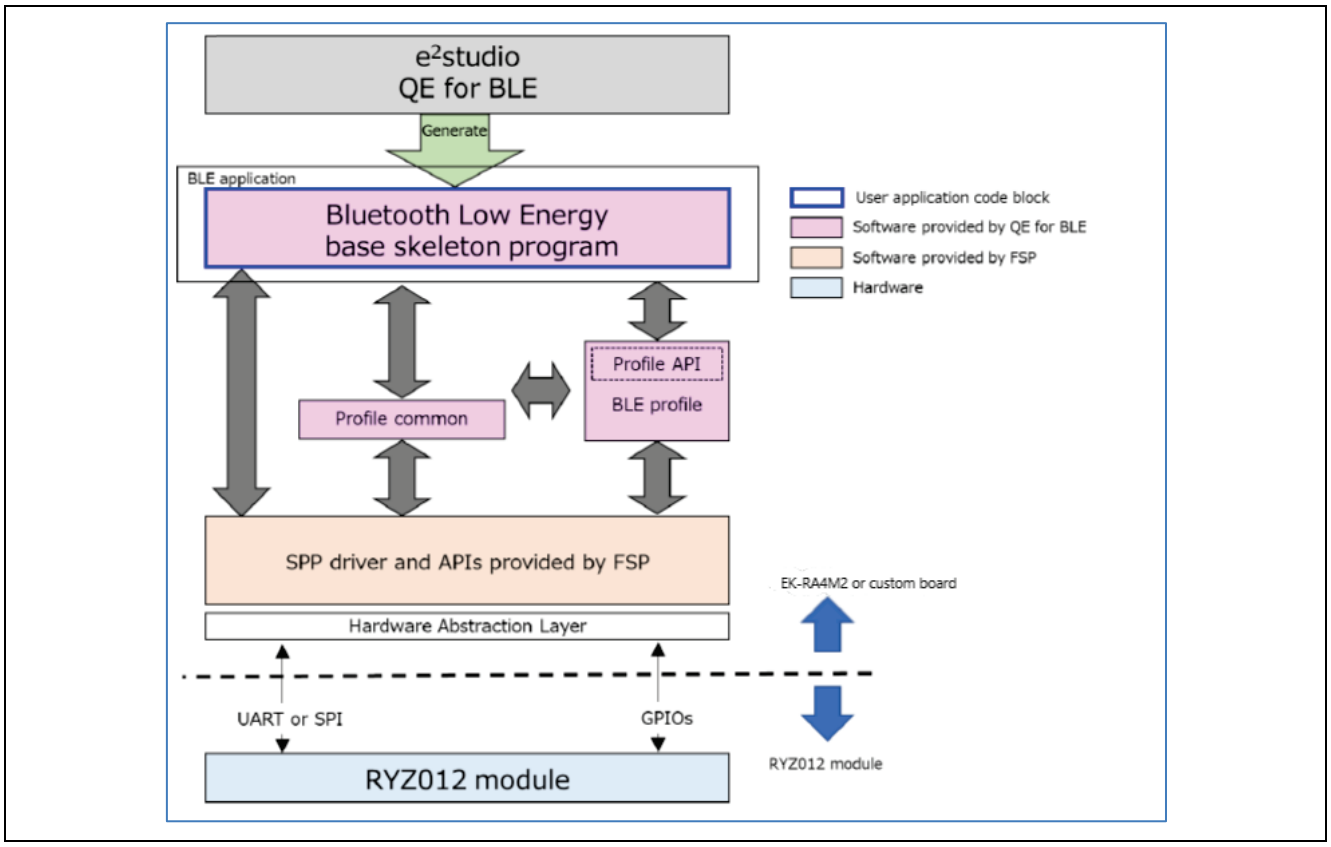


Figure 20. Application Software Architecture (Bare Metal)

Figure 20 shows the software architecture of a Bluetooth LE application in a BareMetal environment. The BLE Application performs initialization and BLE related processing. The QE for BLE tool generates C source code for the Bluetooth LE base skeleton program for the MCU application (located in project **qe_gen/ble/app_main.c**) and the BLE Profile. The base program is extended further for the specific functionality required in the user application. In this case, we are using the FSP BLE Abstraction APIs with SPP APIs to monitor the firmware update of the RYZ012 Module.

For more information on FSP APIs see **Renesas Flexible Software Package (FSP) User's Manual (**R11UM0155**) and section on *SPP Bluetooth Low Energy Abstraction with RYZ012 (rm_ble_abs_spp).***

The FSP SPP APIs are used by the MCU to communicate with the RYZ012 module and monitor the Firmware OTA process with the Mobile Device. Figure 21 shows the software components of the application running on the RA4M2 MCU with a comms interface to the RYZ012. The user application and BLE Profile are customized to add features unique to your application.

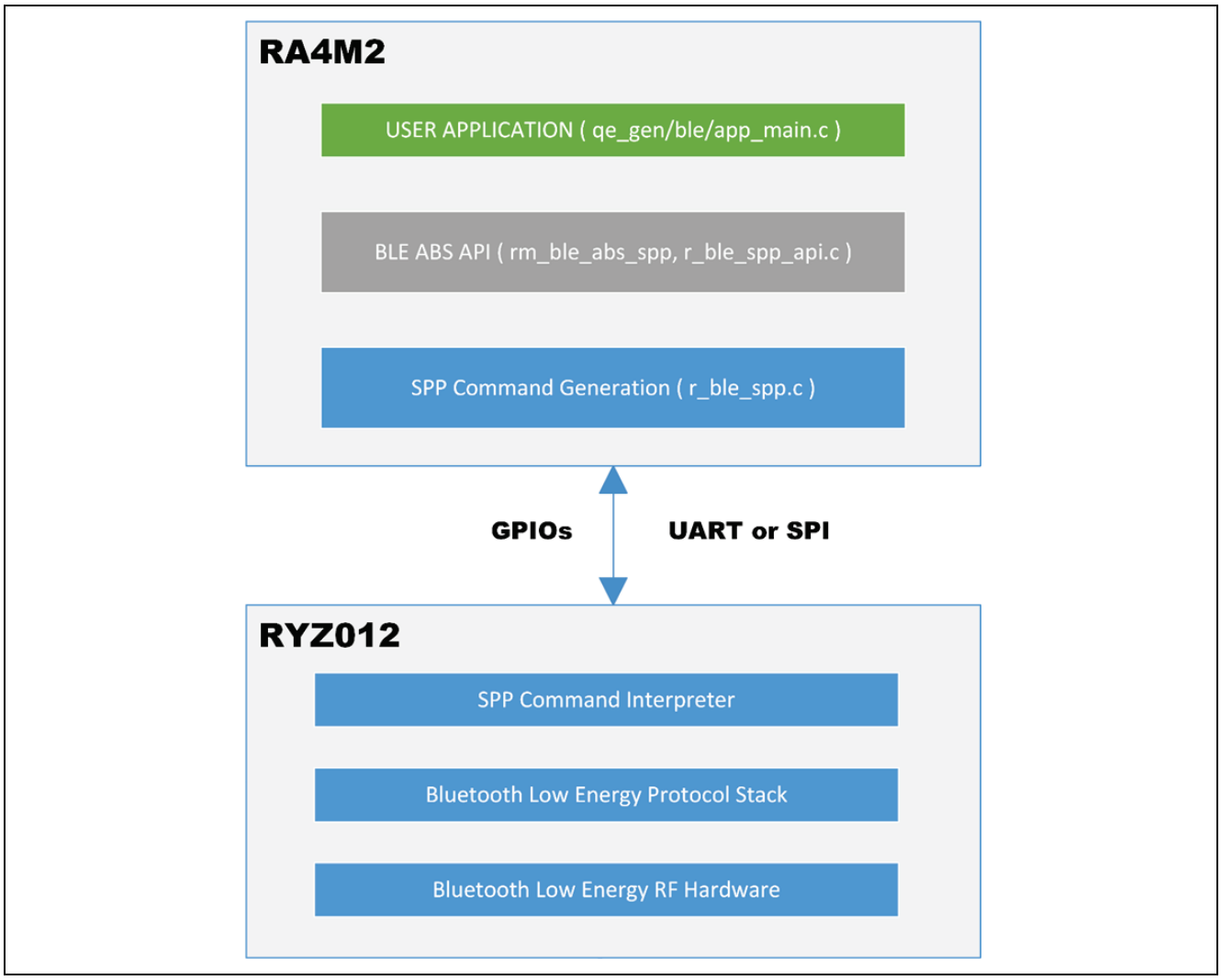


Figure 21. MCU Host and Bluetooth LE Serial Port Profile Interface

Figure 21 provides a more detailed look at the interface of the SPP and BLE APIs provided by FSP. The user application calls the BLE ABS APIs to interact with the RYZ012 BLE Module via the SPP Command Generation (SPP Driver) and SPP Command Interpreter residing on the RYZ012. The MCU communications interface to the RYZ012 PMOD can be UART or SPI. The accompanying software project uses the SPI interface.

The details of the update process are covered in section 1.2, RYZ012 Firmware Update Process.

The details of the application project software implementation are covered in section 4, Application Project Implementation.

3.1 Microcontroller Peripheral Functions

The microcontroller peripheral functions used in the application project are shown below.

Table 3. Microcontroller peripheral functions

Module	Pin	Description
Serial Communication Interface	SCI0(SPI0)	SPI communication with RYZ012 on PMOD1 (J26) MOSI : P207 MISO : P206 SCK : P400 SCLK SS : P401 Chip Select IRQ12 : P008 Interrupt Request RESET : P403 MODE SELECT : P402 SPI Mode (High)

Module	Pin	Description
General-Purpose Timer	GPT0 GPT1 GPT2 GPT3	LED Blue PWM LED Red PWM LED blink timer LED Green PWM
I/O Port	P008	Interrupt Request RYZ012 PMOD1
	P403	Reset pin control of RYZ012 PMOD1
	P402	Mode Select (HIGH = SPI) of RYZ012 PMOD1
	P103	Blinker Timer PWM
	P415	User LED1 (Blue) PWM
	P404	User LED2 (Green) PWM
	P405	User LED3 (Red) PWM
External Interrupt Request	IRQ12	RYZ012 PMOD1 Interrupt Request

3.2 FSP Modules

The FSP modules used in the application project are shown below. See Figure 22, FSP Module Summary for a view of all the modules in the project.

Table 4. FSP Modules

Module Type	Module Name		Usage
System	I/O Port	r_ioport	GPIOs and LED indicators
BLE API	SPP BLE Abstraction	r_ble_abs_spp	RYZ012 BLE
Input	External IRQ	r_icu	RYZ012 PMOD Interrupt Req
Connectivity	SPI	r_sci_spi	RYZ012 PMOD comms
Timers	Timer	r_gpt	Blue, Red, Green Blink LED PWM

Note: This application program is a bare metal version.

