Renesas RA™ Family

Getting Started with Wi-Fi Modules on FSP

Introduction

This application note provides steps for adding support for a new UART-based Wi-Fi module and its associated software/firmware for operations running on RA MCUs. The driver for the new module is developed while referencing the existing Wi-Fi driver provided by FSP as a starting point. The application note also provides an overview of the FSP, its associated pack files, and creating the custom user pack files for a new module driver.

Additionally, this document details steps in developing/porting new Wi-Fi drivers to FSP by utilizing the existing Wi-Fi driver. Upon reading this document and following outlined procedures, you will be able to add support for a Wi-Fi module of your choice to your own design, configure it correctly for the target application, write code, and test the module using the included application project code as a reference and efficient starting point.

Required Resources

- e² studio version 2024-07
- Flexible Software Package (FSP) v5.5.0
- 7-Zip (64-bit Windows x64) version 24.06 or later.
- RA Flexible Software Package Documentation
- Sierra Wireless BX310x Development Board PN: BX3105 DEV KIT_6001182 (https://www.digikey.com/short/zv9m2w)
  - Firmware version 2.7.2 or later for the Bx310x module (https://source.sierrawireless.com/resources/airprime/software/bx310x-firmware/#sthash.VQRX8FUO.dpbs)
  - BX310x Development Board User Guide (https://source.sierrawireless.com/resources/airprime/hardware_specs_user_guides/bx310x-development-board-guide/#sthash.g50DSS2y.dpbs)
  - BX310x AT Command Reference (source.sierrawireless.com/resources/airprime/hardware_specs_user_guides/airprime_bx310x-at-command-reference/)
- EK-RA6M3 Kit Schematics (https://www.renesas.com/us/en/document/swr/ek-ra6m3v1-design-package?language=en)

Target Device

- Renesas RA MCUs (tested on EK-RA6M3 kit).

Contents

1. Introduction

This application note, along with the associated application project, is the starting point for developing/porting new UART-based Wi-Fi drivers to FSP using the existing UART-based FSP Wi-Fi driver. In general, this application note can also be used for custom driver development for FSP.

For more clarity on the Wi-Fi driver development, this application note also provides an overview of the following:

- FSP Wi-Fi driver architecture and its components
- Data and control path of the Wi-Fi driver
- Supported application-level APIs
- Interface to the secured and non-secured sockets for TCP/IP communication
- Module-specific driver APIs to interface the Wi-Fi module
- FreeRTOS-based stream buffer to handle data size of arbitrary lengths
- UART drivers to communicate with the Wi-Fi modules

On top of all required prerequisites to add/modify the new Wi-Fi module to the FSP, a high-level overview of the following topics is also provided:

- Brief introduction to FSP
- FSP pack files
- Organization of FSP pack files
- Creating the user pack files
- Overview of XML changes required for the new modules.

2. FSP Overview

The Renesas FSP is an enhanced software package designed to provide easy-to-use, scalable, high-quality software for embedded system designs using the Renesas RA family of Arm® microcontrollers. FSP provides a versatile way to build secure, connected IoT devices using production-ready drivers, RTOS, and other middleware stacks.

FSP includes HAL drivers and middleware stacks with RTOS integration to ease the implementation of complex modules like communication and security. It uses an open software ecosystem and provides flexibility in using bare-metal programming and RTOS-based applications.

Figure 1 shows the block diagram of the FSP architecture.

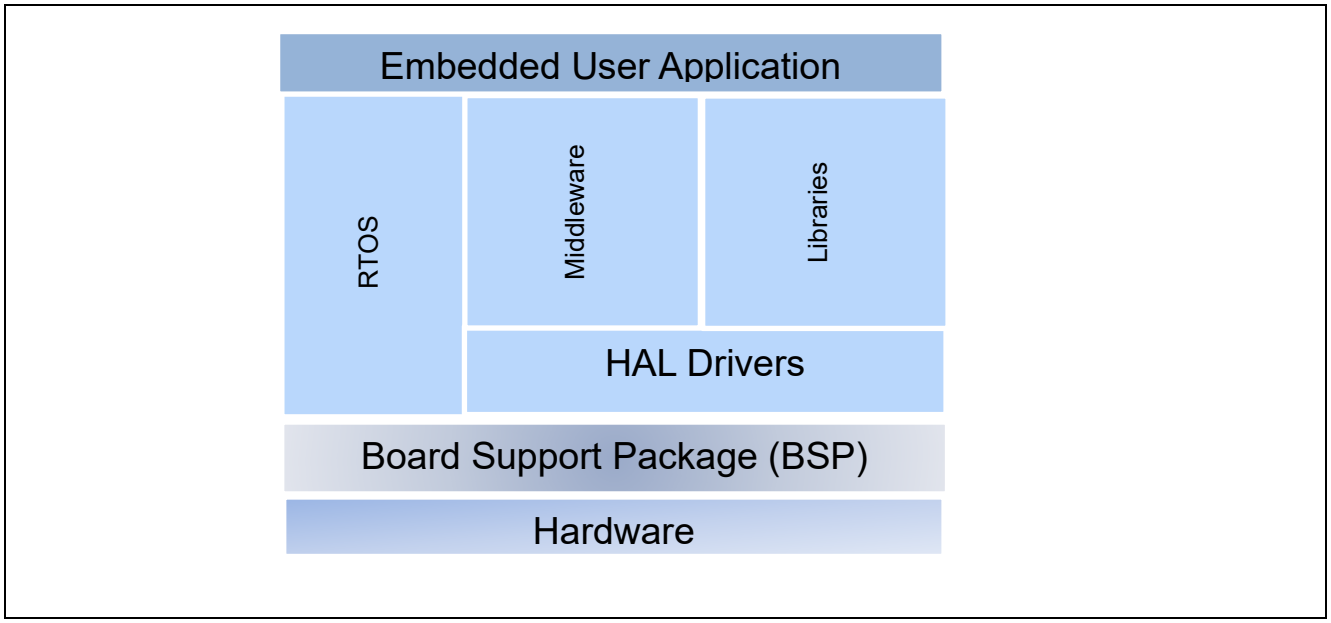

Figure 1. FSP Architecture

2.1 FSP Software Modules

As mentioned above, the FSP is a comprehensive piece of software covering all aspects of embedded systems software development. It includes the parts described in the following sections.

2.1.1 Board Support Package

The **Board Support Package (BSP)** is customized for every RA hardware kit and microcontroller. It includes the startup code for all supported blocks. As a developer using custom hardware, you can take advantage of the BSP, which can be tailored for end products and your board by using the Custom BSP Creator built into e^2 studio.

2.1.2 HAL Drivers

The RTOS-independent **HAL Drivers** provide efficient drivers for all peripherals and systems services. They eliminate the deep study of the underlying hardware in the microcontroller as they abstract the bit-settings and register addresses for you.

2.1.3 Libraries

The **Libraries** containing, for example, specialized software for digital signal processing or security and encryption-related functions also reduce development time and improve the stability of the end application. Even the libraries can be in the form of middleware. For instance, the emWin graphical package from SEGGER is available in FSP in the form of libraries.

2.1.4 Real-Time Operating System

The **RTOS** (Real-Time Operating System) provides a multitasking real-time kernel with preemptive scheduling and a small memory footprint. Amazon FreeRTOS is used as RTOS as part of the FSP.

2.1.5 Middleware

Middleware such as TCP/IP communication, file systems, graphical user interfaces, and USB are available in the FSP. Everything here is optimized and integrated into the FSP.

In summary, FSP is a collection of software modules in packs. The FSP user manual provides more details on the FSP and its components.

2.2 FSP Packs

FSP packs are the delivery mechanism for software components, device parameters, and BSP that comply with the CMSIS standard and can be used across RA Arm® Cortex®-M microcontroller devices.

When FSP is installed, a variety of pack files, also known as packs, are extracted. The FSP packs can be classified into different categories:

- Board support packs
- MCU packs
- Middleware packs
- Third-party or vendor packs
- Sample project packs

Note: For more information on the CMSIS packs, refer to section **9 References** in this document.

2.2.1 Overview of FSP Packs

The installed FSP packs are available under the folder `e2_studio\internal\projectgen\ra\packs`. This folder contains all the required and supported FSP packs to create an embedded application using RA MCUs.

Figure 2 shows a screenshot of different types of pack files as part of the installation. You can see that the pack files start with the name of the vendor such as Amazon, Arm®, and SEGGER. All the Renesas pack files start with the vendor's name such as Renesas. The file name also contains the features and a version associated with the pack.

Name	Date modified	Type	Size
Amazon.CellularInterface.1.3.0+fsp.5.0.0.pack	9/18/2024 1:44 PM	uVision Software Pack	91 KB
Amazon.CellularInterface.1.3.0+fsp.5.3.0.pack	9/18/2024 1:42 PM	uVision Software Pack	91 KB
Amazon.CellularInterface.1.3.0+fsp.5.4.0.pack	9/18/2024 1:42 PM	uVision Software Pack	91 KB
Amazon.CellularInterface.1.3.0+fsp.5.5.0.pack	9/18/2024 1:39 PM	uVision Software Pack	91 KB
Amazon.coreHTTP.3.0.0+renesas.0.fsp.5.0.0.pack	9/18/2024 1:44 PM	uVision Software Pack	78 KB
Amazon.coreHTTP.3.0.0+renesas.0.fsp.5.3.0.pack	9/18/2024 1:42 PM	uVision Software Pack	78 KB
Amazon.coreHTTP.3.0.0+renesas.0.fsp.5.4.0.pack	9/18/2024 1:42 PM	uVision Software Pack	78 KB
Amazon.coreHTTP.3.0.0+renesas.0.fsp.5.5.0.pack	9/18/2024 1:39 PM	uVision Software Pack	78 KB
Amazon.coreJSON.3.2.0+fsp.5.0.0.pack	9/18/2024 1:44 PM	uVision Software Pack	15 KB
Amazon.coreJSON.3.2.0+fsp.5.3.0.pack	9/18/2024 1:42 PM	uVision Software Pack	15 KB
Amazon.coreJSON.3.2.0+fsp.5.4.0.pack	9/18/2024 1:42 PM	uVision Software Pack	15 KB
Amazon.coreJSON.3.2.0+fsp.5.5.0.pack	9/18/2024 1:39 PM	uVision Software Pack	15 KB
Amazon.coreMQTT.2.1.1+fsp.5.0.0.pack	9/18/2024 1:44 PM	uVision Software Pack	78 KB
Amazon.coreMQTT.2.1.1+fsp.5.3.0.pack	9/18/2024 1:42 PM	uVision Software Pack	78 KB
Amazon.coreMQTT.2.1.1+fsp.5.4.0.pack	9/18/2024 1:42 PM	uVision Software Pack	78 KB
Amazon.coreMQTT.2.1.1+fsp.5.5.0.pack	9/18/2024 1:39 PM	uVision Software Pack	78 KB
Amazon.corePKCS11.3.5.0+fsp.5.0.0.pack	9/18/2024 1:44 PM	uVision Software Pack	81 KB
Amazon.corePKCS11.3.5.0+fsp.5.3.0.pack	9/18/2024 1:42 PM	uVision Software Pack	80 KB
Amazon.corePKCS11.3.5.0+fsp.5.4.0.pack	9/18/2024 1:42 PM	uVision Software Pack	80 KB
Amazon.corePKCS11.3.5.0+fsp.5.5.0.pack	9/18/2024 1:39 PM	uVision Software Pack	80 KB

Figure 2. FSP Pack Files

The contents of the pack files typically contain the package description (`.pdsc`) file at the root, which is an XML-based file describing the content of a software pack. The pack also contains a software component under the subfolder, which includes:

- Source code, header files, and software libraries
- Documentation and source code templates
- Device parameters, along with startup code and programming algorithms
- Sample project code

Users can unzip this pack file to see the contents. The contents of the Renesas.RA pack file and its folder/sub-folder structure is listed in the **Figure 3**.

