Renesas Synergy™ Platform

Wi-Fi Porting Guide

R11AN0342EU0100
Rev.1.00
Oct 1, 2018

Introduction

Wi-Fi drivers ported onto Synergy platform should follow SSP guidelines. This document provides guidelines and steps for porting third party Wi-Fi drivers. The document also briefs on design considerations when using ThreadX and SSP architecture.

Required Resources

Hardware

- Any Synergy Kit (renesas.com/us/en/products/synergy/hardware)
- Host PC running Windows 7/8/10

Development tools & software

- e^2 studio ISDE (renesassynergy.com/devtools/e2studio), or IAR Embedded Workbench® for Renesas Synergy™ (renesassynergy.com/devtools/ewsyn)
- Synergy Software Package (SSP) (renesassynergy.com/software/ssp)

Prerequisites and Intended Audience

- This application note assumes that the user has some experience with the Renesas e^2 studio ISDE and Synergy Software Package (SSP). Before porting the Wi-Fi Drivers, follow the procedure in the *SSP User Manual* to build and run the Blinky project. Doing so enables the user to become familiar with the e^2 studio and the SSP. It is also assumed that the user is aware of SSP Best Practices and Guidelines.
- The intended audience are users who want to port Wi-Fi modules onto Synergy Platform. The user should be aware of embedded and ThreadX concepts.

Contents

1. Overview of Templates

Wi-Fi drivers ported onto SSP should be delivered in the form of Pack Files. Pack File templates are provided for the ease of adaptation and creation of custom packs.

There are 2 Wi-Fi templates available with the package depending on where the stack resides.

- Wi-Fi On-MCU: Wi-Fi stack on Synergy MCU
- Wi-Fi On-Module: Wi-Fi Stack on the Wi-Fi module

Each of the templates can be customized to include custom Wi-Fi drivers, add SSP framework, add configurator features and connect to Synergy drivers.

1.1 Pack File Structure

Pack File contains the following:

- Content: This is the code and documentation that is to be distributed with the pack.
- PDSC File: This is the Pack Descriptor File which describes all the files and folders included in the pack.
- XML Configurator: This is the XML file that defines the location of the module and properties of the module in the **Threads** tab of the Synergy Configurator. All the XML files are included in the `.module descriptions` folder.

1.2 Folder Organization

All the code is located in the `/synergy/company` folder.

1.2.1 Folder Organization for On-MCU Case

All the SSP related code in the pack file is organized into `inc` and `src` folders.

SSP related code consists of 4 files below. These files should not contain definitions of APIs, macros or structures of the Wi-Fi module drivers.

1. `sf_wifi_partnumber.h`: This is an instance header file. It consists of structure definitions of the configurations of Wi-Fi module. The configurations include instances to low-level drivers, reset pin, slave select pin, timeouts, external irq instances and other properties that can be configured.

   This file is in the `/synergy/company/inc/framework/instances/` folder.

2. `sf_wifi_partnumber_private_api.h`: This file contains declarations of the API instances supported by the Wi-Fi module.

   This file is in the `/synergy/company/src/framework/sf_wifi_partnumber/` folder.

3. `sf_wifi_partnumber_private.h`: This file contains the structures and macros used in the framework. It also contains the extended cfg structure that is used for user defined configurations for the module.

   This file is in the `/synergy/company/src/framework/sf_wifi_partnumber/` folder.

4. `sf_wifi_partnumber.c`: This file contains the implementation of the API instances declared in `sf_wifi_partnumber_private.h` file.

For detailed explanation on the contents of these files refer to section 4.2.2.

All the Wi-Fi module drivers are in the `vendor_wifi_driver` folder.

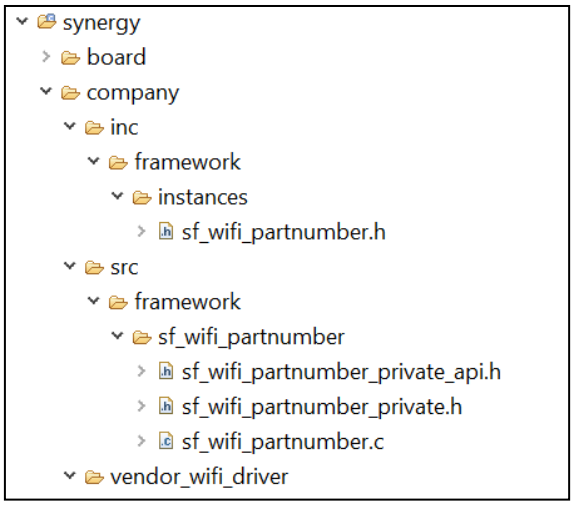

Figure 1 Folder Structure – On-MCU Case

1.2.2 Folder Organization for On-Module Case

For the On-Module case, the pack file contains only vendor drivers or Wi-Fi module drivers.

These drivers reside in `vendor_wifi_driver` folder. There is no requirement on the organization of files in the folder. They may also include library files.

1.3 Pack File naming

The naming of pack files is based on whether the pack file is intended for On-MCU case or for On-Module Case.

The On-MCU pack files are named as <Company>.Synergy_wifi_<partnumber>_OnMCU_<Version>.

The On-Module pack files are named as <Company>.Synergy_wifi_<partnumber>_OnModule_<Version>.

The name of the `.pdsc` file is the same as the pack file and XML file is named described in the pack file. Further details of naming are covered in section 3.

2. Customizing templates

2.1 Overview

The templates should be customized for the following:

- Code
- XML file and the name
- PDSC file and the name
- Pack file name

Throughout the templates the term “company” should be replaced with the name of the company designing the code, “vendor” should be replaced with the vendor of the Wi-Fi module and “partnumber” should be replaced with the part number of the Wi-Fi module.

2.2 Customizing Code

SSP Wi-Fi framework should be used only for the case of On-MCU stack.

2.2.1 On-Module Case

For On-Module case, all the application code and abstraction layer, if any, should be included in the `/src` folder.

2.2.2 On-MCU Case

For On-MCU case, follow the templates provided. All the code should be placed in the appropriate files in the template. Instances for all the APIs supported for On-MCU case should be defined and handled for the Wi-Fi module.

2.3 Customizing XML configurator

XML configurator file is in the `.module_descriptions` folder.

There are two types of elements in the configurator file:

- config
- module

module elements define the properties in the configurator while *config* elements define the module configurations at the application level.

2.3.1 Config elements

Generally, *config* element consists of stack size and preprocessor settings like Parameter Checking. It also consists of the path to the `cfg.h` file where the config parameters are generated. This path should be `synergy_cfg/ssp_cfg/framework/company/sf_wifi_partnumber_cfg.h` for both the On-MCU and On-Module cases.