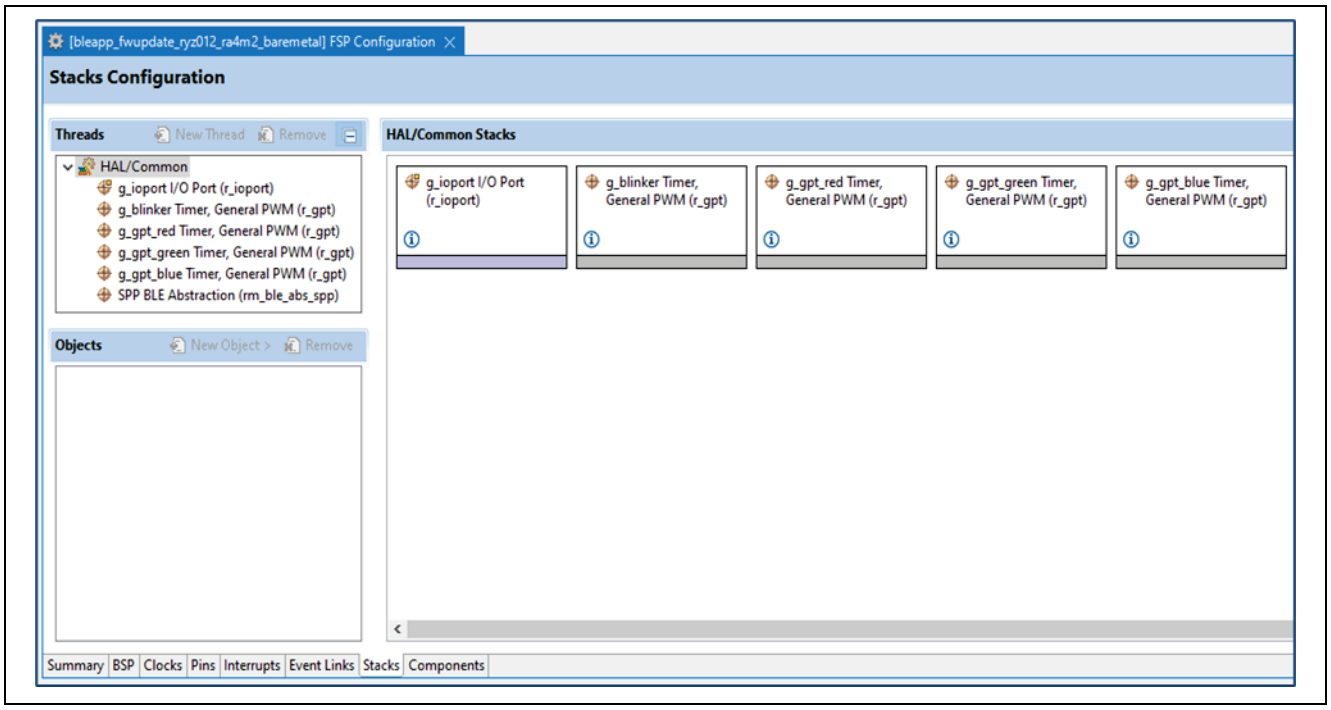
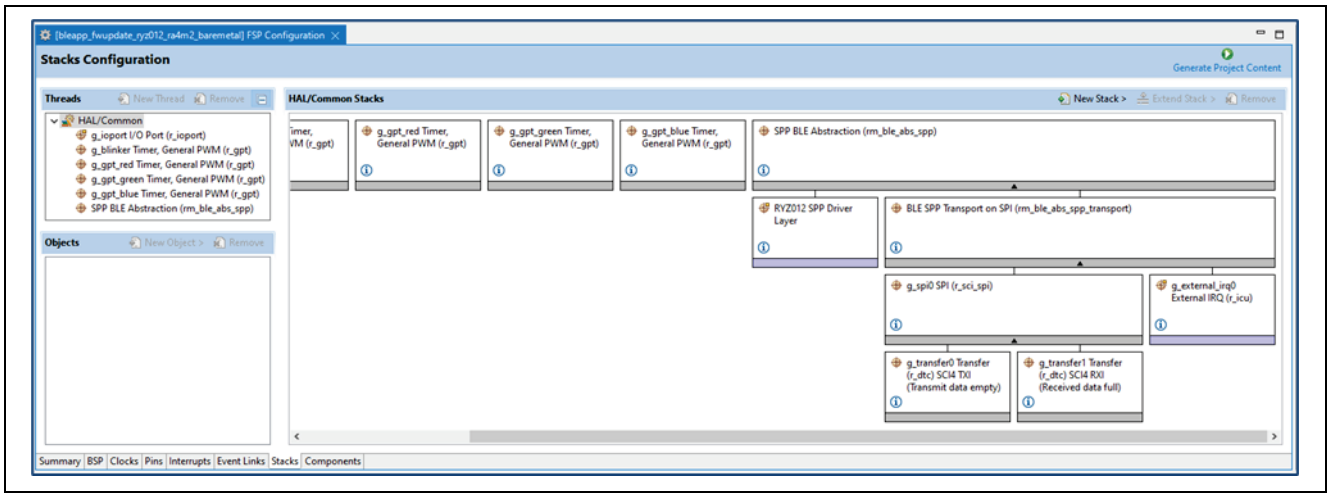


Figure 22. FSP Module Summary



4. Application Project Implementation

This section describes the application project implementation.

The firmware update is implemented in app_main.c. The *app_main( )* includes BLE and system peripheral initialization and the implementation of the main loop.

When using QE for BLE tool, a minimal skeleton code of app_main.c is automatically generated which is customized with FSP APIs and helper functions to add the desired Bluetooth module update functionality.

4.1 Entry Point

In hal_entry.c the function hal_entry() initializes the GPT module for indicator LEDs and calls *app_main( )* to perform the BLE module update as follows:

```
 * File Name    : hal_entry.c
 * DISCLAIMER

#include <stdio.h>
#include <string.h>
#include "hal_entry.h"
#include "common_init.h"
#include "common_utils.h"

/* Function declaration */
void R_BSP_WarmStart(bsp_warm_start_event_t event);
void app_main(void);

 * The RA Configuration tool generates main() and uses it to generate threads if an RTOS is used.  This function is
void hal_entry(void)
{
    fsp_err_t err = FSP_SUCCESS;

    /* Initialize GPT module */
    err = common_init();
    if (FSP_SUCCESS != err)
    {
        /* Turn ON RED LED to indicate fatal error */
        TURN_RED_ON
        APP_ERR_PRINT("\r\n ** Initialize GPT module : failed ** \r\n");
        APP_ERR_TRAP(err);
    }

    /* Start RYZ012A - Use RTT Viewer for Status Logging */
    APP_PRINT("\r\n ** Starting RYZ012 module** \r\n");
    app_main();
}
```

4.2 Main Loop

The *app_main( )* includes the BLE and profile initialization, BLE module firmware update monitoring, and the main loop that processes the BLE events. See the source code below for *app_main( )* with the implementation details. The steps of execution are:

1. Initialize the BLE driver and OTA Profile to start the communications with the RYZ012.
2. Display BLE driver and Profile initialization status on RTT Viewer
3. Get the current firmware version running on the RYZ012.
4. Display the RYZ012 firmware version on RTT Viewer.

5. Instruct user to connect BLE Device to RYZ012 and Start Mobile App OTA.
6. Start Watchdog Monitor and Log the status of the RYZ012 firmware update to RTT Viewer
7. Close the BLE driver and call ble_init() to re-start and initialize the BLE module and Profile, then run the new RYZ012 firmware.
8. Get the RYZ012 firmware version and display on RTT Viewer to confirm updated image is running.
9. Enter the main while loop to operate the RYZ012 and process BLE events.

4.3 BLE Initialization Process

QE for BLE tool was used to create a basic BLE Profile with “RYZ012” set as the advertise name and supported OTA Service. From the tool, the source code of the *ble_init( )* function is automatically generated and placed at the top of the app_main() functions. See below where the placement of **ble_init( )** call must be relocated in the sequence after the firmware update process steps are completed.

```
void app_main(void)
{
/* Hint: Input process that should be done before main loop such as calling initial function or variable definitions */
/* Start user code for process before main loop. Do not edit comment generated here */

    ble_status_t status;
    APP_PRINT("\r\nInit BLE Driver and Profiles\r\n");
    /* Initialize BLE and profiles */
    status = ble_init();
    if (FSP_SUCCESS != status)
    {
        APP_ERR_PRINT("\r\n ** BLE Driver and Profiles Init : Failed ** \r\n");
        APP_ERR_TRAP(status);
    }
    APP_PRINT("\r\nBLE Driver and Profile Init : Success\r\n");

    /* Get RYZ012 FW Version that is running */
    get_fw_version();
    APP_PRINT("\r\nConnect to RYZ012 and Start Mobile App Firmware OTA Update\r\n");
/* End user code. Do not edit comment generated here */


    /* main loop */
    while (1)
    {
        /* Process BLE Event */
        R_BLE_Execute();

/* Hint: Input process that should be done during main loop such as calling processing functions */
/* Start user code for process during main loop. Do not edit comment generated here */
        if (true == GPT_Is_Watchdog_Expired())
        {
            /*
             * monitor FW OTA Transfer watch dog
             * use RYZ012 SPP Vendor Specific Callback to handle OTA timeout
             **/
            vs_cb(BLE_VS_EVENT_OTA_ERROR_NOTIFY, BLE_ERR_RSP_TIMEOUT, NULL);
        }
/* End user code. Do not edit comment generated here */
    }

/* Hint: Input process that should be done after main loop such as calling closing functions */
/* Start user code for process after main loop. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

    /* Terminate BLE */
    RM_BLE_ABS_Close(&g_ble_abs0_ctrl);
}
```

See the details of the ble_init() function implementation below that initializes the BLE module, registers the callback functions for BLE, registers the GATT database, and any other additional services added with the QE for BLE tool during BLE Profile generation.

```
/* Function Name: ble_init */
ble_status_t ble_init(void)
{
    ble_status_t status;
    fsp_err_t err;

    /* Initialize BLE */
    err = RM_BLE_ABS_Open(&g_ble_abs0_ctrl, &g_ble_abs0_cfg);
    if (FSP_SUCCESS != err)
    {
        return err;
    }

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

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

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

    /* Set Prepare Write Queue */
    R_BLE_GATTS_SetPrepareQueue(gs_queue, BLE_GATTS_QUEUE_NUM);

    /* Initialize GATT Service server API */
    status = R_BLE_GATS_Init(gats_cb);
    if (BLE_SUCCESS != status)
    {
        return BLE_ERR_INVALID_OPERATION;
    }
    return status;
}
```

4.4 Register BLE Callback Functions

Registration of the callback functions are performed in **ble_init()** which are required to execute processing according to events from each layer such as GAP, GATT Server, GATT Client, and Profile Server API in the Bluetooth LE protocol stack. See **Bluetooth Low Energy Sample Application (R01AN6116EJ)** for more details of the callback registration process for RYZ012 and RA MCU and refer to **RA Flexible Software Package User's Manual (R11UM0155)** for more information on the type of callback events to handle.