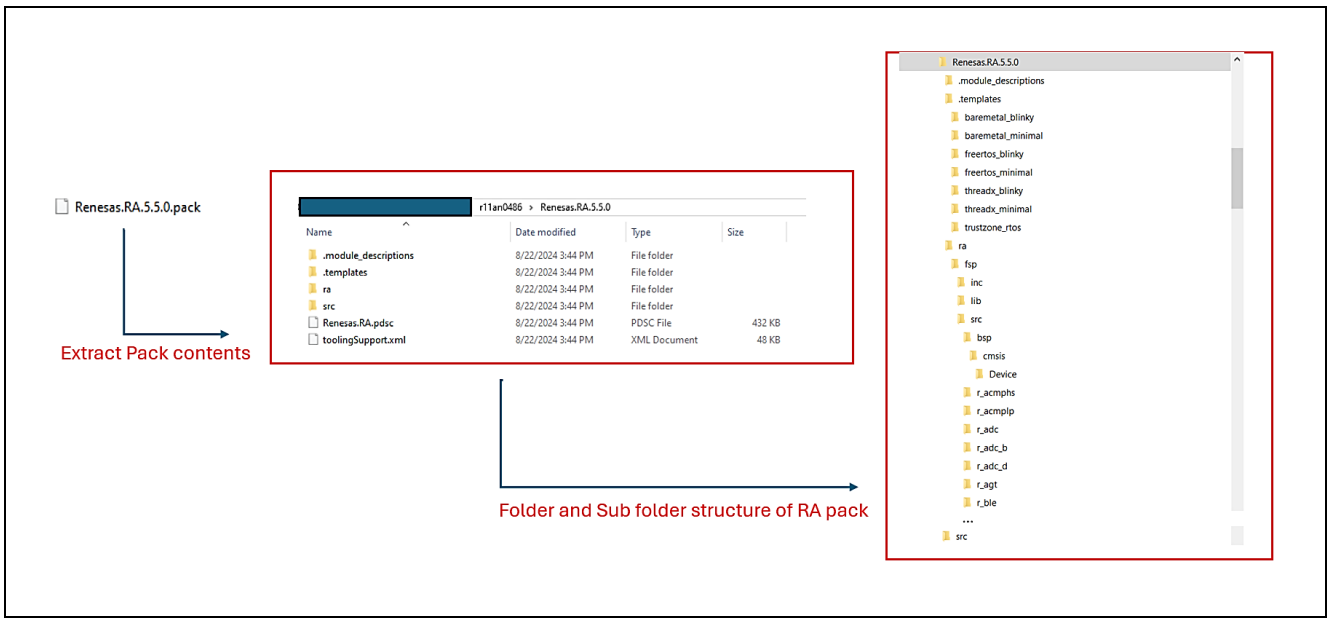

Figure 3. FSP Pack Contents

2.2.2 User-Creatable FSP Packs

Users can create packs to support user-defined modules in addition to those available from FSP. For example, if a company wishes to create a custom board representing its microcontroller-based product, a BSP can be created, verified, and distributed to application developers to speed up development. In the case of a custom communication module, if support is not available in FSP, a separate pack file is required to work with the new module. These pack files play an important role in updating the code between different releases and across the projects. In this application note, we will be creating user packs for a Wi-Fi module in **section 4.8**. The driver for the new Wi-Fi module is distributed as a user pack.

Note: The driver pack is not part of the FSP Pack distribution. But it is bundled as part of the application project.

2.2.3 User Pack Creation Tools

Pack creation can be done in different ways. It can be created through the integrated pack creation utility with e^2 studio, or using pack creation scripts, or by manually modifying the existing FSP packs for the new module. In this application note we will be using the manual user pack creation method to showcase the creation of the user pack for the newly added Wi-Fi module.

3. FSP Wi-Fi Driver Module Architecture

Modules are the core building blocks of FSP. Modules can do many different things, but all modules share the basic concept of providing functionality upwards and requiring functionality from below.

Modules can be layered on top of one another, building an FSP stack. The stacking process is performed by matching what one module provides with what another module requires. The Wi-Fi module, for example, requires a physical layer communication interface for data transfer, which can be provided by the UART driver module. It provides functionality to the secure sockets through the API created by the Wi-Fi driver to the secure sockets at the upper layer.

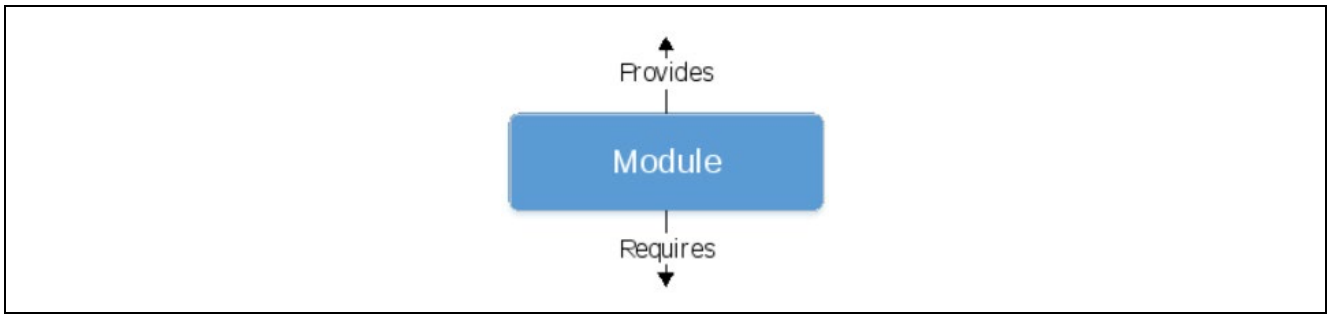

Figure 4. Provides Functionality to the Caller and Requires Functionality from the Lower Level

3.1 FSP Wi-Fi Driver Module Overview

Figure 5 shows the overview of the FSP Wi-Fi module architecture and its components.

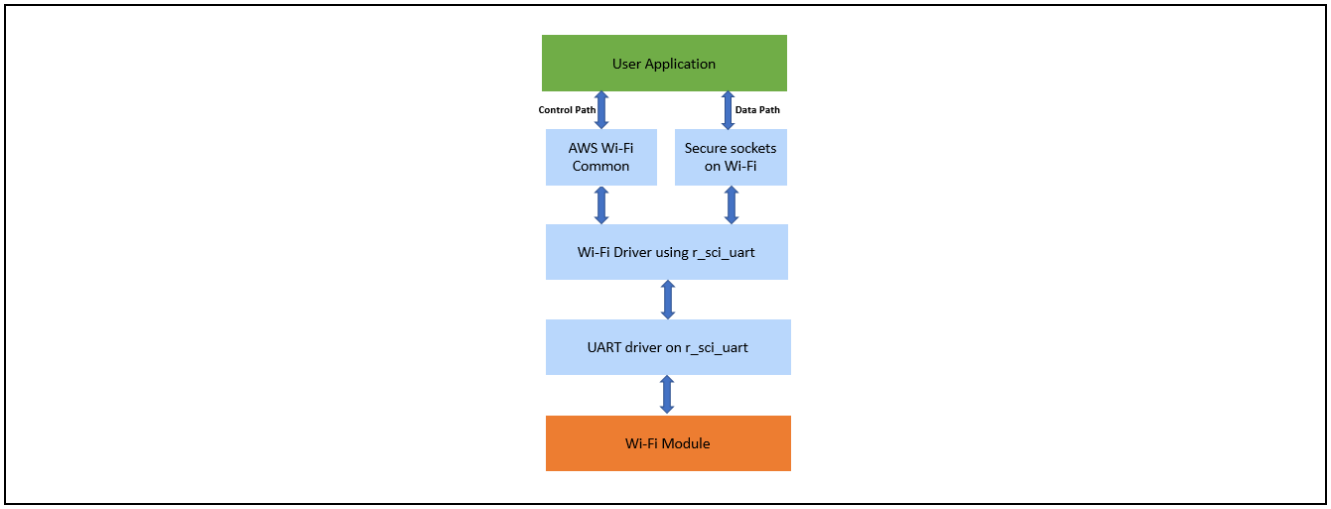

Figure 5. FSP Wi-Fi Driver Architecture

3.2 Silex Wi-Fi Module Directory Structure.

The FSP Wi-Fi driver for the Silex ULPGN Wi-Fi module is packaged as part of the FSP pack(`Renesas.RA.x.x.x.pack`). When the FSP-provided Wi-Fi drivers are used for the application development, the generated code is arranged in multiple folders. The details of the files and folder structure of the code are shown below in the **Figure 6**.

For the Silex ULPGN Wi-Fi module, the configuration-related header file is generated/stored under `ra_cfg\fsp_cfg\rm_wifi_onchip_silex_cfg.h`. This file's contents are generated based on the configurations from the configurator.

Header file `rm_wifi_onchip_silex.h` present inside the `ra\fsp\inc\instances` folder contains the data structure, driver function prototypes for the Silex ULPGN module.

The `rm_wifi_onchip_silex` and `rm_aws_sockets_wrapper_wifi_silex` folder under `ra\fsp\src` contains the secure socket interface, application level interface, FreeRTOS-specific configuration and driver-level Interface code for the module.

It is a good practice to maintain the code structure in the same format when porting or modifying the Wi-Fi drivers for the new module.

Note: The new driver pack for the Sierra Wireless BX310x module is bundled in the same directory format.