In the XML file,the line: `id="config.framework.sf_wifi_partnumber_v2" path="ssp_cfg/framework/company/sf_wifi_partnumber_cfg.h" version="0"` should be modified appropriately to point to the right folder.

It is advised to include parameter checking and enable it to minimize errors.

2.3.2 module elements

There are three parts to the `module` element.

1. Location of the module in the configurator: The following line should be modified to indicate the location of the module in the configurator.

```
config="config.framework.sf_wifi_partnumber_v2"
display="Framework|Networking|Wi-Fi|${module.framework.sf_wifi_v2.name}
CUSTOM Wi-Fi Device Driver on sf_wifi_partnumber"
id="module.framework.sf_wifi_partnumber_v2" version="1"
```

   For On-MCU case, the module should be located in the **Framework|Networking|Wi-Fi** tab and it should be displayed as **Vendor Wi-Fi Device Driver on sf_wifi_partnumber**.

   For On-Module case, the module should be located in the **Framework|Networking|Wi-Fi** tab and it should be displayed as **On-Module Vendor Wi-Fi Driver**.

   The name of the XML defined the location of the module in the components tab of the configurator.

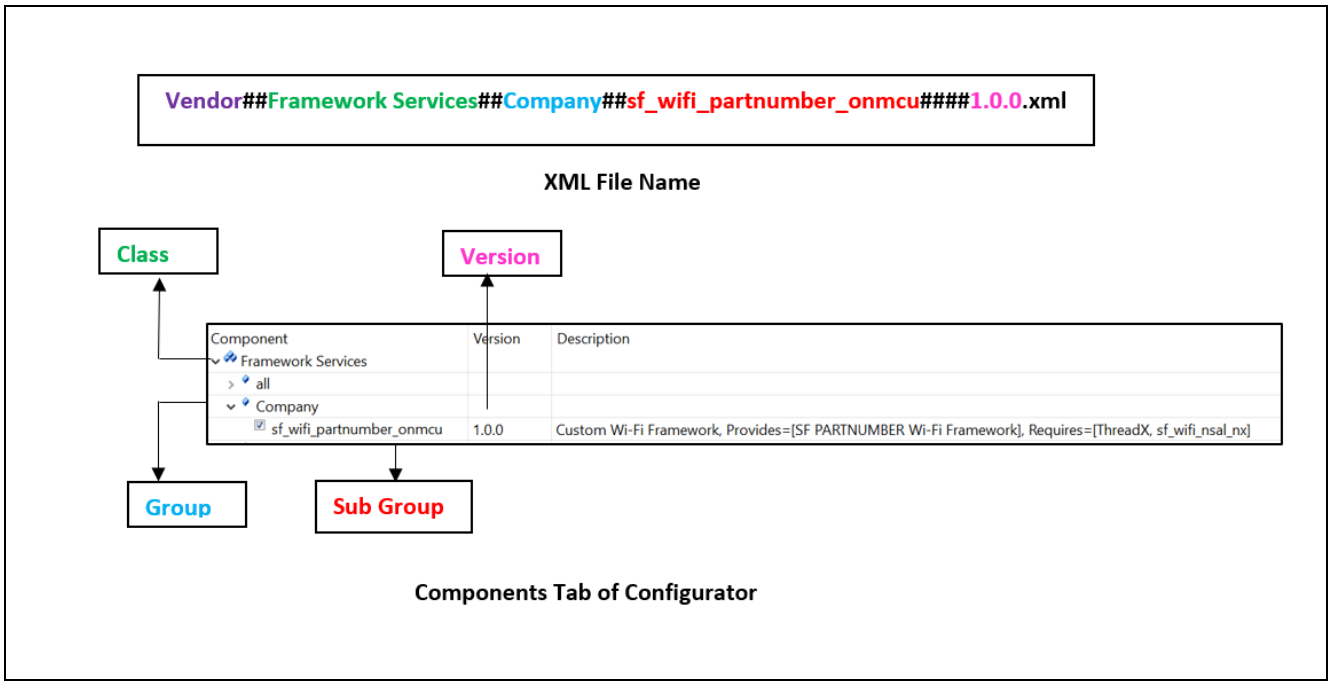

Figure 2 Location of the Module in the Components Tab of the Configurator

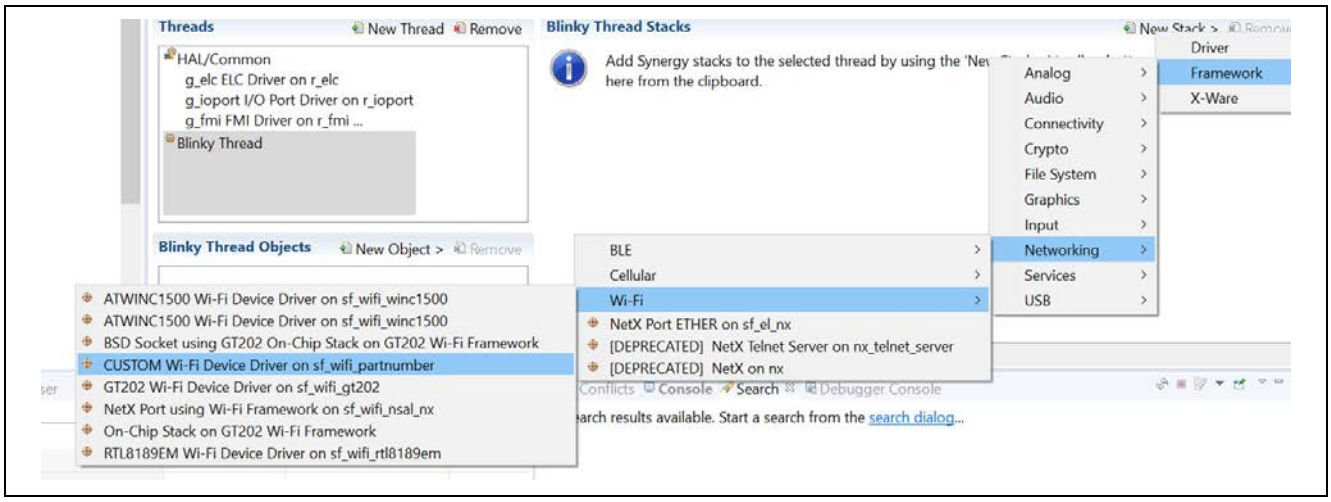

Figure 3 Location of On-MCU Module in the Threads tab in Configurator

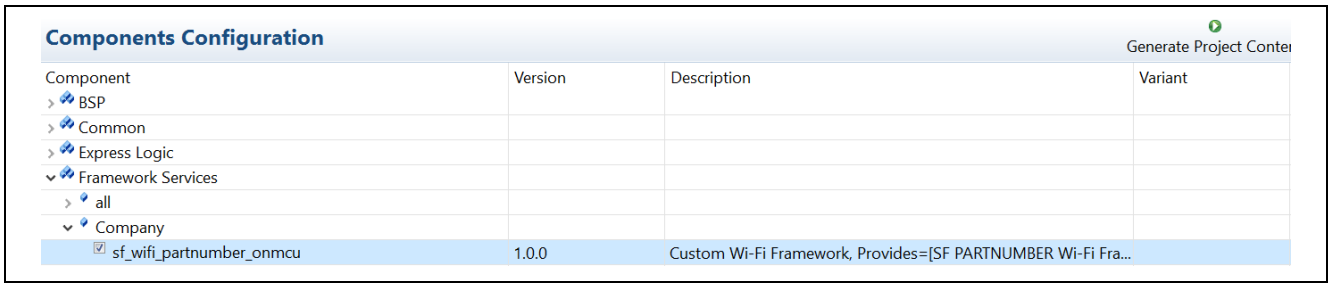

Figure 4 Location of On-MCU Module in the Components tab in Configurator

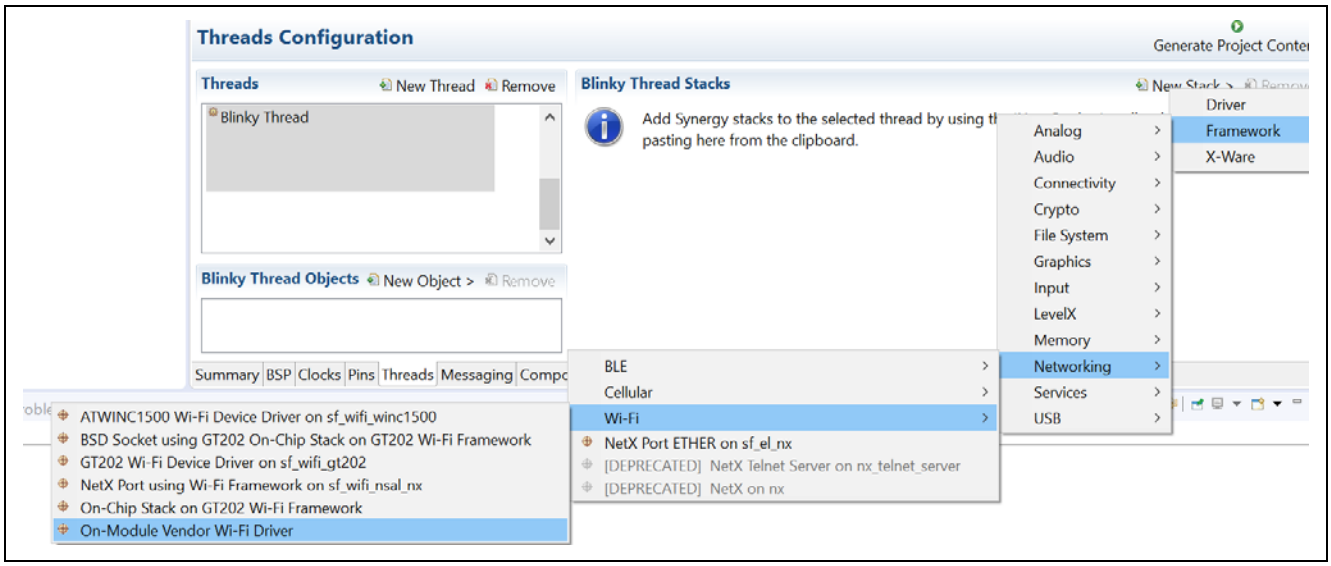