4.5 Get Firmware Version Function

See the source code below for the details of utility function **get_fw_version(void)** which is used to read the firmware version that is currently running on the RYZ012 module. The function calls the FSP provided function **R_BLE_VS_GetFirmwareVersion( )** and uses SPI to get the version response back from the module immediately parses the result for display on RTT Viewer.

```
/* Get the Firmware Version running on RYZ012 Module */
bool get_fw_version(void)
{
    static r_ble_spp_payload_t payload_data;
    r_ble_spp_cmd_rsp_t ret_val = R_BLE_SPP_SUCCESS;
    ble_status_t status;
    #define MIN_FW_VERSION_BYTES  3  /* Payload must have 3 or more bytes */

    status = R_BLE_VS_GetFirmwareVersion();
    if (BLE_SUCCESS != status)
    {
        APP_ERR_PRINT("\r\n ** Get RYZ012 FW Version : Failed ** \r\n");
        return false;
    }

    /* To get the FW version data immediately use R_BLE_SPP_SPI_Read(&payload_data)
     * otherwise, use Process BLE Event by calling R_BLE_Execute() and waiting
     * for callback event BLE_VS_EVENT_GET_FW_VERSION_COMP to be received in vs_cb(...)
     * vs_cb(uint16_t type, ble_status_t result, st_ble_vs_evt_data_t *p_data)
     */
    R_BSP_SoftwareDelay(10, BSP_DELAY_UNITS_MILLISECONDS);
    ret_val = R_BLE_SPP_SPI_Read(&payload_data);
    if ( R_BLE_SPP_SUCCESS == ret_val )
    {
        if ( ( BLE_VS_EVENT_GET_FW_VERSION_COMP == payload_data.event_id ) && ( MIN_FW_VERSION_BYTES <= payload_data.out_len ) )
        {
            /* payload contains fw version data */
            g_ble_version_data.major = payload_data.out_data[1];
            g_ble_version_data.minor = payload_data.out_data[2];
            APP_PRINT("\r\nRYZ012 FW Version v%d.%d \r\n", g_ble_version_data.major, g_ble_version_data.minor);
            return true;
        }

        APP_ERR_PRINT("\r\n ** Get RYZ012 FW Version : Failed ** \r\n");
        return false;
    }
    APP_PRINT("RYZ012 FW Version data read Error: %x\r\n", ret_val);
    return false;
}
```

4.6 SPP BLE OTA Notify Events

The FSP BLE SPP API provides a Vendor Specific Callback **vs_cb(type, result, …)** that handles the OTA Notification Events received during the RYZ012 firmware upgrade process. The callback provides the ability for the MCU to monitor the RYZ012 firmware OTA process status. It is helpful for MCU intervention when issues arise during the Mobile Device firmware OTA transfer.

```
/**********************************************************************************
 * Function Name: vs_cb
 * Description  : Callback function for Vendor Specific API.
 * Arguments    : uint16_t type -
 *                  Event type of Vendor Specific API.
 *              : ble_status_t result -
 *                  Event result of Vendor Specific API.
 *              : st_ble_vs_evt_data_t *p_data -
 *                  Event parameters of Vendor Specific API.
 * Return Value : none
 *********************************************************************************/
void vs_cb(uint16_t type, ble_status_t result, st_ble_vs_evt_data_t *p_data)
{
/* Hint: Input common process of callback function such as variable definitions */
/* Start user code for vender specific callback function common process. Do not edit comment generated here */
    ble_status_t status;
    fsp_err_t err;
    static uint8_t errors_ota;
/* End user code. Do not edit comment generated here */

    R_BLE_SERVS_VsCb(type, result, p_data);
    switch(type)
    {
        case BLE_VS_EVENT_GET_ADDR_COMP:
        {
            /* Start advertising when BD address is ready */
            st_ble_vs_get_bd_addr_comp_evt_t * get_address = (st_ble_vs_get_bd_addr_comp_evt_t *)p_data->p_param;
            memcpy(g_ble_advertising_parameter.own_bluetooth_address, get_address->addr.addr, BLE_BD_ADDR_LEN);
            RM_BLE_ABS_StartLegacyAdvertising(&g_ble_abs0_ctrl, &g_ble_advertising_parameter);
            break;
        }


/* Hint: Add cases of vender specific event macros defined as BLE_VS_XXX */
/* Start user code for vender specific callback function event process. Do not edit comment generated here */
        case BLE_VS_EVENT_GET_FW_VERSION_COMP:
        {
            /* Received BLE Module FW Version info */
            st_ble_vs_get_fw_version_comp_evt_t * get_version = (st_ble_vs_get_fw_version_comp_evt_t *)p_data->p_param;
            g_ble_version_data.major = get_version->major;
            g_ble_version_data.minor = get_version->minor;
            break;
        }
```

```
        /* Handle BLE FW App OTA initiated FW update (Mobile app to RYZ012 Module) */
        case BLE_VS_EVENT_OTA_START_NOTIFY :
        {
            /* Firmware update OTA process has started. */
            APP_PRINT("\r\n BLE_VS_EVENT_OTA_START_NOTIFY event received \r\n");
            APP_PRINT("\r\n FW OTA Started \r\n");
            errors_ota = 0;
            GPT_Start_Watchdog();
            APP_PRINT("\r\n OTA Watchdog Started \r\n");
            break;
        }

        /* Handle BLE FW App OTA completed FW update (Mobile app to RYZ012 Module) */
        case BLE_VS_EVENT_OTA_END_NOTIFY :
        {
            /* Firmware update OTA process has completed. */
            APP_PRINT("\r\n BLE_VS_EVENT_OTA_END_NOTIFY event received \r\n");
            APP_PRINT("\r\n FW OTA Completed \r\n");
            errors_ota = 0;
            GPT_Stop_Watchdog();
            R_BLE_VS_RestartModule();
            break;
        }

        /* Handle BLE FW App OTA error during FW update (Mobile App to radio module) */
        case BLE_VS_EVENT_OTA_ERROR_NOTIFY :
        {
            /* Error reported during Firmware OTA process */
            APP_PRINT("\r\n BLE_VS_EVENT_OTA_ERROR_NOTIFY event received \r\n");
            errors_ota++;
            /* Prevent In-operability of RYZ012 PMOD (Init BLE Driver and Profiles Init : Failed Returned Error Code 0x5)
               which must be cleared by reprogramming FW with Renesas Module Programmer */
            if ((errors_ota >= 2) || (true == GPT_Is_Watchdog_Expired()))
            {
                /* Too many packet transfer errors reported by RYZ012 module
                   Trap the condition and disconnect the Mobile Device to stop FW OTA
                   else the RYZ012 will get "bricked" and will require to be programmed with FW by Renesas Module Programmer */
                if (true == GPT_Is_Watchdog_Expired())
                {
                    APP_PRINT("\r\n OTA Timed Out\r\n");
                }
                else
                {
                    APP_PRINT("\r\n Too Many Failed Packet Transfers\r\n");
                }
                APP_PRINT("\r\n Stopping FW OTA \r\n");
                APP_PRINT("\r\n Restarting RYZ012 : Reconnect Mobile Device to RYZ012 and Restart the FW OTA Update\r\n");
                GPT_Stop_Watchdog();
                /* Stop the Radio Module OTA process */
                R_BLE_VS_RestartModule();
```

```
                /* Stop the Radio Module OTA process */
                R_BLE_VS_RestartModule();
                /* On RYZ012 restart, BLE_VS_EVENT_MODULE_READY_COMP event will be received */
            }
            break;
        }


        /* The module has finished restarting and the MODULE_READY event has been received. */
        case BLE_VS_EVENT_MODULE_READY_COMP:
        {
            APP_PRINT("\r\nRYZ012 READY -> BLE_VS_EVENT_MODULE_READY_COMP event received \r\n");
            /* Close the BLE driver so that it can be re-initialized and run updated firmware. */
            err = RM_BLE_ABS_Close(&g_ble_abs0_ctrl);
            if (FSP_SUCCESS != err)
            {
                APP_ERR_PRINT("\r\n ** Close BLE Driver : Failed ** \r\n");
                APP_ERR_TRAP(err);
            }

            /* Initialize BLE and profiles */
            status = ble_init();
            if (FSP_SUCCESS != status)
            {
                APP_ERR_PRINT("\r\n ** BLE Driver and Profiles Init : Failed ** \r\n");
                APP_ERR_TRAP(status);
            }

            /* Get RYZ012 FW Version that is running */
            get_fw_version();
            break;
        }

        default:
        {
            APP_PRINT("\r\n Un-handled BLE_VS_EVENT Event Received = %d \r\n",type);
            break;
        }
/* End user code. Do not edit comment generated here */
    }
}
```

The steps of execution are:

1. Send command for RYZ012 to start Advertising “RYZ012” and indicate OTA Service is available.
2. The MCU will read the current running RYZ012 firmware version and display on RTT Viewer.
3. The RYZ012 waits for BLE Device connection and notifies the MCU when a device is connected.
4. When the BLE Device OTA app starts a firmware upgrade, the RYZ012 sends an *BLE_VS_EVENT_OTA_START_NOTIFY* event to the MCU which is displayed on RTT Viewer. The OTA Watchdog Timer is started to monitor the upgrade process.
5. If errors are detected by the RYZ012 during the upgrade, *BLE_VS_EVENT_OTA_ERROR_NOTIFY* event is sent to MCU. This event can indicate a BLE Device disconnection, or failed firmware packet transfers. If more than two error events are detected, the OTA upgrade is halted by the MCU with an RYZ012 restart.
6. If the OTA Watchdog Timer expires, then the RYZ012 is restarted to stop the OTA upgrade and clear the error condition. Advertising is restarted on the RYZ012 after reboot is completed. The OTA app can now be re-connected to try the firmware upgrade again.
7. When the Device OTA app completes the firmware upgrade, the RYZ012 sends an *BLE_VS_EVENT_OTA_END_NOTIFY* event to the MCU which is displayed on RTT Viewer. The MCU restarts the RYZ012 module with `R_BLE_VS_RestartModule()` to run the new firmware.
8. After the RYZ012 reboot is completed, a device ready event *BLE_VS_EVENT_MODULE_READY_COMP* is sent by the RYZ012. The RYZ012 firmware version is then read by the MCU to validate the upgrade was successful.

4.7 SPP BLE GAP Events

The FSP BLE SPP API provides a GAP Callback **gap_cb(type, result, …)** that handles the BLE GAP Events received during the RYZ012 firmware upgrade process. The callback provides the ability for the MCU to monitor the RYZ012 BLE connection status to the Mobile Device.

```
void gap_cb(uint16_t type, ble_status_t result, st_ble_evt_data_t *p_data)
{
/* Hint: Input common process of callback function such as variable definitions */
/* Start user code for GAP callback function common process. Do not edit comment generated here */
    // User Specific RTT Viewer event logging for RYZ012 FUOTA
    if ((BLE_GAP_EVENT_CONN_IND == type) && (BLE_SUCCESS == result))
    {
        // BLE Device Connected
        APP_PRINT("\r\n RYZ012 Connected to Mobile Device \r\n");
    }
    else if (BLE_GAP_EVENT_DISCONN_IND == type)
    {
        // BLE Device Disconnected
        APP_PRINT("\r\n RYZ012 Disconnected from Mobile Device \r\n");
    }
/* End user code. Do not edit comment generated here */

    switch(type)
    {
        case BLE_GAP_EVENT_STACK_ON:
        {
            /* Get BD address for Advertising */
            R_BLE_VS_GetBdAddr(BLE_VS_ADDR_AREA_REG, BLE_GAP_ADDR_RAND);
        } break;

        case BLE_GAP_EVENT_CONN_IND:
        {
            if (BLE_SUCCESS == result)
            {
                /* Store connection handle */
                st_ble_gap_conn_evt_t *p_gap_conn_evt_param = (st_ble_gap_conn_evt_t *)p_data->p_param;
                g_conn_hdl = p_gap_conn_evt_param->conn_hdl;
            }
            else
            {
                /* Restart advertising when connection failed */
                RM_BLE_ABS_StartLegacyAdvertising(&g_ble_abs0_ctrl, &g_ble_advertising_parameter);
            }
        } break;
```

5. How to Make and Configure a New Project

This section describes the configuration to create an RA MCU based firmware OTA update application for the RYZ012. The application that is generated will include the system peripheral initialization, BLE Profile, and BLE API framework that will then need to be customized to include the RYZ012 module firmware update sequence. The update sequence is implemented by calling the FSP provided update and utility functions detailed in section 4, Application Project Implementation.

5.1 Create a New Project

Please see the **Renesas Flexible Software Package (FSP) User's Manual (R11UM0155EU)** for instructions on how to create a new project in **e² studio** for a BareMetal environment, choose **BareMetal – Minimal**. Follow the instructions to select the FSP Version, Board, Device, and Toolchains. Continue through all prompts until the project is created. Next the BSP Heap and Stack must be configured.

5.2 BSP Heap and Stack Configuration

Set heap and stack configuration as follows on the FSP configuration **BSP** tab. If the properties tab is not visible, choose **Window** > **Show View** > **Properties** on the **e² studio** menu bar.