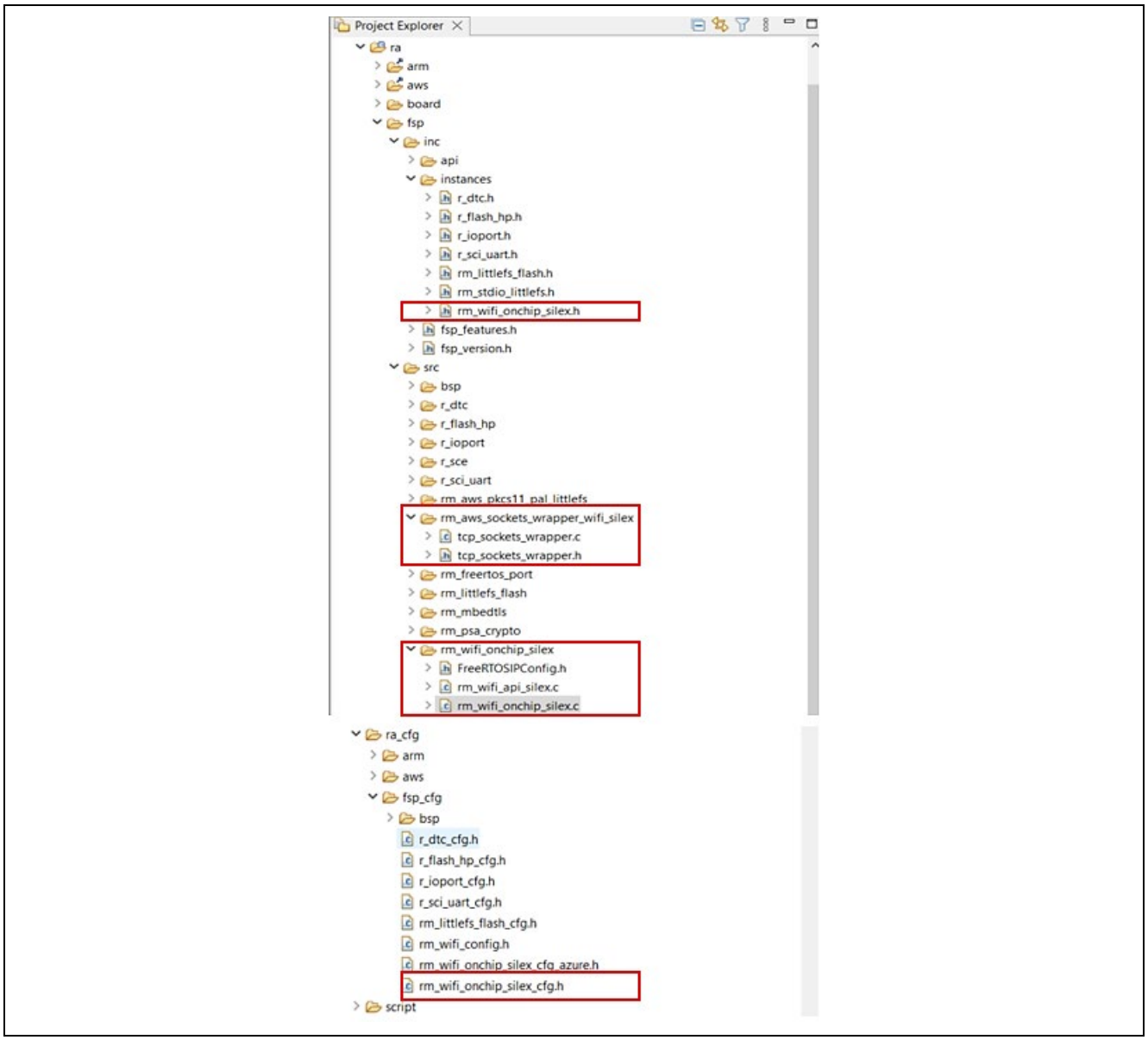

Figure 6. Wi-Fi Driver File/Folder Structure for Silex ULPGN Wi-Fi Module

3.3 Supported APIs for the Application from AWS Wi-Fi Library

Amazon AWS Wi-Fi libraries include the application-level APIs, which can be used for application development. These APIs use a glue logic interface to the Wi-Fi driver-level APIs. As a result, they can easily be ported across different Wi-Fi modules supported with FSP. The AWS-supported APIs found inside the file `rm_wifi_api_silex.c` are listed below:

- `WIFI_On`
- `WIFI_Off`
- `WIFI_ConnectAP`
- `WIFI_Disconnect`
- `WIFI_Reset`
- `WIFI_Scan`
- `WIFI_Ping`
- `WIFI_GetIPInfo`
- `WIFI_GetMAC`
- `WIFI_GetHostIP`
- `WIFI_IsConnected`

3.4 AWS Sockets

The AWS Secure Sockets library provides a socket interface to the embedded applications to communicate securely. The sockets library is based on the Berkeley sockets interface with additional secure communication options provided by mbedTLS.

FSP provides a simple interface for socket connection. The socket connection can be secure or non-secure. The selection is user-programable during the project creation. When developing Wi-Fi applications, these socket APIs are made available to the user from the FSP. The following APIs are available as part of the file `tcp_sockets_wrapper.c`:

- `TCP_Sockets_Connect`
- `TCP_Sockets_Disconnect`
- `TCP_Sockets_Send`
- `TCP_Sockets_Recv`

Note: Usage restriction:

- Only TCP sockets are supported by the FreeRTOS Secure Sockets library. UDP sockets are not supported.
- Only client APIs are supported by the FreeRTOS Secure Sockets library. Server APIs, including Bind, Accept, and Listen, are not supported.

3.5 Wi-Fi Driver API

Driver-level APIs are the entry point for accessing the module through the AT commands. Depending on the module and its supported features, the AT commands are grouped under the driver API. Individual or group AT commands are used under an API to interact with the module. The driver-level APIs use these AT commands to interact with the Wi-Fi module for Wi-Fi configurations, network configurations, and even for sending and receiving data. Before making changes to these APIs, individual AT commands need to be validated to confirm the working behavior as documented. The command and response for some of the modules vary depending on the Wi-Fi chip vendor. If a different chipset from the same vendor is used, most of the AT commands may be reused, supporting the operation without making major changes to the driver-level APIs.

3.5.1 Supported Driver-Level APIs for the Wi-Fi Module

The FSP Wi-Fi driver module supports the following APIs for the Silex ULPGN module. For supporting operation of a new Wi-Fi module, similar APIs need to be developed and tested.

Note: Some of the APIs may not be applicable to the new module, or in some cases, a newer set of APIs are needed to support the new feature present in the module. It is up to the user to add/delete the APIs as required. However, from the driver development perspective, the APIs provided by FSP can be used as reference for the new Wi-Fi module.

- rm_wifi_onchip_silex_open
- rm_wifi_onchip_silex_close
- rm_wifi_onchip_silex_disconnect
- rm_wifi_onchip_silex_socket_connected
- rm_wifi_onchip_silex_network_info_get
- rm_wifi_onchip_silex_connect
- rm_wifi_onchip_silex_mac_addr_get
- rm_wifi_onchip_silex_scan
- rm_wifi_onchip_silex_ping
- rm_wifi_onchip_silex_ip_addr_get
- rm_wifi_onchip_silex_avail_socket_get
- rm_wifi_onchip_silex_socket_status_get
- rm_wifi_onchip_silex_tcp_shutdown
- rm_wifi_onchip_silex_socket_create
- rm_wifi_onchip_silex_tcp_connect
- rm_wifi_onchip_silex_udp_connect
- rm_wifi_onchip_silex_send
- rm_wifi_onchip_silex_recv
- rm_wifi_onchip_silex_socket_disconnect
- rm_wifi_onchip_silex_dns_query

3.6 Stream Buffer

Stream buffers are RTOS objects for inter-task communication and are available from FreeRTOS. They are optimized for single reader, single writer scenarios, such as passing data from an interrupt service routine to a task.

The stream buffer implementation uses direct-to-task notifications. Therefore, calling a stream buffer API function that places the calling task into the blocked state can change the calling task's notification state and value.

In the FSP Wi-Fi driver implementation, stream buffers allow a stream of bytes to be passed from an interrupt service routine to a task. A byte stream can be of arbitrary length and does not necessarily have a beginning or end. Any number of bytes can be written at once, and any number of bytes can be read at once. Data is passed by copy – the data is copied into the buffer by the sender and out of the buffer by the read operation.

3.7 UART Drivers

The UART driver module provides a simple communication interface using the standard UART protocol between the MCU and the Wi-Fi module. The UART module on the MCU side uses the SCI module to communicate with the SCI peripherals and data-transfer (DTC) peripherals on an RA MCU.

The UART HAL module manages data flow using the standard UART protocol. Flow control support is also available for synchronization. DTC support can be optionally added to the module during the configuration so that DTC takes care of the data transfers without using many MCU cycles.

Note: In the Sierra Wi-Fi module implementation, asynchronous UART communication is used without flow control. For achieving higher data throughput, flow control can be used.

4. Adding Support for a New Wi-Fi Module

This section covers considerations to be made when identifying a suitable Wi-Fi module and modifying the existing FSP Wi-Fi drivers to support the new module. This includes creating and modifying the Wi-Fi driver, socket-level API, application-level API, and the user pack contents for the new driver pack.

4.1 Identifying the New Wi-Fi Module

When adding a new Wi-Fi module to FSP by leveraging existing Wi-Fi driver implementation, first identify the new Wi-Fi module that uses the on-module TCP/IP stack, supports AT commands, and supports the UART interface to communicate with the module. On top of the basic minimum criteria, the module also needs to support the TCP/UDP socket interface, DHCP Client, and DNS client. In addition to the bare minimum features, it is up to the users to choose other features supported by the module as value additional features.

Note: This application note only shows how to add/modify the driver for the new Wi-Fi module, which has a UART interface, AT command support, and on-module TCP/IP support. This does not mean that the other modules that have the SPI/I2C interface cannot be added/supported in FSP. With the addition of interface drivers, changes to the control path and data path, and the addition of glue logic to the TCP/IP stack present on the MCU, different Wi-Fi modules can be added.

Note: Module manufacturers usually categorize modules by specific parameters such as IEEE 802.11 a/b/g/n, transmit power, data rate, RF compliance, secure Wi-Fi, and so forth. These play an important role in identifying the Wi-Fi module. While this application note does not include details on this topic, it is advisable to choose the right module for your application with good technical support from module vendors.

4.2 Identify the Driver-Related Changes to New Module

After identifying the module that can be supported with FSP, list the features that are common between the new module and the FSP-supported Silex ULPGN Wi-Fi module. Also, list the new set of features that need to be included.