Figure 5 Location of On-Module Module in the Threads tab in Configurator

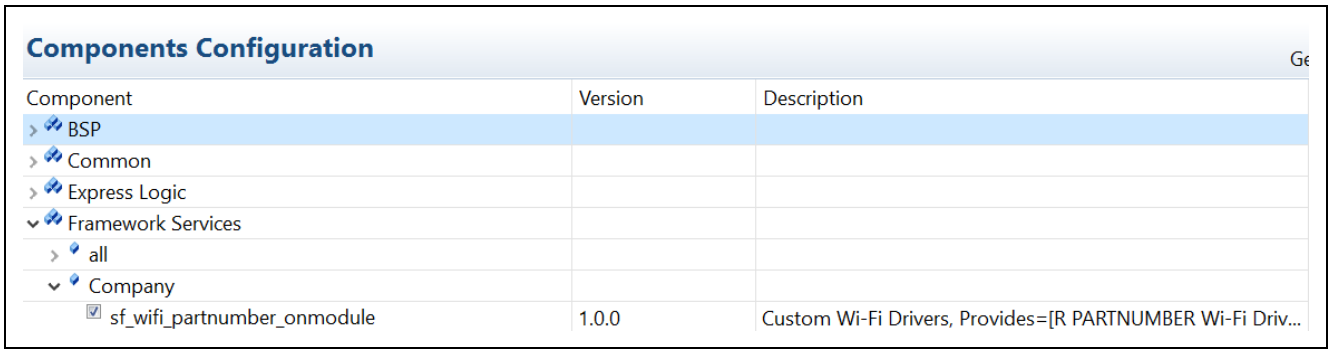

Figure 6 Location of On-Module Module in the Components tab in Configurator

2. Configurator Properties: These include the properties of the module displayed in the configurator and their default values. The 'requires' and 'provides' elements indicate the modules that are a level above and below the Wi-Fi module. The templated XML files have 3 kinds of peripherals – SPI, UART and SDMMC Frameworks- available. One of those should be used based of the module requirements.

   Other properties include Reset Pin, Slave Select Pin, tx power, Thread Priority, Timeouts, callbacks, external IRQs etc as needed by the module.

   Note: The pins configurations should be changed accordingly for the application to work.

3. Defintions: The third part of the XML file is the definitions of the structures to be generated by the configurator and the required header files.

   For the On-MCU case, `sf_wifi_api.h`, `sf_wifi_partnumber.h`, and `sf_wifi_nsal_api.h` files and any other SSP include files, if not included in other files, should be included in the includes section of the file.

```
<includes>
#include &quot;sf_wifi_api.h&quot;
#include &quot;sf_wifi_partnumber.h&quot;
#include &quot;sf_wifi_nsal_api.h&quot;
</includes>
```

   ctrl, cfg structure definitions and the instances of the module are also declared in the XML file. Instance of the module should include api, ctrl and cfg structures.

