RYZ012 and RA MCU

Firmware Update from Host MCU

Introduction

This document describes a sample application that updates the RYZ012 firmware from the host MCU.

The application example works in a configuration that uses the EK-RA4M2 board with RA4M2 as the host MCU and connects the PMODTM Expansion Board for RYZ012 Bluetooth LE Module to the PMOD connector. The steps in this document show the user how to transfer an RYZ012 firmware file to the MCU host, and program the firmware into the RYZ012.

Target Devices

- RA4M2
- RYZ012

Related Documents

- Renesas Flexible Software Package (FSP) User's Manual (R11UM0155)
- e^2 studio 2022-10 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

- e^2 studio IDE v2022-10 (e2studio)
- Flexible Software Package (FSP) v4.2.0 (Flexible Software Package FSP)
- QE for BLE Tool [RA, RE, RX] V1.5.0 for e^2 studio IDE (QE for BLE: Tool for Bluetooth® Low Energy)
- SEGGER J-Link RTT Viewer V7.60f (SEGGER RTT Viewer)
- Tera Term V4.106 (Tera Term Open-Source Project)
- GATTBrowser v1.0.3 or Android or v1.1.4 iOS (see Google Play or iOS App Store)

Hardware

- Renesas RA™ EK-RA4M2 kit (RTK7EKA4M2S00001BE) EK-RA4M2
- PMOD Expansion Board for RYZ012x1 (RTKYZ012A1B00000BE) PMOD Expansion Board
  Must be programmed with v5.4 Firmware or later to support MCU based firmware updates
- PC running Windows® 10
- 2 x Micro USB cables

PMODTM is registered to Digilent Inc.

Contents

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 update. This application note covers an example of updating the RYZ012 module via the host MCU 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 or later to support MCU based firmware updates
Windows® 10 PC	---
2 x Micro USB Cables	for EK-RA4M2 USB Debug J10 connecter (micro-B) and USB FS J11 conn to PC Tera Term (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
e^2 studio IDE	2022-10
GCC Compiler	10.3.1
Renesas FSP	4.2.0
QE for BLE [RA,RE,RX]	V1.5.0 for e^2 studio IDE
SEGGER J-Link RTT Viewer	V7.60f
Tera Term	V4.106
GATT Browser	v1.0.3 Android, or v1.1.4 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.

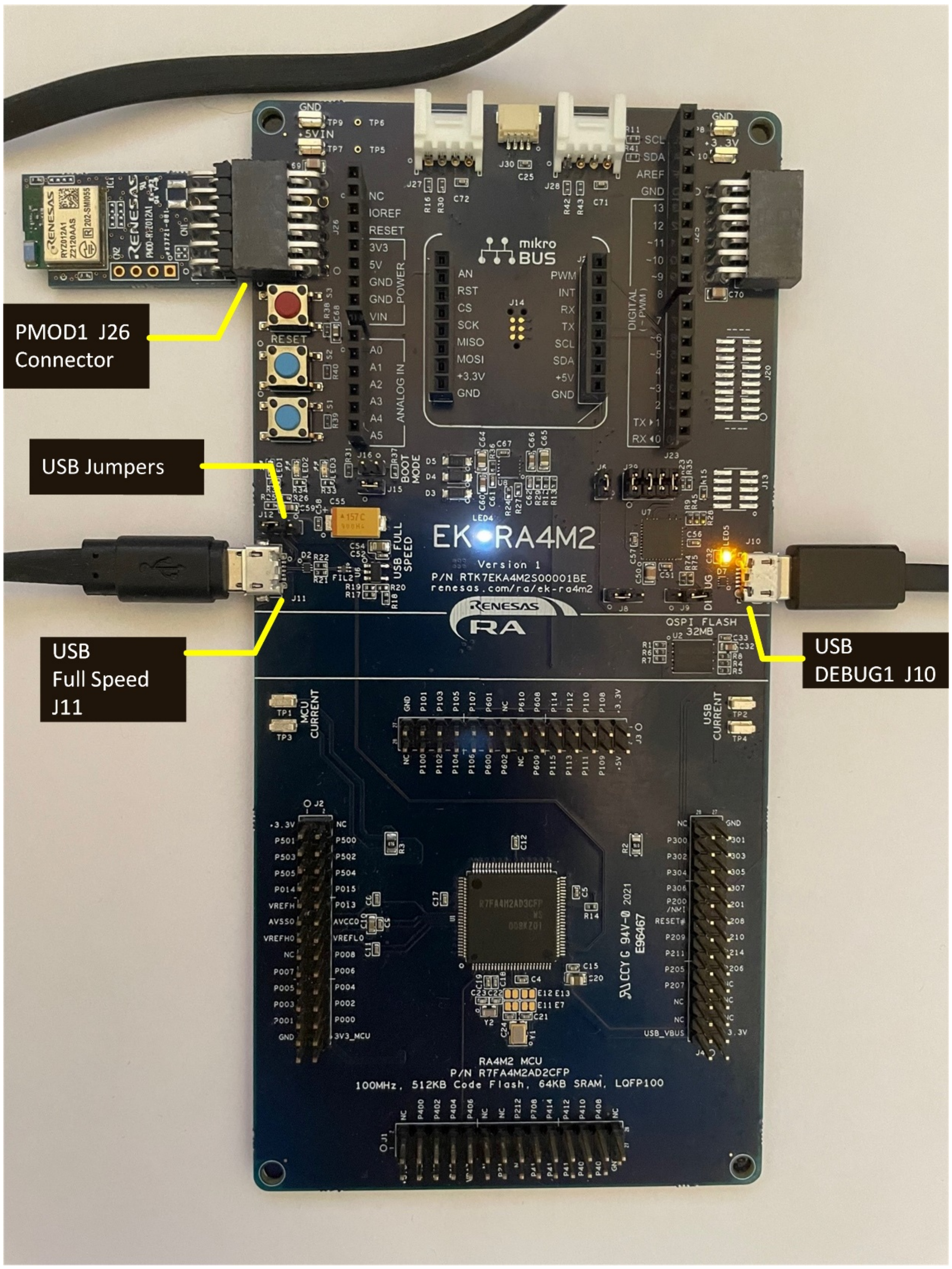


Figure 1. EK-RA4M2 and RYZ012 PMOD Assembly

1.1.4 EK-RA4M2 Jumper Setting

The USB Full Speed interface of EK-RA4M4 is used to communicate with Tera Term for XMODEM file transfer. Set the EK-RA4M2 J11 USB FS option jumpers as follows :

- J12: Jumper installed on pins 2-3
- J15: Jumper installed on pins 1-2

Connect the two micro-B USB cables to USB Full Speed (J11) and DEBUG1 (J10) with other cable ends to PC ports.

1.2 RYZ012 Firmware Update Process

In this application note, the RYZ012 module firmware is upgraded by a host MCU. 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 image files to use are supplied in the accompanying application project. See the root directory of the project that is imported into e^2 studio. 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 opened and transferred to the MCU with Tera Term XMODEM file transfer. The entire RYZ012 firmware image is stored in MCU Flash prior to upgrading the RYZ012 module. Tera Term is then used to trigger the upgrade process where the MCU checks for presence of the firmware image in MCU Flash memory and then updates the RYZ012 with the new image. The status of the upgrade process is shown on the RTT Viewer.

After the RYZ012 is updated, it restarts and runs the new image. Then 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 the following Figure 2 flow diagram.

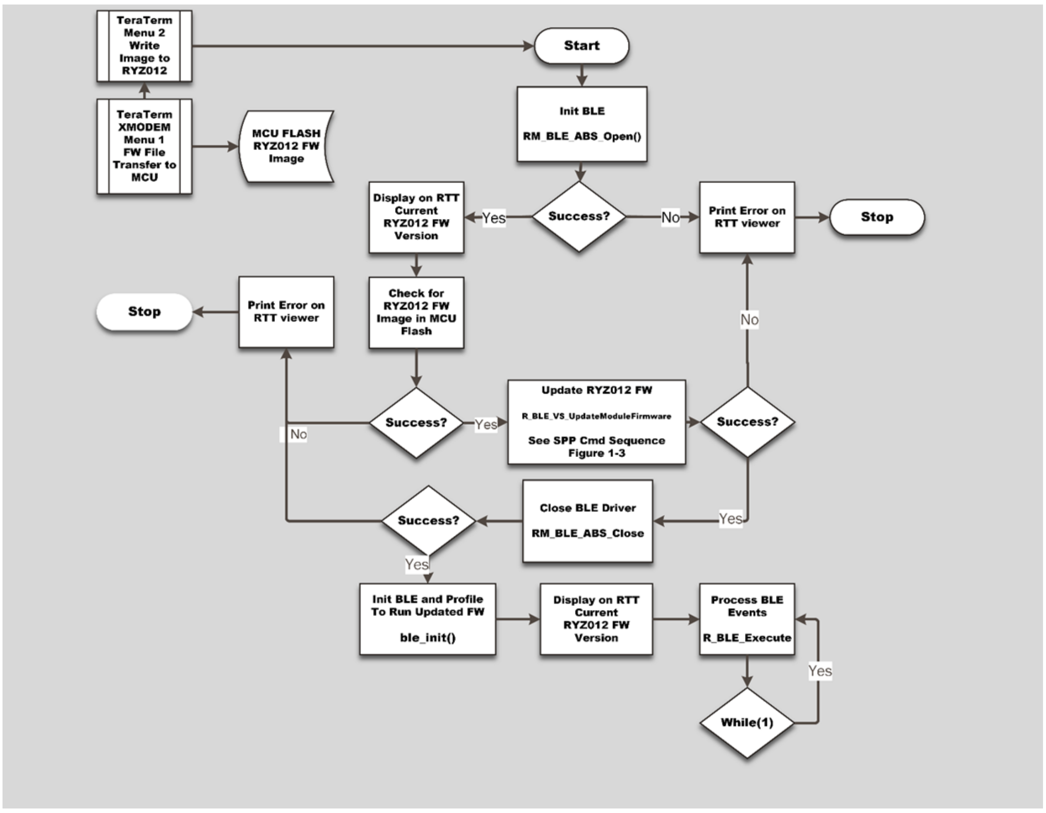

TeraTerm Menu 2 Write Image to RYZ012
TeraTerm XMODEM Menu 1 FW File Transfer to MCU
MCU FLASH RYZ012 FW Image
Start
Init BLE
RM_BLE_ABS_Open()
Success?
Yes
No
Display on RTT Current RYZ012 FW Version
Print Error on RTT viewer
Stop
Check for RYZ012 FW Image in MCU Flash
Print Error on RTT viewer
Stop
No
Success?
Yes
Update RYZ012 FW
R_BLE_VS_UpdateModuleFirmware
See SPP Cmd Sequence Figure 1-3
Success?
No
Yes
Close BLE Driver
RM_BLE_ABS_Close
Success?
No
Yes
Init BLE and Profile To Run Updated FW
ble_init()
Display on RTT Current RYZ012 FW Version
Process BLE Events
R_BLE_Execute
While(1)
Yes

Figure 2. MCU User Application Flow Diagram

Figure 2 shows the MCU user application (**project/qe_gen/ble/app_main.c**) flow diagram for the firmware update process. The FSP provided helper function **R_BLE_VS_UpdateModuleFirmware(…)** simplifies the work required by calling the SPP command sequence required to perform the upgrade.

See Figure 3 for details of the command sequence that is executed.

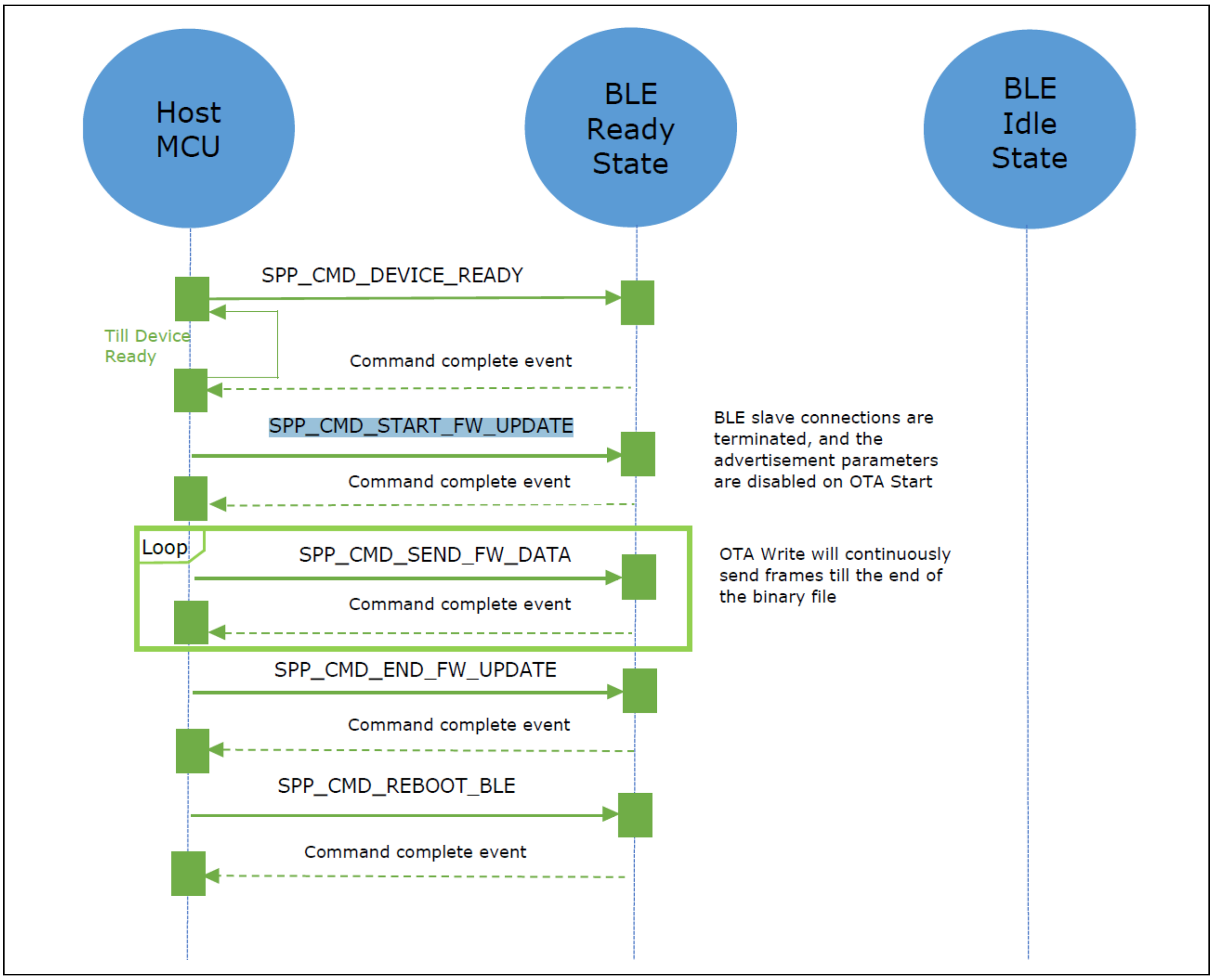


Figure 3. SPP Command Sequence for RYZ012 Firmware Update

- Note that the RYZ012 must receive the **SPP_CMD_REBOOT_BLE** command, 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 with **SPP_CMD_SEND_FW_DATA** commands 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 during the SPP command sequence prior to completing all the update steps, then the RYZ012 will run the roll back image.
- If the RYZ012 loses power during the SPP command sequence 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.

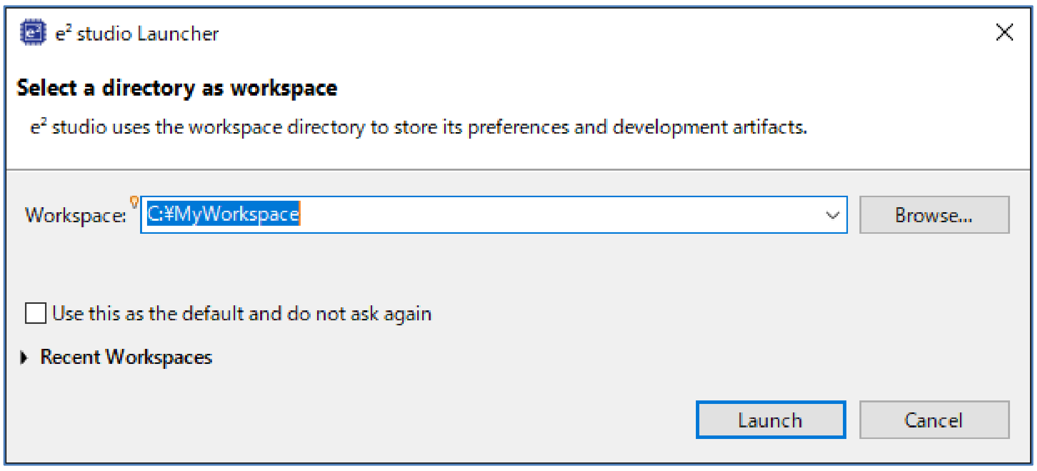
2. Firmware Update Application

2.1 Importing the Application Project

The steps to import the application project into e^2 studio are shown in the following sections.

2.1.1 Specify e^2 studio Workspace

Launch e^2 studio, specify the workspace directory, and click the **Launch** button.


e^2 studio Launcher

Select a directory as workspace

e^2 studio uses the workspace directory to store its preferences and development artifacts.

Workspace: C:¥MyWorkspace

Figure 4. e^2 studio Workspace

2.1.2 Import Project

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

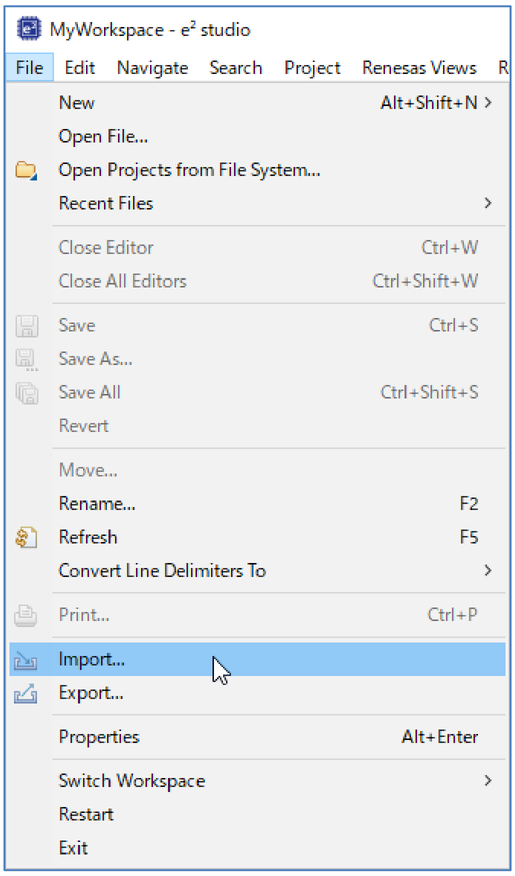


Figure 5. Importing Project

2.1.3 Select Existing Project

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

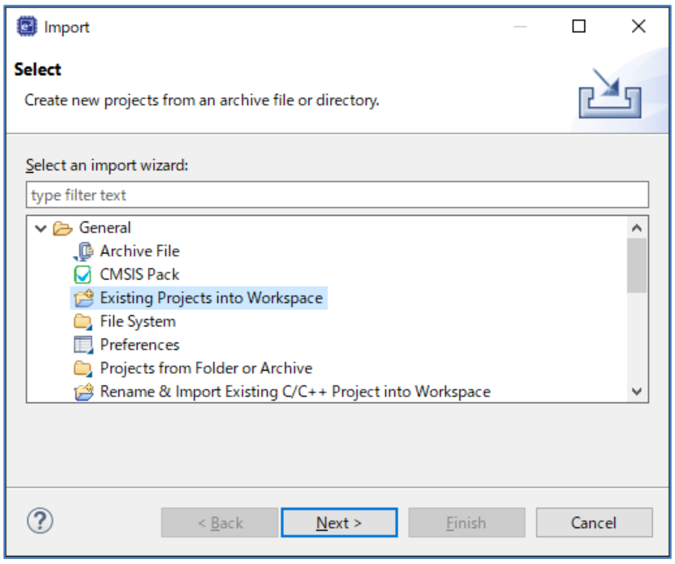

Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
General
Archive File
CMSIS Pack
Existing Projects into Workspace
File System
Preferences
Projects from Folder or Archive
Rename & Import Existing C/C++ Project into Workspace
< Back
Next >
Finish
Cancel

Figure 6. 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.

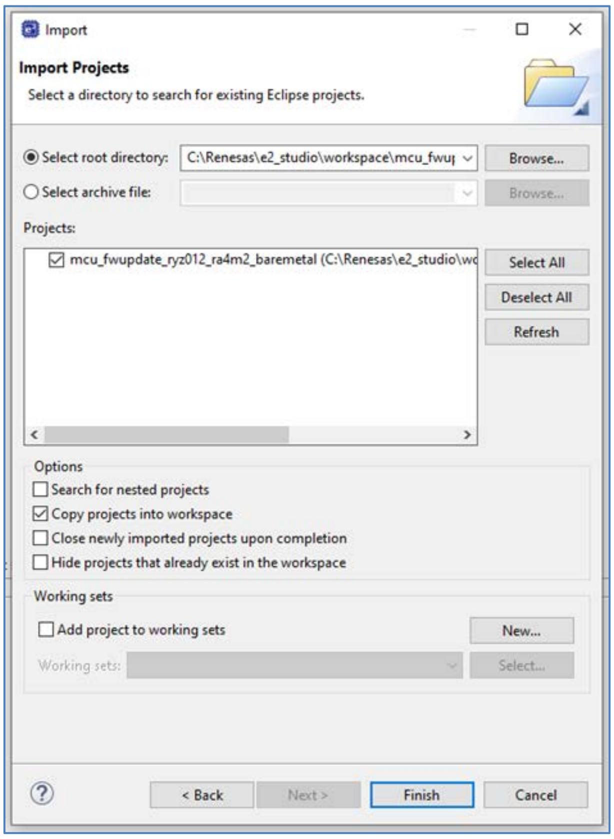

Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory: C:\Renesas\e2_studio\workspace\mcu_fwu
Browse...
Select archive file:
Browse...
Projects:
mcu_fwupdate_ryz012_ra4m2_baremetal (C:\Renesas\e2_studio\wo
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Close newly imported projects upon completion
Hide projects that already exist in the workspace
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel

Figure 7. Selecting Existing Projects

2.2 Project Build and Download

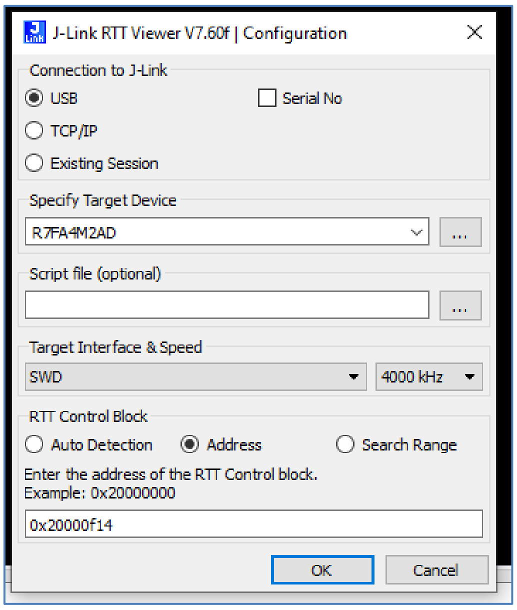
1. Select **Project > Build Project** from the menu bar or click the Build icon to build the project.
2. Make sure that the hardware is connected according to section 1.1, Operating Environment. Click the debug icon to launch the project. When the project starts, the application will be downloaded to 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, 0x20000f14



J-Link RTT Viewer V7.60f | Configuration

Connection to J-Link
[✓ USB]   Serial No
TCP/IP
Existing Session

Specify Target Device
R7FA4M2AD

Script file (optional)

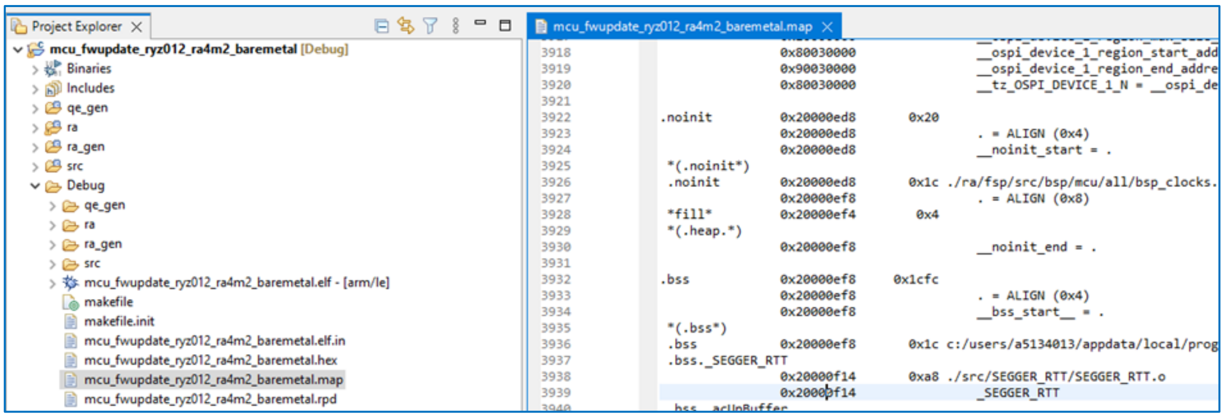
Target Interface & Speed
SWD   4000 kHz

RTT Control Block
Auto Detection   [✓ Address]   Search Range
Enter the address of the RTT Control block.
Example: 0x20000000
0x20000f14

OK   Cancel

Figure 8. 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.