List what features you need to support in your driver implementation. For example, some Wi-Fi modules support both Access Point mode and Station mode. Choose modes your driver needs to support, such as both Access Point mode and Station mode or just Station mode. You can select the appropriate AT commands for the driver modification or feature addition.

For basic Wi-Fi connectivity, Wi-Fi and network-specific basic AT commands for the new Sierra module are listed in **Table 1** and **Table 2**. These AT commands are used in the new driver.

Table 1. Wi-Fi Specific AT Commands

AT Commands	Description
+SRWAPCFG	Configure local device's Wi-Fi access point
+SRWSTACFG	Configure/display Wi-Fi station connection information
+SRWSTACON	Connect/disconnect to Wi-Fi access point
+SRWSTANETCFG	Configure/ local device Wi-Fi station interface network IP address
+SRWSTASCN	Scan for Wi-Fi access points

Table 2. Network-Specific AT Commands

AT Commands	Description
+KTCPCFG	TCP session (Connection) configuration
+KTCPCNX	Start TCP connection
+KTCPCLOSE	Close TCP connection
+SRWSTANETCFG	Configure/ local device Wi-Fi station interface network IP address
+KTCPSND	Send data through a TCP connection
+KPING	Ping an IP address

Above are some of the AT commands that must be added or modified in the driver APIs. Details of the modification is shown in the next section.

Note: The Silex ULPGN Wi-Fi module uses the Qualcomm QCA4010 System-on-Chip (SoC). If the identified module is also based on the Qualcomm SoC, the changes to the driver can be minimized by leveraging the existing driver. However, if the module identified is different, the AT commands and response data/strings may be different and must be handled differently in the driver APIs.

4.3 Modify the Driver APIs

Refer to the module datasheet and user manual for the Sierra Wi-Fi and Silex Wi-Fi modules. List the AT commands for the identified features on the new Wi-Fi modules and compare them to the existing FSP-supported module. Check that the basic API support for the new module is available in the existing Wi-Fi driver as part of FSP (reference **section 3.5**). If available, identify the API and change the AT command and response code/string for the API to support the new module. These AT commands can be found in the module's AT command reference manual of the new module.

If the equivalent APIs are not available in the FSP, create a new API to accommodate the new feature and its associated AT commands and responses.

Also, it is important to identify all the APIs that have the supporting equivalent AT commands required by the module. For instance, for some of the modules, the AT command support may be limited, or a feature may not be supported (for example, the stack in the Sierra Wireless module does not have support for an AT command to perform DNS lookup, which is provided by the Silex Wi-Fi and used frequently by any application using socket programming). Such limitations can be solved by using the workaround suggested by the module vendor.

Note: The existing FSP Wi-Fi driver module has string parsing routines. Leveraging the string parsing routines can be beneficial for easy porting for the new module.

Now let us look into the header file and source code-related changes for the new module.

For changing the driver APIs in the header file, open `rm_wifi_onchip_silex.h` under the `/ra/fsp/inc/instances` folder add/modify the macros, enums, and driver-specific data structures as applicable. Rename the function prototypes or any references made to the Silex module to Sierra specific as required.

For instance, some of the sample changes are shown as follows:

1. Change the enums
   `WIFI_ONCHIP_SILEX_CFG_MAX_NUMBER_UART_PORTS` can be changed to
   `WIFI_ONCHIP_SIERRA_CFG_MAX_NUMBER_UART_PORTS`
2. Change the data structures

```
/** User configuration structure, used in open function */
typedef struct st_wifi_onchip_cfg
{
    const uint32_t                          num_uarts;
    const uint32_t                          num_sockets;
    const bsp_io_port_pin_t                 reset_pin;
    const uart_instance_t                 * uart_instances[WIFI_ONCHIP_SILEX_CFG_MAX_NUMBER_UART_PORTS];
    const wifi_onchip_silex_sntp_enable_t sntp_enabled;
    const uint8_t                         * sntp_server_ip;
    const int32_t  sntp_timezone_offset_from_utc_hours;
    const uint32_t sntp_timezone_offset_from_utc_minutes;
    const wifi_onchip_silex_sntp_daylight_savings_enable_t sntp_timezone_use_daylight_savings;
    void const * p_context;
    void const * p_extend;
} wifi_onchip_silex_cfg_t;

/** User configuration structure, used in open function */
typedef struct st_wifi_onchip_cfg
{
    const uint32_t                          num_uarts;
    const uint32_t                          num_sockets;
    const bsp_io_port_pin_t                 reset_pin;
    const uart_instance_t                 * uart_instances[WIFI_ONCHIP_SIERRA_CFG_MAX_NUMBER_UART_PORTS];
    const wifi_onchip_sierra_sntp_enable_t sntp_enabled;
    const uint8_t                         * sntp_server_ip;
    const int32_t  sntp_timezone_offset_from_utc_hours;
    const uint32_t sntp_timezone_offset_from_utc_minutes;
    const wifi_onchip_sierra_sntp_daylight_savings_enable_t sntp_timezone_use_daylight_savings;
    void const * p_context;
    void const * p_extend;
} wifi_onchip_sierra_cfg_t;
```

Figure 7. Sample Enum and Data structure change for the Sierra Wi-Fi Module

3. Change the API function prototype name
   `fsp_err_t rm_wifi_onchip_silex_open(wifi_onchip_silex_cfg_t const * const p_cfg);`
   to
   `fsp_err_t rm_wifi_onchip_sierra_open(wifi_onchip_sierra_cfg_t const * const p_cfg);`

For the changes related to driver API inside the source file, open `rm_wifi_onchip_silex.c` under the `ra\fsp\src\rm_wifi_onchip_silex` folder, and add/modify the API carefully as applicable. Some of

the code may be removed as it is not required for the new module. In some cases, additional code can be added to support the new AT commands for the driver APIs.

A sample screenshot of the source code changes from the Silex module to the Sierra module is shown as follows. This is just for reference and as a user you need to change all the code as desired for your module.

```
if (FSP_SUCCESS == err)
{
    err = rm_wifi_onchip_silex_send_basic(p_instance_ctrl,
                                          p_instance_ctrl->curr_cmd_port,
                                          "ATS108=1\r",
                                          WIFI_ONCHIP_SILEX_TIMEOUT_100MS,
                                          WIFI_ONCHIP_SILEX_TIMEOUT_8SEC,
                                          WIFI_ONCHIP_SILEX_RETURN_OK);
}
```

```
if (FSP_SUCCESS == err)
{
    err = rm_wifi_onchip_sierra_send_basic(p_instance_ctrl,
                                           p_instance_ctrl->curr_cmd_port,
                                           "AT+SRWCFG=1,2\r",
                                           WIFI_ONCHIP_SIERRA_TIMEOUT_3MS,
                                           WIFI_ONCHIP_SIERRA_TIMEOUT_200MS,
                                           WIFI_ONCHIP_SIERRA_RETURN_OK);
}
```

Figure 8. Sample Driver Code Change for the Sierra Wi-Fi Module

Note: Many more changes are required in this file. The sample shown here is for reference only.

Note: Change the file name and the folder name for the new set of drivers. These are required to differentiate the Silex module driver from the Sierra module drivers. Also, maintain the same directory structure.

Note: After all the changes are done, make a backup of these files and folders. Also, make changes to the files and folders as read-only. This will help prevent the file from getting overwritten by FSP during the project generation.

4.4 Files and Folder Structure for the New Module

The screenshot in **Figure 9** shows the directory structure and the list of files modified as part of the new Wi-Fi module driver.

The screenshots are similar to the Silex module packs that are available as part of the FSP packs. When you create a pack for the new (Sierra wireless module) Wi-Fi driver, these will not be part of the FSP pack. The pack will be an independent user pack. This makes it easy to port across different FSP releases.

The file and folder structure are kept similar to the Silex module to ensure compatibility and to maintain the same structure for ease of use.

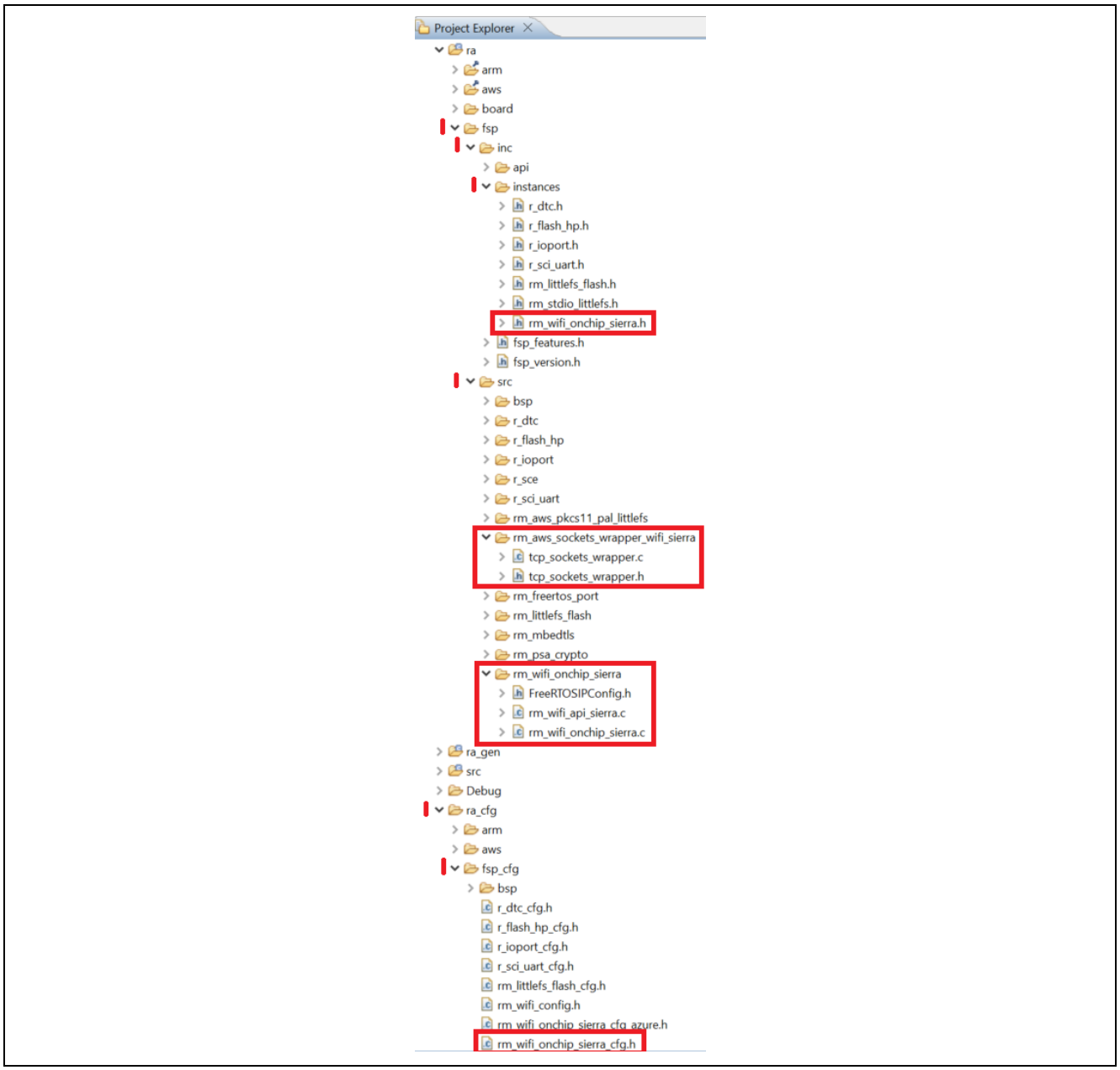

Figure 9. Wi-Fi Driver File/Folder Structure for Sierra Wi-Fi Module

4.5 Modifying the AWS API

To accommodate the application level Wi-Fi APIs calling the right driver-level APIs, you need to make changes in the `rm_wifi_api_silex.c` file under the folder `ra\fsp\inc\src\rm_wifi_onchip_silex` and call the driver API as applicable.