   In addition to the above configurations, zero-copy feature, that is supported by the NSAL layer of Wi-Fi framework, is also declared in the configuration. Refer to section 4.2 for more information on zero-copy feature.

   Refer to the SSP Module Guide for detailed explanation on how to modify the XML file further.

2.4 Customizing pdsc file

The pdsc file includes:

- Support for GCC compiler
- Support for IAR compiler
- Description of XML configurator file. This also defines the location of the module in the Components tab of the configurator
- Declaration of all the files and folders included in the pack

The folder to be included should be listed under `category="include";` header files should be listed under `category="header"` and source files should be included under `category="source"`.

There are 2 important considerations while creating a pdsc file.

- XML file description in the .pdsc file should match the name of the XML file.
  Figure 7 is the example of the xml file description in PDSC file and the corresponding XML file name

```
<component Cclass="Framework Services" Cgroup="Company"
Csub="sf_wifi_partnumber_onmcu" Cvendor="Vendor"
Cversion="1.0.0" condition="">
```

Figure 7 XML Description in the PDSC File

```
Vendor##Framework
Services##Company##sf_wifi_partnumber_onmcu####1.0.0.xml
```

Figure 8 XML File Name

- Name of the pdsc file should be the same as the name of the pack file without the version number.
  For example: If Pack File is named `Company.Synergy_wifi_partnumber_OnMCU.1.0.0.pack`, the pdsc file should be named `Company.Synergy_wifi_partnumber_OnMCU.pdsc`.

If any of the above conditions are mishandled, the pack file becomes non-functional.

2.5 Version Control

Version of the pack file is independent of the SSP version. Initial version of the pack file should be 1.0.0. The version should be incremented with any changes to the pack file. Decisions on incrementing major, minor and patch versions are left to the discretion of the developer.

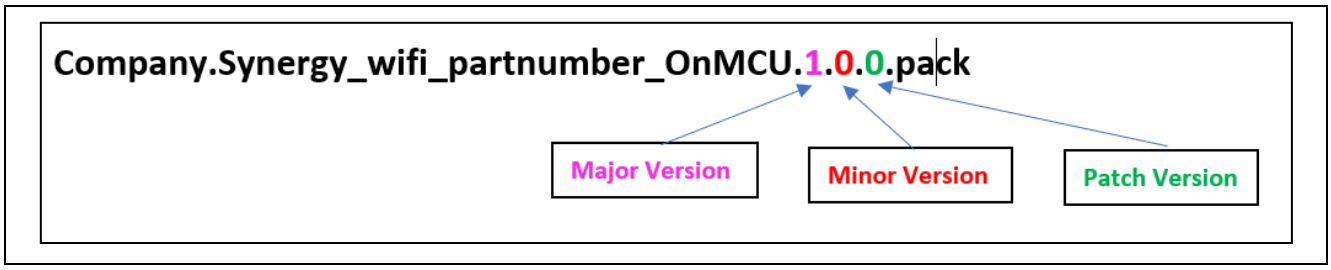

Figure 9 Pack File Version Schema

Version of the files included in the pack file do not have to follow the version of the pack. This allows tracking of changes to the files independently of the packs.

```
<component Cclass="Framework Services" Cgroup="Company" Csub="sf_wifi_partnumber_onmcu" Cvendor="Vendor" Cversion="1.5.0" condition="">
<description>Custom Wi-Fi Framework, Provides=[SF PARTNUMBER Wi-Fi Framework], Requires=[ThreadX, sf_wifi_nsal_nx]</description>
<files>
    <file category="include" name="synergy/company/inc/framework/instances/"/>
    <file category="include" name="synergy/company/src/framework/sf_wifi_partnumber/"/>
    <file category="include" name="synergy/company/vendor_wifi_driver"/>
    <file category="header" condition="" name="inc/framework/instances/sf_wifi_partnumber.h" select="" src="" version="1.0.0"/>
    <file category="header" condition="" name="vendor_wifi_driver/example_folder/include/nm_bsp.h" select="" src="" version="1.2.0"/>
    <file category="source" condition="" name="vendor_wifi_driver/example_folder/source/nm_bsp.c" select="" src="" version="1.5.0"/>
</files>
</component>
```

Figure 10 PDSC File (Ver. 1.5.0) with files of different versions

When more than one pack file of the same name is included in the `/packs` folder, the pack file with the highest version number is selected by default by the configurator. To change this, the desired pack file should be manually selected in the Components tab and the other pack file should be unselected.

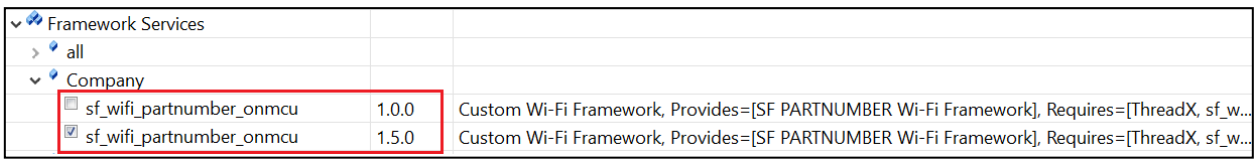

Figure 11 Pack with Higher Version is automatically selected in the configurator

2.6 Generating the pack file

- Select all the folders and files to be included in the pack as shown in Figure 12.
- Right click and zip