Project Explorer

```
mcu_fwupdate_ryz012_ra4m2_baremetal [Debug]
  Binaries
  Includes
  qe_gen
  ra
  ra_gen
  src
  Debug
    qe_gen
    ra
    ra_gen
    src
    mcu_fwupdate_ryz012_ra4m2_baremetal.elf - [arm/le]
    makefile
    makefile.init
    mcu_fwupdate_ryz012_ra4m2_baremetal.elf.in
    mcu_fwupdate_ryz012_ra4m2_baremetal.hex
    mcu_fwupdate_ryz012_ra4m2_baremetal.map
    mcu_fwupdate_ryz012_ra4m2_baremetal.rpd
```

mcu_fwupdate_ryz012_ra4m2_baremetal.map

```
                0x80030000                __ospi_device_1_region_start_add
                0x90030000                __ospi_device_1_region_end_addre
                0x80030000                __tz_OSPI_DEVICE_1_N = __ospi_de

.noinit         0x20000ed8       0x20
                0x20000ed8                . = ALIGN (0x4)
                0x20000ed8                __noinit_start = .
 *(.noinit*)
 .noinit        0x20000ed8       0x1c ./ra/fsp/src/bsp/mcu/all/bsp_clocks.
                0x20000ef8                . = ALIGN (0x8)
 *fill*         0x20000ef4        0x4
 *(.heap.*)
                0x20000ef8                __noinit_end = .

.bss            0x20000ef8     0x1cfc
                0x20000ef8                . = ALIGN (0x4)
                0x20000ef8                __bss_start__ = .
 *(.bss*)
 .bss           0x20000ef8       0x1c c:/users/a5134013/appdata/local/prog
 .bss._SEGGER_RTT
                0x20000f14       0xa8 ./src/SEGGER_RTT/SEGGER_RTT.o
                0x20000f14                _SEGGER_RTT
```

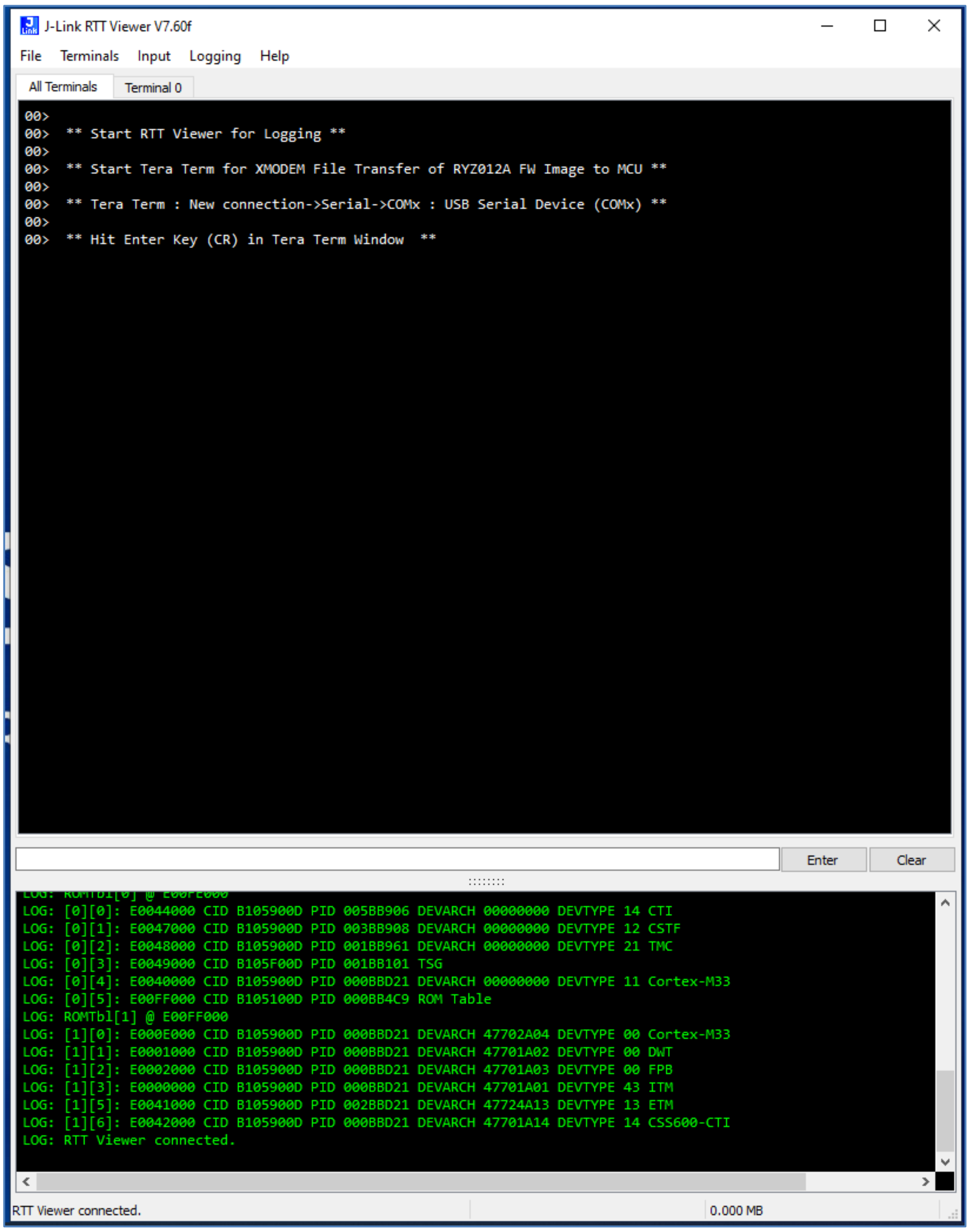
Figure 9. RTT Control Block Address

2.3.2 Start the Application Execution

Click the resume icon in the e^2 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/off to indicate that the program is running. In Figure 10, observe the instructions to start Tera Term.

If the LEDs fail to blink and the white LED 4 is not lighted, then verify a USB cable is plugged into USB Full Speed J11 (with other end connected to PC) and the jumpers are set correctly for J12, J15. See section 1.1.4, EK-RA4M2 Jumper Setting for more details.



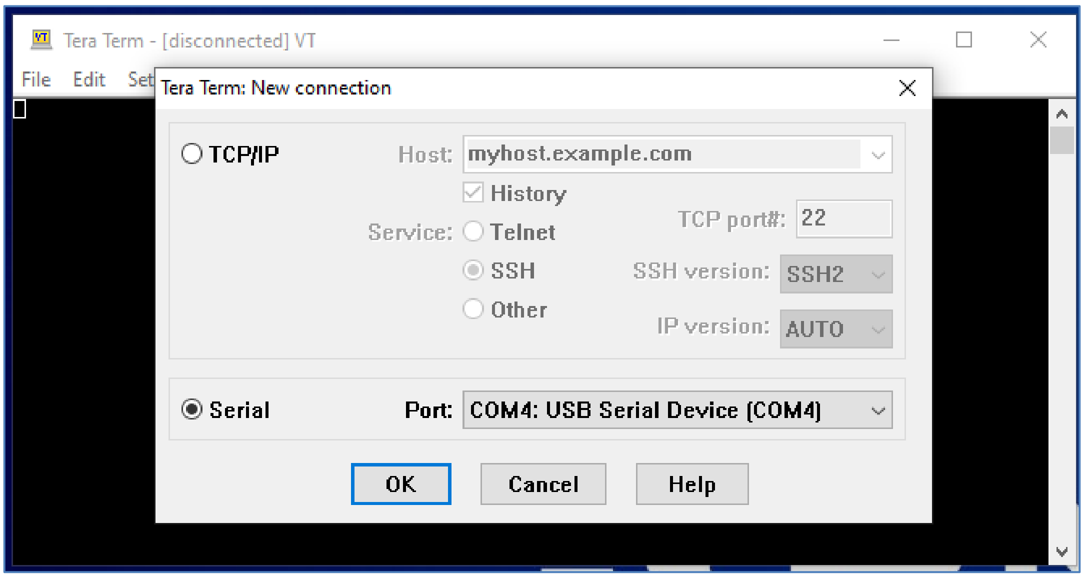
```
00>
00>  ** Start RTT Viewer for Logging **
00>
00>  ** Start Tera Term for XMODEM File Transfer of RYZ012A FW Image to MCU **
00>
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
```

```
LOG: ROMTbl[0] @ E00FE000
LOG: [0][0]: E0044000 CID B105900D PID 005BB906 DEVARCH 00000000 DEVTYPE 14 CTI
LOG: [0][1]: E0047000 CID B105900D PID 003BB908 DEVARCH 00000000 DEVTYPE 12 CSTF
LOG: [0][2]: E0048000 CID B105900D PID 001BB961 DEVARCH 00000000 DEVTYPE 21 TMC
LOG: [0][3]: E0049000 CID B105F00D PID 001BB101 TSG
LOG: [0][4]: E0040000 CID B105900D PID 000BBD21 DEVARCH 00000000 DEVTYPE 11 Cortex-M33
LOG: [0][5]: E00FF000 CID B105100D PID 000BB4C9 ROM Table
LOG: ROMTbl[1] @ E00FF000
LOG: [1][0]: E000E000 CID B105900D PID 000BBD21 DEVARCH 47702A04 DEVTYPE 00 Cortex-M33
LOG: [1][1]: E0001000 CID B105900D PID 000BBD21 DEVARCH 47701A02 DEVTYPE 00 DWT
LOG: [1][2]: E0002000 CID B105900D PID 000BBD21 DEVARCH 47701A03 DEVTYPE 00 FPB
LOG: [1][3]: E0000000 CID B105900D PID 000BBD21 DEVARCH 47701A01 DEVTYPE 43 ITM
LOG: [1][5]: E0041000 CID B105900D PID 002BBD21 DEVARCH 47724A13 DEVTYPE 13 ETM
LOG: [1][6]: E0042000 CID B105900D PID 000BBD21 DEVARCH 47701A14 DEVTYPE 14 CSS600-CTI
LOG: RTT Viewer connected.
```

Figure 10. RTT Viewer – Instructions to Start Tera Term

2.3.3 Launch Tera Term

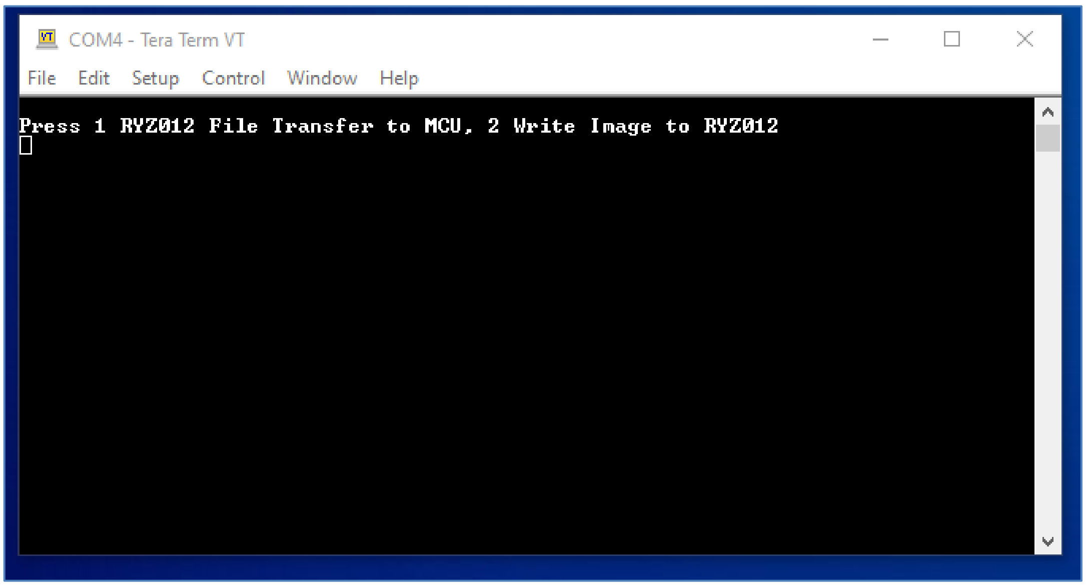
Launch Tera Term, set as follows, and click the **OK** button. The Serial COM port may be different for your PC. Choose the **Port > COMx: USB Serial Device (COMx)**, where X is the number assigned by your PC. See Figure 11 for details.



TCP/IP Host: myhost.example.com
History
Service: Telnet / SSH / Other
TCP port#: 22
SSH version: SSH2
IP version: AUTO
Serial Port: COM4: USB Serial Device (COM4)

Figure 11. Tera Term Connect Configuration

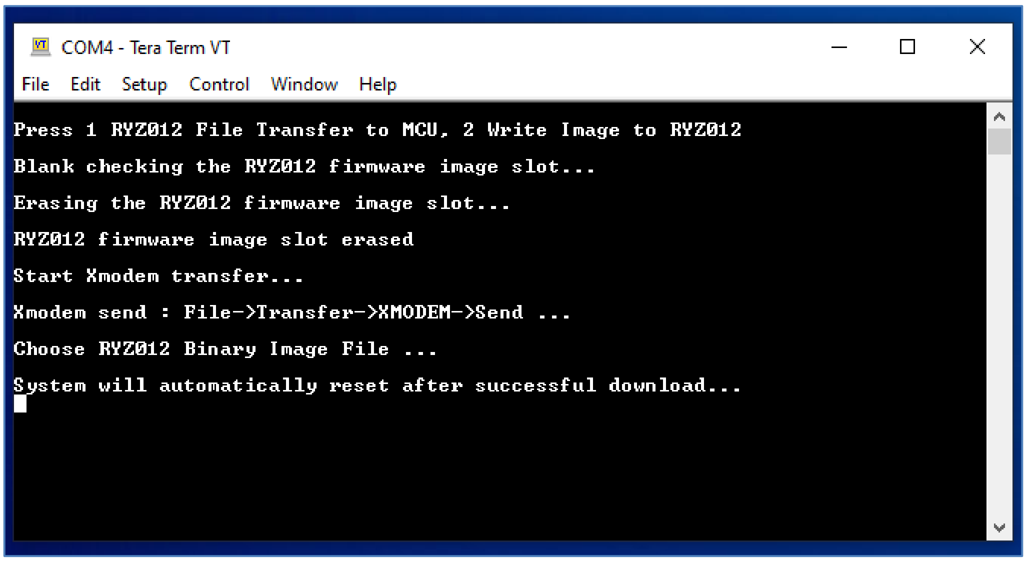
Select and click inside the Tera Term window to receive keyboard input and press the **Enter** key to get the Option Menu shown below in Figure 12.



```
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
```

Figure 12. Tera Term Menu

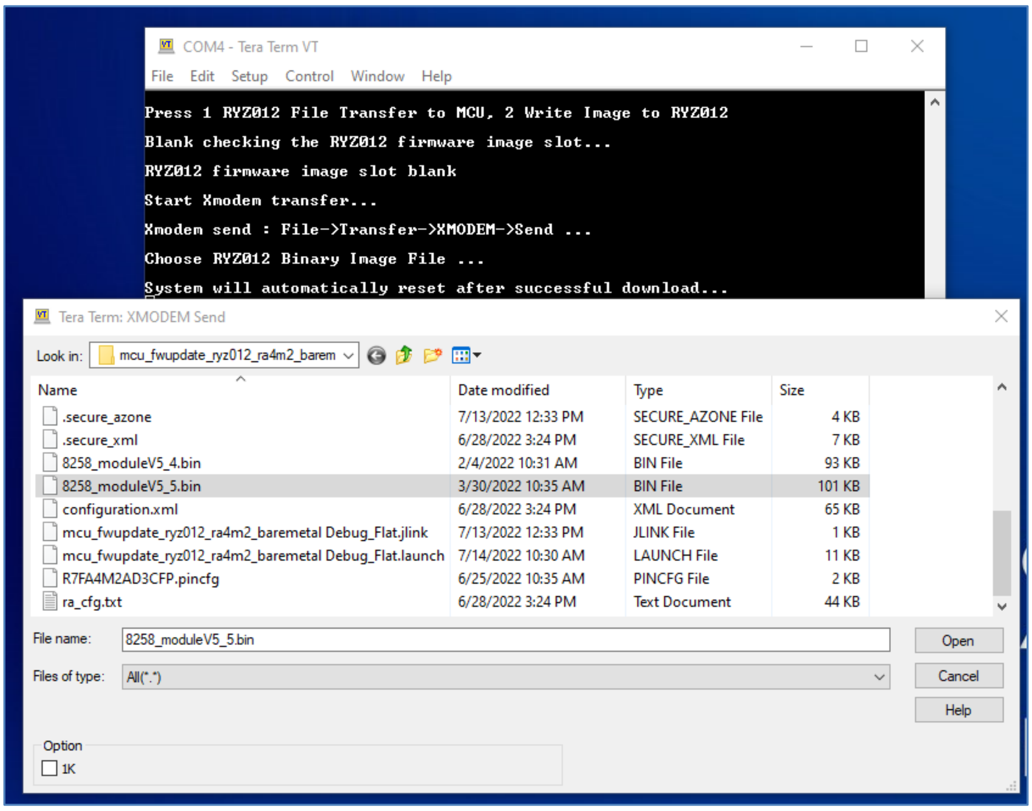
Press **1** to select the RYZ012 firmware image file to transfer to the MCU Flash. See Figure 13.



```
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Blank checking the RYZ012 firmware image slot...
Erasing the RYZ012 firmware image slot...
RYZ012 firmware image slot erased
Start Xmodem transfer...
Xmodem send : File->Transfer->XMODEM->Send ...
Choose RYZ012 Binary Image File ...
System will automatically reset after successful download...
```

Figure 13. Tera Term – Press 1 File Transfer to MCU

Navigate to the **root directory** of the Application Project imported in the e^2 studio workspace and select one of the RYZ012 firmware files such as **8258_moduleV5_5.bin** and Click **Open**. See Figure 14.



```
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Blank checking the RYZ012 firmware image slot...
RYZ012 firmware image slot blank
Start Xmodem transfer...
Xmodem send : File->Transfer->XMODEM->Send ...
Choose RYZ012 Binary Image File ...
System will automatically reset after successful download...
```

Look in: mcu_fwupdate_ryz012_ra4m2_barem

Name	Date modified	Type	Size
.secure_azone	7/13/2022 12:33 PM	SECURE_AZONE File	4 KB
.secure_xml	6/28/2022 3:24 PM	SECURE_XML File	7 KB
8258_moduleV5_4.bin	2/4/2022 10:31 AM	BIN File	93 KB
8258_moduleV5_5.bin	3/30/2022 10:35 AM	BIN File	101 KB
configuration.xml	6/28/2022 3:24 PM	XML Document	65 KB
mcu_fwupdate_ryz012_ra4m2_baremetal Debug_Flat.jlink	7/13/2022 12:33 PM	JLINK File	1 KB
mcu_fwupdate_ryz012_ra4m2_baremetal Debug_Flat.launch	7/14/2022 10:30 AM	LAUNCH File	11 KB
R7FA4M2AD3CFP.pincfg	6/25/2022 10:35 AM	PINCFG File	2 KB
ra_cfg.txt	6/28/2022 3:24 PM	Text Document	44 KB

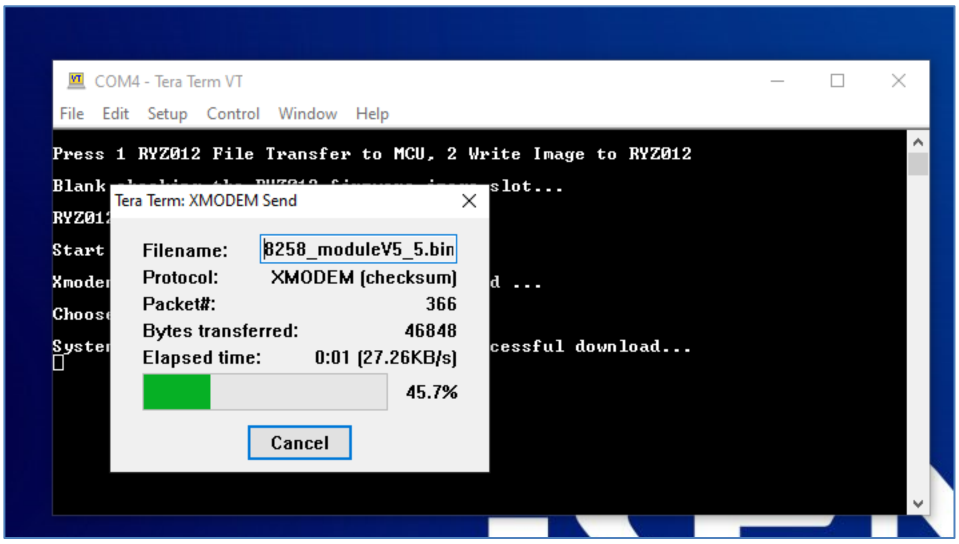
File name: 8258_moduleV5_5.bin
Files of type: All(*.*)

Figure 14. Tera Term – Select File XMODEM to Send

The file will start transferring to the EK-RA4M2 MCU Flash. If the file transfer does not start after the file is selected, then either **Press 1** option was not performed prior to choosing the file or too much time elapsed after **Press 1** option to the time the file was chosen and you may see the error message shown in Figure 16.

In this case, hit the enter key in Tera Term to start over and press 1 again, navigate to and select the file to transfer.