For instance, the `WIFI_On` API calls the Driver API `rm_wifi_onchip_silex_open` for the Silex module. This needs to be changed to `rm_wifi_onchip_sierra_open` to support the Sierra Wi-Fi module. Change it for all the applicable APIs as required.

The APIs to be changed are listed in the section **3.3**.

Note: Some of the Wi-Fi APIs may not have support at the driver-level interface. If the support in the module is present, you can take advantage of it by adding the new driver APIs and linking to the application-level APIs.

4.6 Modifying the AWS Sockets API

To accommodate the socket-level APIs calling the right driver-level APIs, you need to make changes in the `tcp_sockets_wrapper.c` file under the folder `ra\fsp\inc\src\rm_aws_sockets_wrapper_wifi_silex` and call the driver API as applicable.

For instance, `TCP_Sockets_Connect` API calls the driver API `rm_wifi_onchip_silex_tcp_connect` for the Silex module. This needs to be changed to `rm_wifi_onchip_sierra_tcp_connect` for the Sierra module. Such calls need to be changed for all the applicable APIs as required.

The APIs to be changed are listed in section **3.4**.

4.7 Modifying the XML Files for the New Wi-Fi Module

XML file changes under the `.module_descriptions` folder are required for the FSP configurator. Before you start making changes to this XML file, you need to understand its organization. The XML file contains the following listed tags; under <raModuleDescription>, you will see the <module> and its associated <config>.

Under the <module> tag, you will see the ID, display data, version, and URL details. It also has the tags for <requires> and <provides>.

Under the <config> tag, you will see the configuration; you will notice the property and contents for the FSP configuration. The module tag also contains the constraint, header, includes, and so forth.

This XML file needs to be copied and modified for the new module to prepare the pack file.

Note: For more details, refer to the XML file included in the pack files for the Silex module/Sierra module.

Part of the FSP RA Pack folder contains the XML files under the `.module_descriptions` folder. The XML file `Renesas##HAL Drivers##all##rm_wifi_onchip_silex####x.y.z.xml` is key for the configuration parameters for the Wi-Fi module configuration on the FSP configurator. Users are required to modify this XML file for the new Wi-Fi module-specific configurations.

Copy `Renesas##HAL Drivers##all##rm_wifi_onchip_silex####x.y.z.xml` from the RA packs folder to a temporary location. Change the name of the file. The resulting file will be `Renesas##HAL Drivers##all##rm_wifi_onchip_sierra####x.y.z.xml`.

Note: Make sure the version numbers match the FSP version. In this case, ‘x.y.z’ should be ‘5.5.0’.

Open the file and change the contents. Find and replace instances of ‘silex’ with ‘sierra’ and ‘SILEX’ with ‘SIERRA’.

Save the file and continue to the next steps of pack creation.

Note: Do the same with the other .xml files if necessary. For Sierra module, you also need to change from `Renesas##Middleware##all##rm_aws_sockets_wrapper_silex####x.y.z.xml` file to `Renesas##Middleware##all##rm_aws_sockets_wrapper_sierra####x.y.z.xml` file.

4.8 Pack Creation for the New Wi-Fi Module

Pack files can be created in two ways:

- Manual pack creation
- Using the e^2 studio RA Pack Creator utility

4.8.1 Manual User Pack Creation

To create the user pack manually, you need to validate the modified driver code on the new module. Once the code is validated and ready, it can be added as part of the user pack to make it available for configuration and application use using the FSP configurator. Creating the pack involves the code to be arranged with the proper folder structure format, as shown in the **Figure 10**.

Once the driver code changes are ready, new changes also need to be added to the XML file under **.module_description** folder (as explained in the **section 4.7**), the pack descriptor file (.pdsc) file for the description of the pack contents, and the tooling support file for tooling info. These are the pack-specific changes.

Note: Refer to the Silex Wi-Fi module/Sierra Wi-Fi module pack for detailed information on the contents of the pack file.

- Create a folder with the format `Vendor.VendorFeature.Version`. For the Sierra Wi-Fi module, we have `Renesas.RA_wifi_onchip_sierra.5.5.0` as the folder name. Place the modified `toolingSupport.xml` and `Reneas.RA_wifi_onchip_sierra.pdsc` file inside the root of the newly created folder (`Renesas.RA_wifi_onchip_sierra.5.5.0`).
- Create a folder with a name like `.module_descriptions`, and place the modified module description file under this folder. Here, in the case of the Sierra Wi-Fi module, `Renesas##HAL Drivers##all##rm_wifi_onchip_sierra####5.5.0.xml` and `Renesas##Middleware##all##rm_aws_sockets_wrapper_sierra####5.5.0.xml` are used as module description file.
- For source and header files, create a set of folders in the same hierarchy `ra\fsp\inc\instances` and `ra\fsp\src` under `Renesas.RA_wifi_onchip_sierra.5.5.0`, as shown in the snapshot, and place the modified header files and source files.

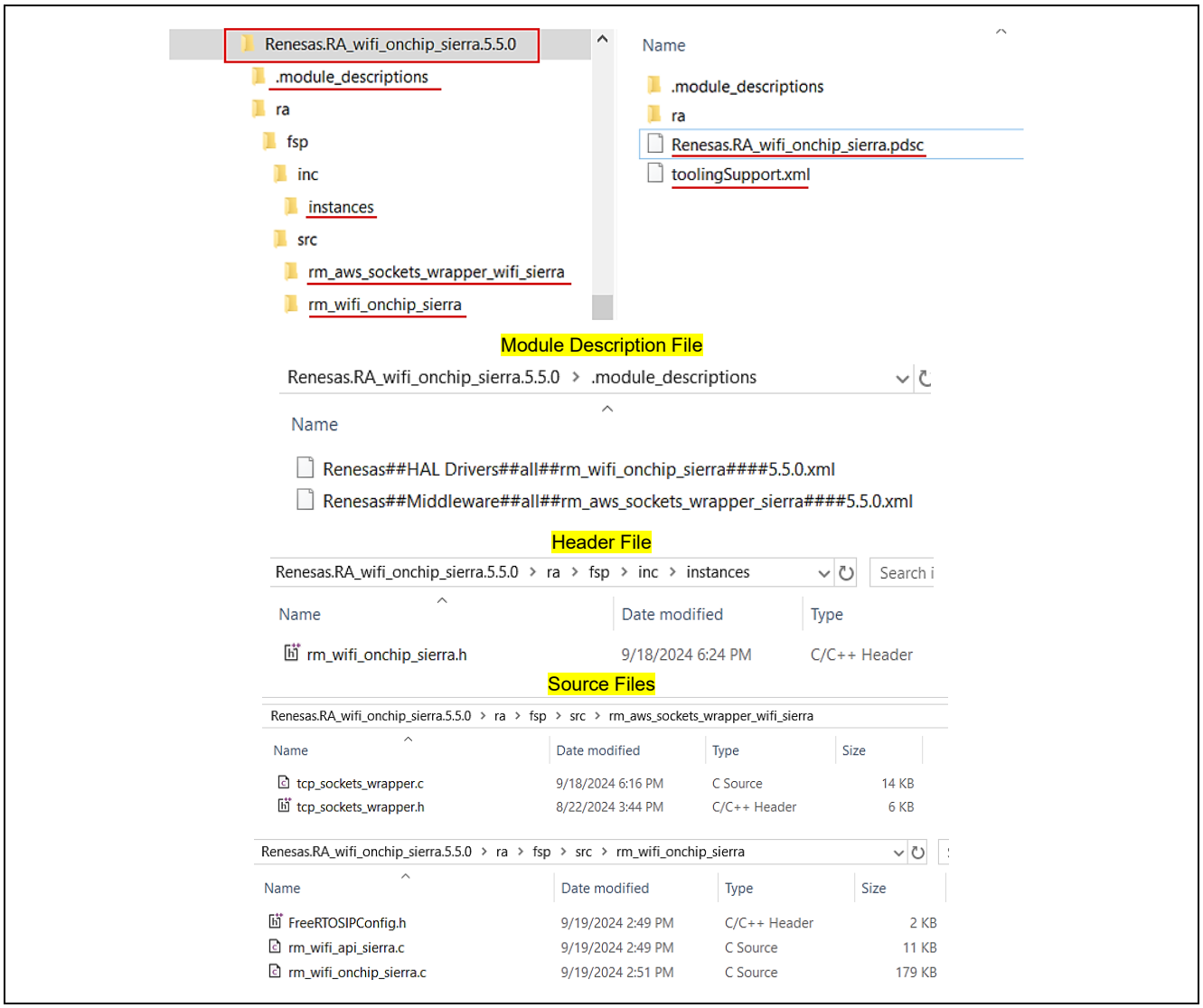

Figure 10. Folder Structure and File Information for the New Wi-Fi Driver Pack

- After all the changes are in place, you need to create a zip file using the 7-zip utility. The method for creating the `.zip` file and later renaming as `.pack` file is shown in **Figure 11**.