- [RA Common] > [Main stack size (bytes)] : 0x4000
- [RA Common] > [Heap size (bytes)] : 0x1000

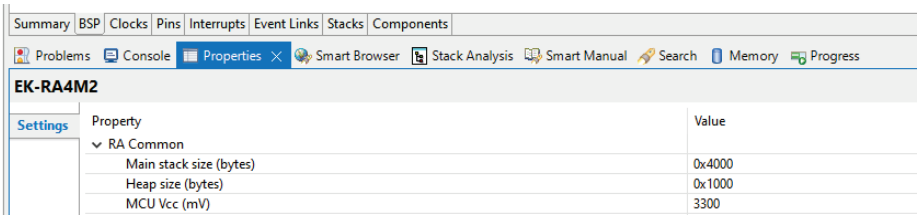


Figure 23. BSP Configuration

5.3 Add the I/O Port Stack

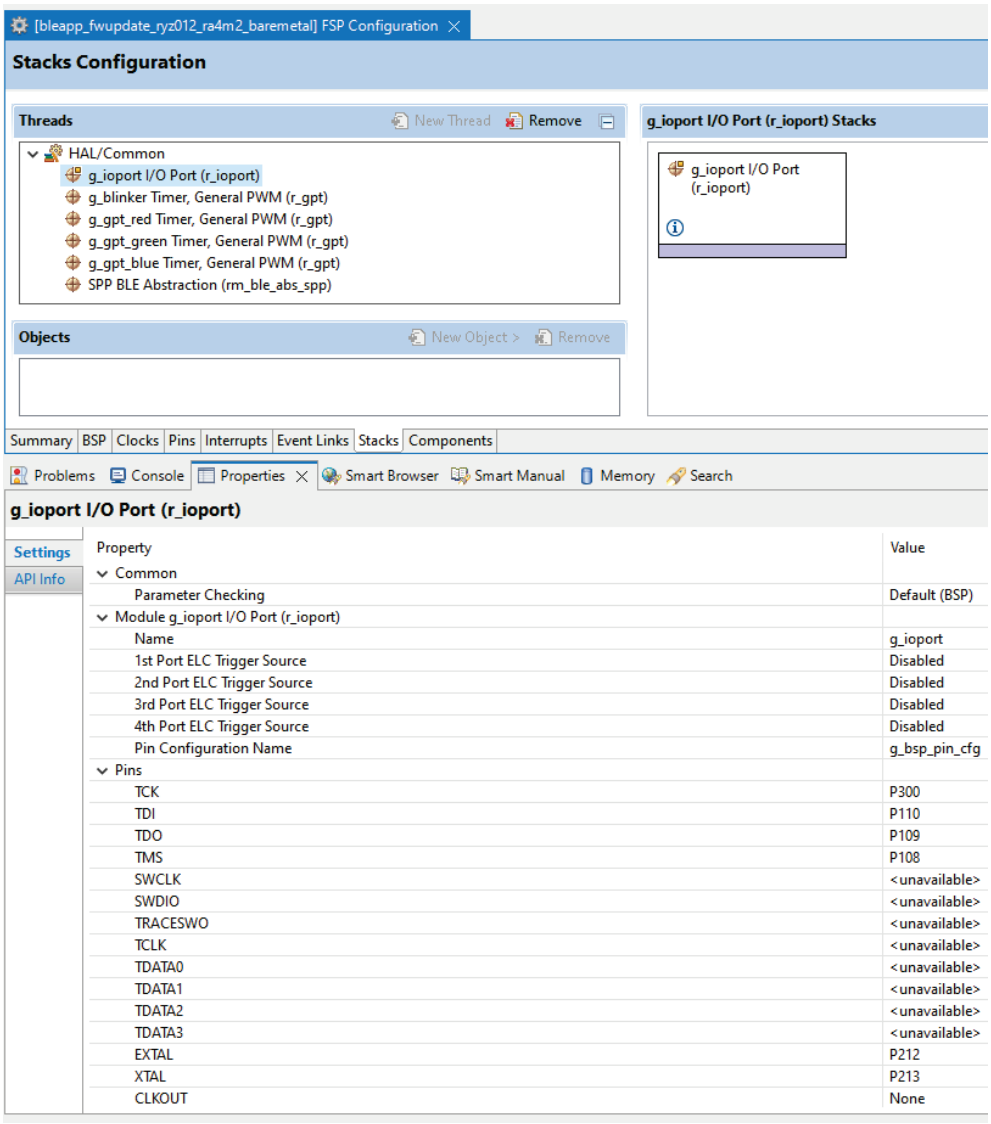


Figure 24. Adding I/O Stack

5.4 Add and Configure the SPP BLE Module

This section describes how to add and configure SPP BLE Abstraction Driver for communicating with the RYZ012 module in the application. Open **configuration.xml** in the project and add / configure **SPP BLE Abstraction Driver** on FSP configuration **Stacks** tab.

The procedure to add the SPP BLE Abstraction Driver is different for BareMetal and FreeRTOS environments. This document describes the procedure for BareMetal environment.

Add RYZ012 module in BareMetal environment

Click **New Stack** and add **Networking > SPP BLE Abstraction (rm_ble_abs_spp)** to **HAL/Common**. If the stack cannot be found use the **Search** box in e^2 studio since stack categories and organization may change in future FSP releases.

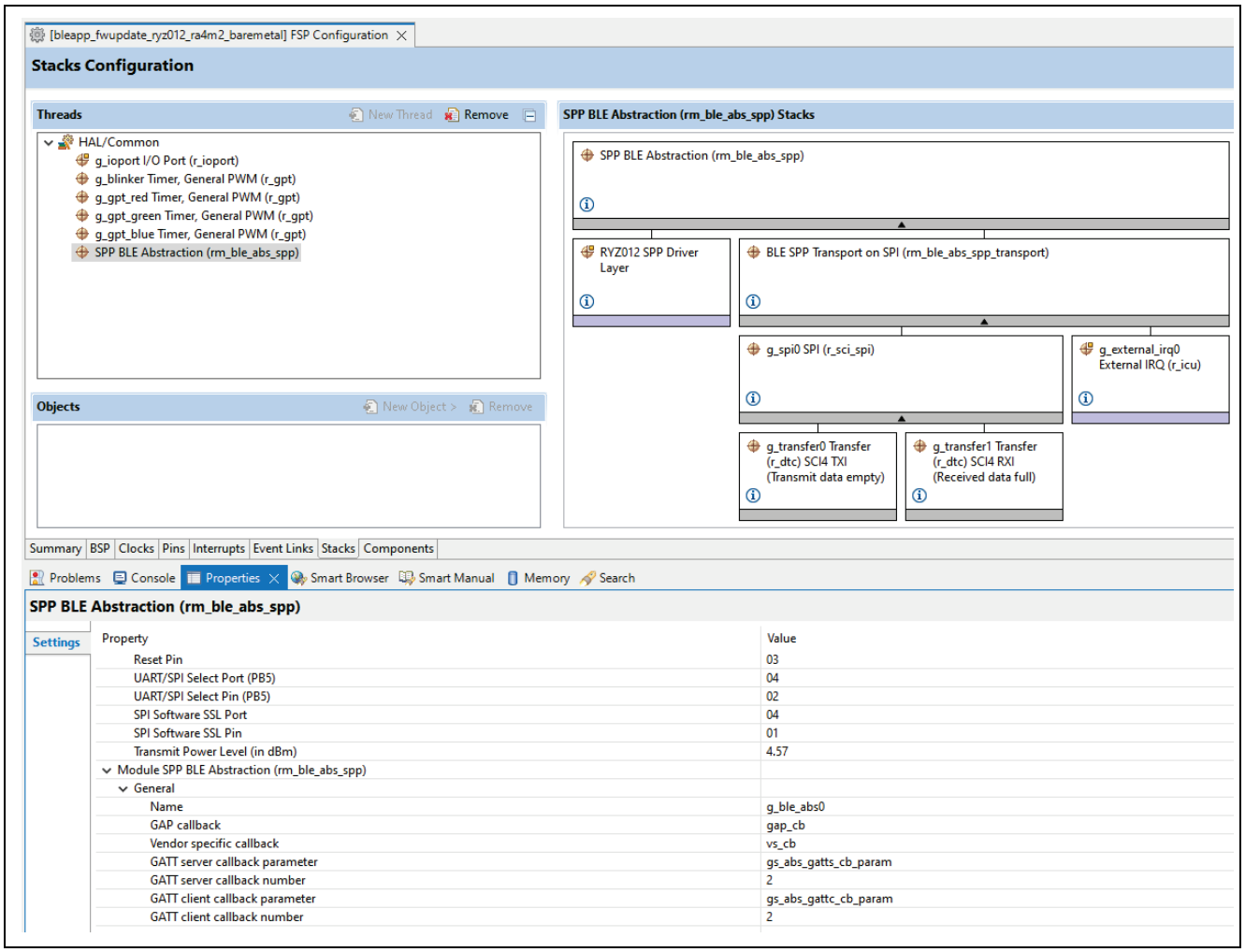


Figure 25. SPP BLE ABS Module

5.4.1 SPP BLE Abstraction Driver Configuration

This section describes the SPP BLE Abstraction Driver configuration options and related modules. SPP BLE Abstraction Driver includes the following configuration for the **EK-RA4M2** and **SPI comms** interface to **PMOD1 with RYZ012**. Users can modify these configurations according to their own specific hardware requirements. See Figure 25, SPP BLE ABS Module for more details.

Table 5. SPP Module Configuration

Configuration options	Comment
Reset port EK-RA4M2 case: 04	Specify port number of Reset Pin
Reset Pin EK-RA4M2 case: 03	Specify pin number of Reset Pin
UART / SPI Select Port (PB5) EK-RA4M2 case: 04	Specify port number of PMOD1 Mode Select Pin.
UART / SPI Select Pin (PB5) EK-RA4M2 case: 02	Specify pin number of PMOD1 Mode Select Pin. Configure mode as SPI
SPI Software SSL Port EK-RA4M2 case: 04	Specify port number of SPI Slave Select (SS)
SPI Software SSL Pin EK-RA4M2 case: 01	Specify pin number of SPI Slave Select (SS)
Transmit Power Level (in dBm) EK-RA4M2 case: 4.57dBm	Specify required transmit power level.

Gap callback Default: gap_cb	Do NOT change.
Vendor specific callback Default: vs_cb	Do NOT change.
GATT server callback parameter Default: gs_abs_gatts_cb_param	Do NOT change.
GATT server callback number Default: 2	Do NOT change.
GATT client callback parameter Default: gs_abs_gattc_cb_param	Do NOT change.
GATT client callback number Default: 2	Do NOT change.

5.4.2 Configure Peripherals for SPP BLE Abstraction Driver

The SPP BLE Abstraction Driver uses the SPI Driver on r_sci_spi to communicate with RYZ012 module attached to the RA4M2 with PMOD1 connector (J26). This section describes how to configure the SPI driver. See additional information in section 6, Next Steps for using the SPP BLE Driver with other communication interfaces such as UART.