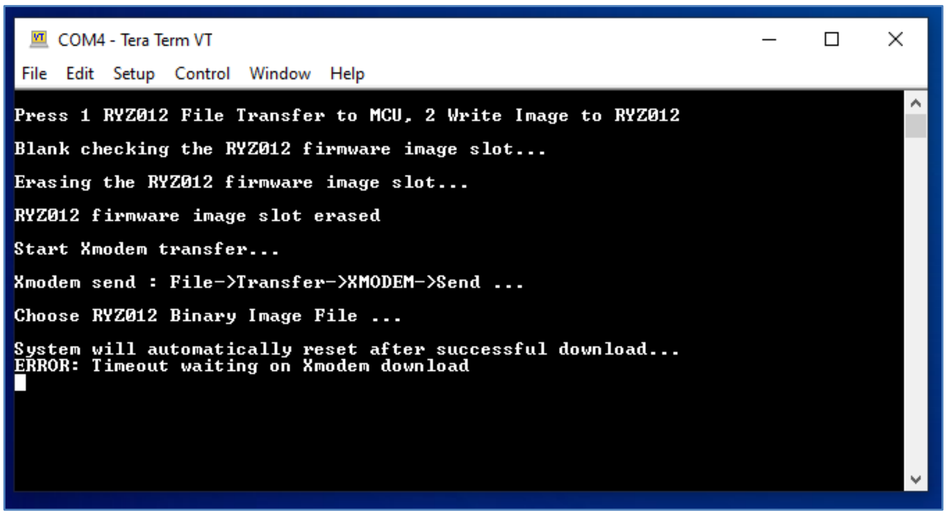
When the file transfer starts, the transfer progress dialog will be shown as in Figure 15.



```
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Blank                                   slot...
RYZ01
Start
Xmode                                   d ...
Choos
Syste                                   cessful download...
```

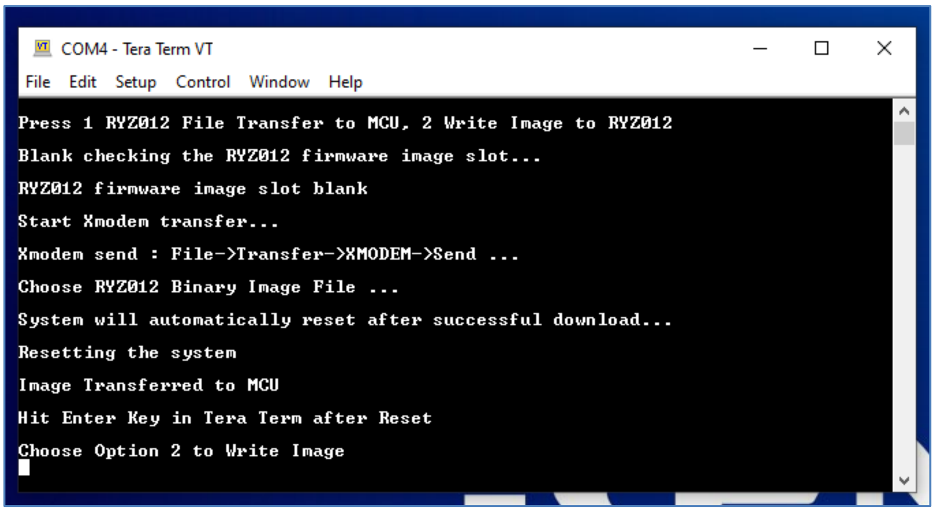
```
Filename:          8258_moduleV5_5.bin
Protocol:          XMODEM (checksum)
Packet#:                        366
Bytes transferred:            46848
Elapsed time:       0:01 (27.26KB/s)
                              45.7%
```

Figure 15. Tera Term – XMODEM File Send in Progress



```
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Blank checking the RYZ012 firmware image slot...
Erasing the RYZ012 firmware image slot...
RYZ012 firmware image slot erased
Start Xmodem transfer...
Xmodem send : File->Transfer->XMODEM->Send ...
Choose RYZ012 Binary Image File ...
System will automatically reset after successful download...
ERROR: Timeout waiting on Xmodem download
```

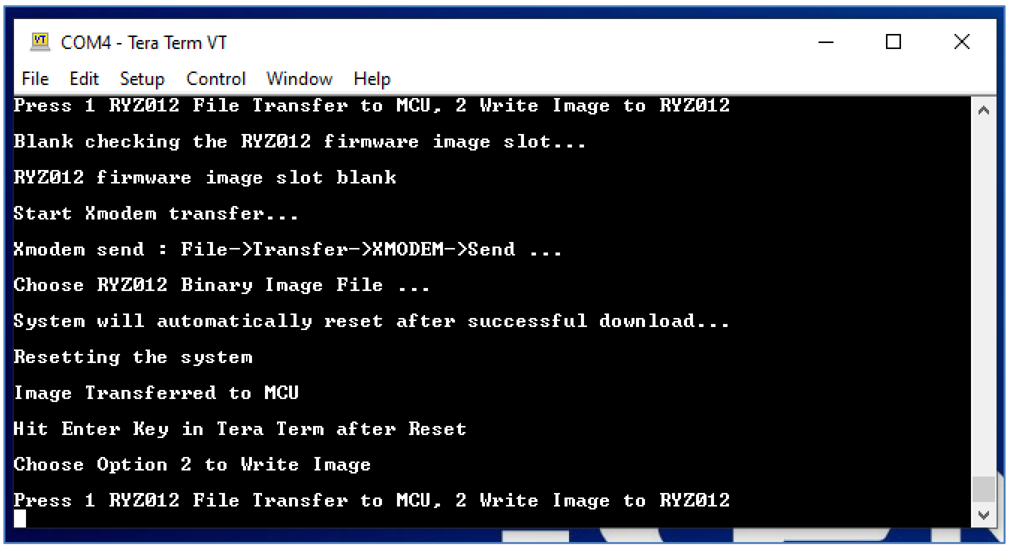
Figure 16. Tera Term – XMODEM Timeout Waiting on File



```
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Blank checking the RYZ012 firmware image slot...
RYZ012 firmware image slot blank
Start Xmodem transfer...
Xmodem send : File->Transfer->XMODEM->Send ...
Choose RYZ012 Binary Image File ...
System will automatically reset after successful download...
Resetting the system
Image Transferred to MCU
Hit Enter Key in Tera Term after Reset
Choose Option 2 to Write Image
```

Figure 17. Tera Term – XMODEM Image Transferred to MCU

After the file transfer has completed, “`Image Transferred to MCU`” will appear in Tera Term and you will be prompted to **“Hit Enter Key in Tera Term**” and then **“Choose Option 2 to Write the Image”** to the RYZ012. See Figure 17.

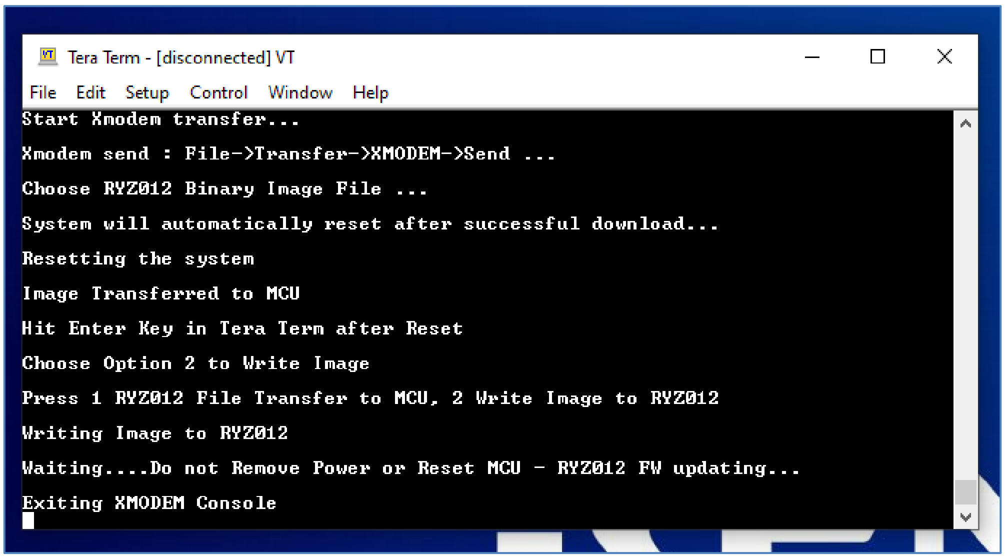


```
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Blank checking the RYZ012 firmware image slot...
RYZ012 firmware image slot blank
Start Xmodem transfer...
Xmodem send : File->Transfer->XMODEM->Send ...
Choose RYZ012 Binary Image File ...
System will automatically reset after successful download...
Resetting the system
Image Transferred to MCU
Hit Enter Key in Tera Term after Reset
Choose Option 2 to Write Image
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
```

Figure 18. Tera Term – Choose Option 2 Write Image to RYZ012

Hit **Enter** key in Tera Term window to get the Option menu, then “**Press 2 Write Image to RYZ012”**. See Figure 18.

Next you will see Tera Term logging update as shown in Figure 19.

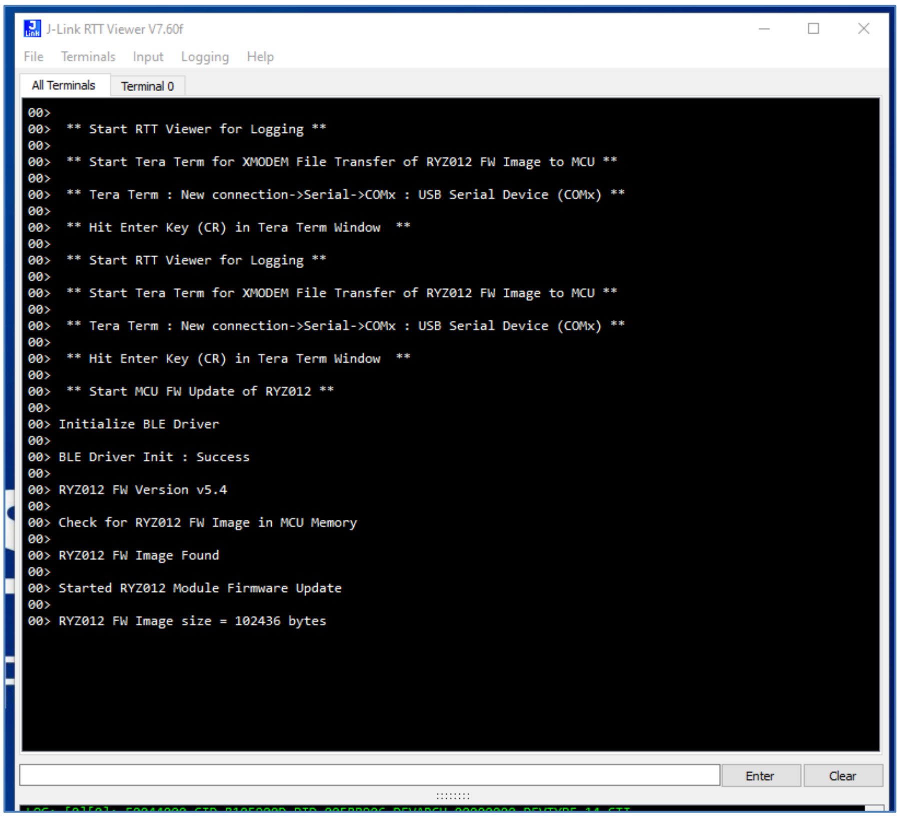

Tera Term - [disconnected] VT
File Edit Setup Control Window Help

```
Start Xmodem transfer...
Xmodem send : File->Transfer->XMODEM->Send ...
Choose RYZ012 Binary Image File ...
System will automatically reset after successful download...
Resetting the system
Image Transferred to MCU
Hit Enter Key in Tera Term after Reset
Choose Option 2 to Write Image
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Writing Image to RYZ012
Waiting....Do not Remove Power or Reset MCU - RYZ012 FW updating...
Exiting XMODEM Console
```

Figure 19. Tera Term – Option 2 Writing Image to RYZ012 In Progress

At this point, you are done using Tera Term so move your attention to RTT Viewer to see the firmware upgrade status there. You should observe logging similar to Figure 20.

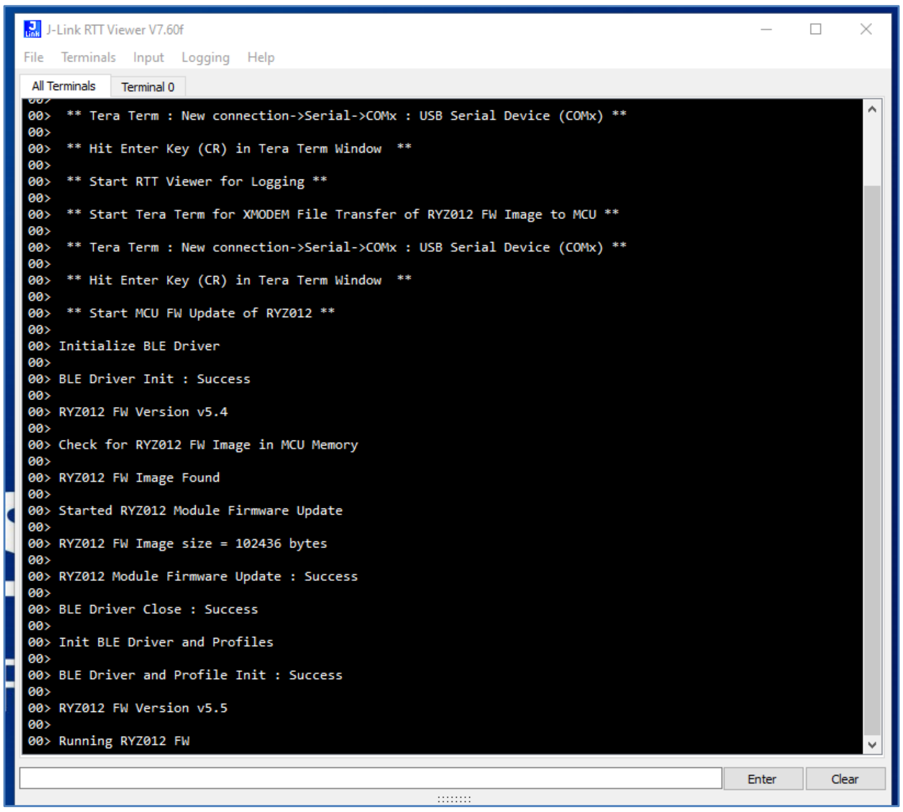
RTT Viewer will show the results of the RYZ012 firmware upgrade process.



```
00>
00>  ** Start RTT Viewer for Logging **
00>
00>  ** Start Tera Term for XMODEM File Transfer of RYZ012 FW Image to MCU **
00>
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start RTT Viewer for Logging **
00>
00>  ** Start Tera Term for XMODEM File Transfer of RYZ012 FW Image to MCU **
00>
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start MCU FW Update of RYZ012 **
00>
00> Initialize BLE Driver
00>
00> BLE Driver Init : Success
00>
00> RYZ012 FW Version v5.4
00>
00> Check for RYZ012 FW Image in MCU Memory
00>
00> RYZ012 FW Image Found
00>
00> Started RYZ012 Module Firmware Update
00>
00> RYZ012 FW Image size = 102436 bytes
```

Figure 20. RTT Viewer – Option 2 Writing Image to RYZ012 In Progress

When the RYZ012 firmware upgrade completes with success, RTT Viewer will show logging as shown in Figure 21.



```
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start RTT Viewer for Logging **
00>
00>  ** Start Tera Term for XMODEM File Transfer of RYZ012 FW Image to MCU **
00>
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start MCU FW Update of RYZ012 **
00>
00> Initialize BLE Driver
00>
00> BLE Driver Init : Success
00>
00> RYZ012 FW Version v5.4
00>
00> Check for RYZ012 FW Image in MCU Memory
00>
00> RYZ012 FW Image Found
00>
00> Started RYZ012 Module Firmware Update
00>
00> RYZ012 FW Image size = 102436 bytes
00>
00> RYZ012 Module Firmware Update : Success
00>
00> BLE Driver Close : Success
00>
00> Init BLE Driver and Profiles
00>
00> BLE Driver and Profile Init : Success
00>
00> RYZ012 FW Version v5.5
00>
00> Running RYZ012 FW
```

Figure 21. RTT Viewer – RYZ012 Firmware Image Update Success

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

To verify that the RYZ012 is functioning use the GATT Browser app on a mobile device and scan for BLE devices. The RYZ012 module will advertise as "RYZ012" when the firmware upgrade has completed.

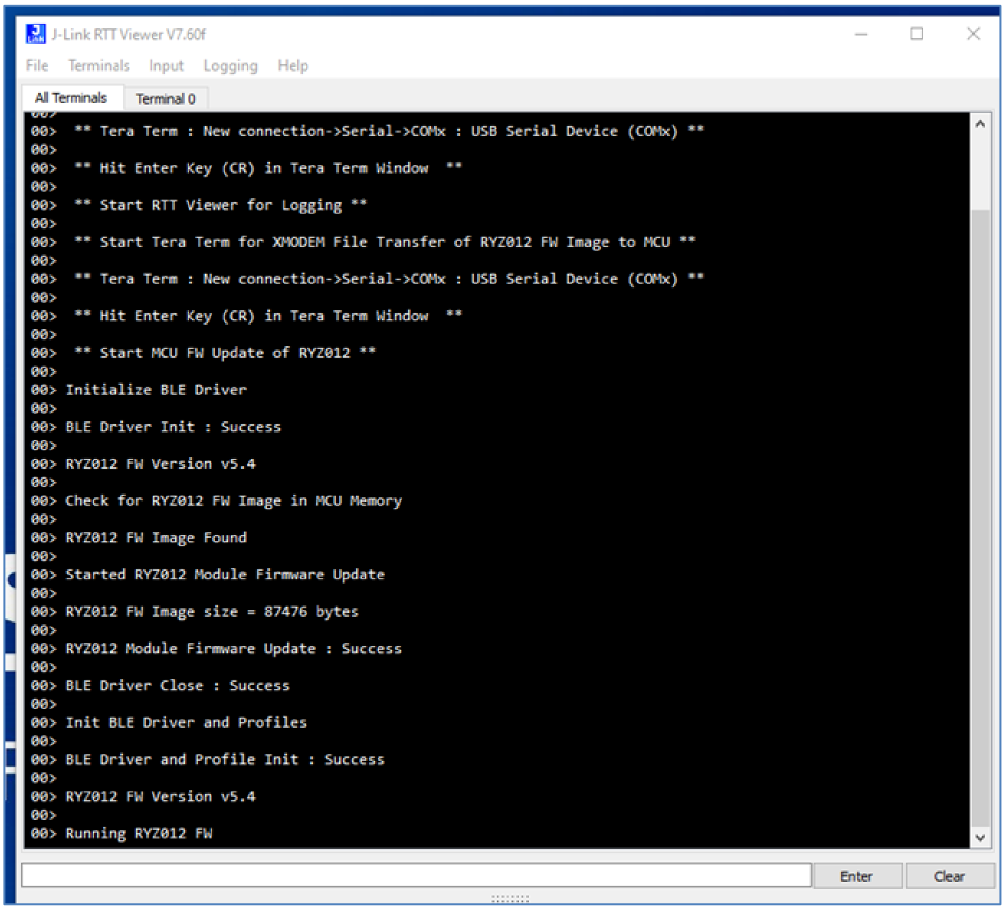
In the event that the firmware upgrade fails with error messages shown in RTT Viewer, the next section 2.4, RYZ012 Firmware Update Considerations will help you to troubleshoot and resolve the issues.

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 **mcu_fwupdate_ryz012_ra4m2_baremetal/8258_moduleV5_4.bin** or **mcu_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 it not possible to roll back to a previous 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 RYZ012 Firmware Image Invalid / Update Failed



```
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start RTT Viewer for Logging **
00>
00>  ** Start Tera Term for XMODEM File Transfer of RYZ012 FW Image to MCU **
00>
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start MCU FW Update of RYZ012 **
00>
00> Initialize BLE Driver
00>
00> BLE Driver Init : Success
00>
00> RYZ012 FW Version v5.4
00>
00> Check for RYZ012 FW Image in MCU Memory
00>
00> RYZ012 FW Image Found
00>
00> Started RYZ012 Module Firmware Update
00>
00> RYZ012 FW Image size = 87476 bytes
00>
00> RYZ012 Module Firmware Update : Success
00>
00> BLE Driver Close : Success
00>
00> Init BLE Driver and Profiles
00>
00> BLE Driver and Profile Init : Success
00>
00> RYZ012 FW Version v5.4
00>
00> Running RYZ012 FW
```

Figure 22. RTT Viewer – RYZ012 Firmware Image Invalid / Update Failed

In Figure 22, notice that the RYZ012 firmware version is v5.4 before the upgrade starts and is at v5.4 after the upgrade ends. This indicates that the firmware image was either corrupted, had an invalid signature, or not all 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 transferred image failed integrity check, the old image v5.4 (roll back image) was retained and selected by the RYZ012 to execute.

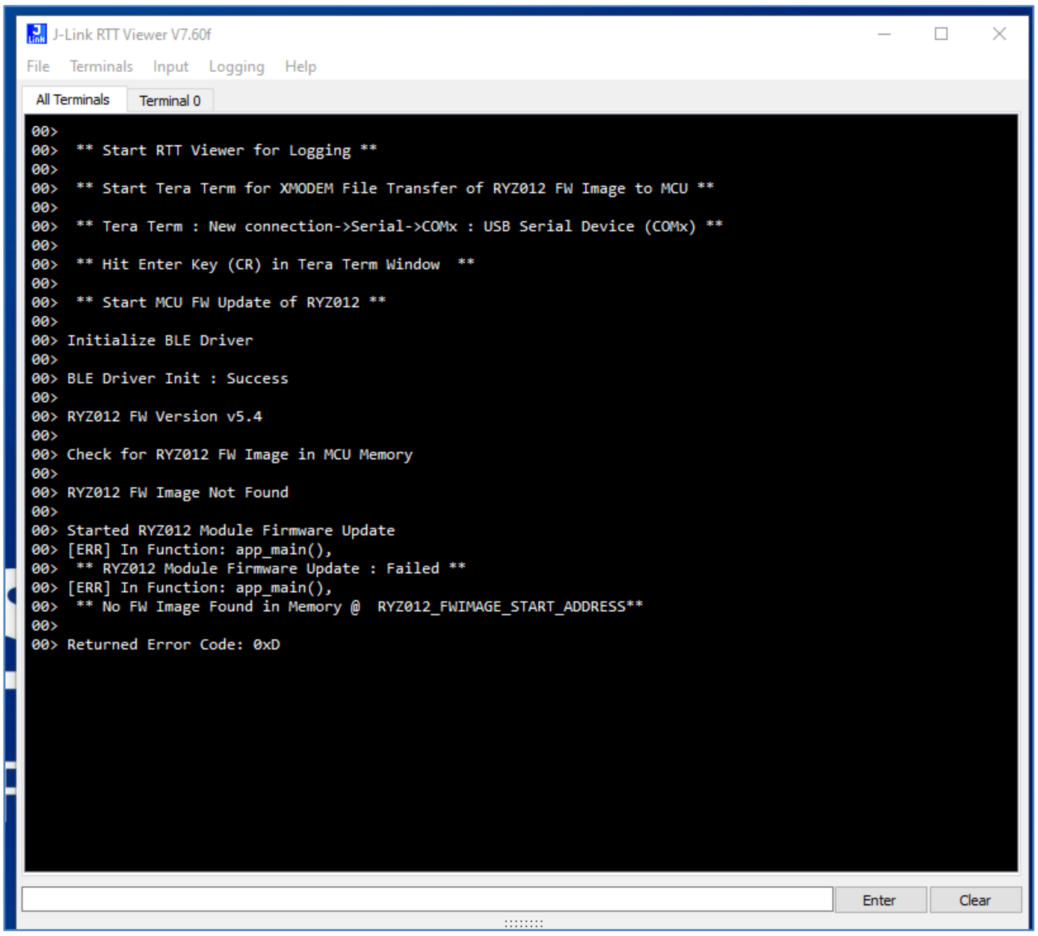
Make sure that the EK-RA4M2 board remains powered during the file transfer and write to RYZ012. If the EK-RA4M2 board is reset during the firmware upgrade then the file integrity check will fail on the transfer and the RYZ012 will run the last known good image version (roll back image).

2.4.2 RYZ012 Image Not Found

For the case that a RYZ012 firmware image is not found, see Figure 23. This can be due to the XMODEM firmware transfer to MCU Flash failing or the image file did not fit into the MCU Flash space allocated.

First reset the EK-RA4M2 with Red Reset button (S3), then select the Tera Term window, and Press 1 to Select the RYZ012 FW image file to transfer to the MCU Flash. See Figure 13 to confirm all the steps of the XMODEM file transfer.