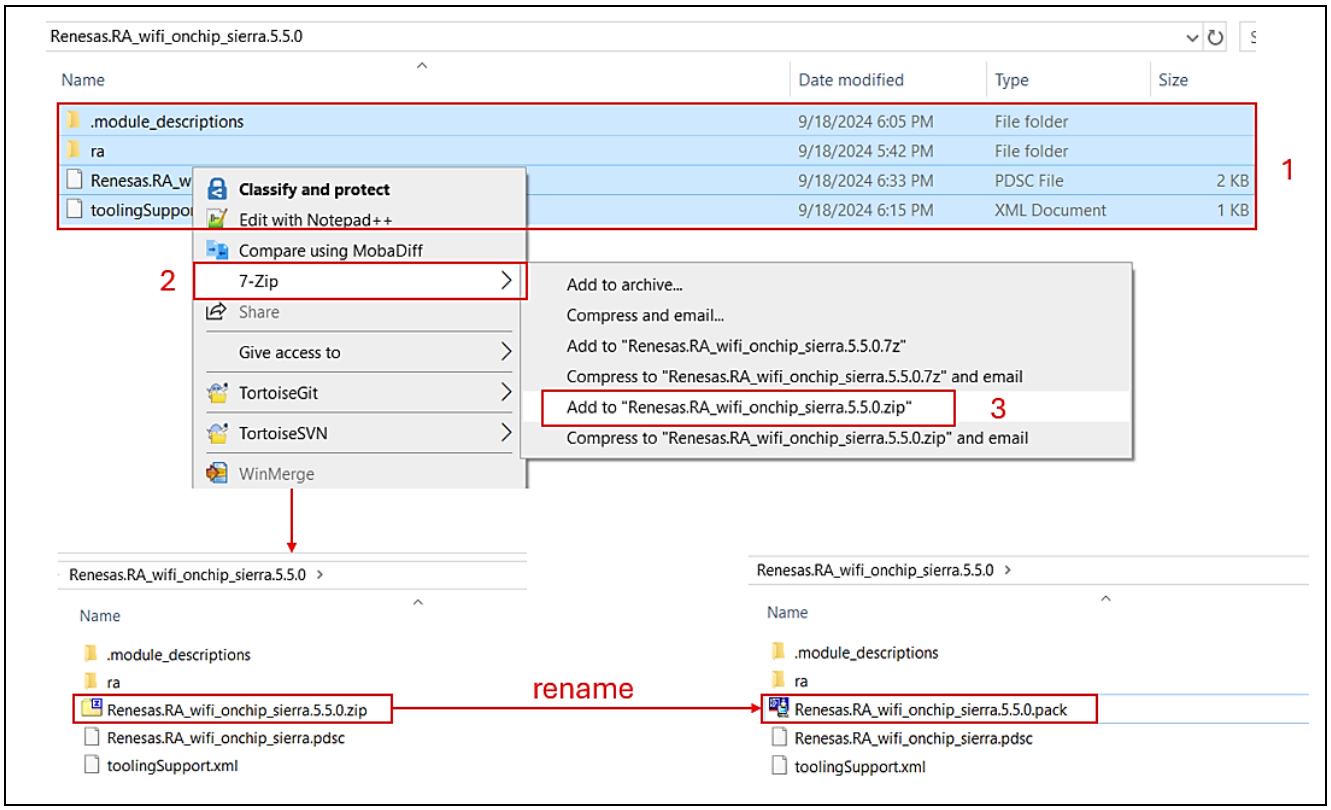

Figure 11. Pack Creation from the Zip File for the New Wi-Fi Driver Pack

4.8.2 Creating User Pack Using e² studio Utility

User pack creation using e² studio can be done for the new Wi-Fi driver via the utility tool. This application note will not be covering the pack creation using the e² studio utility tool. The steps for creating the user pack using the e² studio utility are available as part of the e² studio **Help** tab. More details on the usage steps can be found under the e² studio help section by searching for “Creating a RA CMSIS user pack”.

4.9 Importing New User Pack to the Project

Once the final user pack is built, it needs to be copied to the pack's folder. Installing the pack file can be done in two different ways:

- Manually copy the new user pack to the packs folder of the e² studio installation. The packs folder under the e² studio installation folder is `e2_studio\internal\projectgen\ra\packs`.
- Or use the import feature of e² studio to import the pack (**Project → Right-Click → Import → General → CMSIS Pack**).

After the packs are installed by manually copying or importing using the e² studio tool, you will notice the e² studio IDE detects the new packs and updates its database.

Note: If the pack files are detected by the e² studio IDE, close and restart the e² studio for the installed packs to take effect.

You can check whether the e² studio installed the pack file successfully by accessing through **e² studio → Help → About e² studio → Installation Details → Support folders → e² studio Support Area link**. This is available in the support area under the `internal\projectgen\ra\packs` folder. The screenshot for the same is shown in **Figure 12**.

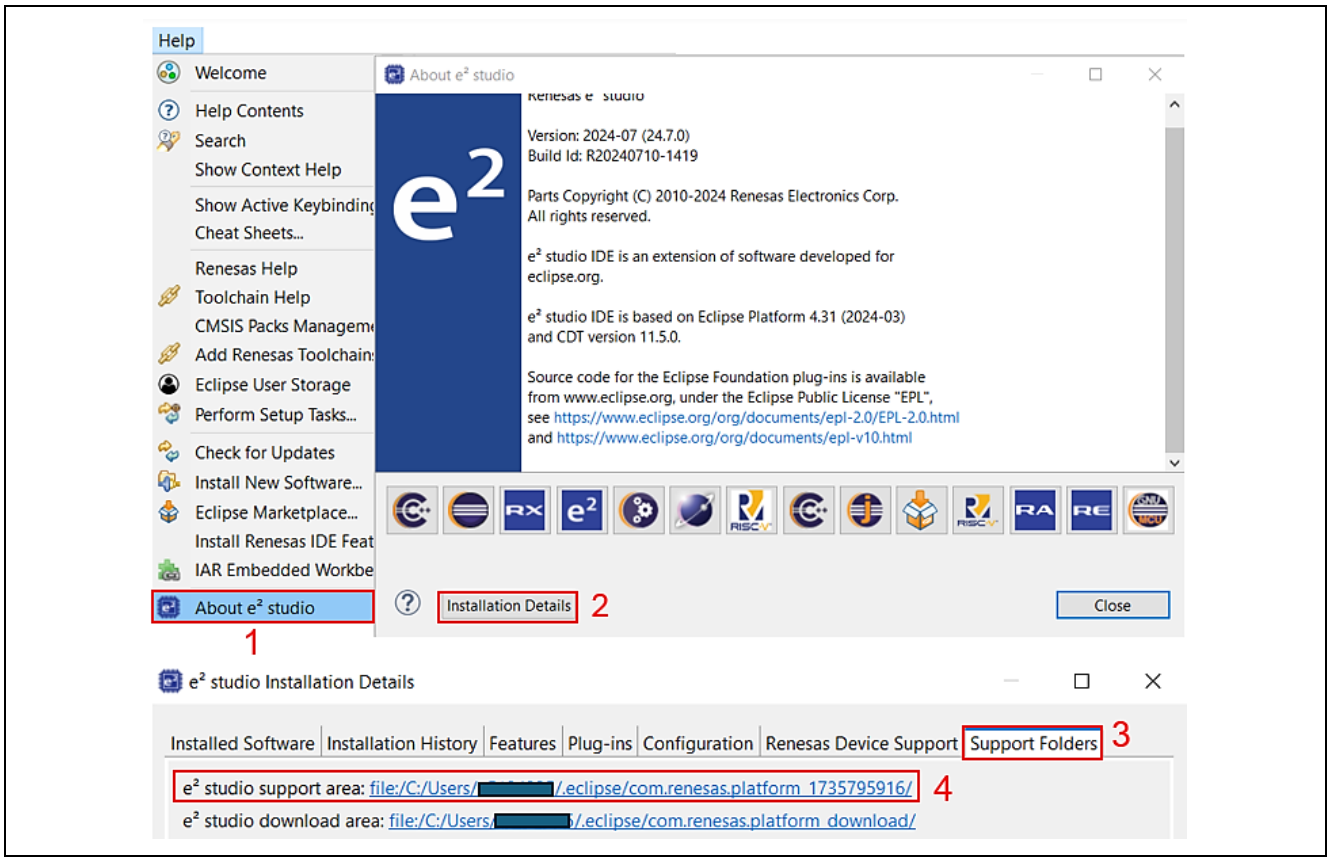

Figure 12. Location of Cached e² studio Packs

In the next section, we show how this Wi-Fi module can be included in the project using the FSP configurator, how to configure the new module using the property window, and how to change its configuration parameter values.

5. Building Application with New Wi-Fi Module

After the pack is installed successfully, the new application project can be created by adding the new Wi-Fi module and configuring the module via the FSP configurator.

5.1 FSP Configuration

FSP configuration for the project is done through the e² studio's graphically guided tool. Refer to the FSP User's Manual section on Configuring a Project, which has details on how the FSP configuration can be done for the individual modules and different configuration settings as needed.

5.2 Including the Module in the Project

To include the Wi-Fi Module in an RA Project, in the FSP configurator, under the created thread, choose the **Stacks** tab, **New Stack → Networking → AWS Core HTTP.** After the AWS Core HTTP stack is created**,** click to **Add Sockets Wrapper → New → AWS Sierra WiFi Sockets Wrapper (rm_aws_sockets_wrapper_sierra)**. This will add the Sierra Wi-Fi module to the thread. Adding the Wi-Fi module and Wi-Fi module to the project is shown in **Figure 13** and **Figure 14**, respectively.