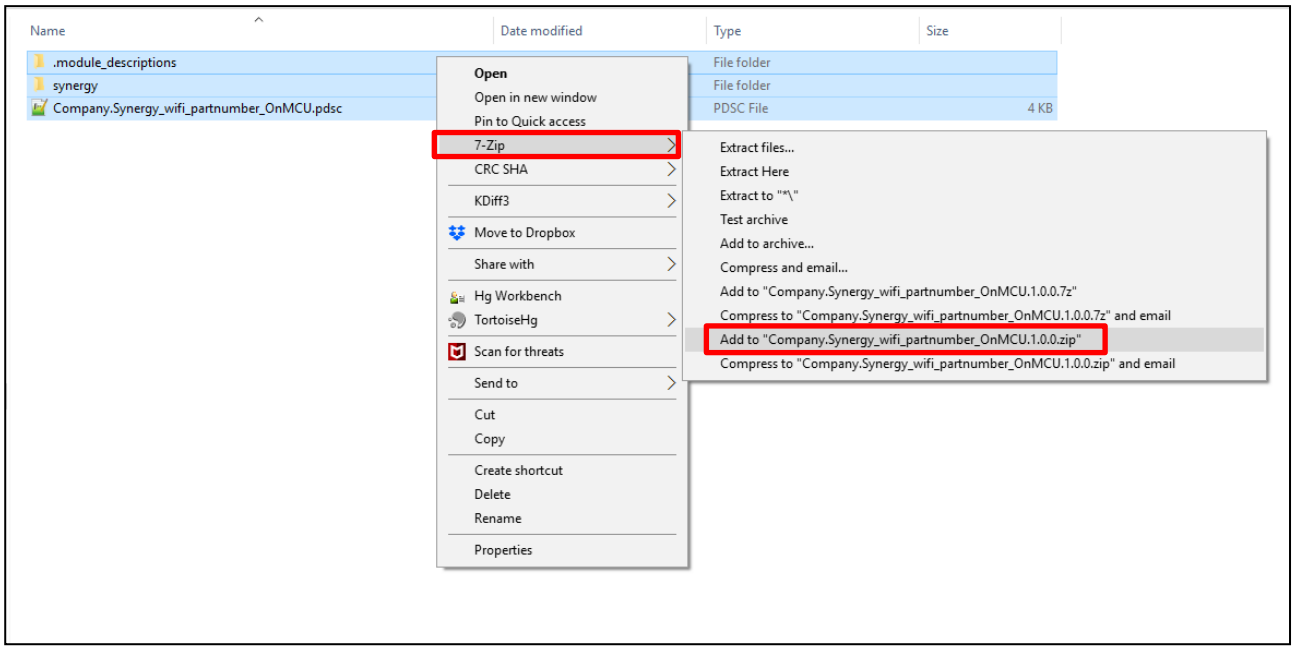

Figure 12 Pack File Creation

- Change the file suffix to `.pack`
  For example: Company.Synergy_wifi_partnumber_OnMCU.1.0.0.**zip** to Company.Synergy_wifi_partnumber_OnMCU.1.0.0.**pack**
- Copy the .pack file to the `/Synergy/e2studio/internal/projectgen/arm/packs/` folder for e² studio or `Synergy/ssc/internal/projectgen/arm/packs/` folder for IAR Workbench.

2.7 Migration of Pack Files with new SSP and e² studio

Since the pack file version in not dependent on the SSP version, pack files do not need to be modified with new SSP versions. A change in the pack file would be needed only if there is a change in the code. However, with every new e² studio or IAR Workbench, pack file should be copied to the corresponding `/packs` folder.

Refer to the Reference Pack File for details on creating a pack file.

3. Importing Pack File

A .pack file can be installed in two ways.

- Manually copy the pack file into `/Synergy/e2studio/internal/projectgen/arm/packs/` folder if using e² studio or copy the pack file into `/Synergy/ssc/internal/projectgen/arm/packs/` folder if using IAR Workbench.
- Install using the 'Import' option in e² studio as shown below.
  - Choose File -> Import -> General -> CMSIS Pack
  - Click Next
  - Browse to the location of the pack file, choose the file
  - Click Finish

It is highly recommended that e² studio and SSC are closed when pack files are installed or removed.