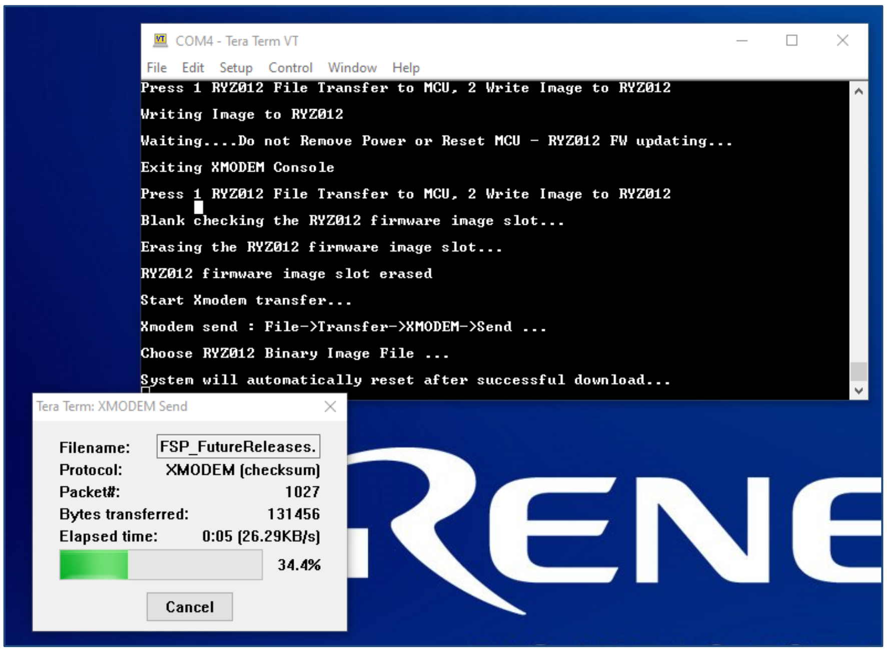
If the issue continues, then verify that the MCU Flash size can hold the entire firmware image and check the **#defines** match your chosen MCU and the required firmware image size in section 6, Next Steps, step 4.



```
00>
00>  ** Start RTT Viewer for Logging **
00>
00>  ** Start Tera Term for XMODEM File Transfer of RYZ012 FW Image to MCU **
00>
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start MCU FW Update of RYZ012 **
00>
00> Initialize BLE Driver
00>
00> BLE Driver Init : Success
00>
00> RYZ012 FW Version v5.4
00>
00> Check for RYZ012 FW Image in MCU Memory
00>
00> RYZ012 FW Image Not Found
00>
00> Started RYZ012 Module Firmware Update
00> [ERR] In Function: app_main(),
00>  ** RYZ012 Module Firmware Update : Failed **
00> [ERR] In Function: app_main(),
00>  ** No FW Image Found in Memory @  RYZ012_FWIMAGE_START_ADDRESS**
00>
00> Returned Error Code: 0xD
```

Figure 23. RTT Viewer – RYZ012 Image Not Found

2.4.3 Tera Term XMODEM Transfer Halt



```
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Writing Image to RYZ012
Waiting....Do not Remove Power or Reset MCU - RYZ012 FW updating...
Exiting XMODEM Console
Press 1 RYZ012 File Transfer to MCU, 2 Write Image to RYZ012
Blank checking the RYZ012 firmware image slot...
Erasing the RYZ012 firmware image slot...
RYZ012 firmware image slot erased
Start Xmodem transfer...
Xmodem send : File->Transfer->XMODEM->Send ...
Choose RYZ012 Binary Image File ...
System will automatically reset after successful download...
```

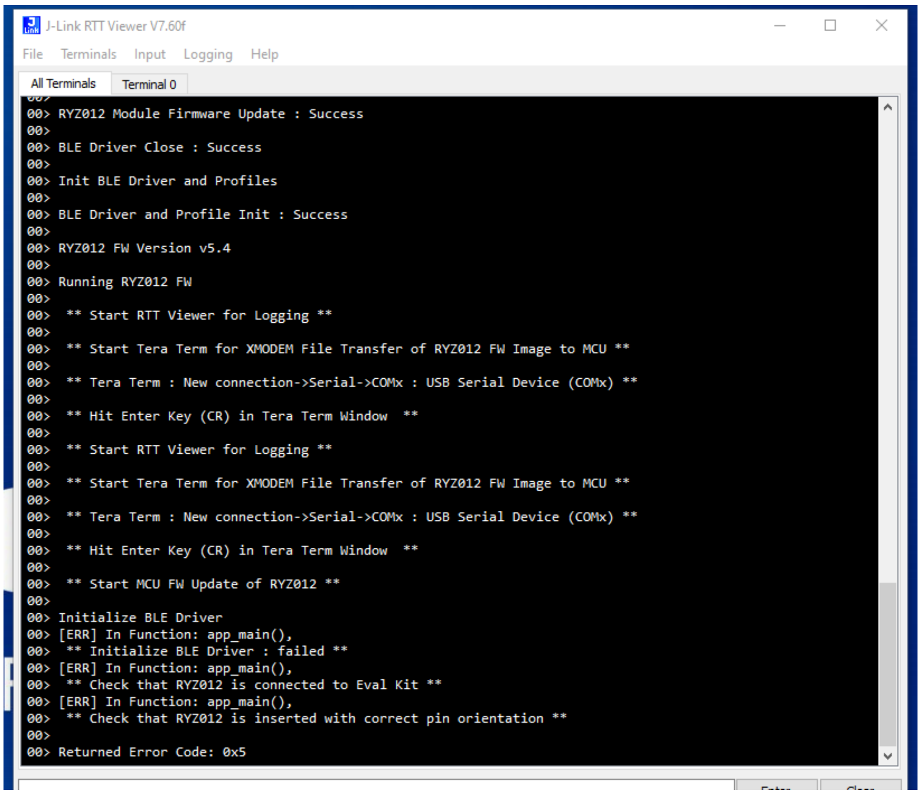
Filename:	FSP_FutureReleases.
Protocol:	XMODEM (checksum)
Packet#:	1027
Bytes transferred:	131456
Elapsed time:	0:05 (26.29KB/s)
	34.4%

Figure 24. Tera Term XMODEM Transfer Halt : Image Size Exceeds MCU Flash Allocated

Figure 24 shows the result when the RYZ012 firmware image XMODEM transfer halts unexpectedly. Notice that the transfer progress % (green) stops, and the entire file is not transferred to the MCU. This occurs because the MCU Flash does not have enough space allocated to write the entire RYZ012 firmware image.

Verify that the MCU Flash size can hold the entire firmware image and check that the **#defines** match your chosen MCU and required firmware image size in section 6**,** Next Steps, step 4.

2.4.4 Initialize BLE Driver Failed



```
00> RYZ012 Module Firmware Update : Success
00>
00> BLE Driver Close : Success
00>
00> Init BLE Driver and Profiles
00>
00> BLE Driver and Profile Init : Success
00>
00> RYZ012 FW Version v5.4
00>
00> Running RYZ012 FW
00>
00>  ** Start RTT Viewer for Logging **
00>
00>  ** Start Tera Term for XMODEM File Transfer of RYZ012 FW Image to MCU **
00>
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start RTT Viewer for Logging **
00>
00>  ** Start Tera Term for XMODEM File Transfer of RYZ012 FW Image to MCU **
00>
00>  ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) **
00>
00>  ** Hit Enter Key (CR) in Tera Term Window  **
00>
00>  ** Start MCU FW Update of RYZ012 **
00>
00> Initialize BLE Driver
00> [ERR] In Function: app_main(),
00>  ** Initialize BLE Driver : failed **
00> [ERR] In Function: app_main(),
00>  ** Check that RYZ012 is connected to Eval Kit **
00> [ERR] In Function: app_main(),
00>  ** Check that RYZ012 is inserted with correct pin orientation **
00>
00> Returned Error Code: 0x5
```

Figure 25. 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 image programmed before attempting module upgrade.

3. Application Software Architecture

The application software architecture for the MCU based RYZ012 firmware upgrade project is covered in this section.

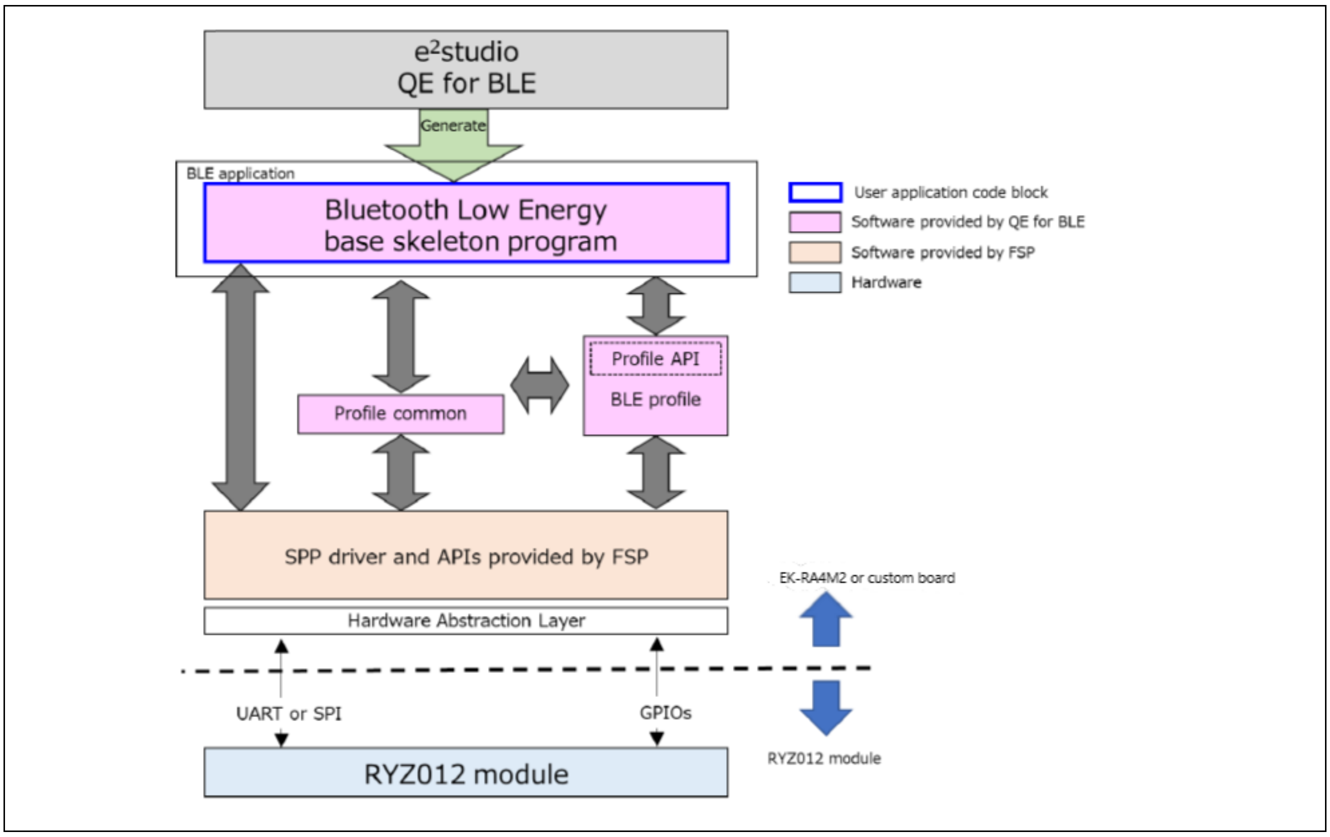


Figure 26. Application Software Architecture (Bare Metal)

Figure 26 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 functionality required in the user application. In this case, we are using the FSP BLE Abstraction APIs with SPP APIs to update the firmware 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. **Figure 3-2** 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.

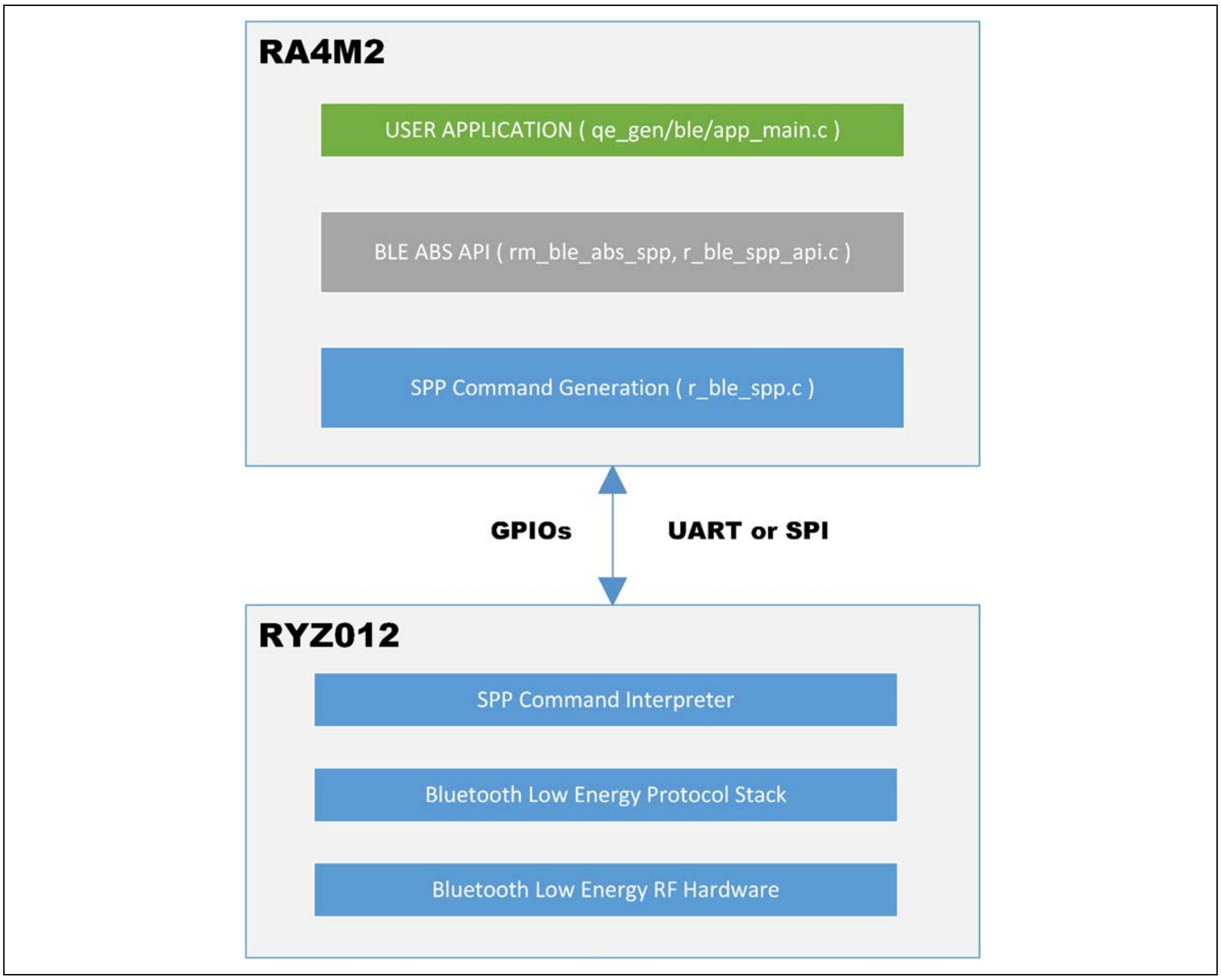

RA4M2
USER APPLICATION (qe_gen/ble/app_main.c)
BLE ABS API (rm_ble_abs_spp, r_ble_spp_api.c)
SPP Command Generation (r_ble_spp.c)
GPIOs
UART or SPI
RYZ012
SPP Command Interpreter
Bluetooth Low Energy Protocol Stack
Bluetooth Low Energy RF Hardware

Figure 27. MCU Host and Bluetooth LE Serial Port Profile Interface

Figure 27 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)
General-Purpose Timer	GPT0 GPT1 GPT2 GPT3	LED Blue PWM LED Red PWM LED blink timer LED Green PWM
USB 2.0 Full-Speed Module	USBFS	Communication with Tera Term for XMODEM transfer and User Input
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
Flash Memory	FLASH0	MCU storage for RYZ012 firmware image
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 28, 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
Connectivity	USB PCDC	r_usb_pcdc r_usb_basic	Tera Term XMODEM File Transfer & FW Upgrade
Storage	Flash High Performance	r_flash_hp	MCU Storage for RYZ012 FW
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.

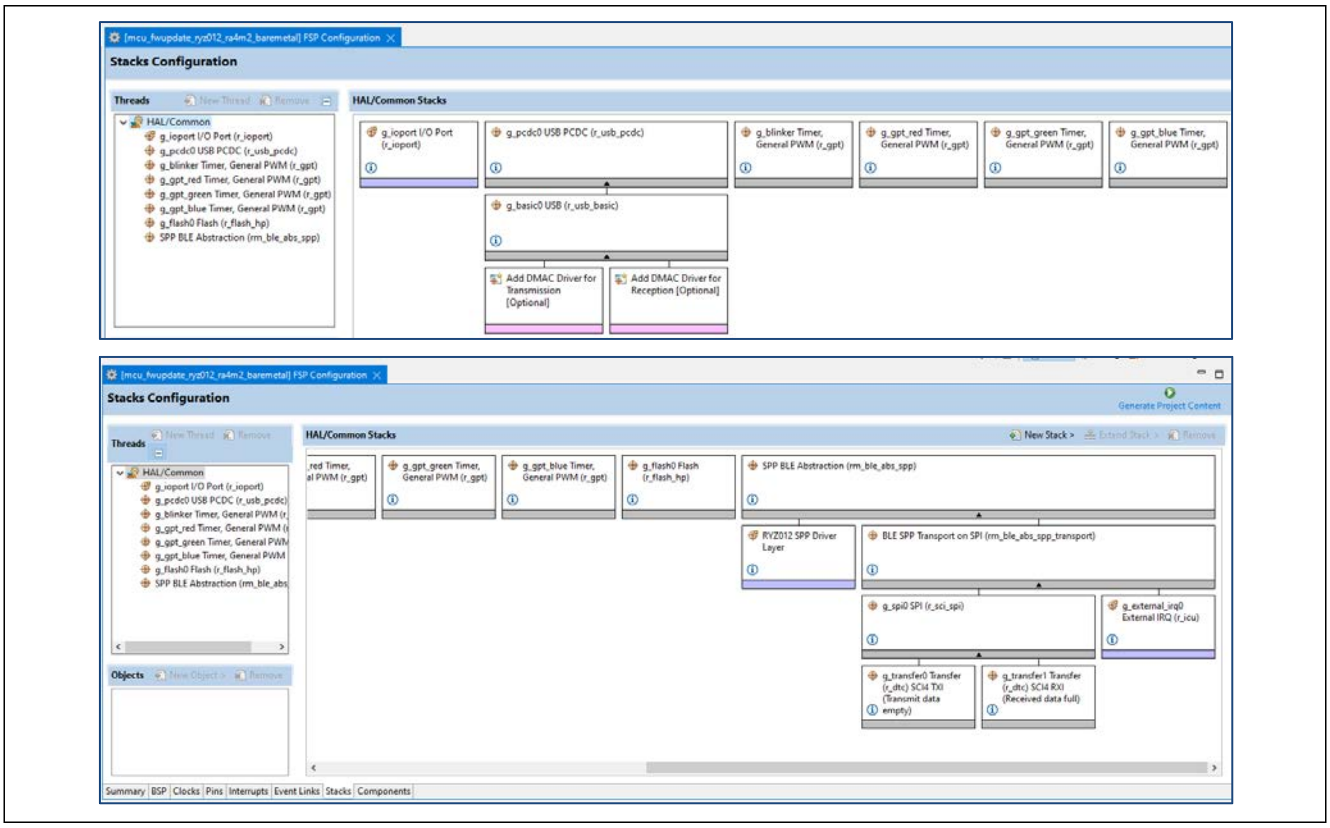

[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration
Stacks Configuration
Threads
New Thread
Remove
HAL/Common Stacks
HAL/Common
g_ioport I/O Port (r_ioport)
g_pcdc0 USB PCDC (r_usb_pcdc)
g_blinker Timer, General PWM (r_gpt)
g_gpt_red Timer, General PWM (r_gpt)
g_gpt_green Timer, General PWM (r_gpt)
g_gpt_blue Timer, General PWM (r_gpt)
g_flash0 Flash (r_flash_hp)
SPP BLE Abstraction (rm_ble_abs_spp)
g_basic0 USB (r_usb_basic)
Add DMAC Driver for Transmission [Optional]
Add DMAC Driver for Reception [Optional]
Generate Project Content
New Stack >
Extend Stack >
RYZ012 SPP Driver Layer
BLE SPP Transport on SPI (rm_ble_abs_spp_transport)
g_spi0 SPI (r_sci_spi)
g_external_irq0 External IRQ (r_icu)
g_transfer0 Transfer (r_dtc) SCI4 TXI (Transmit data empty)
g_transfer1 Transfer (r_dtc) SCI4 RXI (Received data full)
Objects
New Object >
Summary
BSP
Clocks
Pins
Interrupts
Event Links
Stacks
Components

Figure 28. FSP Module Summary

4. Application Project Implementation

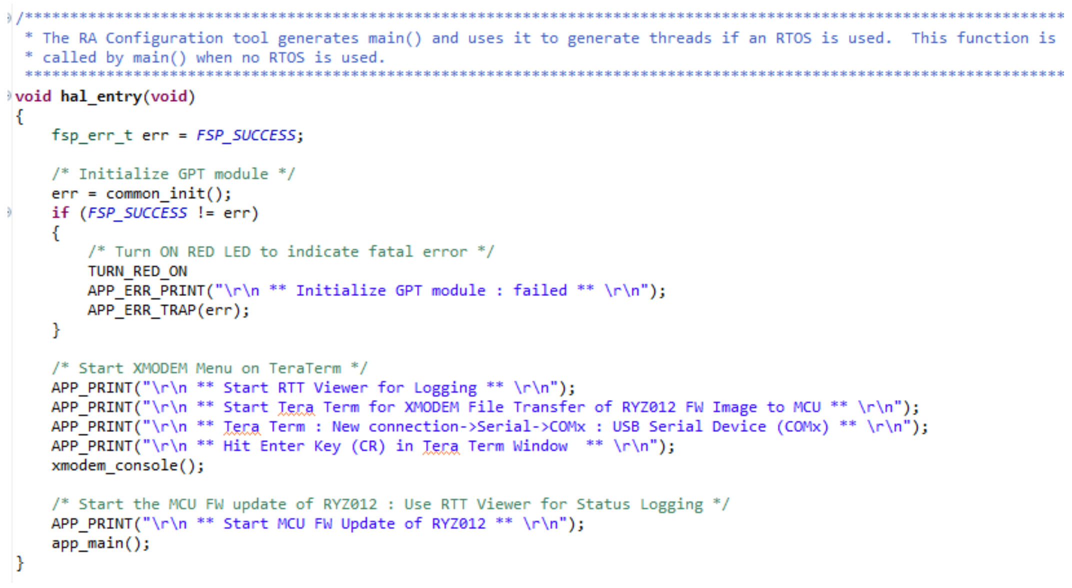
This section describes the application project implementation.