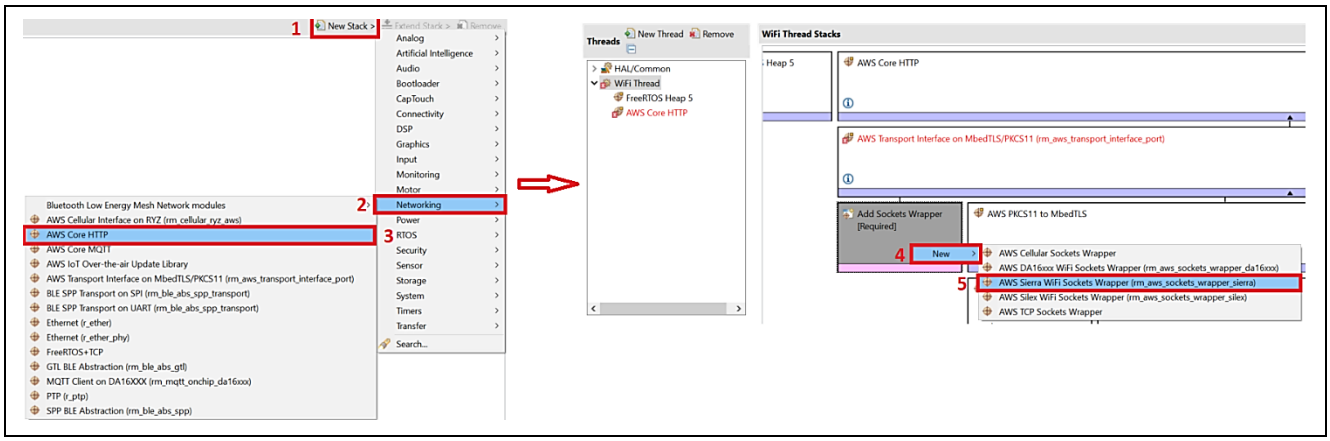

Figure 13. Adding Sierra Wi-Fi Module to the Project

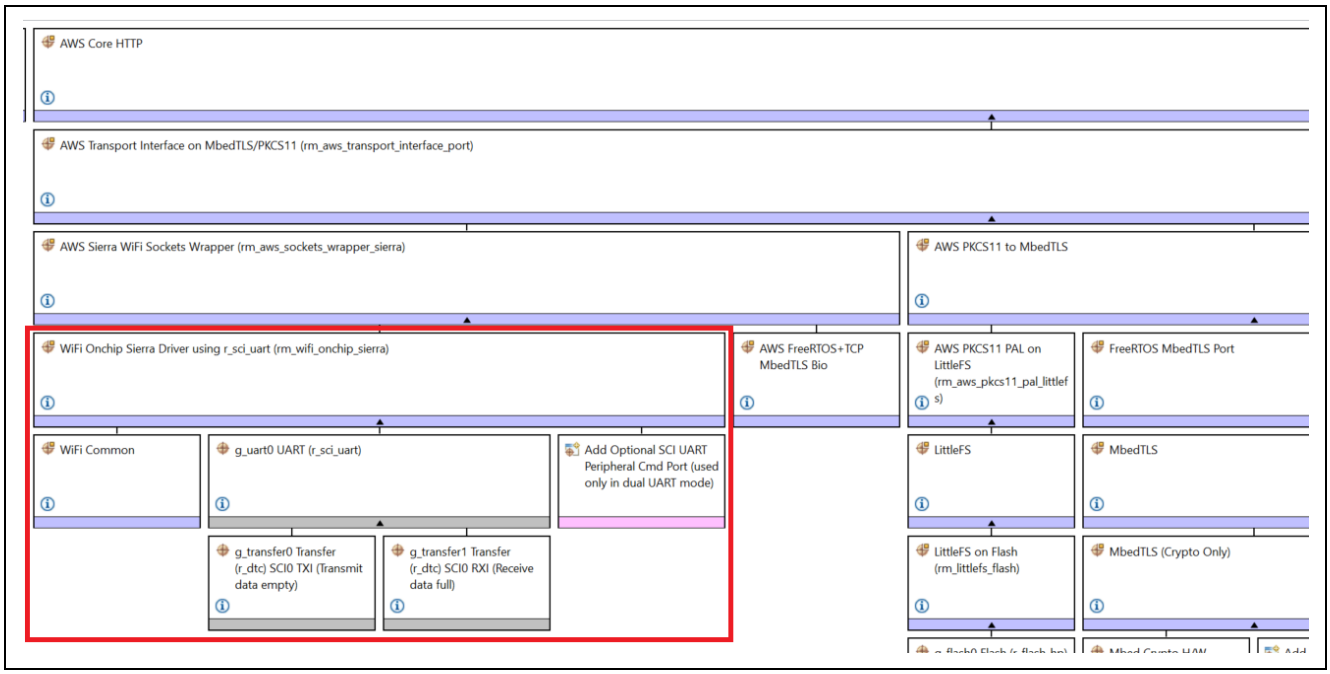

Figure 14. Sierra Wi-Fi Module Added to the Project

5.3 Module Configuration

The first step is adding the module to the project. The next step is making the module operate in the desired way through module-specific configuration. Each block of the module has an associated property window where the configuration values can be changed from the default values.

The details of the individual module configuration for each block are not explained here. The included project provides more details. For additional details, refer to the FSP user manual for the Wi-Fi module configuration.

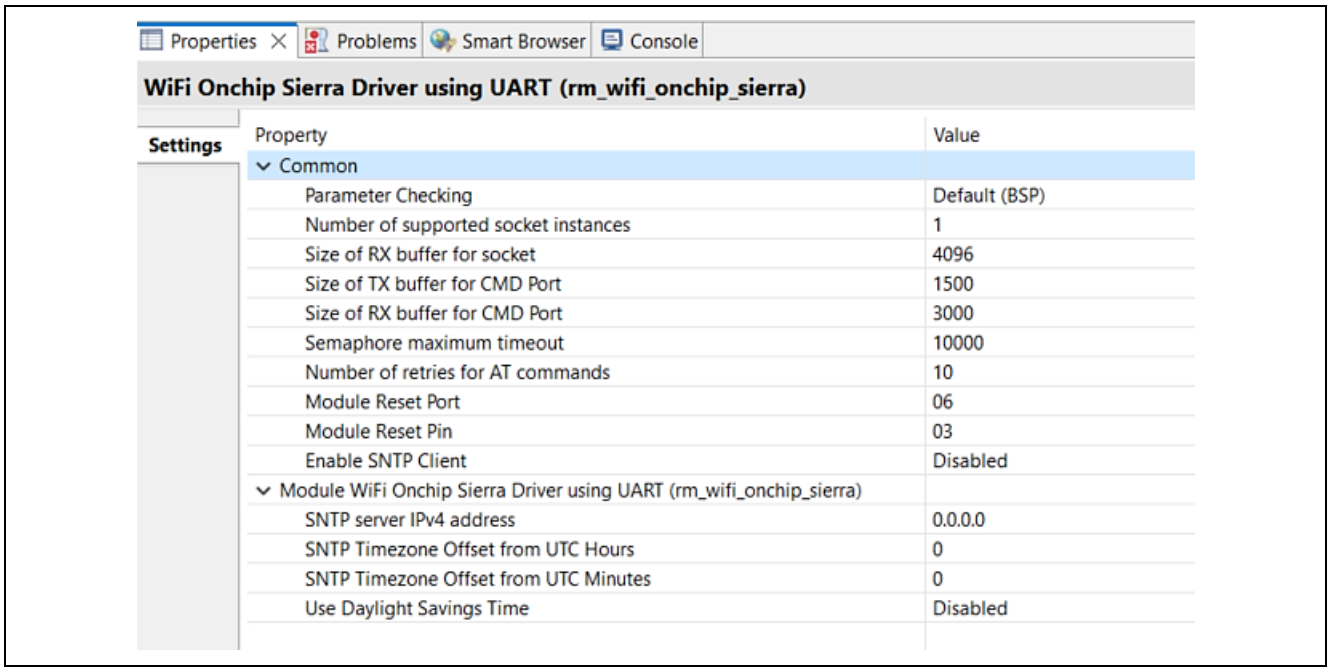

Figure 15. Properties Window for Modifying the Wi-Fi Module Configuration

6. Importing and Building the Project

Prior to importing the project bundled as part of the application note, make sure that the new pack you created is installed to the packs folder (for reference and quick running of the attached Application Project, pre-created "`Renesas.RA_wifi_onchip_sierra.5.5.0.pack`" file is attached as part of the bundle).

- Add the included user pack to the pack's folder `e2studio\internal\projectgen\ra\packs`
- To import the included project, follow the standard import steps documented as part of the *RA FSP User's manual,* Starting Development, e² studio User's Guide, Importing an Existing Project into e² studio.

After importing the project, make sure the FSP configurator shows the new module without any warnings or errors.

Open the `wifi_app.h` header file under the `src` folder, make the changes to SSID and password as applicable to your access point.

```
#define WIFI_SSID "Renesas"
#define WIFI_PW   "@Renesas123"
```

Generate the project content and build the project. The build should be error-free and warning-free with reference to the application code.

This completes the successful importing and building of the module.

7. Running the Application

Before running the application on EK-RA6M3, it is necessary to make the connections shown in this section.

7.1 Board Setups

Before running the project, make sure the Sierra Wireless BX310x Development Board is connected to the PMOD1 of the EK-RA6M3 board using the connection diagram as shown in **Table 5**.

Note: Sierra Wireless BX310x Development Board has a jumper setting to select the UART or the USB (connecting to PC). Make sure the UART setting is selected.

The BX310x development board provides two sources for accessing the BX310x modules.

Table 3. 4-wire UART Interface

USB accessed via USB (UART FTDI IC)	Can be used for debugging and testing in the initial stages of validation of the module
UART header	Accessed via UART header – this is the interface used with RA6M3

A 3-pin header is used to select the UART interface.

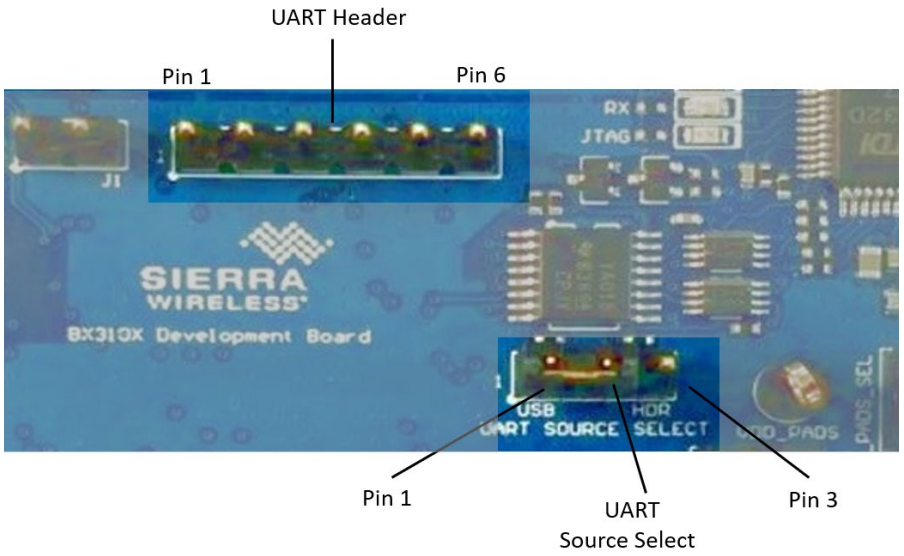

Figure 16. UART Header and UART Interface Selection

Table 4. UART Details of the Sierra Wireless Module

Component	PCB Label	Description
3-pin header	UART SOURCE SELECT	Selects the hardware source used to access the BX310x module's UART interface, based on jumper position: • Pins 1 and 2—USB via FTDI USB, UART IC • Pins 2 and 3—UART header
UART Header Pins		
6-pin UART Header Pins		
1	None	Ground
2	None	UART Clear to Send
3	None	No connect
4	None	UART Transmit Data
5	None	UART Receive Data
6	None	UART Ready to Send

Note: UART signals are named from the host perspective, with the module acting as a slave device (for example, UART_HD_TXD is Host Transmit/Module Receive)

Table 5. UART Connection Diagram from BX310x Development Board to RA6M3 PMOD1

UART Header Pin	PMOD1 Pin
1- (Ground)	5- (Ground)
2- (CTS)	
3- (No Connection)	
4- Transmit	2- Receive
5- Receive	3- Transmit
6- RTS	

To power the RA6M3 and Sierra Wireless BX310x Development Board, connect the USB cable to the micro-USB connector of the EK-RA6M3 kit (J10) and Sierra Wireless BX310x Development Board (micro-B USB connector).