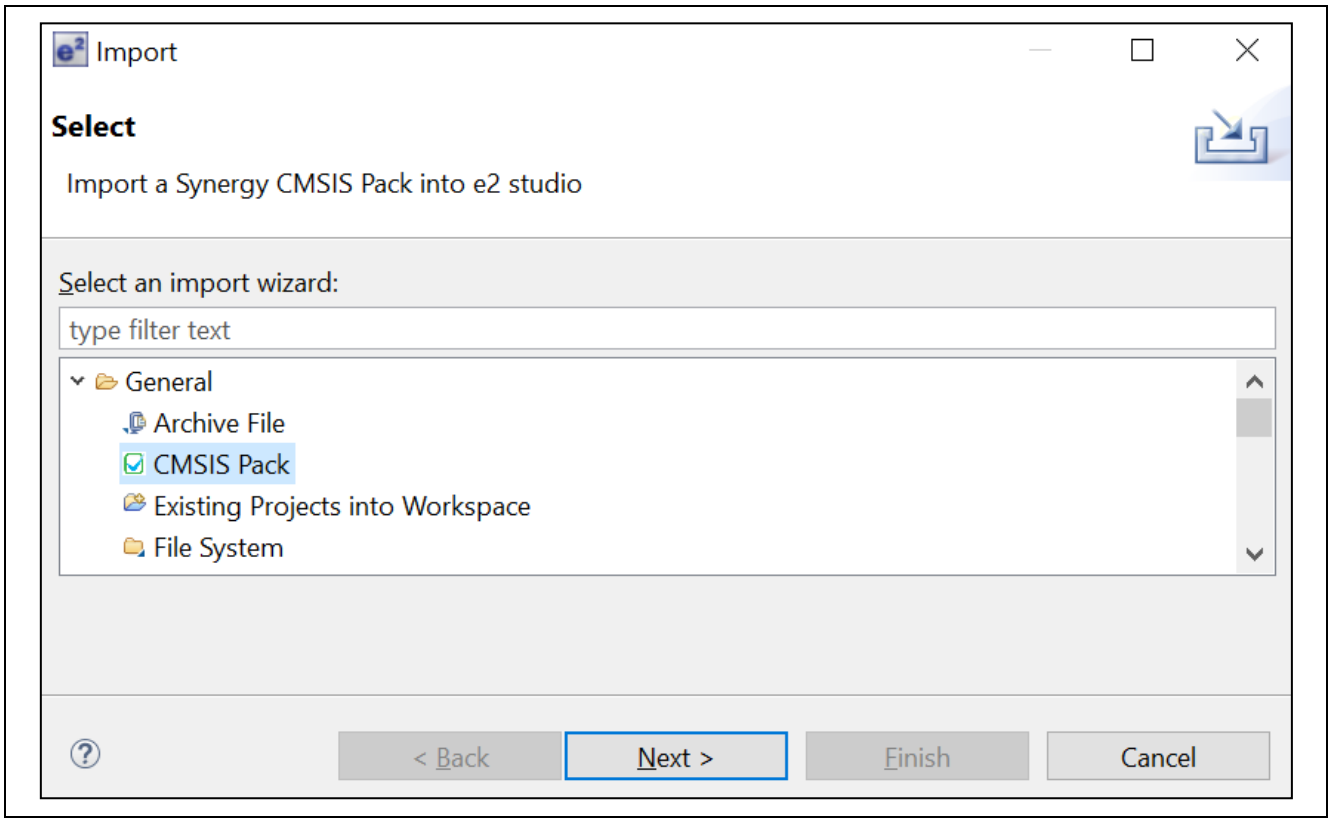

Figure 13 Importing Pack File from e² studio

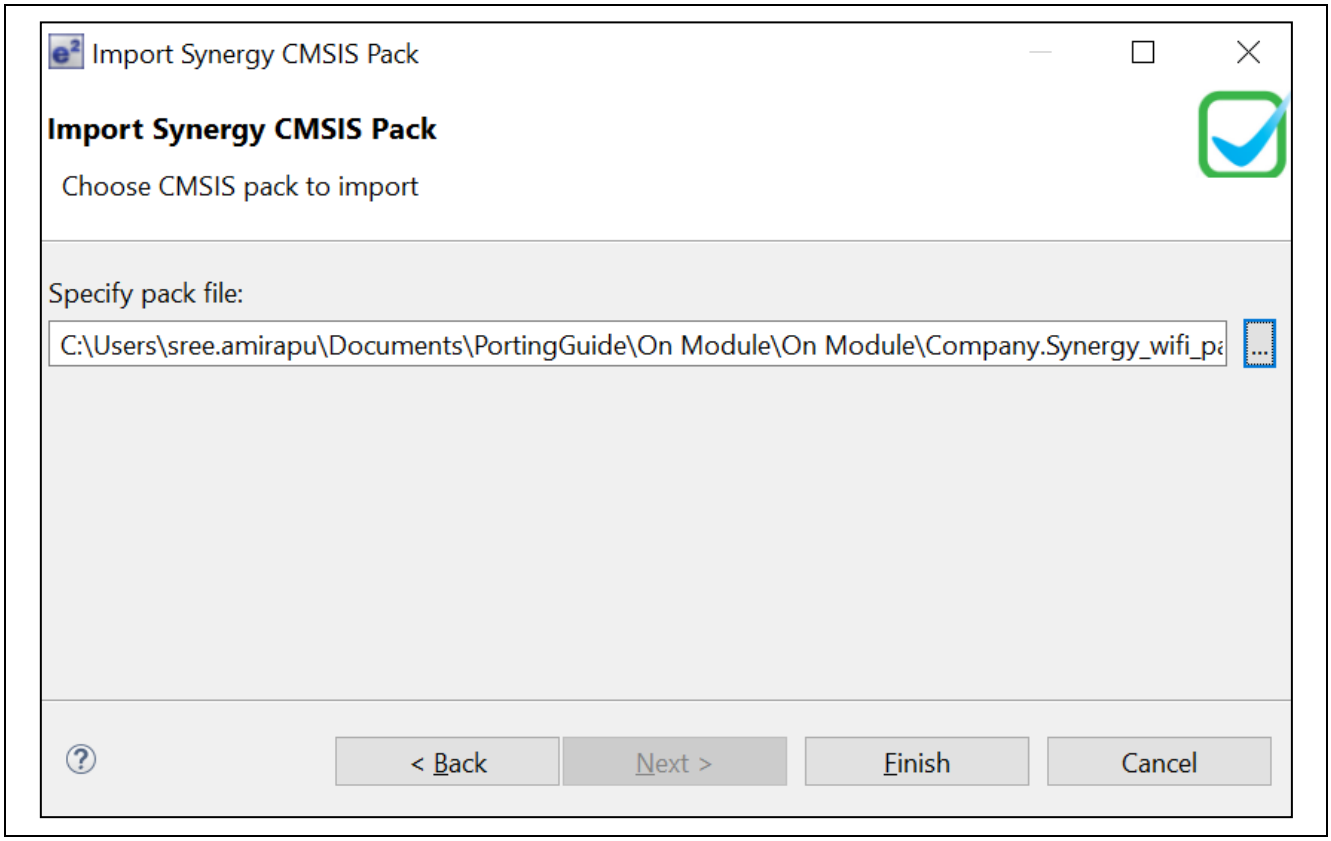

Figure 14 Importing Pack File from e² studio

Uninstalling a pack file is manual. The pack file to be uninstalled should be deleted from the `/packs/` folder for IAR Workbench and e² studio.

For both installation and uninstallation of packs, e² studio or IAR Workbench should be restarted to ensure the new pack file is available.

It is sometimes required to delete the 'supportfilepacks.xml' file in the `/packs/` folder and then restart e² studio or IAR workbench.

4. Overview of Driver Adaptation for Synergy Platform

4.1 RTOS Adaptation

This section gives a high-level overview of working with ThreadX and Synergy Platform. It is, however, assumed that the user is aware of general RTOS concepts and this section does not get into the details of RTOS concepts.

Synergy platform is preconfigured with ThreadX which is included as a library. This greatly improves the compilation time since the ThreadX Source does not need to be compiled every time with the application. Below are some of the basics of ThreadX and Synergy Platform that should be considered for transitioning to ThreadX.

4.1.1 Compiler Dependency

- ThreadX is written in ANSI C
- Source code of ThreadX is in ASCII format
- ThreadX is endian neutral

4.1.2 Threads

Threads in SSP can be added through the configurator where the properties of the thread like stack size, priority, time slicing interval can be configured.

4.1.2.1 Scheduling

ThreadX scheduling is based on the priority. Round-robin scheduling is used for scheduling threads of same priority.

4.1.2.2 Thread Synchronization

Message Queues: Message queues are the primary means of thread synchronization in ThreadX. Message size is at least 32 bits (1 word) and can be of a maximum size of 16 words. Messages greater than 16 words should be passed as pointers. This also optimizes memory usage. Message queues can be created from Synergy Configurator as shown in Figure 15. Message size and queue size can be configured from the **Properties** tab.