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, starts the XMODEM console used by Tera Term, and calls *app_main( )* to perform the BLE module update as follows:

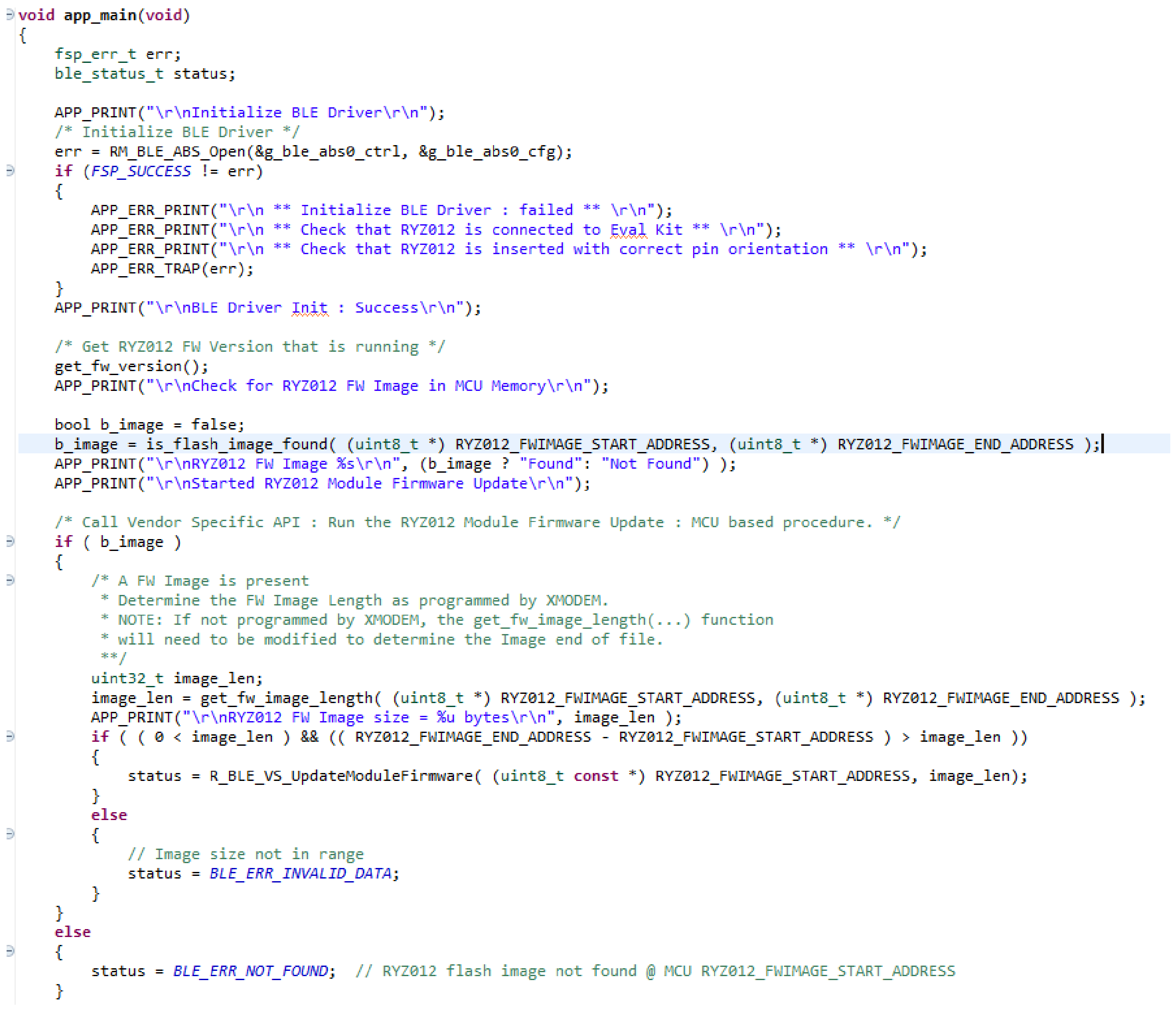


```
/*******************************************************************************************************************
 * The RA Configuration tool generates main() and uses it to generate threads if an RTOS is used.  This function is
 * called by main() when no RTOS is used.
 *******************************************************************************************************************
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 XMODEM Menu on TeraTerm */
    APP_PRINT("\r\n ** Start RTT Viewer for Logging ** \r\n");
    APP_PRINT("\r\n ** Start Tera Term for XMODEM File Transfer of RYZ012 FW Image to MCU ** \r\n");
    APP_PRINT("\r\n ** Tera Term : New connection->Serial->COMx : USB Serial Device (COMx) ** \r\n");
    APP_PRINT("\r\n ** Hit Enter Key (CR) in Tera Term Window  ** \r\n");
    xmodem_console();

    /* Start the MCU FW update of RYZ012 : Use RTT Viewer for Status Logging */
    APP_PRINT("\r\n ** Start MCU FW Update of RYZ012 ** \r\n");
    app_main();
}
```

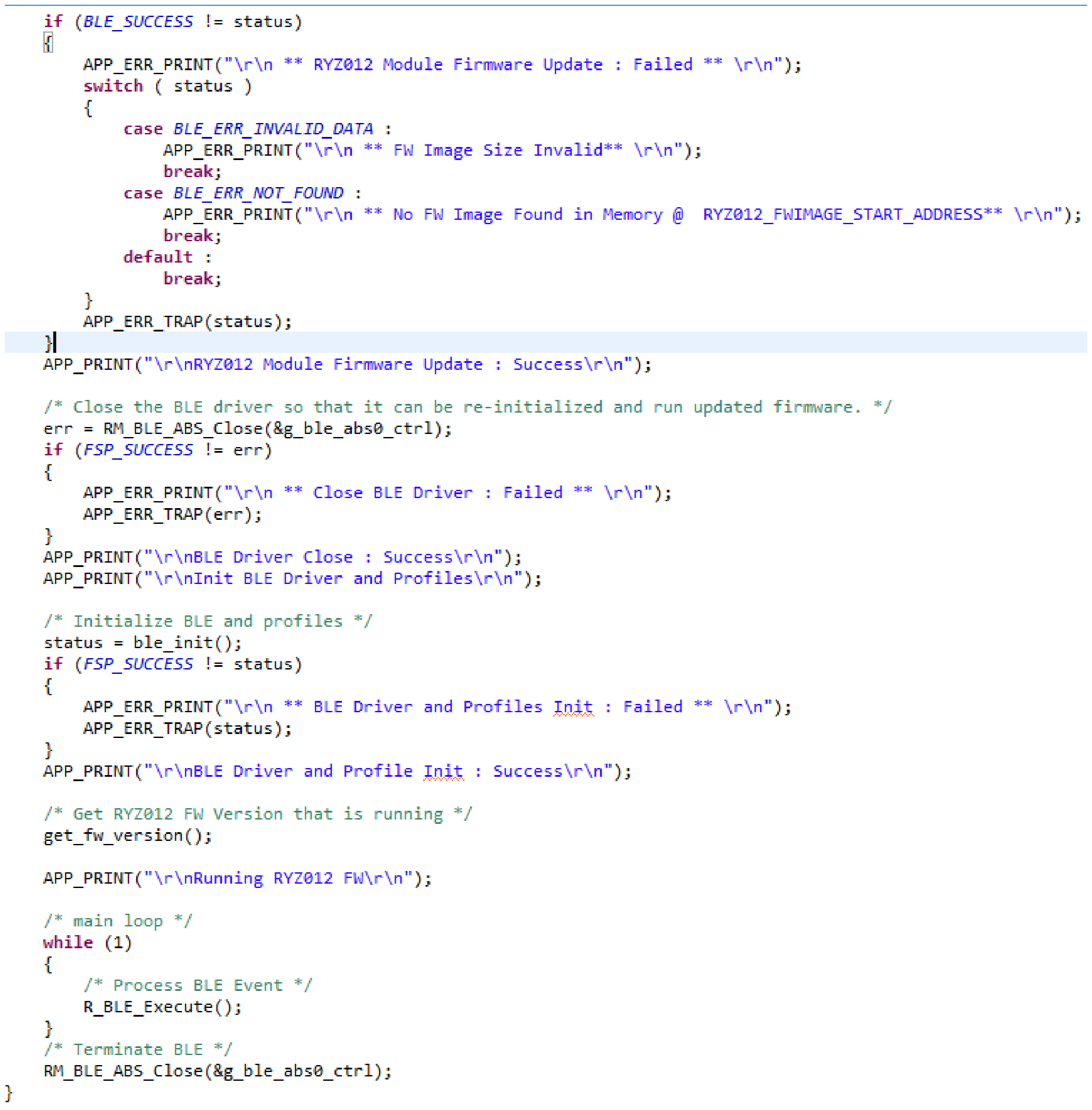
4.2 Main Loop

The *app_main( )* includes the BLE and profile initialization, BLE module firmware update, 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 to start the communications with the RYZ012.
2. Get the current firmware version running on the RYZ012.
3. Check for a RYZ012 firmware image stored in EK-RA4M2 flash and display firmware version.
4. If a firmware image is found, determine the length of the file and update the RYZ012 with **R_BLE_VS_UpdateModuleFirmware**(…).
5. Log the status of the firmware update to RTT Viewer
6. Close the BLE driver and call ble_init() to initialize the BLE module and BLE Profile, then run the new RYZ012 firmware.
7. Get the RYZ012 firmware version and display to confirm updated image is running.
8. Enter the main while loop to operate the RYZ012 and process BLE events.



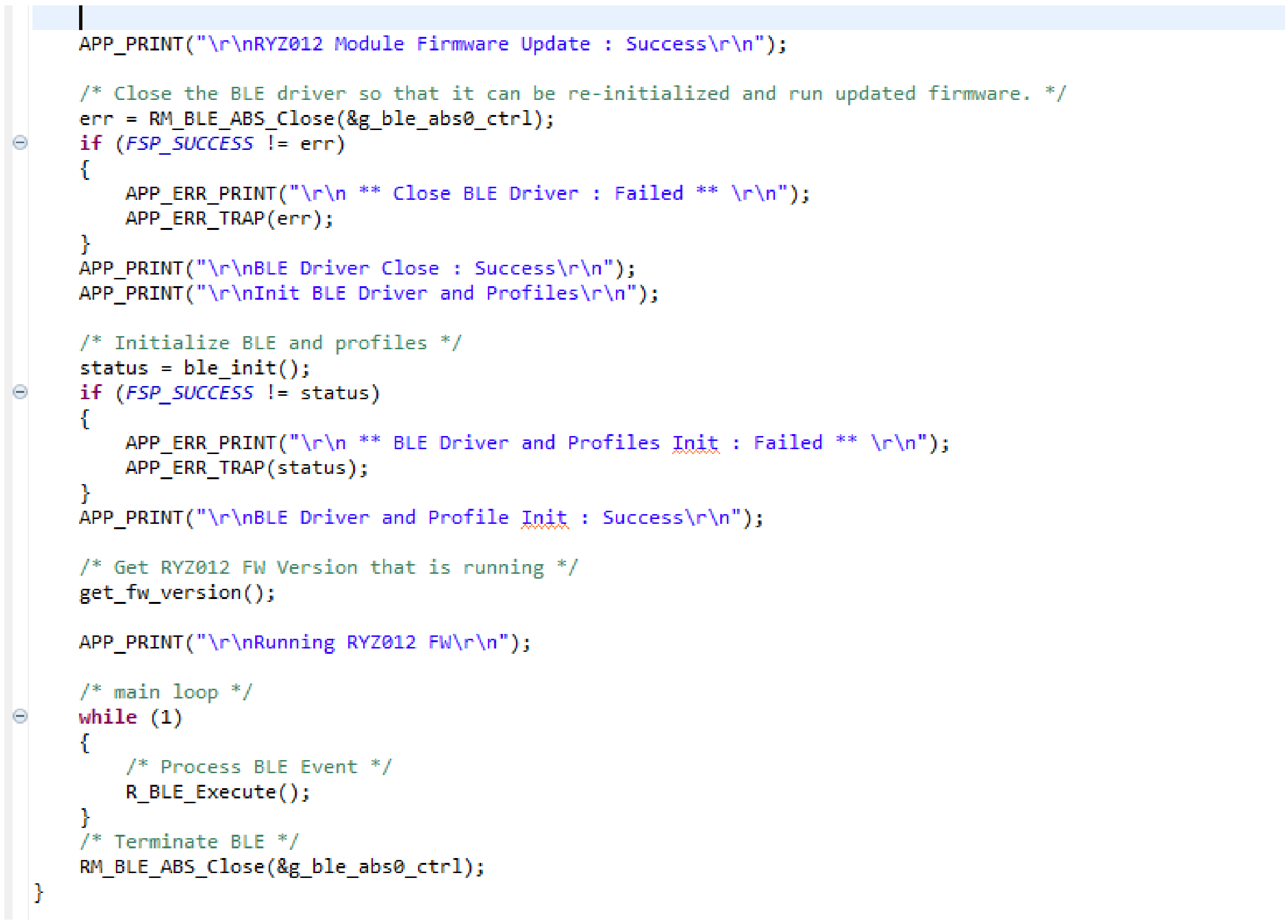
```
void app_main(void)
{
    fsp_err_t err;
    ble_status_t status;

    APP_PRINT("\r\nInitialize BLE Driver\r\n");
    /* Initialize BLE Driver */
    err = RM_BLE_ABS_Open(&g_ble_abs0_ctrl, &g_ble_abs0_cfg);
    if (FSP_SUCCESS != err)
    {
        APP_ERR_PRINT("\r\n ** Initialize BLE Driver : failed ** \r\n");
        APP_ERR_PRINT("\r\n ** Check that RYZ012 is connected to Eval Kit ** \r\n");
        APP_ERR_PRINT("\r\n ** Check that RYZ012 is inserted with correct pin orientation ** \r\n");
        APP_ERR_TRAP(err);
    }
    APP_PRINT("\r\nBLE Driver Init : Success\r\n");

    /* Get RYZ012 FW Version that is running */
    get_fw_version();
    APP_PRINT("\r\nCheck for RYZ012 FW Image in MCU Memory\r\n");

    bool b_image = false;
    b_image = is_flash_image_found( (uint8_t *) RYZ012_FWIMAGE_START_ADDRESS, (uint8_t *) RYZ012_FWIMAGE_END_ADDRESS );
    APP_PRINT("\r\nRYZ012 FW Image %s\r\n", (b_image ? "Found": "Not Found") );
    APP_PRINT("\r\nStarted RYZ012 Module Firmware Update\r\n");

    /* Call Vendor Specific API : Run the RYZ012 Module Firmware Update : MCU based procedure. */
    if ( b_image )
    {
        /* A FW Image is present
         * Determine the FW Image Length as programmed by XMODEM.
         * NOTE: If not programmed by XMODEM, the get_fw_image_length(...) function
         * will need to be modified to determine the Image end of file.
         **/
        uint32_t image_len;
        image_len = get_fw_image_length( (uint8_t *) RYZ012_FWIMAGE_START_ADDRESS, (uint8_t *) RYZ012_FWIMAGE_END_ADDRESS );
        APP_PRINT("\r\nRYZ012 FW Image size = %u bytes\r\n", image_len );
        if ( ( 0 < image_len ) && (( RYZ012_FWIMAGE_END_ADDRESS - RYZ012_FWIMAGE_START_ADDRESS ) > image_len ))
        {
            status = R_BLE_VS_UpdateModuleFirmware( (uint8_t const *) RYZ012_FWIMAGE_START_ADDRESS, image_len);
        }
        else
        {
            // Image size not in range
            status = BLE_ERR_INVALID_DATA;
        }
    }
    else
    {
        status = BLE_ERR_NOT_FOUND;  // RYZ012 flash image not found @ MCU RYZ012_FWIMAGE_START_ADDRESS
    }
```



```
    if (BLE_SUCCESS != status)
    {
        APP_ERR_PRINT("\r\n ** RYZ012 Module Firmware Update : Failed ** \r\n");
        switch ( status )
        {
            case BLE_ERR_INVALID_DATA :
                APP_ERR_PRINT("\r\n ** FW Image Size Invalid** \r\n");
                break;
            case BLE_ERR_NOT_FOUND :
                APP_ERR_PRINT("\r\n ** No FW Image Found in Memory @  RYZ012_FWIMAGE_START_ADDRESS** \r\n");
                break;
            default :
                break;
        }
        APP_ERR_TRAP(status);
    }
    APP_PRINT("\r\nRYZ012 Module Firmware Update : Success\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);
    }
    APP_PRINT("\r\nBLE Driver Close : Success\r\n");
    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\nRunning RYZ012 FW\r\n");

    /* main loop */
    while (1)
    {
        /* Process BLE Event */
        R_BLE_Execute();
    }
    /* Terminate BLE */
    RM_BLE_ABS_Close(&g_ble_abs0_ctrl);
}
```

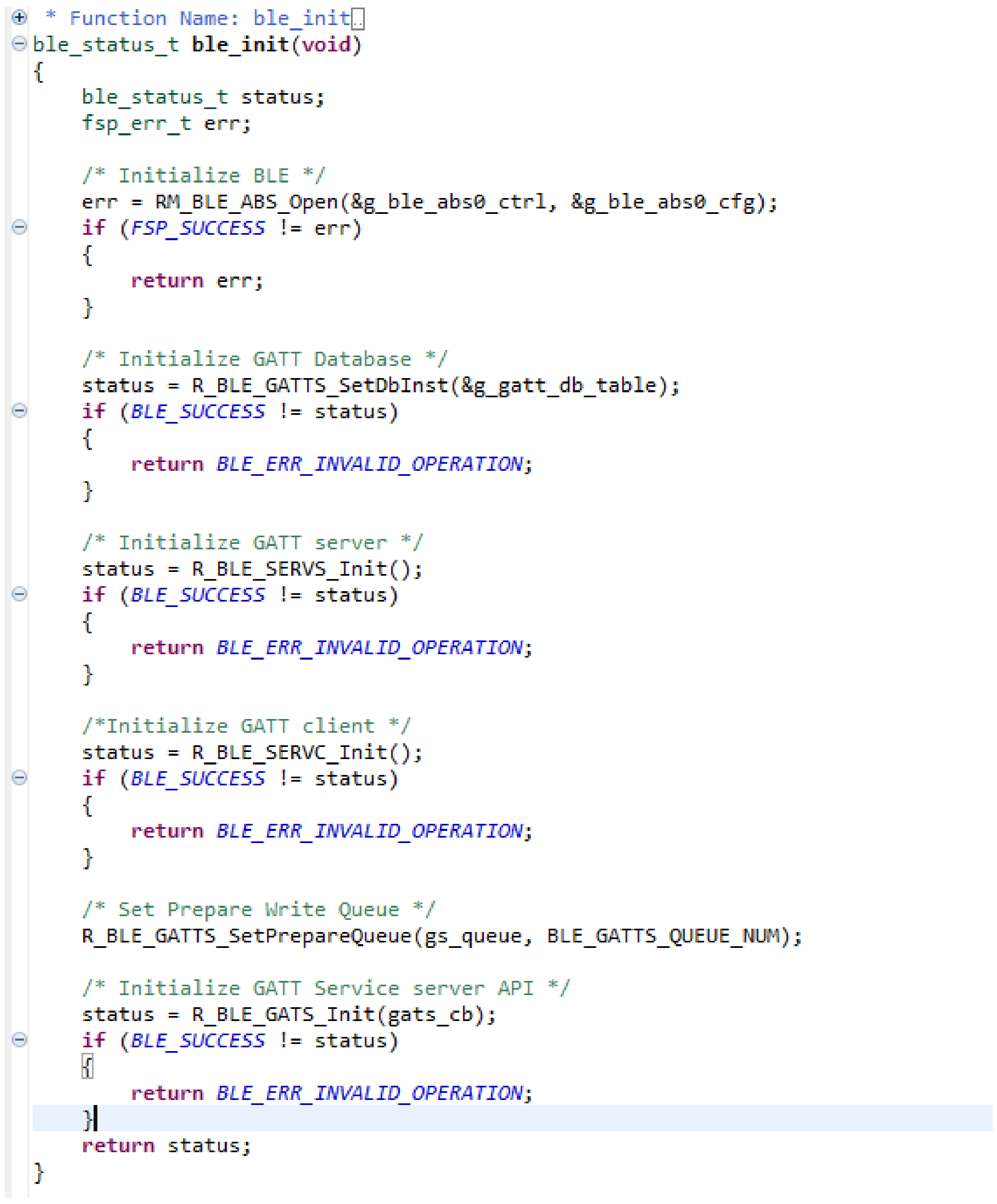
4.3 BLE Initialization Process

QE for BLE tool was used to create a basic BLE Profile with “RYZ012” set as the advertise name. 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.



```
    
    APP_PRINT("\r\nRYZ012 Module Firmware Update : Success\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);
    }
    APP_PRINT("\r\nBLE Driver Close : Success\r\n");
    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\nRunning RYZ012 FW\r\n");

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



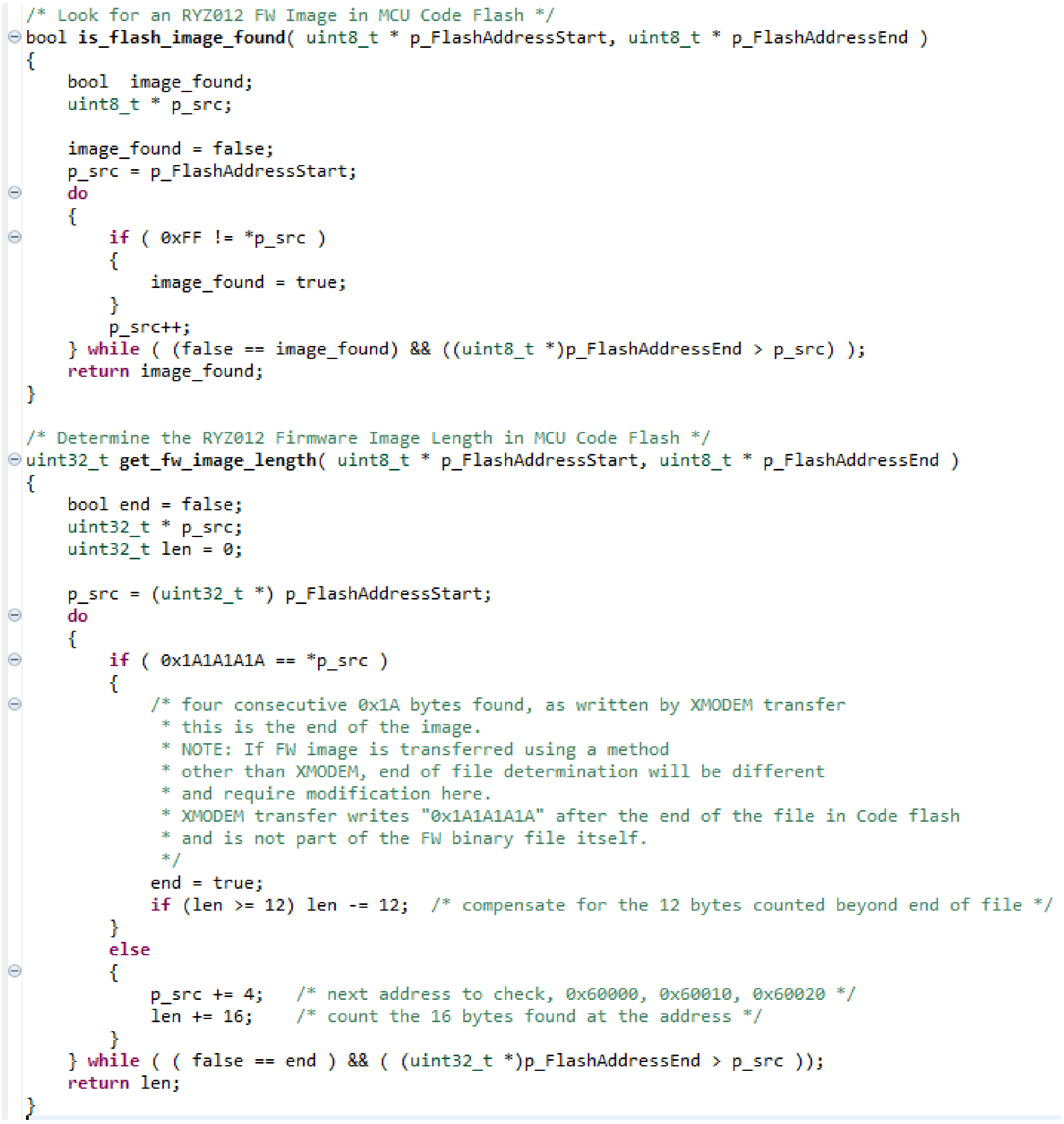
```
 * Function Name: ble_init
ble_status_t ble_init(void)
{
    ble_status_t status;
    fsp_err_t err;

    /* 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 Firmware Image Helper Functions