1. Click g_spi0 SPI Driver on r_sci_spi.

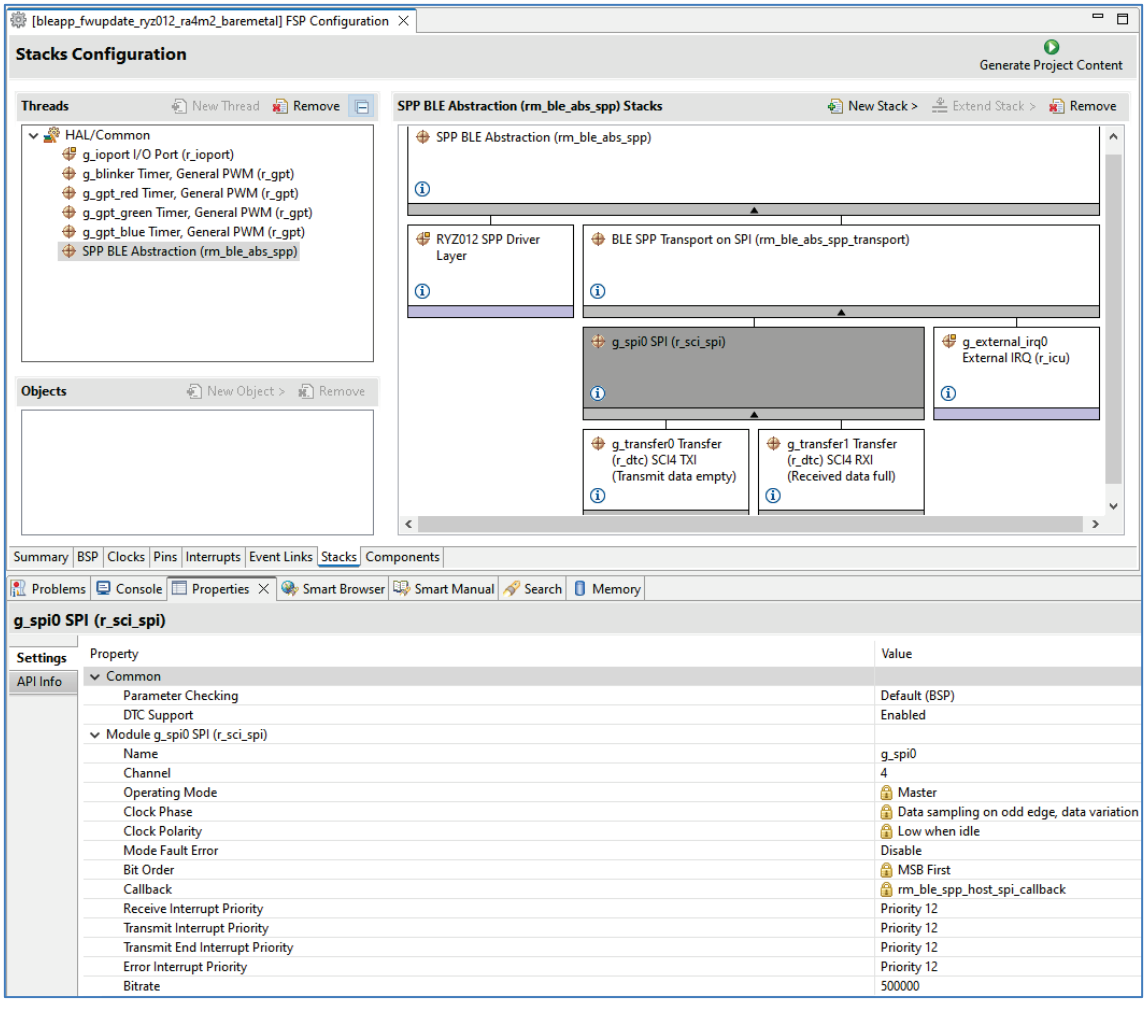


Figure 26. SPI Module Properties

See Figure 26, SPI Module Properties. For the SPI configuration, it is necessary to specify which SCI channel to use. Match the SCI channel used and configure it as the SPI interface for PMOD1 in the **Pins Tab** see Figure 28, Pin Configuration > SCI to configure SPI Mode and Pins. Here SCI Channel 4 is being used

on the RA4M2 in SPI Mode with the Pins assigned to PMOD1 (conn J26). See the Schematic for EK-RA4M2 for more details on the PMOD1 signals and pin assignments.

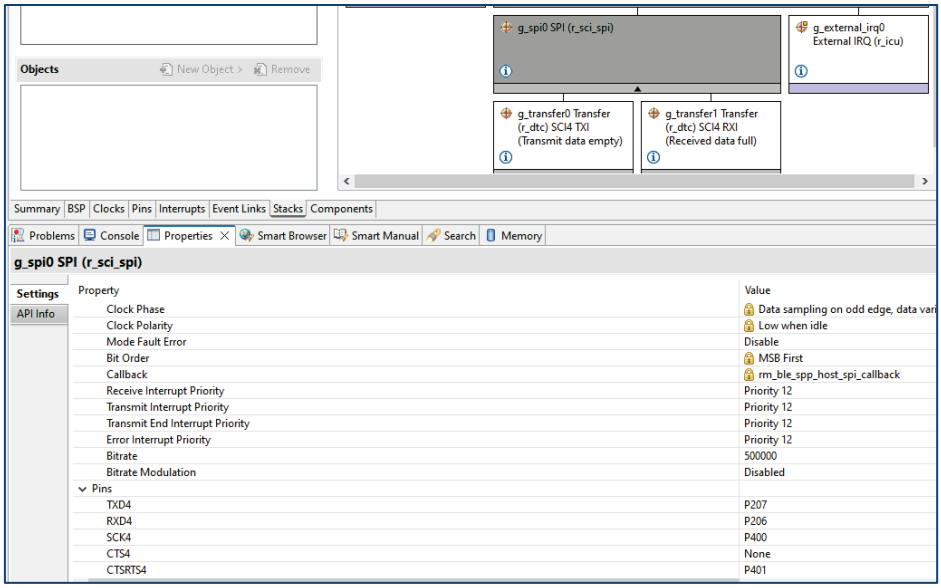


Figure 27. SPI Module Properties – Pin Assignments

2. Change the following SPI Configuration Properties in the properties window as in Figure 26 and Figure 27.

Table 6. SPI configuration

Property	Changed Value	Default Value
Common→DTC Support	Enabled	Disabled
Module g_spi0 SPI→Channel	4 for EK-RA4M2 case	0
Pins→TXD4	P207	None
Pins→RXD4	P206	None
Pins→SCK4	P400	None
Pins→CTS4	None	None
Pins→CTSRTS4	P401	None

3. Add the DTC driver (r_dtc) to both transmit and receive. See Figure 26, SPI Module Properties, `g_transfer0 SCI4 TXI` and `g_transfer1 SCI4 RXI` for more details.

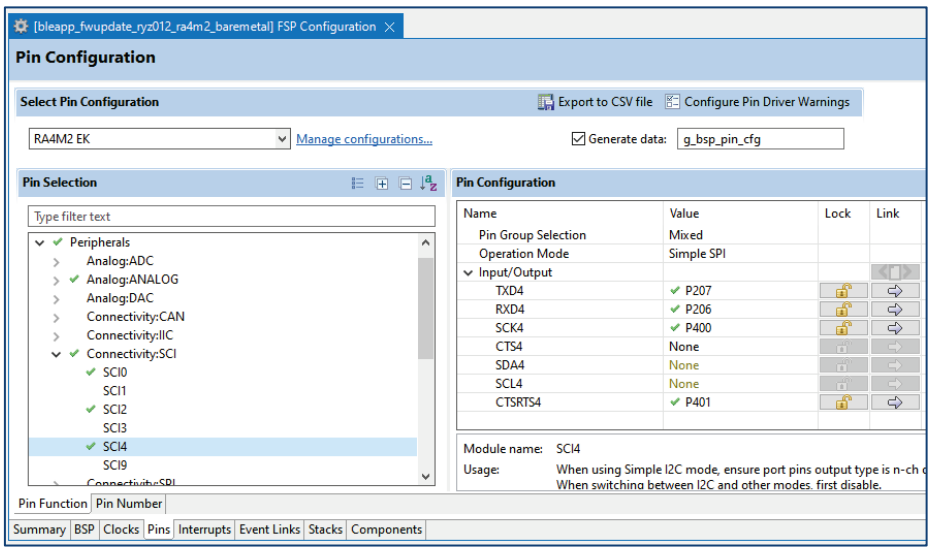


Figure 28. Pin Configuration > SCI

For the `TXD_MOSI` and `RXD_MISO` pin assignments, it is necessary to specify which SCI channel to use on the **Pins** tab. For the EK-RA4M2 with RYZ012 PMOD connected to PMOD1 (conn J26), SPI is assigned to SCI Channel 4 (SCI4) and the **Operation Mode** set to **Simple SPI**. The Input/Output Pins are assigned as specified in Figure 28, Pin Configuration > SCI. If your application uses a different RA MCU or PMOD connector then refer to the RA MCU Schematic to determine the Pin assignments for the PMOD connector.

In the **Pin Configuration**>**Ports**, confirm that the **Ports** > **P4 Pin assignments** match the SPI Configuration in Table 5**,** SPP Module Configuration. These Pin assignments are for the EK-RA4M2 and RYZ012 connected to PMOD1 connector (J26). You will need to modify these assignments when using a different PMOD connector or evaluation kit / board type.