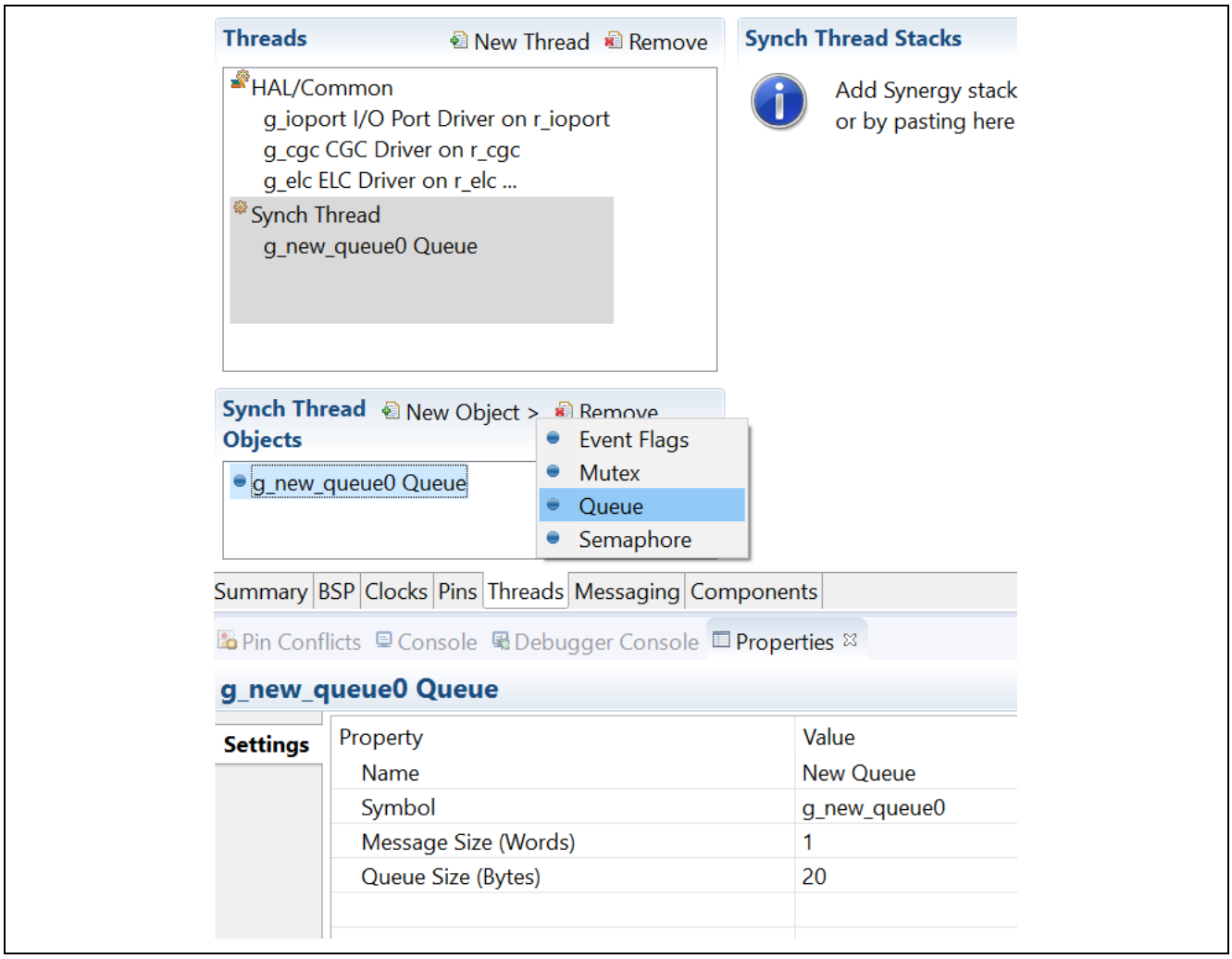

Figure 15 Thread Synchronization Objects

Mutexes: Mutexes also can be created by the configurator. Select the thread and create a mutex just like message queue. “Priority Inheritance” is disabled by default. Enable priority inheritance to avoid priority inversion. Also, make sure to use a timeframe for priority inheritance.

Semaphores: Semaphores in ThreadX are 32-bit counting semaphores. Semaphores can also be used for event notification. Semaphores are created from the configurator as well. `tx_semaphore_get` and `tx_semaphore_put` are the 2 APIs used for increasing and decreasing the semaphore count.

Event Flags: Based on the application, event flags might be the most efficient method of thread synchronization. Event flags in ThreadX are a group of 32 individual bits at the same memory location. Setting of event flags in logical AND/OR operation. So, event Flags are preferred over semaphores for optimized memory and speed usage.

4.1.3 Memory

Static memory usage of ThreadX is determined by the compiler and linker while the dynamic memory is determined by the application. This means that the memory associated with stacks, queues etc can be reside anywhere.

ThreadX has built-in memory pools – Memory Block Pools and Memory Byte Pools. Block pools consist of fixed size blocks. Block pools are the most preferred method of dynamic memory allocation as there are no associated fragmentation issues. Byte pools are similar to heap and so can result in fragmentation. However, they come with an advantage of flexibility in usage. Application threads can suspend on a pool until the memory in the pool is available.

4.1.4 ISR

Applications can manage the interrupts in ThreadX. Some ThreadX APIs can be called from within ISRs.

4.1.5 Other Properties

System timer frequency by default is set to 100 ticks, which is 10 ms. However, this value can be configured as needed by the application by adding **ThreadX Source** to the **HAL/Common Thread** in the configurator as shown below. In reality, ThreadX applications can run without a system timer at all.

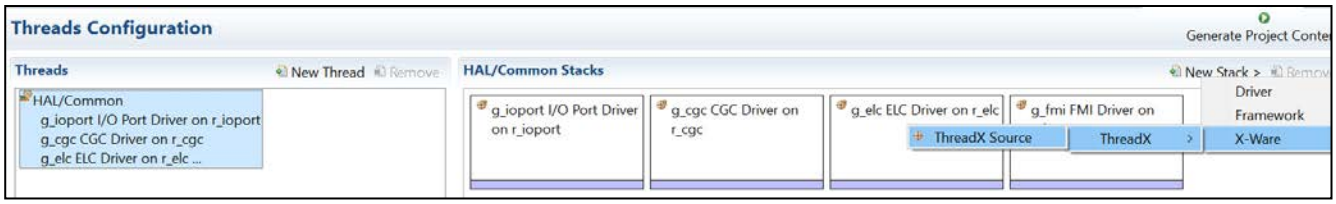

Figure 16 Customizing ThreadX Source

There are several other properties of ThreadX that can be modified in ThreadX source like minimum stack size, preemption threshold, event trace, performance info, etc.

Enabling preemption threshold is highly recommended to minimize resource starvation. A running thread can only be preempted if the preempting thread has a priority higher than the running thread's PreemptionThreshold.

4.1.6 Error Codes

There are two kinds of error codes

- Common Error Codes that are common to all the modules. These are found in the `ssp_common_api.h` file located in the `/synergy/SSP folder.`
- Module specific error codes. These are defined in the module API header files.

Refer to the *ThreadX User Manual* for further reading.

4.2 Porting to SSP

The first design consideration for porting the Wi-Fi drivers onto SSP is to have the stack running either on Synergy MCU or on the Wi-Fi module. There are advantages to both the approaches.