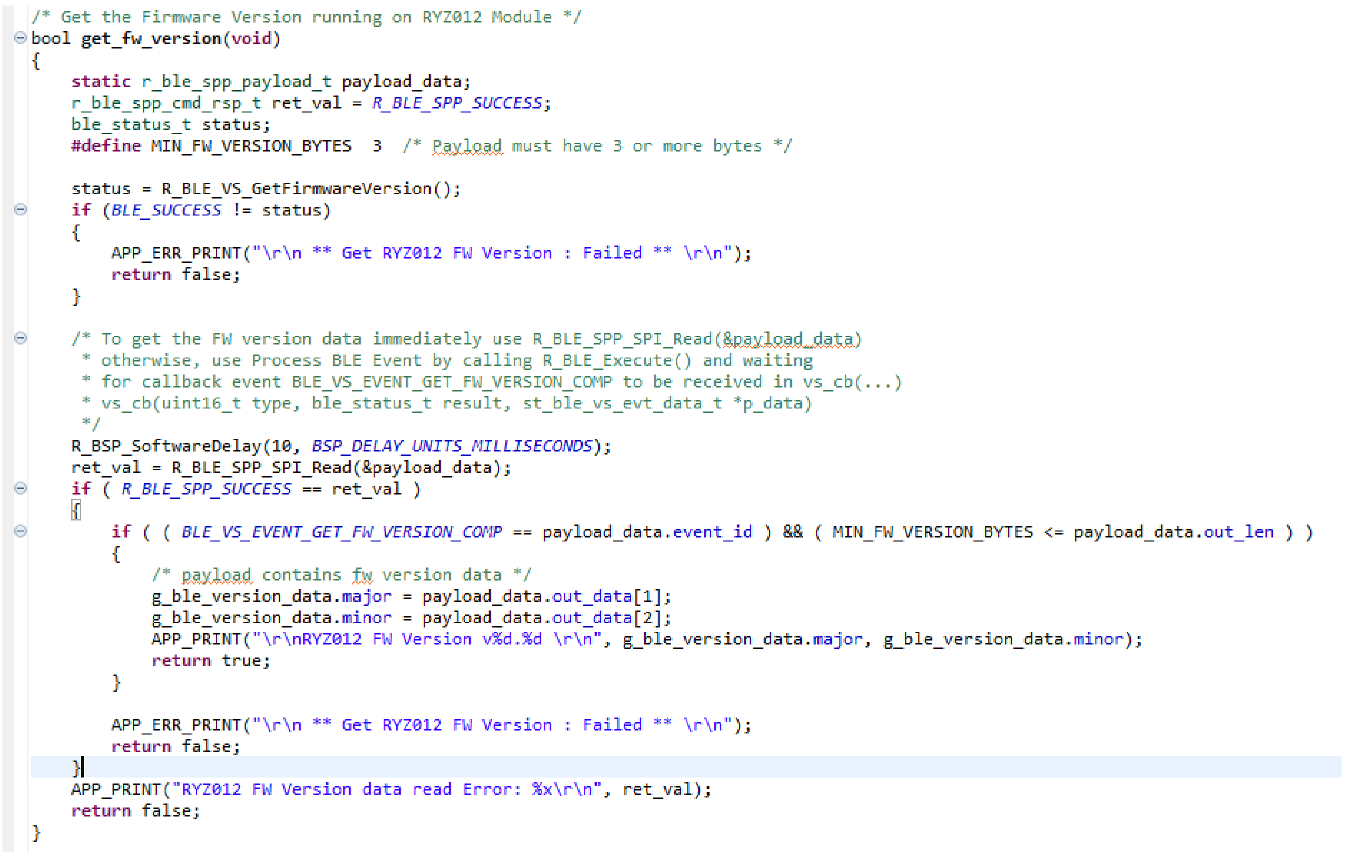
The following source code provides helper functions **is_flash_image_found(…)** and **get_fw_image_length(…)** which determine whether there is an RYZ012 firmware image in MCU flash and the length of the firmware image file in bytes.



```c
/* Look for an RYZ012 FW Image in MCU Code Flash */
bool is_flash_image_found( uint8_t * p_FlashAddressStart, uint8_t * p_FlashAddressEnd )
{
    bool  image_found;
    uint8_t * p_src;

    image_found = false;
    p_src = p_FlashAddressStart;
    do
    {
        if ( 0xFF != *p_src )
        {
            image_found = true;
        }
        p_src++;
    } while ( (false == image_found) && ((uint8_t *)p_FlashAddressEnd > p_src) );
    return image_found;
}

/* Determine the RYZ012 Firmware Image Length in MCU Code Flash */
uint32_t get_fw_image_length( uint8_t * p_FlashAddressStart, uint8_t * p_FlashAddressEnd )
{
    bool end = false;
    uint32_t * p_src;
    uint32_t len = 0;

    p_src = (uint32_t *) p_FlashAddressStart;
    do
    {
        if ( 0x1A1A1A1A == *p_src )
        {
            /* four consecutive 0x1A bytes found, as written by XMODEM transfer
             * this is the end of the image.
             * NOTE: If FW image is transferred using a method
             * other than XMODEM, end of file determination will be different
             * and require modification here.
             * XMODEM transfer writes "0x1A1A1A1A" after the end of the file in Code flash
             * and is not part of the FW binary file itself.
             */
            end = true;
            if (len >= 12) len -= 12;  /* compensate for the 12 bytes counted beyond end of file */
        }
        else
        {
            p_src += 4;   /* next address to check, 0x60000, 0x60010, 0x60020 */
            len += 16;    /* count the 16 bytes found at the address */
        }
    } while ( ( false == end ) && ( (uint32_t *)p_FlashAddressEnd > p_src ));
    return len;
}
```

4.6 Get Firmware Version Function

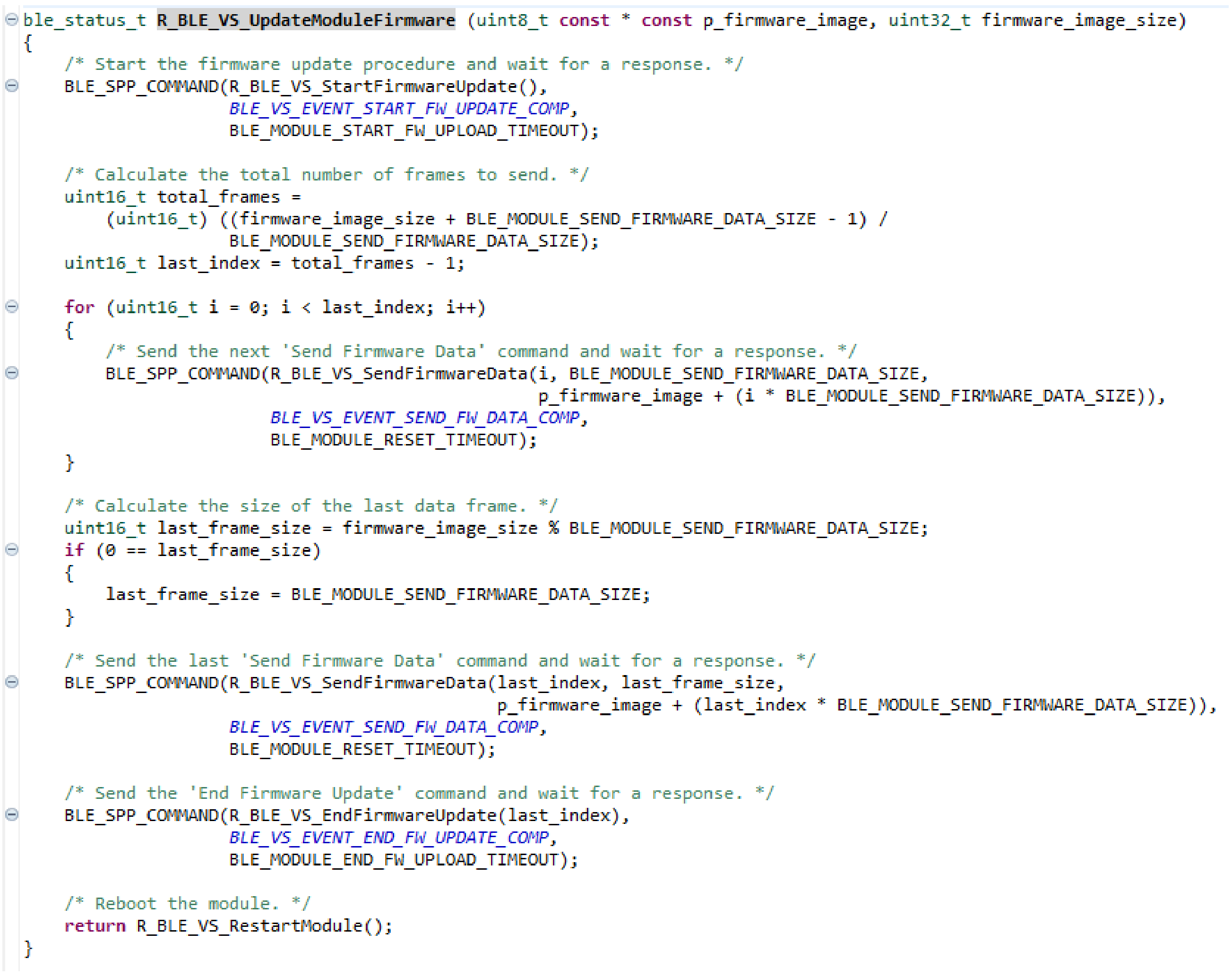
See the source code below to for the details of utility function **get_fw_version(void)** which is used to read the firmware version that is currently running on the RYZ012 module. The function calls the FSP provided function **R_BLE_VS_GetFirmwareVersion( )** and uses SPI to get the version response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



```c
/* 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;
}
```

4.7 Firmware Update Function

The FSP provided function **R_BLE_VS_UpdateModuleFirmware(…)** implements the execution steps of the firmware update to the RYZ012. The steps are covered in section 1.2, RYZ012 Firmware Update Process**.**



```
ble_status_t R_BLE_VS_UpdateModuleFirmware (uint8_t const * const p_firmware_image, uint32_t firmware_image_size)
{
    /* Start the firmware update procedure and wait for a response. */
    BLE_SPP_COMMAND(R_BLE_VS_StartFirmwareUpdate(),
                    BLE_VS_EVENT_START_FW_UPDATE_COMP,
                    BLE_MODULE_START_FW_UPLOAD_TIMEOUT);

    /* Calculate the total number of frames to send. */
    uint16_t total_frames =
        (uint16_t) ((firmware_image_size + BLE_MODULE_SEND_FIRMWARE_DATA_SIZE - 1) /
                    BLE_MODULE_SEND_FIRMWARE_DATA_SIZE);
    uint16_t last_index = total_frames - 1;

    for (uint16_t i = 0; i < last_index; i++)
    {
        /* Send the next 'Send Firmware Data' command and wait for a response. */
        BLE_SPP_COMMAND(R_BLE_VS_SendFirmwareData(i, BLE_MODULE_SEND_FIRMWARE_DATA_SIZE,
                                                  p_firmware_image + (i * BLE_MODULE_SEND_FIRMWARE_DATA_SIZE)),
                        BLE_VS_EVENT_SEND_FW_DATA_COMP,
                        BLE_MODULE_RESET_TIMEOUT);
    }

    /* Calculate the size of the last data frame. */
    uint16_t last_frame_size = firmware_image_size % BLE_MODULE_SEND_FIRMWARE_DATA_SIZE;
    if (0 == last_frame_size)
    {
        last_frame_size = BLE_MODULE_SEND_FIRMWARE_DATA_SIZE;
    }

    /* Send the last 'Send Firmware Data' command and wait for a response. */
    BLE_SPP_COMMAND(R_BLE_VS_SendFirmwareData(last_index, last_frame_size,
                                              p_firmware_image + (last_index * BLE_MODULE_SEND_FIRMWARE_DATA_SIZE)),
                    BLE_VS_EVENT_SEND_FW_DATA_COMP,
                    BLE_MODULE_RESET_TIMEOUT);

    /* Send the 'End Firmware Update' command and wait for a response. */
    BLE_SPP_COMMAND(R_BLE_VS_EndFirmwareUpdate(last_index),
                    BLE_VS_EVENT_END_FW_UPDATE_COMP,
                    BLE_MODULE_END_FW_UPLOAD_TIMEOUT);

    /* Reboot the module. */
    return R_BLE_VS_RestartModule();
}
```

The steps of execution are:

1. Send the SPP command to Start Firmware Update with `R_BLE_VS_StartFirmwareUpdate()` and wait for a response.
2. Calculate the total number of frames to transfer to RYZ012 based on image file size.
3. Send the data frames of image to RYZ012 with SPP command `R_BLE_VS_SendFirmwareData()` and wait for a response. Repeat sending frames until last data frame remains.
4. Send the last data frame of image with `R_BLE_VS_SendFirmwareData()` and wait for a response.
5. Send the End Firmware Update to RYZ012 with SPP command `R_BLE_VS_EndFirmwareUpdate()` and wait for a response.
6. Reboot the RYZ012 module with `R_BLE_VS_RestartModule()`.

5. How to Make and Configure a New Project

This section describes the configuration to create an RA MCU based firmware 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.

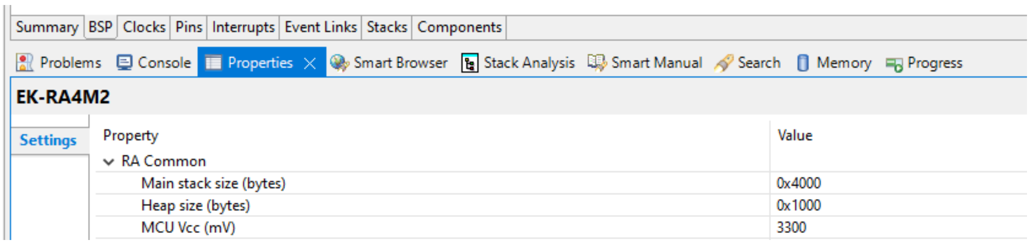
5.1 Create a New Project

Please see the **Renesas Flexible Software Package (FSP) User's Manual (R11UM0155EU)** for instructions on how to create a new project in **e^2 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 **e^2 studio** menu bar.

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

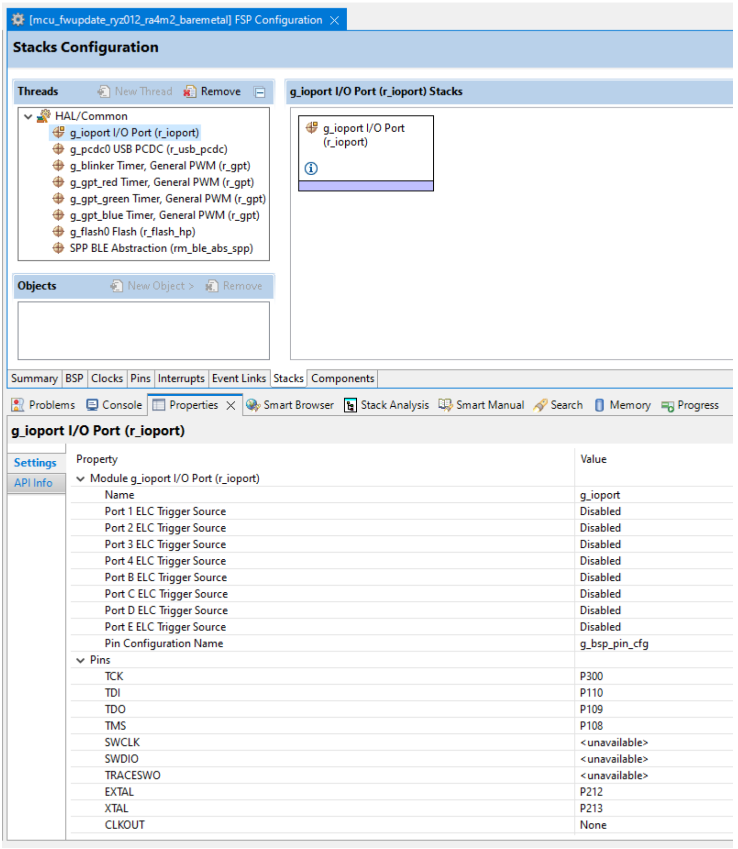


EK-RA4M2

Property	Value
RA Common	
Main stack size (bytes)	0x4000
Heap size (bytes)	0x1000
MCU Vcc (mV)	3300

Figure 29. BSP Configuration

5.3 Add the I/O Port Stack



HAL/Common
- g_ioport I/O Port (r_ioport)
- g_pcdc0 USB PCDC (r_usb_pcdc)
- 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)
- g_flash0 Flash (r_flash_hp)
- SPP BLE Abstraction (rm_ble_abs_spp)

g_ioport I/O Port (r_ioport)

Property	Value
Module g_ioport I/O Port (r_ioport)	
Name	g_ioport
Port 1 ELC Trigger Source	Disabled
Port 2 ELC Trigger Source	Disabled
Port 3 ELC Trigger Source	Disabled
Port 4 ELC Trigger Source	Disabled
Port B ELC Trigger Source	Disabled
Port C ELC Trigger Source	Disabled
Port D ELC Trigger Source	Disabled
Port E ELC Trigger Source	Disabled
Pin Configuration Name	g_bsp_pin_cfg
Pins	
TCK	P300
TDI	P110
TDO	P109
TMS	P108
SWCLK	<unavailable>
SWDIO	<unavailable>
TRACESWO	<unavailable>
EXTAL	P212
XTAL	P213
CLKOUT	None

Figure 30. Adding I/O Stack

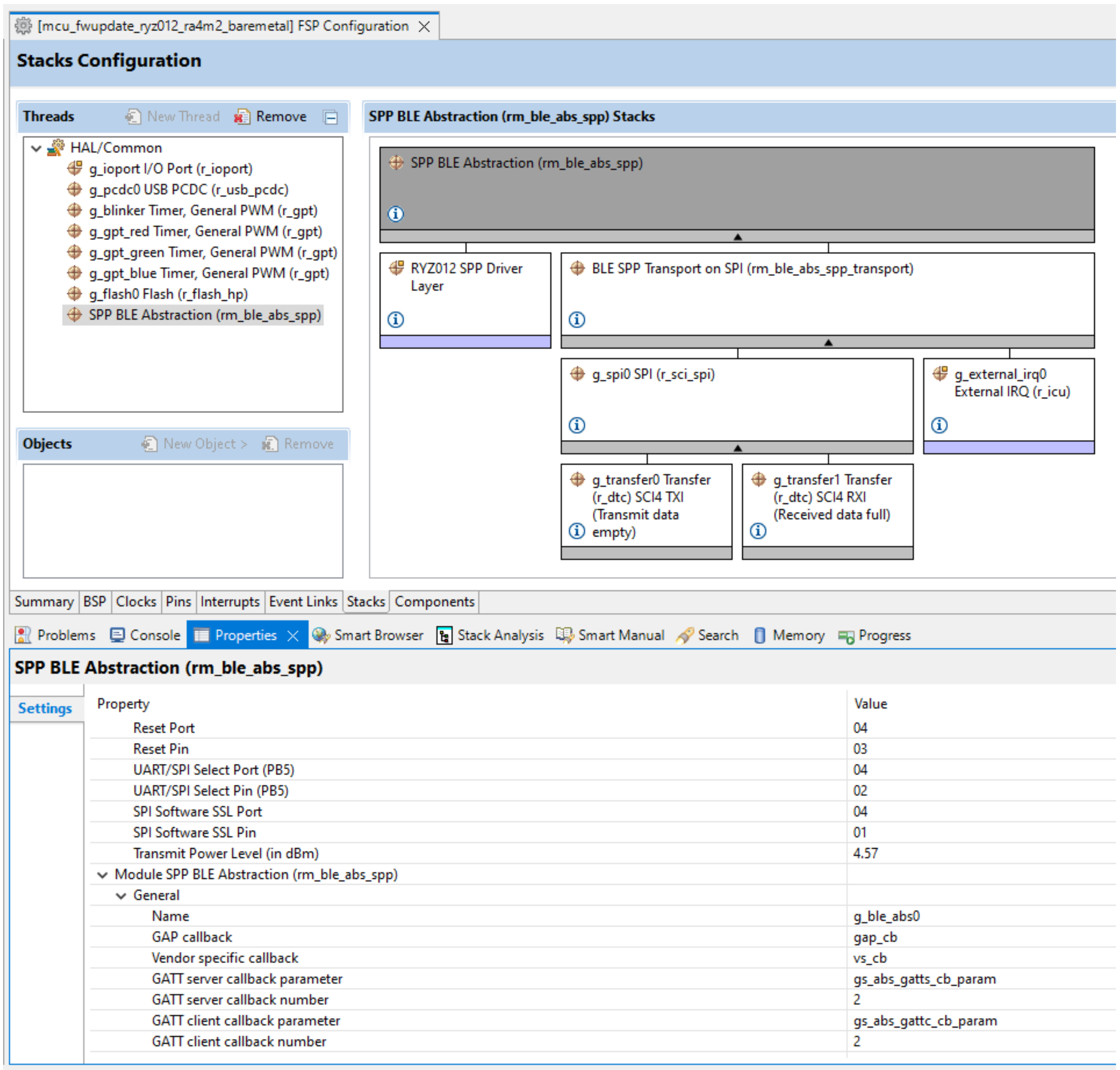
5.4 Add and Configure the SPP BLE Module

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.


[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration
Stacks Configuration
Threads
New Thread
Remove
HAL/Common
g_ioport I/O Port (r_ioport)
g_pcdc0 USB PCDC (r_usb_pcdc)
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)
g_flash0 Flash (r_flash_hp)
SPP BLE Abstraction (rm_ble_abs_spp)
Objects
New Object >
Remove
SPP BLE Abstraction (rm_ble_abs_spp) Stacks
SPP BLE Abstraction (rm_ble_abs_spp)
RYZ012 SPP Driver Layer
BLE SPP Transport on SPI (rm_ble_abs_spp_transport)
g_spi0 SPI (r_sci_spi)
g_external_irq0 External IRQ (r_icu)
g_transfer0 Transfer (r_dtc) SCI4 TXI (Transmit data empty)
g_transfer1 Transfer (r_dtc) SCI4 RXI (Received data full)
Summary
BSP
Clocks
Pins
Interrupts
Event Links
Stacks
Components
Problems
Console
Properties
Smart Browser
Stack Analysis
Smart Manual
Search
Memory
Progress
SPP BLE Abstraction (rm_ble_abs_spp)
Settings

Property	Value
Reset Port	04
Reset Pin	03
UART/SPI Select Port (PB5)	04
UART/SPI Select Pin (PB5)	02
SPI Software SSL Port	04
SPI Software SSL Pin	01
Transmit Power Level (in dBm)	4.57
Module SPP BLE Abstraction (rm_ble_abs_spp)	
General	
Name	g_ble_abs0
GAP callback	gap_cb
Vendor specific callback	vs_cb
GATT server callback parameter	gs_abs_gatts_cb_param
GATT server callback number	2
GATT client callback parameter	gs_abs_gattc_cb_param
GATT client callback number	2

Figure 31. 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 31, SPP BLE ABS Module for more details.