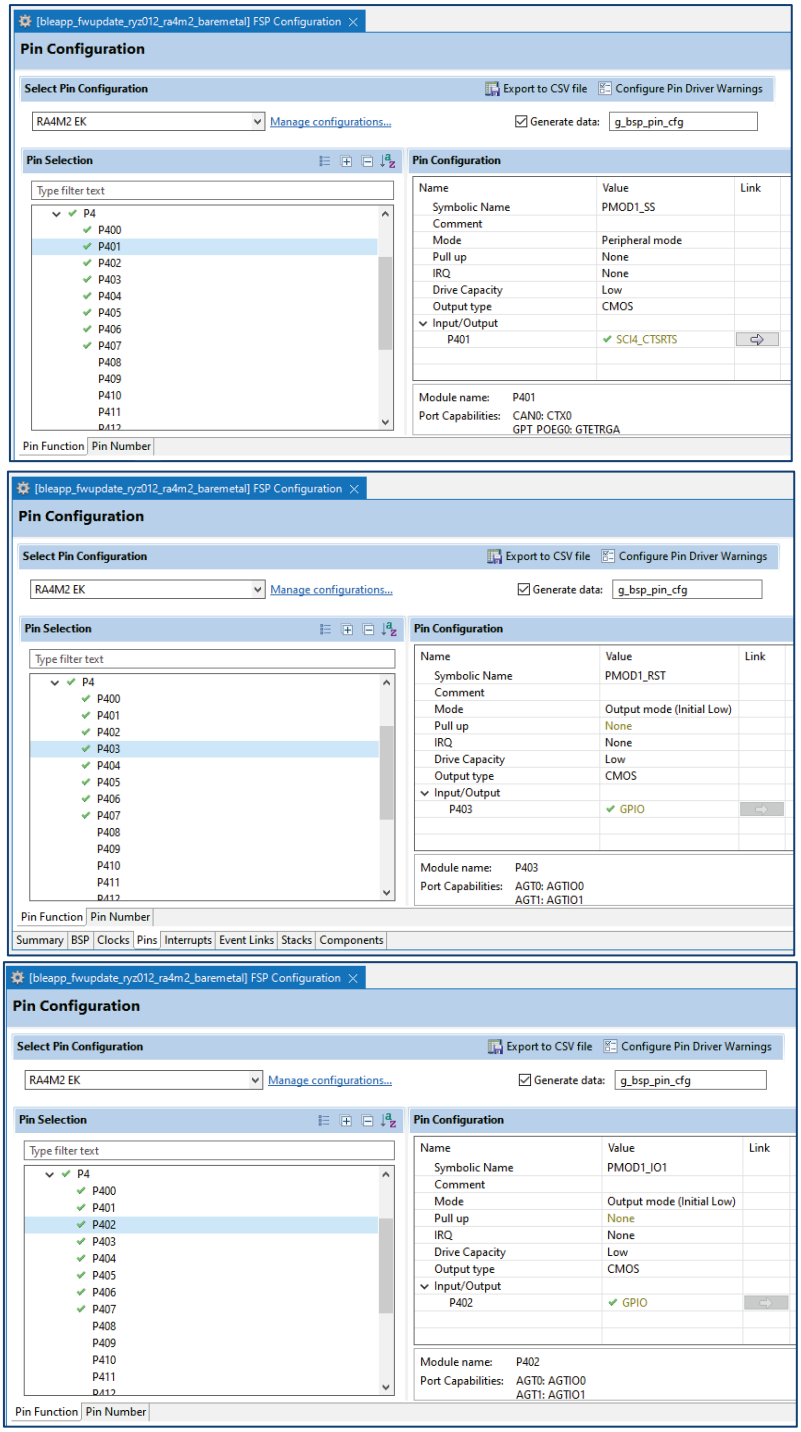
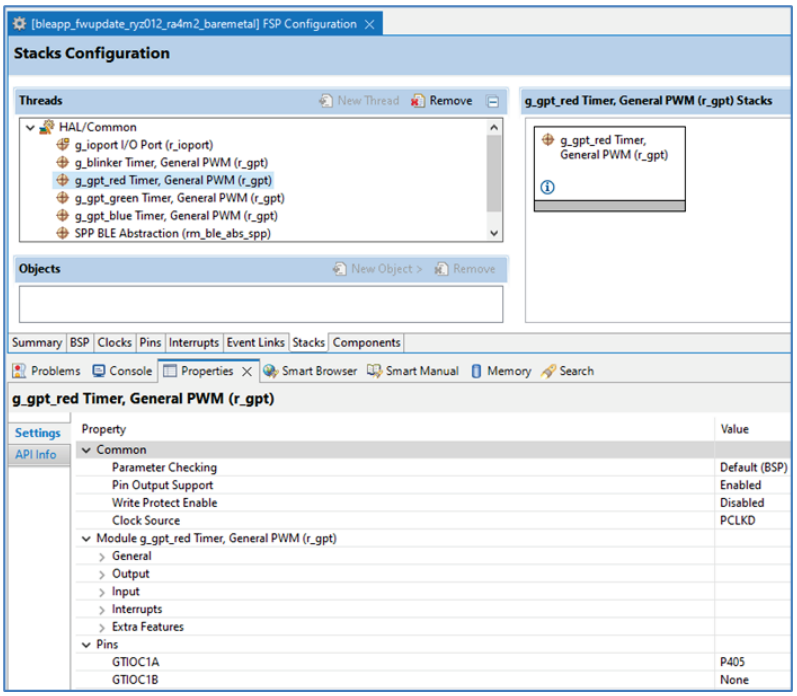
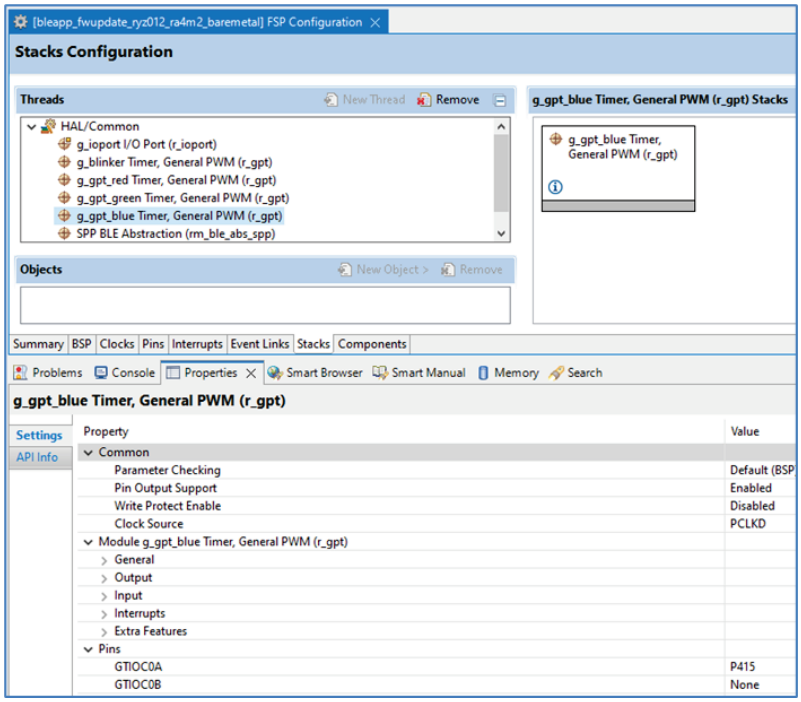


Figure 29. Pin Configuration > Ports


[bleapp_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration
Stacks Configuration
Threads
New Thread
Remove
HAL/Common
g_ioport I/O Port (r_ioport)
g_blinker Timer, General PWM (r_gpt)
g_gpt_red Timer, General PWM (r_gpt)
g_gpt_green Timer, General PWM (r_gpt)
g_gpt_blue Timer, General PWM (r_gpt)
SPP BLE Abstraction (rm_ble_abs_spp)
g_gpt_red Timer, General PWM (r_gpt) Stacks
g_gpt_red Timer, General PWM (r_gpt)
Objects
New Object >
Remove
Summary BSP Clocks Pins Interrupts Event Links Stacks Components
Problems Console Properties Smart Browser Smart Manual Memory Search
g_gpt_red Timer, General PWM (r_gpt)
Settings
API Info
Property
Value
Common
Parameter Checking
Default (BSP)
Pin Output Support
Enabled
Write Protect Enable
Disabled
Clock Source
PCLKD
Module g_gpt_red Timer, General PWM (r_gpt)
General
Output
Input
Interrupts
Extra Features
Pins
GTIOC1A
P405
GTIOC1B
None


[bleapp_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration
Stacks Configuration
Threads
New Thread
Remove
HAL/Common
g_ioport I/O Port (r_ioport)
g_blinker Timer, General PWM (r_gpt)
g_gpt_red Timer, General PWM (r_gpt)
g_gpt_green Timer, General PWM (r_gpt)
g_gpt_blue Timer, General PWM (r_gpt)
SPP BLE Abstraction (rm_ble_abs_spp)
g_gpt_blue Timer, General PWM (r_gpt) Stacks
g_gpt_blue Timer, General PWM (r_gpt)
Objects
New Object >
Remove
Summary BSP Clocks Pins Interrupts Event Links Stacks Components
Problems Console Properties Smart Browser Smart Manual Memory Search
g_gpt_blue Timer, General PWM (r_gpt)
Settings
API Info
Property
Value
Common
Parameter Checking
Default (BSP
Pin Output Support
Enabled
Write Protect Enable
Disabled
Clock Source
PCLKD
Module g_gpt_blue Timer, General PWM (r_gpt)
General
Output
Input
Interrupts
Extra Features
Pins
GTIOC0A
P415
GTIOC0B
None

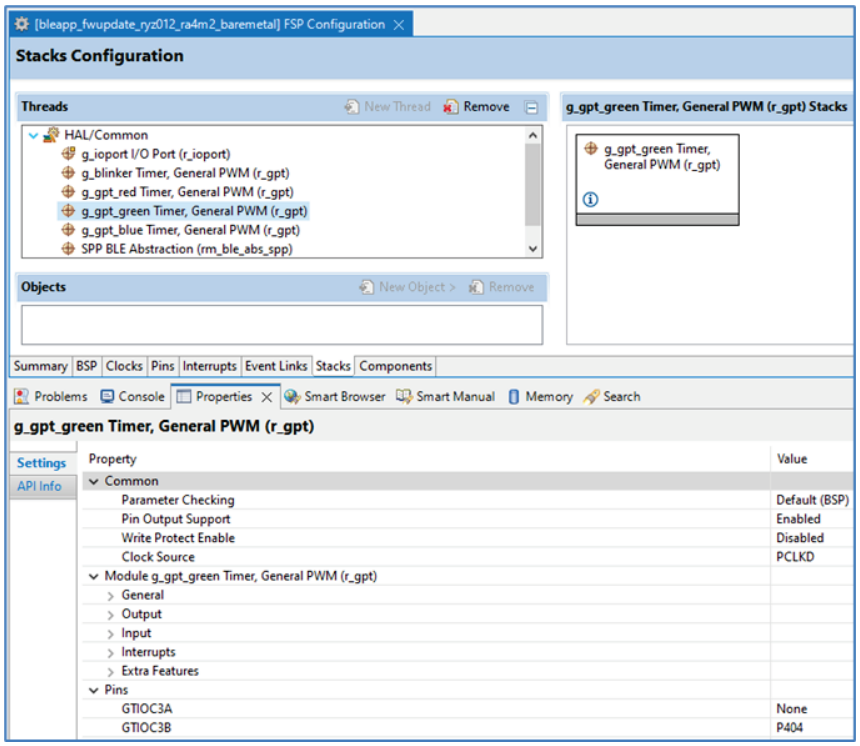
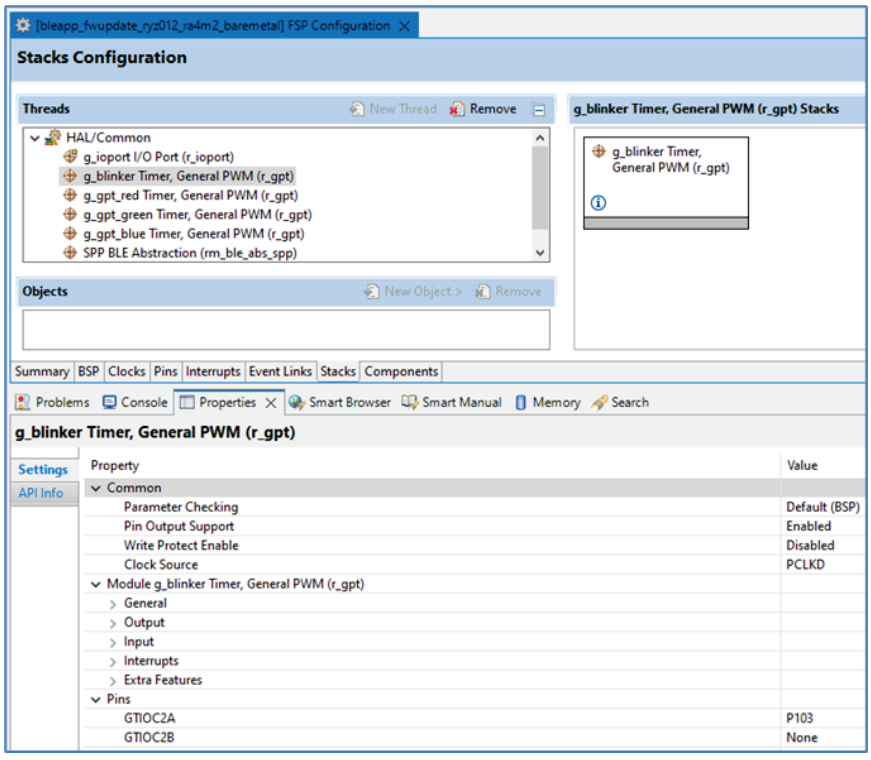


Figure 30. Stack Configuration Settings



5.5 Add General PWM Timers

This step is required for Firmware Update OTA Watchdog functionality. In addition, it provides an indication on the board LEDs that the application is running. Add the General PWM Timers (`r_gpt`) for the LED blinker timer, watchdog, and board LEDs. See Figure 30, Stack Configuration Settings

5.6 Add RYZ012 BLE OTA Service Profile

QE for BLE can generate a custom BLE profile and the Bluetooth LE application skeleton code. You must modify this source code according to your application BLE functional requirements. See Renesas QE for BLE Tool information QE BLE Tool Dev Assistance Documents about the usage of QE for BLE.

The BLE Profile created in this application project is as follows. From the e² studio Menu go to **Renesas Views** > **Renesas QE > R_BLE Custom Profile RA,RE,RX (QE).** Once the BLE Custom Profile opens, then select the **Project:** <YourProject> and **Module:** RYZ012 from the drop-down boxes.