7.2 User Interface

Once the application is running, open the RTT viewer to see the banner message and initialization sequence log messages at each step. If the Wi-Fi module is connected to the access point, the DHCP client gets the IP address, which is displayed on the console. The user can ping from the PC to the new IP address and validate the connectivity.

Also, upon a successful connection, you can notice the blinking green and blue LEDs lighting up, indicating the connection to the access point.

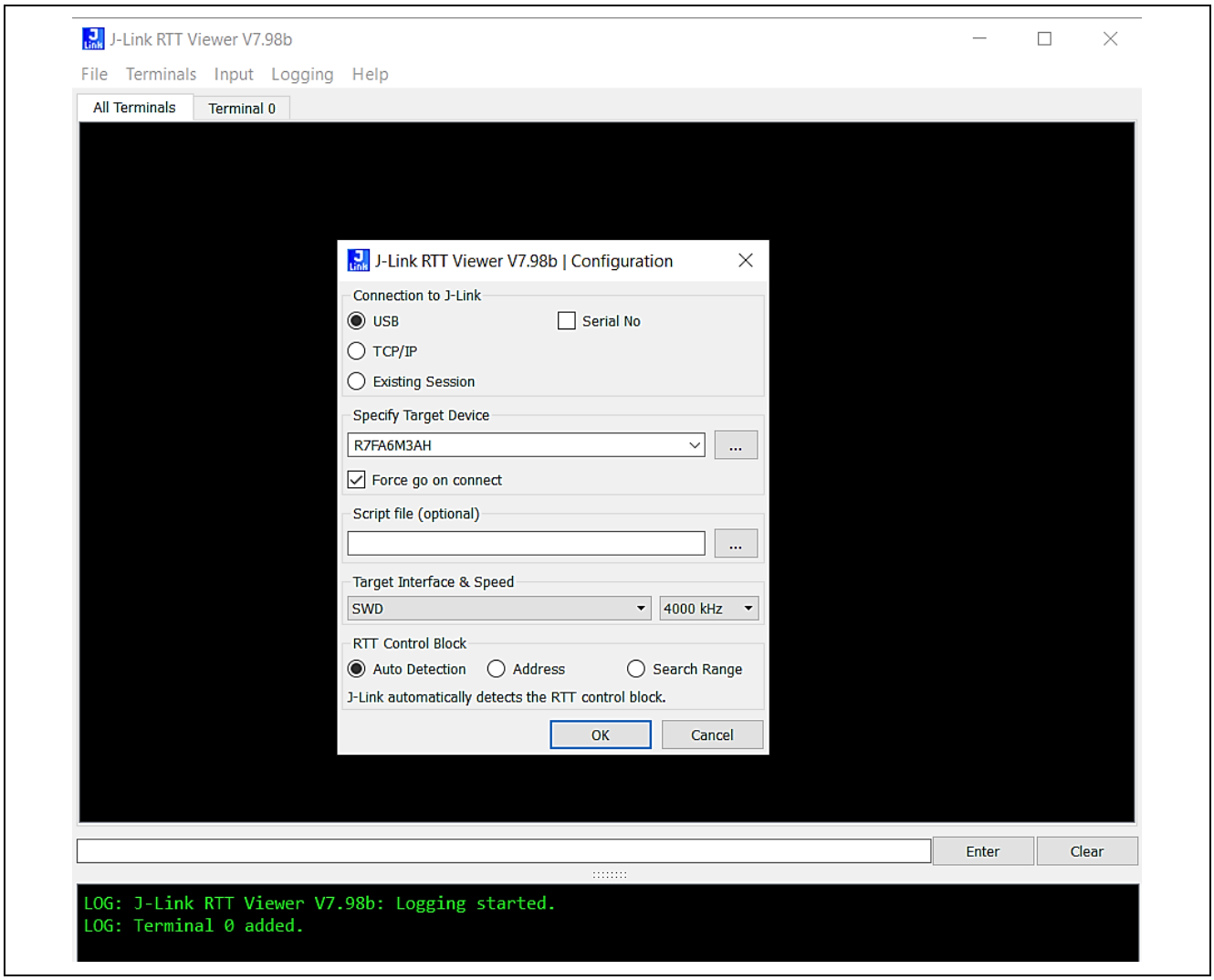

Figure 17. Snapshot of RTT Viewer Settings

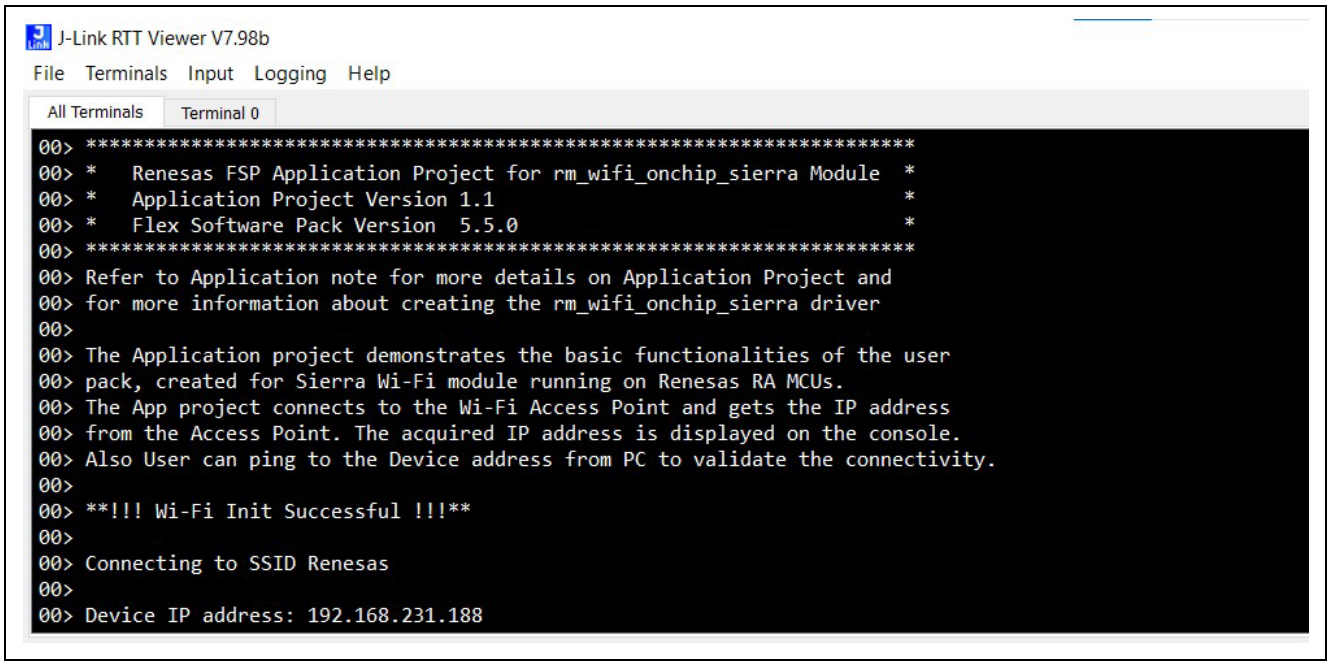

Figure 18. Snapshot of the User Interface for the Application

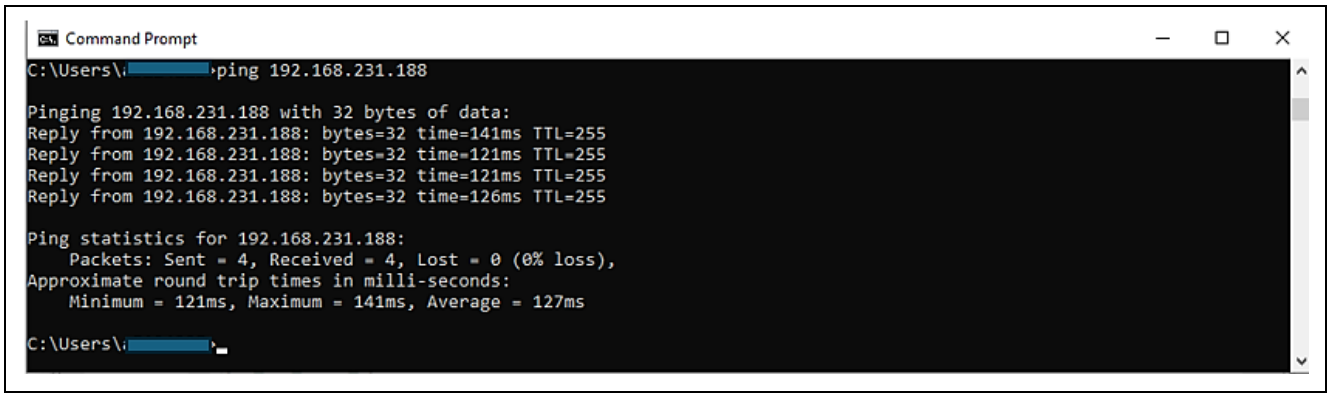

Figure 19. Snapshot of Validating the Connectivity to the Module

8. Known Issues

The Sierra Wireless BX310x Development Board does not provide an AT command for a DNS feature. But there is a workaround to make it available for the application given as part of **section 9**.

9. References

- RA Flexible Software Package (FSP) Documentation: https://renesas.github.io/fsp
- CMSIS-Pack Documentation: https://www.keil.com/pack/doc/cmsis/Pack/html/index.html
- FreeRTOS Stream Buffer: https://www.freertos.org/RTOS-stream-message-buffers.html
- Suggested Workaround for DNS Client: *Application Note for Setting Up a DNS Service in a Private Network-Rev1.1*: https://forum.sierrawireless.com/uploads/short-url/cDpnrH63tlv7jBIsy0s5jjcgNPn.pdf
- Custom BSP Creation: https://en-support.renesas.com/knowledgeBase/19427072
- RA6M3 MCU: https://www.renesas.com/us/en/products/microcontrollers-microprocessors/ra-cortex-m-mcus/ra6m3-32-bit-microcontrollers-120mhz-usb-high-speed-ethernet-and-tft-controller

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 Support	www.renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.16.20	-	Initial version
1.10	Jan.02.24	-	Update to FSPv5.0.0
1.20	Oct.03.24	-	Update to FSPv5.5.0

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

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/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.