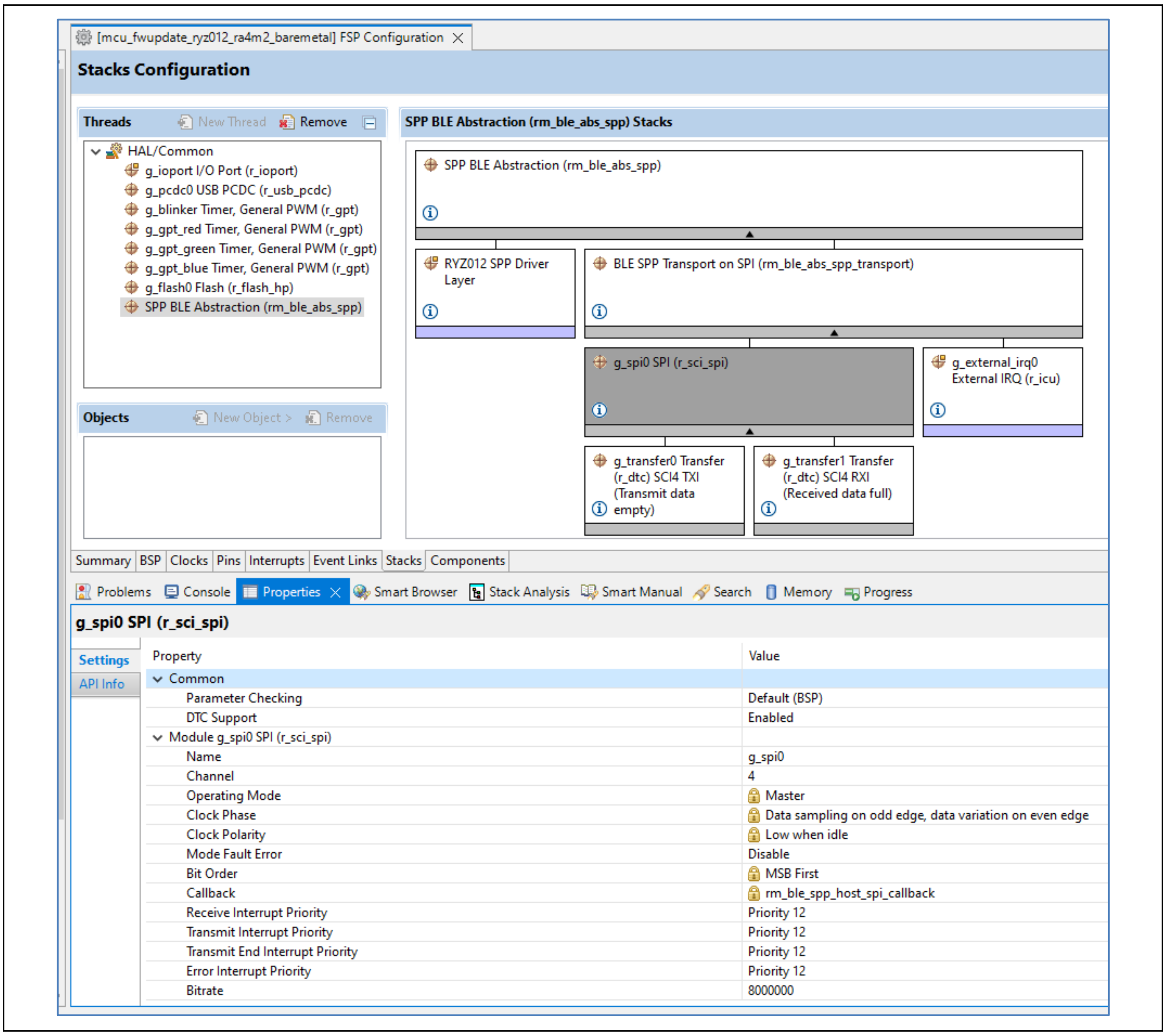
Table 5. SPP Module Configuration

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.

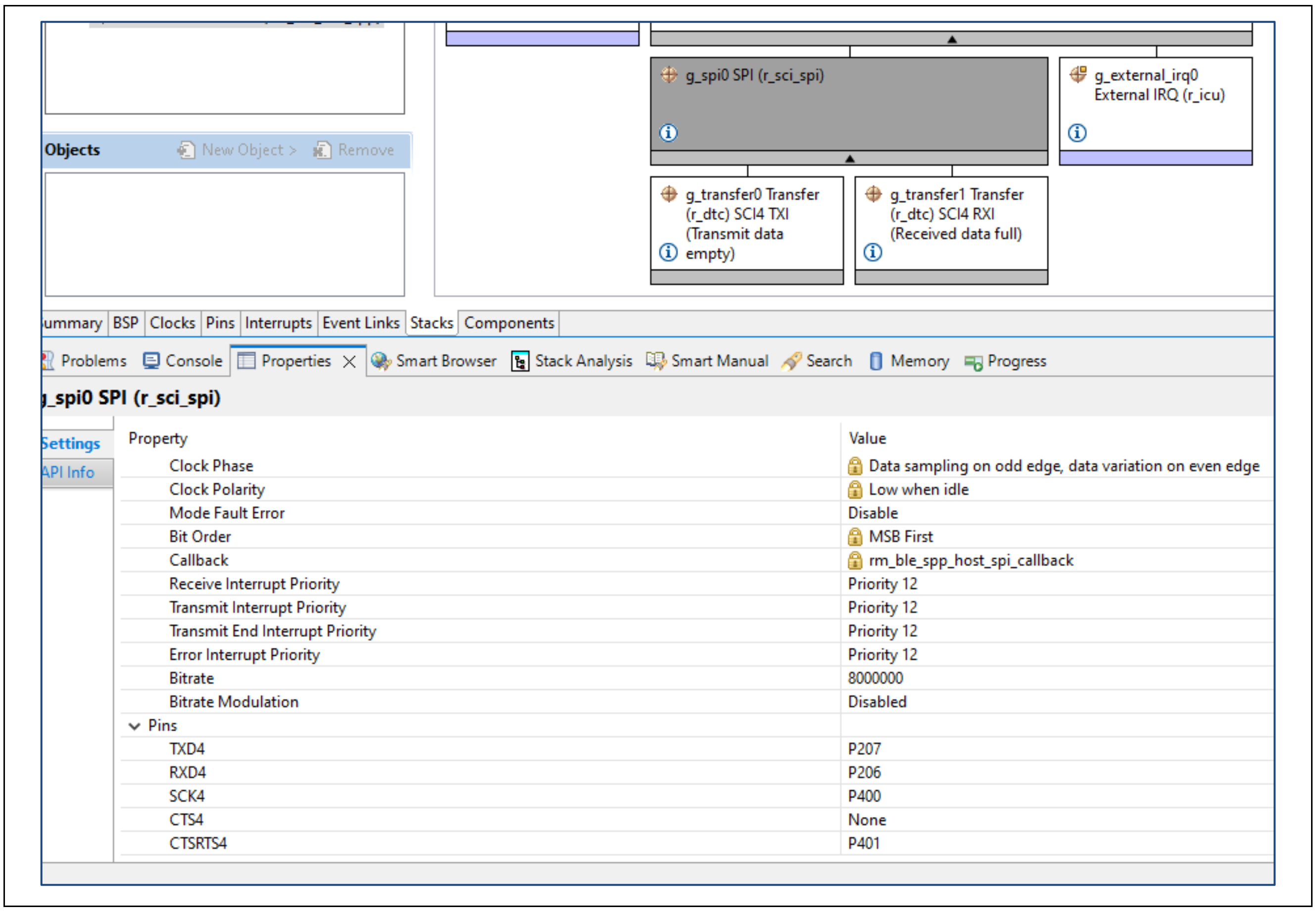
1. Click g_spi0 SPI Driver on r_sci_spi.


[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration
Stacks Configuration
Threads
New Thread
Remove
HAL/Common
g_ioport I/O Port (r_ioport)
g_pcdc0 USB PCDC (r_usb_pcdc)
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)
g_flash0 Flash (r_flash_hp)
SPP BLE Abstraction (rm_ble_abs_spp)
Objects
New Object >
Remove
SPP BLE Abstraction (rm_ble_abs_spp) Stacks
SPP BLE Abstraction (rm_ble_abs_spp)
RYZ012 SPP Driver Layer
BLE SPP Transport on SPI (rm_ble_abs_spp_transport)
g_spi0 SPI (r_sci_spi)
g_external_irq0 External IRQ (r_icu)
g_transfer0 Transfer (r_dtc) SCI4 TXI (Transmit data empty)
g_transfer1 Transfer (r_dtc) SCI4 RXI (Received data full)
Summary
BSP
Clocks
Pins
Interrupts
Event Links
Stacks
Components
Problems
Console
Properties
Smart Browser
Stack Analysis
Smart Manual
Search
Memory
Progress
g_spi0 SPI (r_sci_spi)
Settings
API Info

Property	Value
Common	
Parameter Checking	Default (BSP)
DTC Support	Enabled
Module g_spi0 SPI (r_sci_spi)	
Name	g_spi0
Channel	4
Operating Mode	Master
Clock Phase	Data sampling on odd edge, data variation on even edge
Clock Polarity	Low when idle
Mode Fault Error	Disable
Bit Order	MSB First
Callback	rm_ble_spp_host_spi_callback
Receive Interrupt Priority	Priority 12
Transmit Interrupt Priority	Priority 12
Transmit End Interrupt Priority	Priority 12
Error Interrupt Priority	Priority 12
Bitrate	8000000

Figure 32. SPI Module Properties

See Figure 32, 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 34, 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.


Objects
New Object >
Remove
g_spi0 SPI (r_sci_spi)
g_external_irq0 External IRQ (r_icu)
g_transfer0 Transfer (r_dtc) SCI4 TXI (Transmit data empty)
g_transfer1 Transfer (r_dtc) SCI4 RXI (Received data full)
ummary BSP Clocks Pins Interrupts Event Links Stacks Components
Problems Console Properties Smart Browser Stack Analysis Smart Manual Search Memory Progress
Settings
API Info

g_spi0 SPI (r_sci_spi)

Property	Value
Clock Phase	Data sampling on odd edge, data variation on even edge
Clock Polarity	Low when idle
Mode Fault Error	Disable
Bit Order	MSB First
Callback	rm_ble_spp_host_spi_callback
Receive Interrupt Priority	Priority 12
Transmit Interrupt Priority	Priority 12
Transmit End Interrupt Priority	Priority 12
Error Interrupt Priority	Priority 12
Bitrate	8000000
Bitrate Modulation	Disabled
Pins	
TXD4	P207
RXD4	P206
SCK4	P400
CTS4	None
CTSRTS4	P401

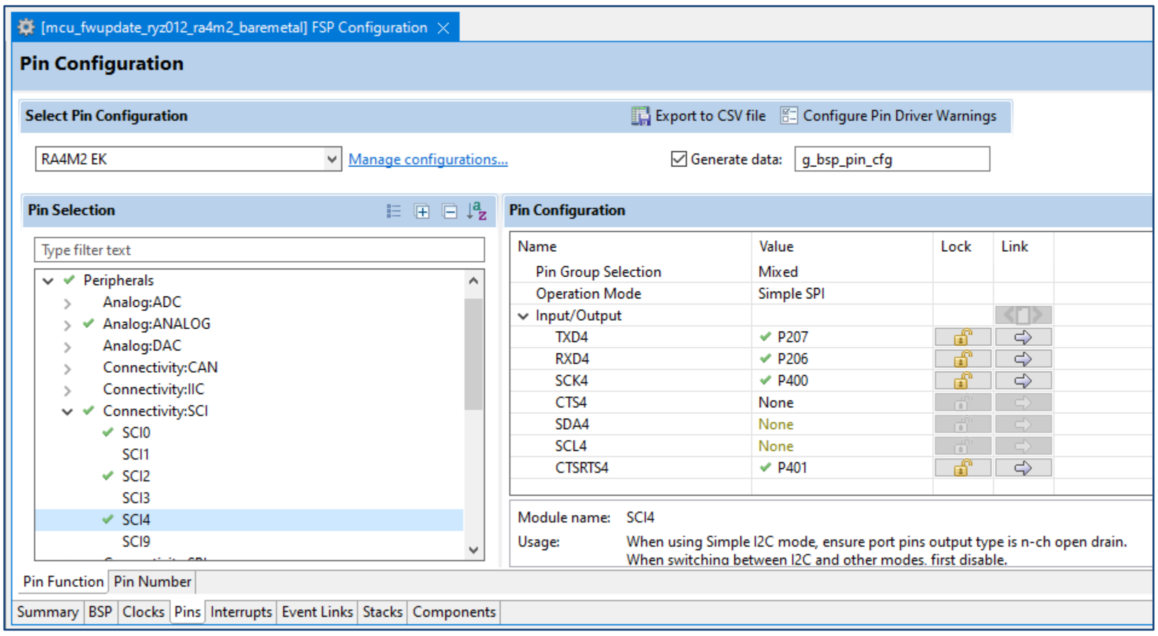
Figure 33. SPI Module Properties – Pin Assignments

2. Change the following SPI Configuration Properties in the properties window as in Figure 32 and Figure 33.

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 32, SPI Module Properties, `g_transfer0 SCI4 TXI` and `g_transfer1 SCI4 RXI` for more details.


[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration

Pin Configuration

Select Pin Configuration — Export to CSV file — Configure Pin Driver Warnings

RA4M2 EK — Manage configurations... — ✓ Generate data: g_bsp_pin_cfg

Pin Selection

- Peripherals ✓
  - Analog:ADC
  - Analog:ANALOG ✓
  - Analog:DAC
  - Connectivity:CAN
  - Connectivity:IIC
  - Connectivity:SCI ✓
    - SCI0 ✓
    - SCI1
    - SCI2 ✓
    - SCI3
    - SCI4 ✓
    - SCI9

Pin Configuration

Name	Value	Lock	Link
Pin Group Selection	Mixed		
Operation Mode	Simple SPI		
Input/Output			
TXD4	✓ P207		
RXD4	✓ P206		
SCK4	✓ P400		
CTS4	None		
SDA4	None		
SCL4	None		
CTSRTS4	✓ P401		

Module name: SCI4
Usage: When using Simple I2C mode, ensure port pins output type is n-ch open drain. When switching between I2C and other modes, first disable.

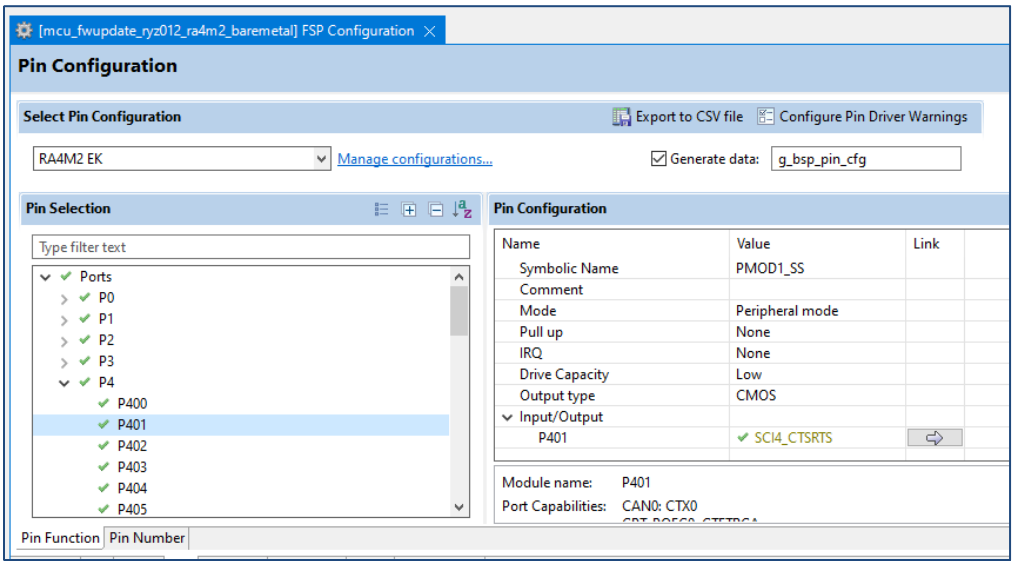
Pin Function | Pin Number

Summary | BSP | Clocks | Pins | Interrupts | Event Links | Stacks | Components

Figure 34. 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 34, 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.



[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration

Pin Configuration

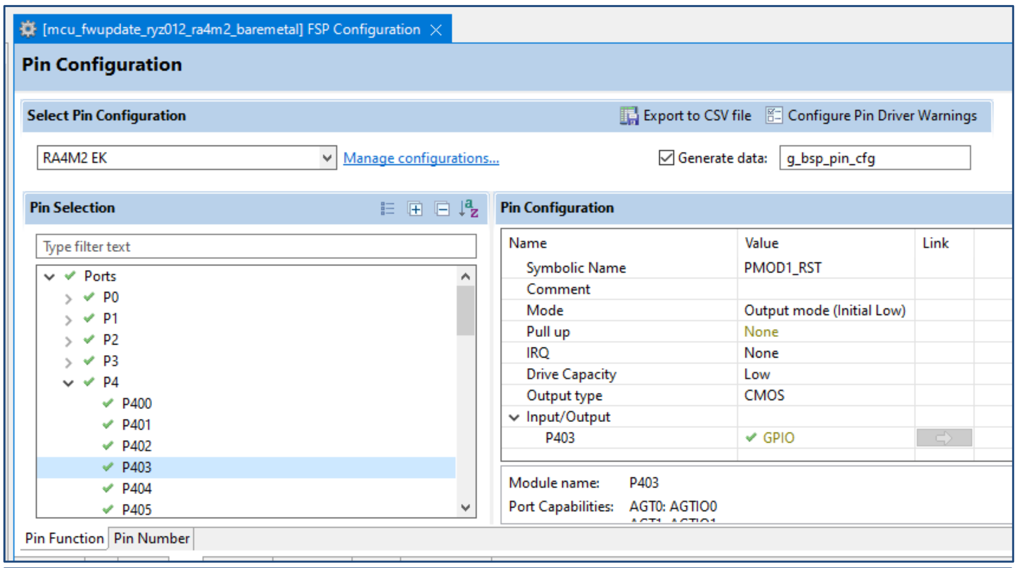
Select Pin Configuration: RA4M2 EK — Manage configurations... — Export to CSV file — Configure Pin Driver Warnings — ☑ Generate data: g_bsp_pin_cfg

Pin Selection: Ports > P0, P1, P2, P3, P4 > P400, **P401**, P402, P403, P404, P405

Name	Value	Link
Symbolic Name	PMOD1_SS	
Comment		
Mode	Peripheral mode	
Pull up	None	
IRQ	None	
Drive Capacity	Low	
Output type	CMOS	
Input/Output		
P401	✓ SCI4_CTSRTS	

Module name: P401
Port Capabilities: CAN0: CTX0

Pin Function | Pin Number



[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration

Pin Configuration

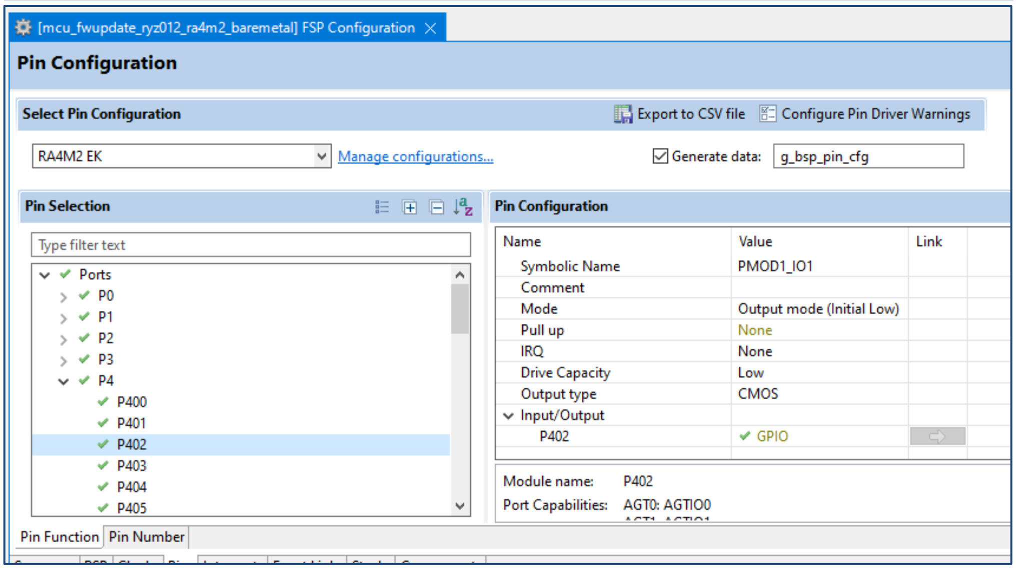
Select Pin Configuration: RA4M2 EK — Manage configurations... — Export to CSV file — Configure Pin Driver Warnings — ☑ Generate data: g_bsp_pin_cfg

Pin Selection: Ports > P0, P1, P2, P3, P4 > P400, P401, P402, **P403**, P404, P405

Name	Value	Link
Symbolic Name	PMOD1_RST	
Comment		
Mode	Output mode (Initial Low)	
Pull up	None	
IRQ	None	
Drive Capacity	Low	
Output type	CMOS	
Input/Output		
P403	✓ GPIO	

Module name: P403
Port Capabilities: AGT0: AGTIO0

Pin Function | Pin Number



[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration

Pin Configuration

Select Pin Configuration: RA4M2 EK — Manage configurations... — Export to CSV file — Configure Pin Driver Warnings — ☑ Generate data: g_bsp_pin_cfg

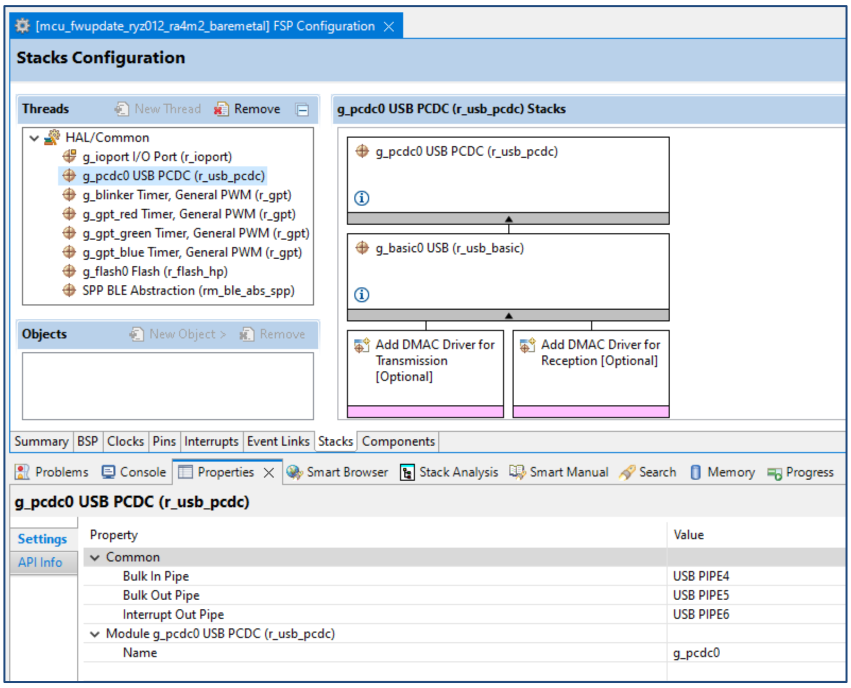
Pin Selection: Ports > P0, P1, P2, P3, P4 > P400, P401, **P402**, P403, P404, P405

Name	Value	Link
Symbolic Name	PMOD1_IO1	
Comment		
Mode	Output mode (Initial Low)	
Pull up	None	
IRQ	None	
Drive Capacity	Low	
Output type	CMOS	
Input/Output		
P402	✓ GPIO	

Module name: P402
Port Capabilities: AGT0: AGTIO0

Pin Function | Pin Number

Figure 35. Pin Configuration > Ports


[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration

Stacks Configuration

Threads
- HAL/Common
  - g_ioport I/O Port (r_ioport)
  - g_pcdc0 USB PCDC (r_usb_pcdc)
  - 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)
  - g_flash0 Flash (r_flash_hp)
  - SPP BLE Abstraction (rm_ble_abs_spp)

g_pcdc0 USB PCDC (r_usb_pcdc) Stacks
- g_pcdc0 USB PCDC (r_usb_pcdc)
- g_basic0 USB (r_usb_basic)
- Add DMAC Driver for Transmission [Optional]
- Add DMAC Driver for Reception [Optional]

g_pcdc0 USB PCDC (r_usb_pcdc)

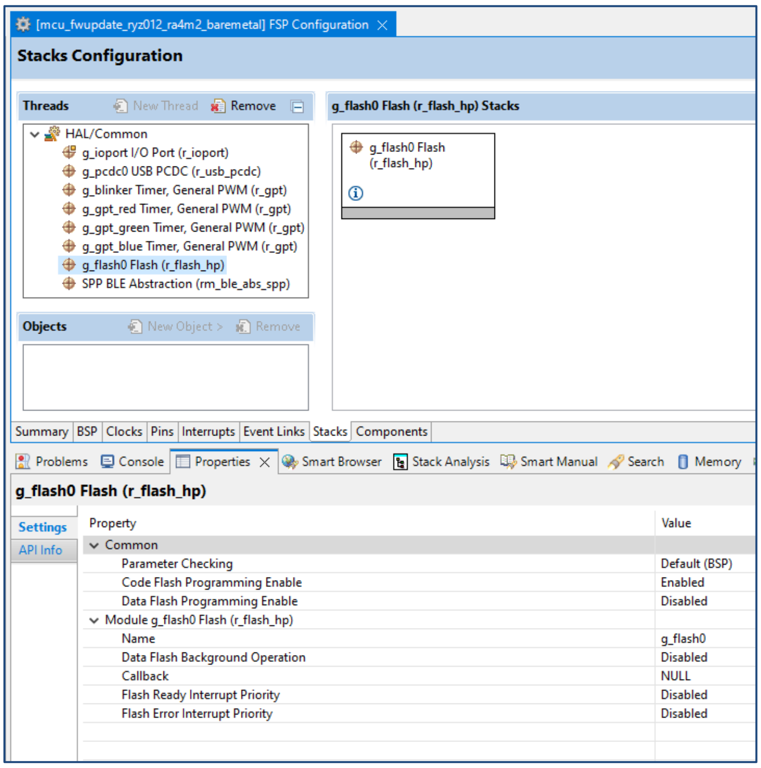
Property	Value
Common	
Bulk In Pipe	USB PIPE4
Bulk Out Pipe	USB PIPE5
Interrupt Out Pipe	USB PIPE6
Module g_pcdc0 USB PCDC (r_usb_pcdc)	
Name	g_pcdc0

Figure 36. Stack Configuration > g_pcdc0 USB PCDC (r_usb_pcdc)

5.5 Add USB PCDC for Tera Term Menu and XMODEM File Transfer

Add the USB PCDC stack for the USB Full Speed Port (J11) on the EK-RA4M2 and Tera Term communications.