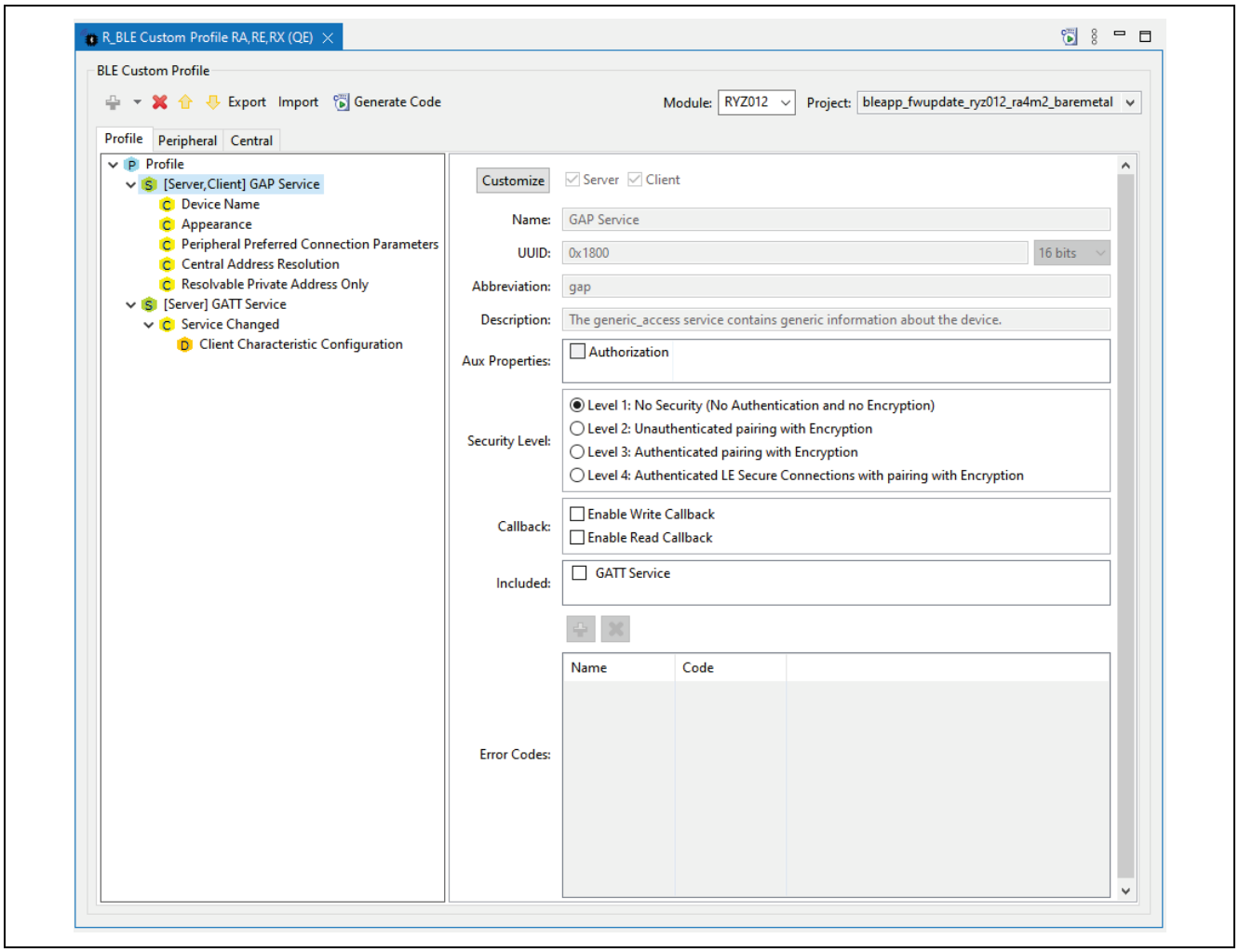


Figure 31. Creating BLE Profile with QE for BLE

The BLE Profile is shown above in Figure 31 is required for the OTA Firmware Update of the RYZ012A. Select the **Peripheral** tab to change **Local Name** to use for BLE Advertising in the RYZ012. Set **Complete local name** to "**RYZ012**". See Figure 32 Setting Complete Local Name.

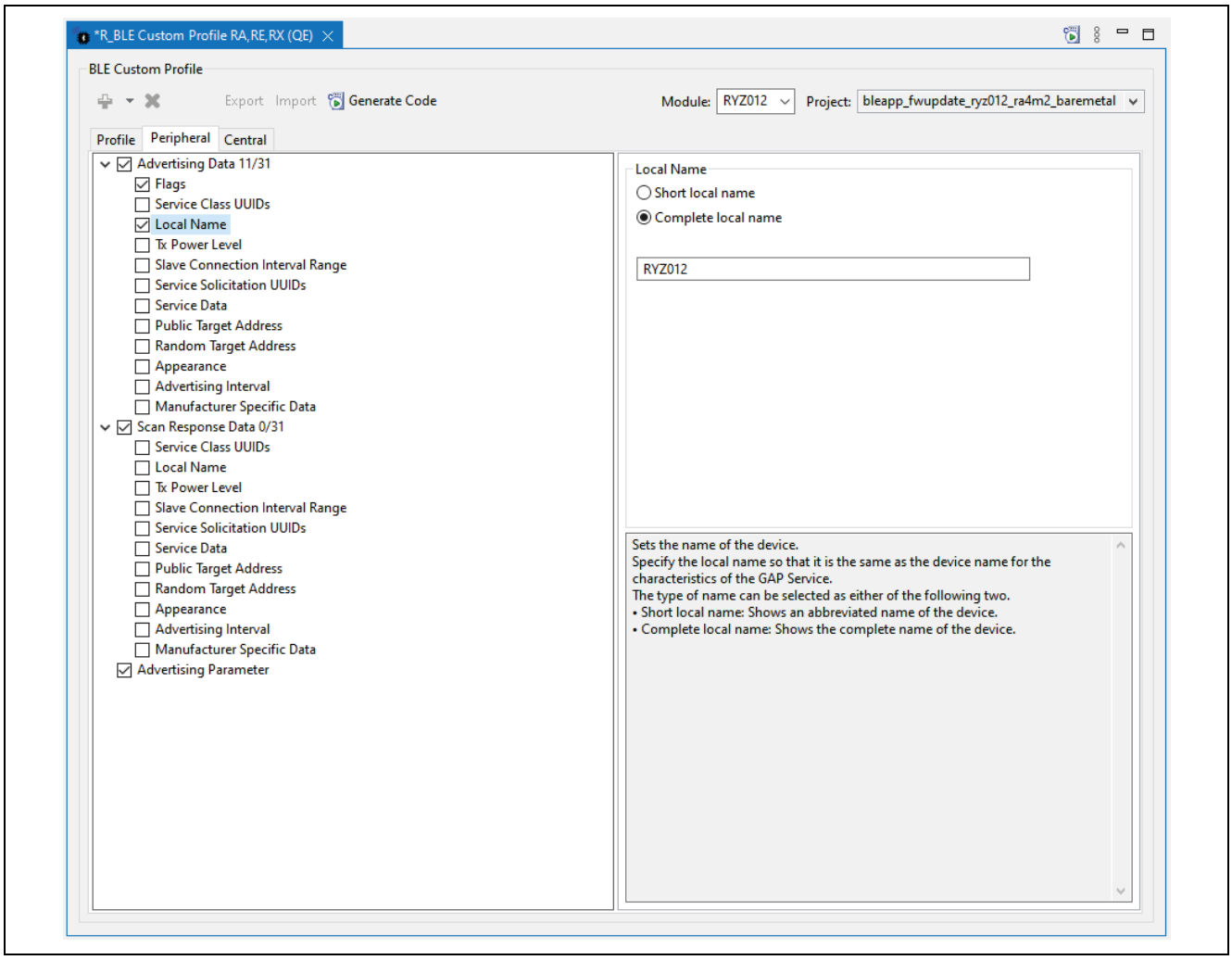


Figure 32. Setting Complete Local Name

The OTA app or GATT Browser app on the mobile device can be used to verify that the RYZ012 is running (advertising) after a firmware update.

After the BLE Profile is created for your application, press **Generate Code** to create the BLE application skeleton code and BLE Profile. The tool generated source code will appear under **/qe_gen/ble/app_main.c** and **qe_ble_profile.c.** Edit **qe_ble_profile.h** and **qe_ble_profile.c** files with the following code additions to add the OTA Service to the Profile sent to the RYZ012 at BLE initialization.

```
* DISCLAIMER
* File Name    : qe_ble_profile.c
* History      : MM/DD/YYYY Version Description

#include "qe_ble_profile.h"

const attribute_t qe_ble_profile[QE_ATTRIBUTE_HANDLE_PROFILE_END] =
{

    // Profile Declaration
    [0] =
    {
        .handle = 0,
        .encapsulated_attributes = QE_ATTRIBUTE_HANDLE_PROFILE_END,    // added OTA Service
        .permissions = 0x00,
        .uuid_length = 0x00,
        .value_length = 0x00,
        .notify_write = 0,
        .notify_read = 0
    },
    // Service Declaration: GAP Service
    [1] =
    {
        .handle = 1,
        .encapsulated_attributes = 11,
        .permissions = 0x01,
        .uuid_length = QE_BLE_PROFILE_UUID_SIZE_ADOPTED,
        .value_length = 0x02,
        .uuid = (uint8_t []){0x00,0x28},
        .value = (uint8_t []){0x00,0x18},
        .notify_write = 0,
        .notify_read = 0
    },
```

…. <Lines of Code are Omitted Here>

```
    // Descriptor: Client Characteristic Configuration
    [15] =
    {
        .handle = 15,
        .encapsulated_attributes = 0,
        .permissions = 0x03,
        .uuid_length = QE_BLE_PROFILE_UUID_SIZE_ADOPTED,
        .value_length = 0x02,
        .uuid = (uint8_t []){0x02,0x29},
        .value = (uint8_t []){0x00,0x00},
        .notify_write = 0,
        .notify_read = 0
    },
    // 16 :   OTA Service and Characteristic
    //!< OTA Service : required to perform RYZ012 FW Update with TelinkOTA mobile app, choose this Service on mobile
   [QE_ATTRIBUTE_HANDLE_SERVICE_DECL_OTA_SERVICE] =
   {
       .handle = QE_ATTRIBUTE_HANDLE_SERVICE_DECL_OTA_SERVICE,
       .encapsulated_attributes = 4,
       .permissions = 0x01,
       .uuid_length = QE_BLE_PROFILE_UUID_SIZE_ADOPTED,
       .value_length = QE_BLE_PROFILE_UUID_SIZE_CUSTOM,
       .uuid = (uint8_t []){0x00,0x28},
       .value = (uint8_t []){0x12,0x19,0x0d,0x0c,0x0b,0x0a,0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01,0x00},
       .notify_write = 0,
       .notify_read = 0
   },
   //!< OTA Service : character value
   [QE_ATTRIBUTE_HANDLE_CHARACTERISTIC_DECL_OTA_CHARVAL] =
   {
       .handle = QE_ATTRIBUTE_HANDLE_CHARACTERISTIC_DECL_OTA_CHARVAL,
       .encapsulated_attributes = 0,
       .permissions = 0x01,
       .uuid_length = QE_BLE_PROFILE_UUID_SIZE_ADOPTED,
       .value_length = 0x0013,
       .uuid = (uint8_t []){0x03,0x28},
       .value = (uint8_t []){ 0x02|0x04,0x48,0x00,0x12,0x2B,0x0d,0x0c,0x0b,0x0a,0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01,0x00},
       .notify_write = 0,
       .notify_read = 0
   },


    //!< OTA UUID : SPP data required to perform RYZ012 FW Update with TelinkOTA mobile app, choose this Characteristic on mobile
    [QE_ATTRIBUTE_HANDLE_CHARACTERISTIC_VALUE_OTA_UUID] =
    {
        .handle = QE_ATTRIBUTE_HANDLE_CHARACTERISTIC_VALUE_OTA_UUID,
        .encapsulated_attributes = 0,
        .permissions = 0x01 | 0x02,
        .uuid_length = QE_BLE_PROFILE_UUID_SIZE_CUSTOM,
        .value_length = 0x0001,
        .uuid = (uint8_t []){0x12,0x2B,0x0d,0x0c,0x0b,0x0a,0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01,0x00},
        .value = (uint8_t []){0x00},
        .notify_write = 0x02,
        .notify_read = 0x02
    },
    //!< OTA Name
    [QE_ATTRIBUTE_HANDLE_DESCRIPTOR_OTA_NAME] =
    {
        .handle = QE_ATTRIBUTE_HANDLE_DESCRIPTOR_OTA_NAME,
        .encapsulated_attributes = 0,
        .permissions = 0x01,
        .uuid_length = QE_BLE_PROFILE_UUID_SIZE_ADOPTED,
        .value_length = 0x0003,
        .uuid = (uint8_t []){0x01, 0x29},
        .value = (uint8_t []){'O', 'T', 'A'},
        .notify_write = 0,
        .notify_read = 0
     }
};
```

5.7 Build and Modify Application Skeleton Code

- Select **Project > Build Project** from the menu bar or click the Build icon to build the project.
- Click the debug icon to launch the project. When the project starts, the sample application will be downloaded to EK-RA4M2.