5.6 Add MCU Flash Driver for XMODEM File Storage


[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration

Stacks Configuration

Threads
- HAL/Common
  - g_ioport I/O Port (r_ioport)
  - g_pcdc0 USB PCDC (r_usb_pcdc)
  - 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)
  - g_flash0 Flash (r_flash_hp)
  - SPP BLE Abstraction (rm_ble_abs_spp)

g_flash0 Flash (r_flash_hp) Stacks
- g_flash0 Flash (r_flash_hp)

g_flash0 Flash (r_flash_hp)

Property	Value
Common	
Parameter Checking	Default (BSP)
Code Flash Programming Enable	Enabled
Data Flash Programming Enable	Disabled
Module g_flash0 Flash (r_flash_hp)	
Name	g_flash0
Data Flash Background Operation	Disabled
Callback	NULL
Flash Ready Interrupt Priority	Disabled
Flash Error Interrupt Priority	Disabled

Figure 37. Stack Configuration > g_flash0 Flash (r_flash_hp)

Common > Code Flash Programming Enable must be set to **Enabled**.

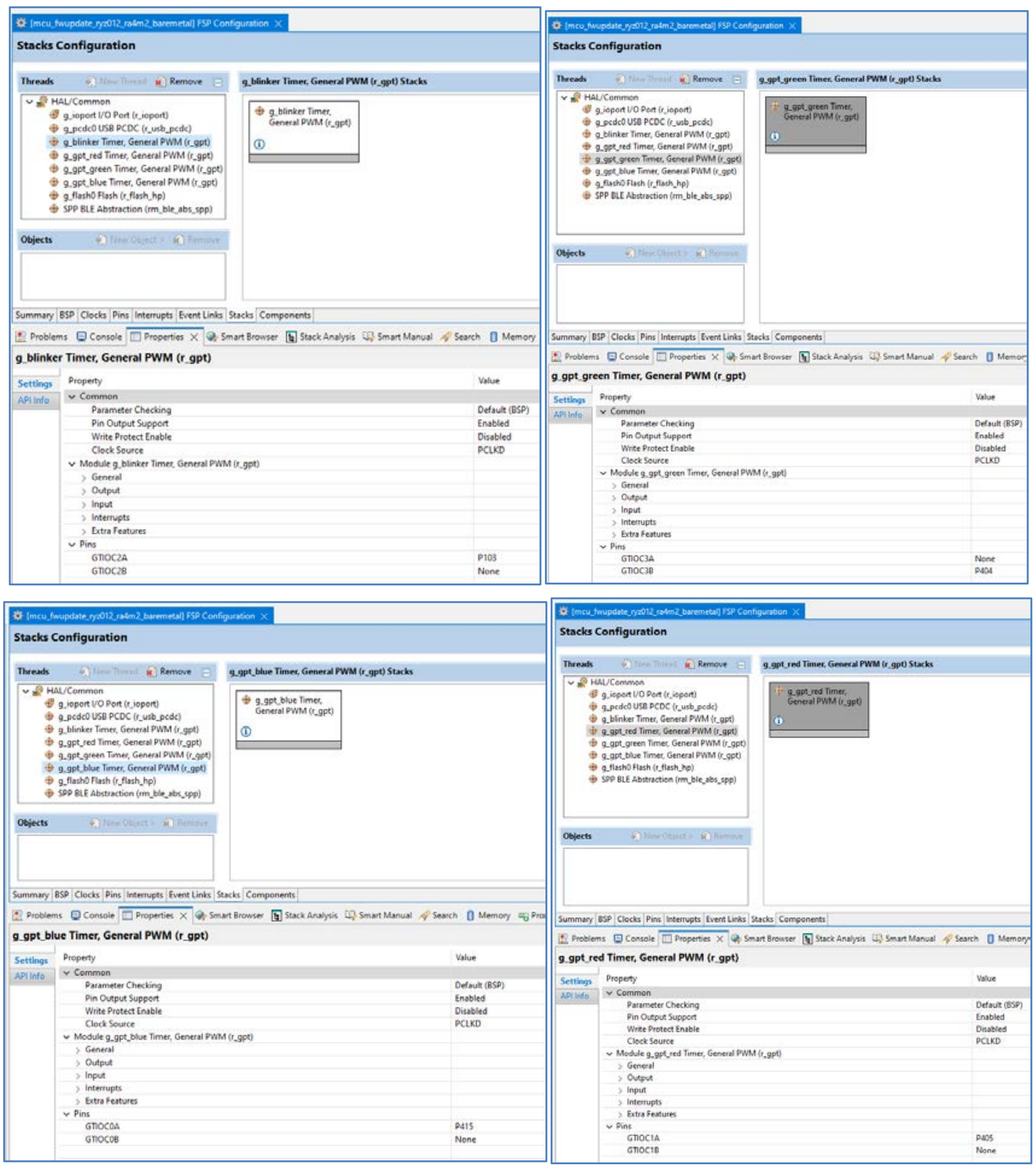
The MCU Flash size must be large enough for the RYZ012A firmware binary image file. For the EK-RA4M2, in this sample project (see file `src/common_init.h`), the code flash used for the RYZ012 firmware image starts at

`#define RYZ012_FWIMAGE_START_ADDRESS 0x00060000 which is above the user application space. Make sure you adjust the RYZ012_FWIMAGE_START_ADDRESS and RYZ012_FWIMAGE_END_ADDRESS address for your custom application.`

The largest RYZ012A firmware image used in this sample application is **102436 bytes** for **8258_moduleV5_5.bin**.

The RYZ012A firmware image size can increase in future RYZ012 firmware releases as new features are added. Be sure to increase the number of Flash Blocks required to store the firmware image file as the RYZ012A image size increases.

See file `src/common_init.h` `#define RYZ012_FWIMAGE_END_ADDRESS 0x00080000` and #define `RYZ012_FWIMAGE_NUM_BLOCKS.`


[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration
Stacks Configuration
Threads
New Thread
Remove
HAL/Common
g_ioport I/O Port (r_ioport)
g_pcdc0 USB PCDC (r_usb_pcdc)
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)
g_flash0 Flash (r_flash_hp)
SPP BLE Abstraction (rm_ble_abs_spp)
Objects
New Object >
Summary
BSP
Clocks
Pins
Interrupts
Event Links
Stacks
Components
Problems
Console
Properties
Smart Browser
Stack Analysis
Smart Manual
Search
Memory
Settings
API Info

g_blinker Timer, General PWM (r_gpt) Stacks

g_blinker Timer, General PWM (r_gpt)

Property	Value
Common	
Parameter Checking	Default (BSP)
Pin Output Support	Enabled
Write Protect Enable	Disabled
Clock Source	PCLKD
Module g_blinker Timer, General PWM (r_gpt)	
General	
Output	
Input	
Interrupts	
Extra Features	
Pins	
GTIOC2A	P103
GTIOC2B	None

g_gpt_green Timer, General PWM (r_gpt) Stacks

g_gpt_green Timer, General PWM (r_gpt)

Property	Value
Common	
Parameter Checking	Default (BSP)
Pin Output Support	Enabled
Write Protect Enable	Disabled
Clock Source	PCLKD
Module g_gpt_green Timer, General PWM (r_gpt)	
General	
Output	
Input	
Interrupts	
Extra Features	
Pins	
GTIOC3A	None
GTIOC3B	P404

g_gpt_blue Timer, General PWM (r_gpt) Stacks

g_gpt_blue Timer, General PWM (r_gpt)

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

g_gpt_red Timer, General PWM (r_gpt) Stacks

g_gpt_red Timer, General PWM (r_gpt)

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

Figure 38. Stack Configuration Settings

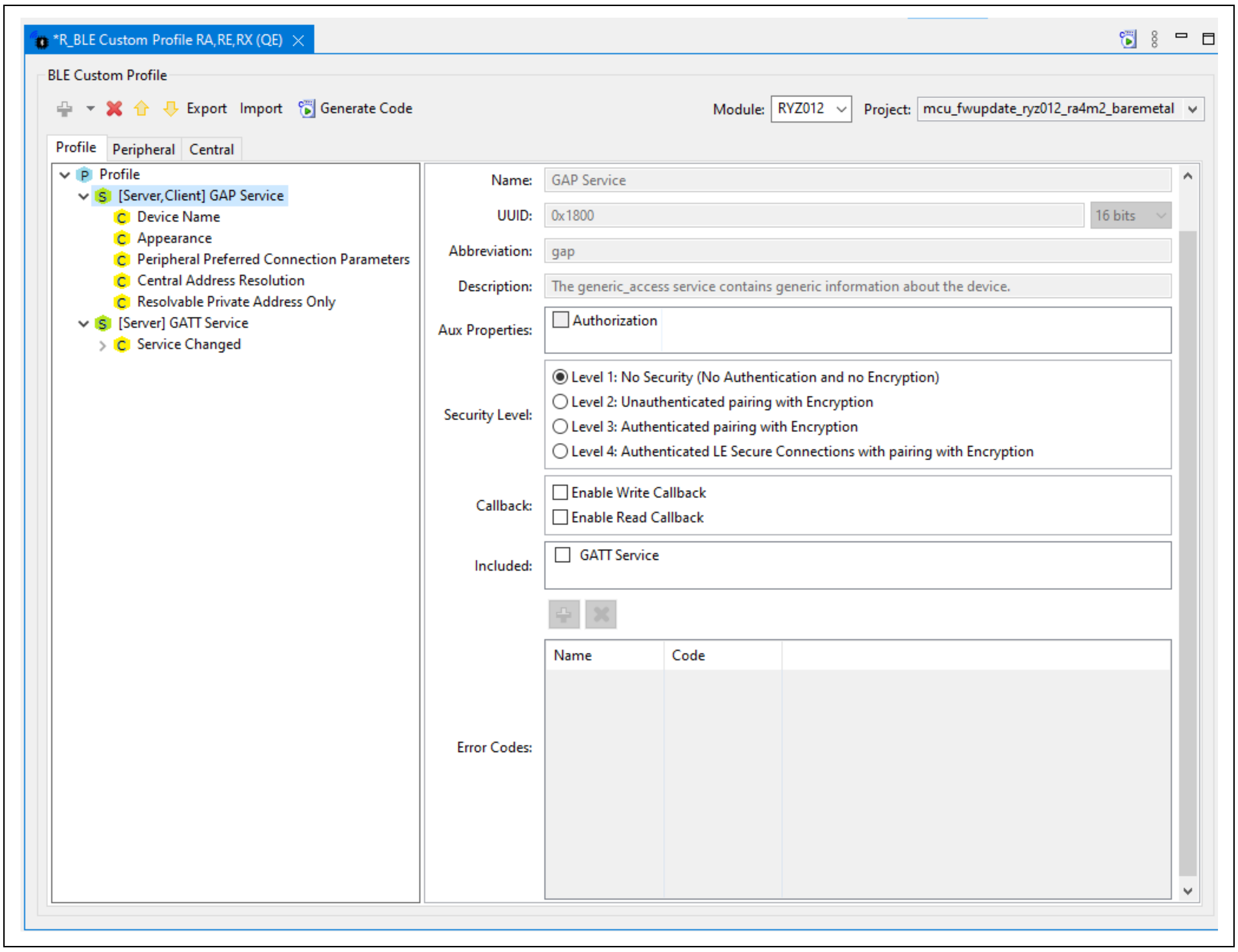
5.7 Add General PWM Timers for LED Indications

This step is optional and is not required for Firmware Update functionality. 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 and board LEDs.

5.8 Create BLE Profile with QE for BLE

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 e^2 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.


*R_BLE Custom Profile RA,RE,RX (QE)
BLE Custom Profile
Export
Import
Generate Code
Profile
Peripheral
Central
Profile
[Server,Client] GAP Service
Device Name
Appearance
Peripheral Preferred Connection Parameters
Central Address Resolution
Resolvable Private Address Only
[Server] GATT Service
Service Changed

Module: RYZ012
Project: mcu_fwupdate_ryz012_ra4m2_baremetal

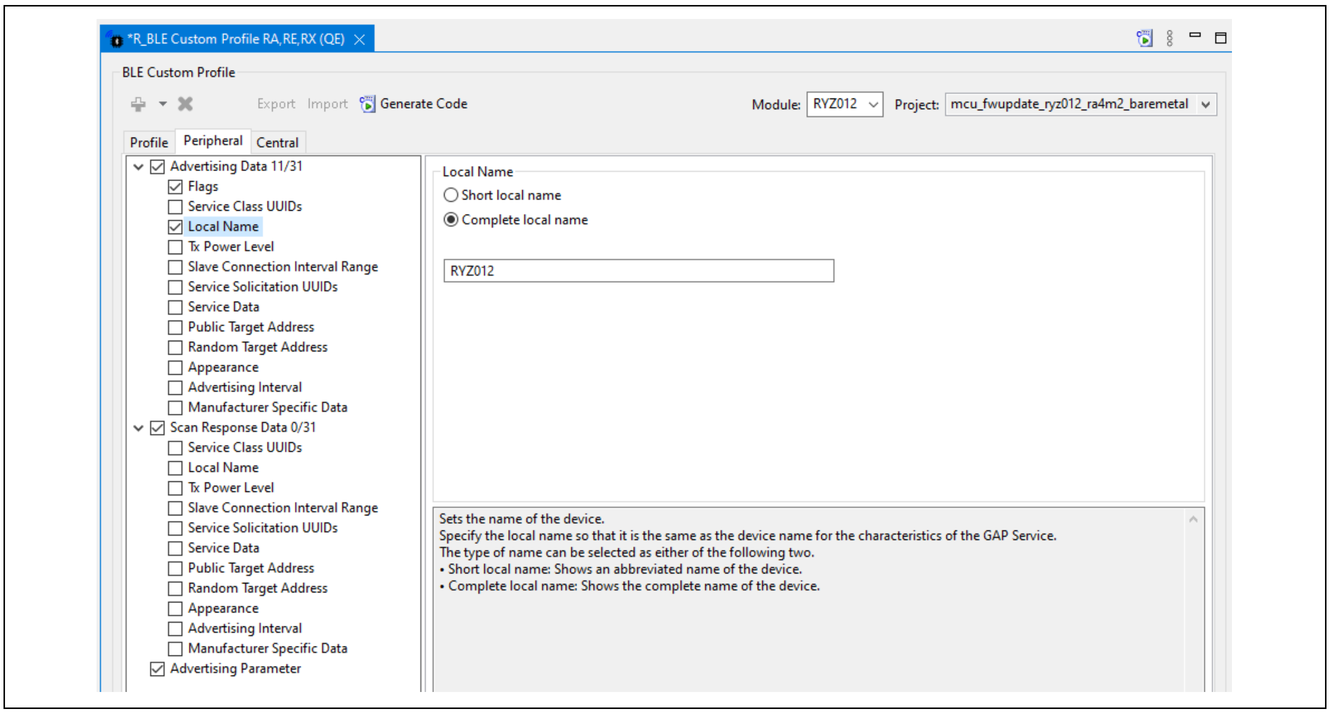
Name: GAP Service
UUID: 0x1800 16 bits
Abbreviation: gap
Description: The generic_access service contains generic information about the device.
Aux Properties: ☐ Authorization
Security Level:
- ● Level 1: No Security (No Authentication and no Encryption)
- ○ Level 2: Unauthenticated pairing with Encryption
- ○ Level 3: Authenticated pairing with Encryption
- ○ Level 4: Authenticated LE Secure Connections with pairing with Encryption

Callback: ☐ Enable Write Callback ☐ Enable Read Callback
Included: ☐ GATT Service
Error Codes:

Name	Code

Figure 39. Creating BLE Profile with QE for BLE

The minimum BLE Profile is shown above and nothing specific is needed for the MCU based Firmware Update of the RYZ012A. The only optional modification required is to select the **Peripheral** tab to change **Local Name** to use for BLE Advertising in the RYZ012. Set **Complete local name** to “**RYZ012**”.



Module: RYZ012
Project: mcu_fwupdate_ryz012_ra4m2_baremetal

- [✓] Advertising Data 11/31
  - [✓] Flags
  - [ ] Service Class UUIDs
  - [✓] Local Name
  - [ ] Tx Power Level
  - [ ] Slave Connection Interval Range
  - [ ] Service Solicitation UUIDs
  - [ ] Service Data
  - [ ] Public Target Address
  - [ ] Random Target Address
  - [ ] Appearance
  - [ ] Advertising Interval
  - [ ] Manufacturer Specific Data
- [✓] Scan Response Data 0/31
  - [ ] Service Class UUIDs
  - [ ] Local Name
  - [ ] Tx Power Level
  - [ ] Slave Connection Interval Range
  - [ ] Service Solicitation UUIDs
  - [ ] Service Data
  - [ ] Public Target Address
  - [ ] Random Target Address
  - [ ] Appearance
  - [ ] Advertising Interval
  - [ ] Manufacturer Specific Data
- [✓] Advertising Parameter

Local Name
○ Short local name
● Complete local name

RYZ012

Sets the name of the device.
Specify the local name so that it is the same as the device name for the characteristics of the GAP Service.
The type of name can be selected as either of the following two.
• Short local name: Shows an abbreviated name of the device.
• Complete local name: Shows the complete name of the device.

Figure 40. Setting Complete Local Name

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

After the BLE Profile is created and customized 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.**

5.9 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.

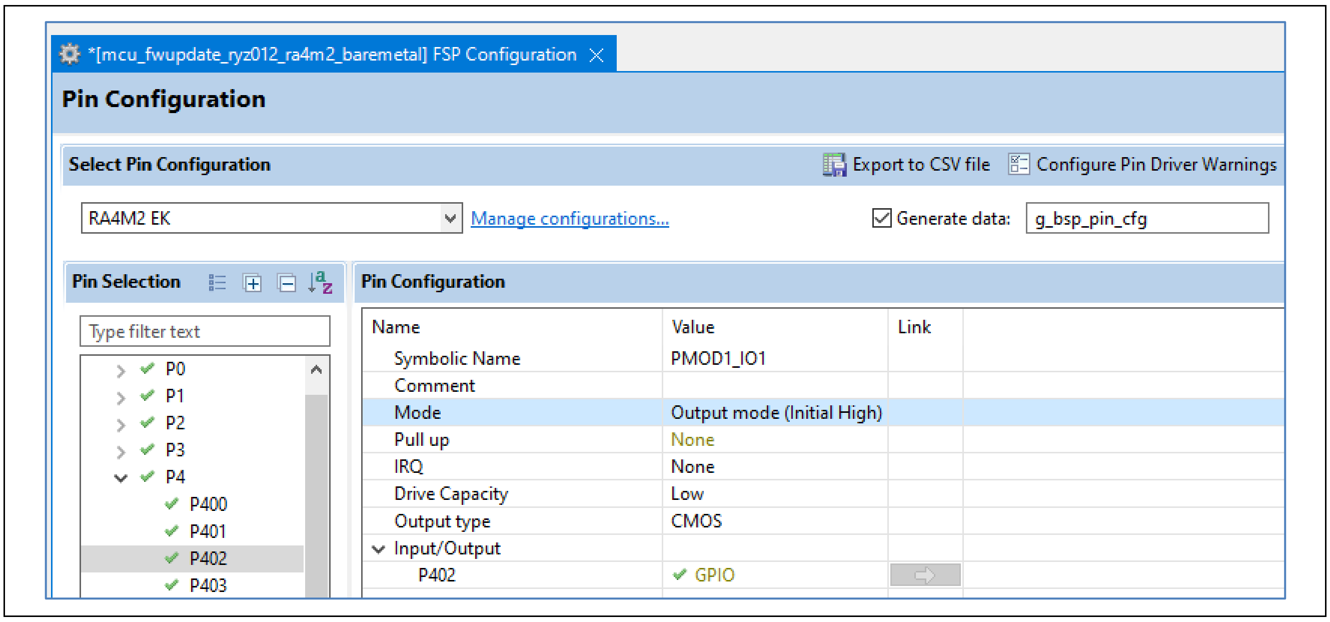
6. Next Steps

Now that you have a basic understanding of the process to update the RYZ012 firmware with the EK-RA4M2 MCU 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 34, Pin Configuration > SCI, set the Operation Mode to **Asynchronous UART**.
   As a reference, see Figure 35, 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).**



*[mcu_fwupdate_ryz012_ra4m2_baremetal] FSP Configuration

Pin Configuration

Select Pin Configuration: RA4M2 EK — Manage configurations... — Export to CSV file — Configure Pin Driver Warnings — ☑ Generate data: g_bsp_pin_cfg

Pin Selection: P0, P1, P2, P3, P4 (P400, P401, P402, P403)

Name	Value	Link
Symbolic Name	PMOD1_IO1	
Comment		
Mode	Output mode (Initial High)	
Pull up	None	
IRQ	None	
Drive Capacity	Low	
Output type	CMOS	
Input/Output		
P402	✓ GPIO	⇨

Figure 41. Pin Configuration > Mode

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 5.7 Add General PWM Timers for LED Indications 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 e^2 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 (renesas.com)
3. Renesas FSP v3.8.0 and later must be used to support the RYZ012 (FW SPP SDK v5.4 and later) for MCU based firmware update.

The XMODEM file transfer depends on the MCU code flash size available. The MCU code flash size requirement can change as future RYZ012 firmware versions are released.

The MCU Flash size must be large enough to hold the entire RYZ012A firmware binary image file. For the EK-RA4M2, in this sample project (See file `src/common_init.h`), the code flash used for the RYZ012 firmware image starts at

```
#define RYZ012_FWIMAGE_START_ADDRESS 0x00060000 which is above the user application
space. Make sure you adjust the RYZ012_FWIMAGE_START_ADDRESS and
RYZ012_FWIMAGE_END_ADDRESS address for your custom application.
```

Currently, the largest RYZ012A firmware image used in this sample application is **102436 bytes** for **8258_moduleV5_5.bin**. The RYZ012A firmware image size can increase in future RYZ012 FW releases as new features are added. Be certain to increase the number of MCU Flash Blocks required to store the firmware image file as the RYZ012A image size increases. See file src/common_init.h `#define RYZ012_FWIMAGE_END_ADDRESS 0x00080000` and #define `RYZ012_FWIMAGE_NUM_BLOCKS.`

Make sure the #define `RYZ012_FWIMAGE_START_ADDRESS` resides in free space above the MCU application code flash used. Otherwise, you may erase MCU application code during the XMODEM file transfer flash erase step.

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	www.renesas.com/ra
RA Product Support Forum	www.renesas.com/ra/forum
RA Flexible Software Package	www.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	www.renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb.10.23	—	Initial release

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.

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/.