6. Next Steps

Now that you have a basic understanding of the process to update the RYZ012 firmware with a BLE Device App you may want to customize the application. Please consider this information as you make those changes:

1. If you need to change the RYZ012 to a different PMOD connector, use a different RA kit, or use UART comms instead of SPI with the RYZ012, then:

   To change RYZ012 Communications Interface Mode

   — To change the RYZ012 comms interface to UART see section 5.4**,** Add and Configure the SPP BLE Module**.** When adding the SPP BLE Abstraction Stack, add the UART for "**Transport Interface for communicating with the module"**.
   Similar to configuring SPI, see section 5.4.2, Configure Peripherals for SPP BLE Abstraction Driver, to configure the UART Ports and Pins required for the PMOD connector on your board. In the step where Input/Output Pins are assigned as specified in Figure 28, Pin Configuration > SCI, set the Operation Mode to **Asynchronous UART**.
   As a reference, see Figure 29, Pin Configuration > Ports, the PMOD Mode Select GPIO (UART or SPI mode is set by GPIO PMODx_IOx) must be configured to **UART** by setting the Pin Configuration > Mode to **Output mode (Initial High).**
   Consult the schematic for your particular board to determine the correct Ports / Pins for the PMOD connector signals used in your application. Configure and confirm the **Peripherals** > **SCIx settings** and **Ports** > **Px** > **Pin Configuration** settings for all the PMOD signals.
   If the PMOD signal pins do not appear as available in the Pin Configuration or Peripheral SCI channel, it may be required to disable unused SCI channels to gain access to the required PMOD Pins on your desired SCIx channel. Use **Peripherals** > **SCIx** > **Operation mode = Disabled**

   To change RYZ012 PMOD Connector

   — To use a different PMOD connector other than PMOD1 (J26):
   Consult the schematic for your particular board to determine the correct Ports / Pins for the PMOD connector signals used in your application.
   Follow section 5.4.2, Configure Peripherals for SPP BLE Abstraction Driver to configure the Ports & Pins and comms mode (UART or SPI) required for the PMOD connector on your board.
   Configure and confirm the **Peripherals** > **SCIx** settings and **Ports > Px > Pin Configuration** settings for all the PMOD signals.
   If the PMOD signal pins do not appear as available in the Pin Configuration or Peripheral SCI channel, it may be required to disable unused SCI channels to gain access to the required PMOD Pins on your desired SCIx channel. Use **Peripherals > SCIx > Operation mode** = **Disabled**

   To change the RA MCU kit or Custom Board

   — To use a different RA MCU kit or custom board design it may be required to reassign the communications interface Port / Pins for the RYZ012 PMOD connector on your board. You will also be required to change the e^2 studio FSP Configuration BSP settings for the RA MCU kit or custom board Device used.
   See **To change RYZ012 PMOD Connector** above.
   It may be required to change the board LED port and pin assignments.

See section 0

   Add General PWM Timer to change the LED pins.
   Keep in mind that your RA MCU must have enough Code Flash to contain the entire RYZ012 firmware image. See Item 4 below for more details.

2. To convert this e² studio project to use with IAR or Keil see the App Note for converting the project to IAR. **Converting Applications from e² studio to IAR or Keil for RA – Application Note** (R11AN0555EU0100)
   Renesas FSP v3.8.0 and higher must be used to support the RYZ012 (FW SPP SDK v5.4 and higher) for BLE Radio based firmware update.

7. Appendix - Windows PC OTA App

An alternative to using a mobile device app is to use a PC with Web-based app Telink OTA (pvvx.github.io) to upgrade the firmware with a Windows 10 PC's built-in Bluetooth support. Note this Web-based app is shown for illustrative purposes only and is not supported by Renesas. Make sure Bluetooth is enabled on the PC.

- To use the OTA app, open with web-browser at Telink OTA (pvvx.github.io).
- Press "**Connect**" to search for "RYZ012" and select Pair. See Figure 33.

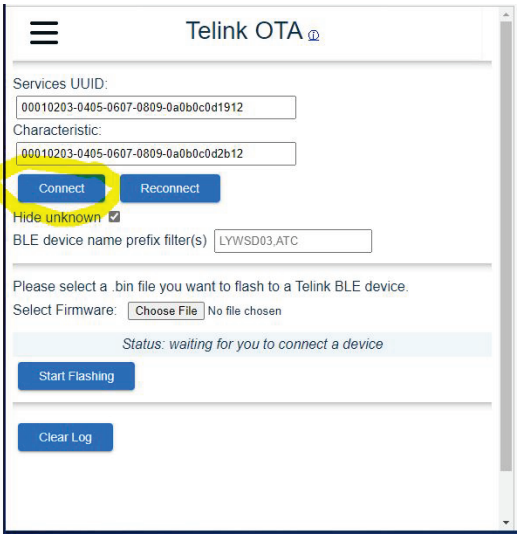


Figure 33. Windows PC Update – Press Connect and Search Devices

- Use "**Choose File**" to select Firmware BIN file from the e² studio project workspace in the project directory **bleapp_fwupdate_ryz012_ra4m2_baremetal/8258_moduleV5_4.bin** or **8258_moduleV5_5.bin.** See Figure 34.
- Start the firmware update with "**Start Flashing**". See Figure 35.
- Monitor firmware update progress on RTT Viewer and OTA app. See Figure 11 and 36.

In the event that the firmware upgrade fails with error messages shown in RTT Viewer, the section 2.4, RYZ012 Firmware Update Considerations will help you to troubleshoot and resolve the issues.

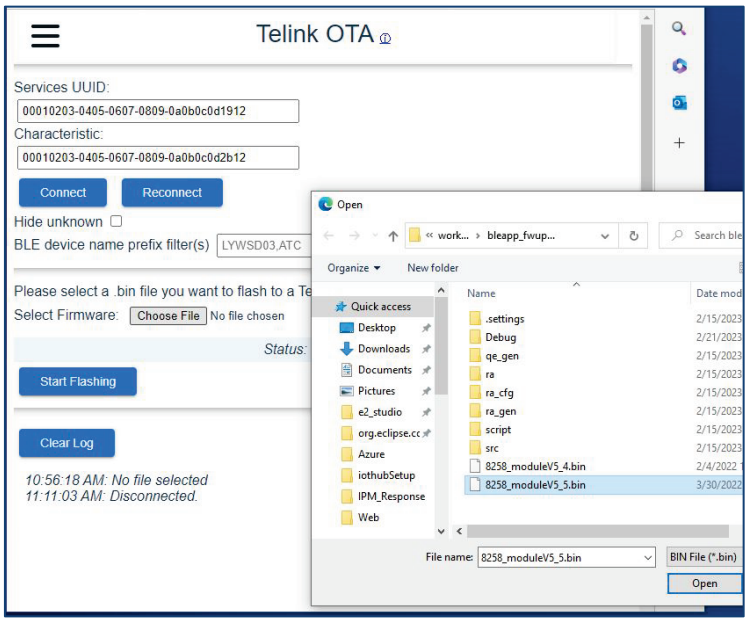


Figure 34. Windows PC Update – Choose Firmware Bin File

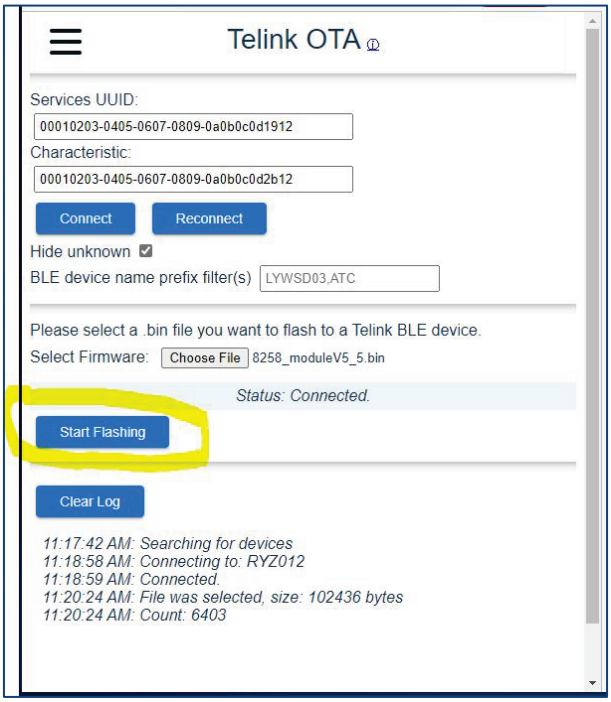


Figure 35. Windows PC Update – Start Firmware Update

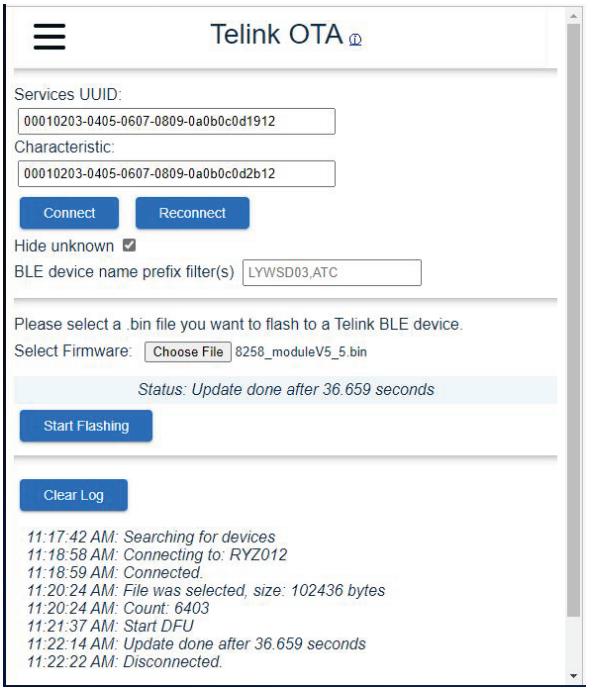


Figure 36. Windows PC Update – Firmware Upgrade Completed

Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas RA™ EK-RA4M2 kit	renesas.com/ra/ek-ra4m2
RYZ012x1 Bluetooth LE Module	renesas.com/ryz012x1-bluetooth-le-module
PMOD Expansion Board RYZ012x1	renesas.com/pmod-expansion-board-ryz012x1
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.09.23	—	Initial release
1.01	Sep.26.23	18	Revised section 2.4 RYZ012 Firmware Update Considerations

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/.