On-MCU case utilizes SSP Wi-Fi Framework. Wi-Fi framework includes Wi-Fi APIs, NetX Stack, Network Stack Abstraction Layer (NSAL) and Wi-Fi module interface. On-MCU case provides the ease of building applications on Synergy Platform without having to understand the details of the module driver. Also, it provides consistency in integrating the drivers which allows the application to work with multiple modules.

On-Module case is ideal for cases where memory is a constraint. Also, for cases where the application requires third-party stack integration, On-Module case is preferred.

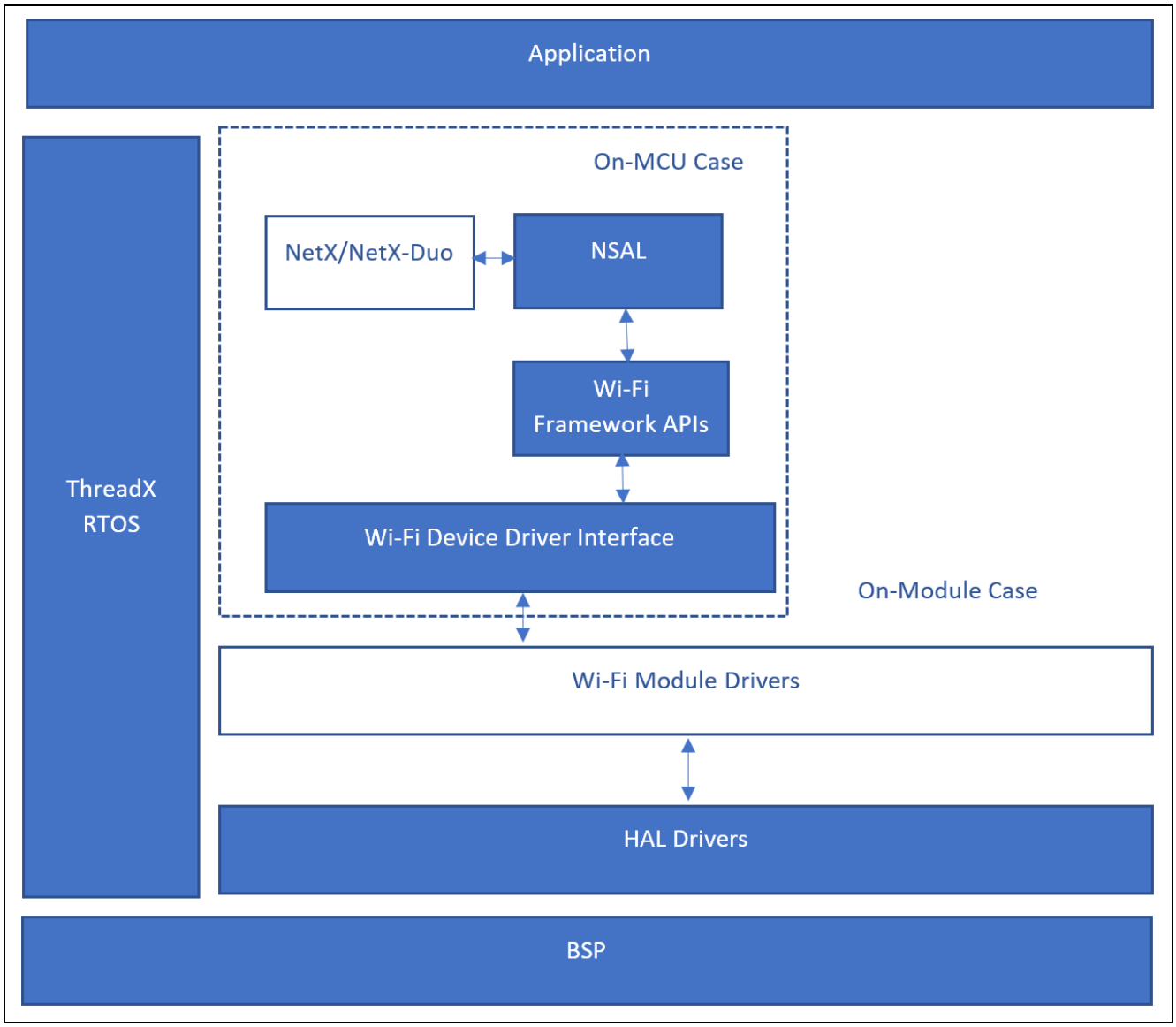

Figure 17 Wi-Fi Module on SSP – On-MCU and On-Module Cases

4.2.1 SSP Wi-Fi Framework Feature Support

Enums are defined for most of the generic Wi-Fi features in SSP Framework. These enums are defined in `sf_wifi_api.h` file. Below are some of these features.

4.2.1.1 Wi-Fi standards

SSP Wi-Fi Framework supports 802.11a, 802.11b, 802.11g and 802.11n standards. It supports both types BSS (Basic Service Set): Infrastructure and Independent (ad-hoc) BSS

4.2.1.2 Wi-Fi Modes

SSP Wi-Fi framework supports both Access Point mode and Client (Station) mode.

4.2.1.3 Security Types

SSP framework supports WEP, WPA2, WPA, and Open security. There is support for TKIP, CCMP, and WEP encryption types. Both ASCII and Hex Key formats are acceptable for WEP keys.

4.2.1.4 Power Management

There are no specific APIs for power management in SSP framework. User has the capability of configuring the Tx power of the module and this can be in the range of 1–17 dBm. However,the `.close` API de-initializes the network interface and puts it is either in low-power mode or power it off when the operation is module is disabled.

4.2.1.5 Other features

- Access Control List Management: Allows the application to control the devices the can be connected to an access point
- Multicast Filter List Management: Allows the application to leave or join multicast group
- Zero-Copy: This feature allows for optimized memory usage for modules that can support NetX packet pools and structures. If zero-copy is enabled, the packets are directly copied from Wi-Fi module to NetX and from NetX to the module without passing through the driver buffers for processing. Because of this there are no multiple copies of the data and processing time is reduced significantly.

4.2.2 Framework Usage for On-MCU case

This section gives an overview of SSP Wi-Fi Framework. Refer to the Wi-Fi Framework Module Guide for a detailed explanation of the framework.

There are 3 main components to Wi-Fi framework

- APIs
- Network Stack Abstraction Layer (NSAL)
- NetX library

  NSAL layer provides an interface between the Wi-Fi driver's MAC layer data frames and NetX stack.

  Every SSP Framework instance uses three kinds of structures
  - `sf_wifi_ctrl_t`: pointer to the user defined Wi-Fi control structure
  - `sf_wifi_cfg_t`: pointer to the user defined Wi-Fi configurations
  - `sf_wifi_api_t`: contains pointers to the Wi-Fi framework APIs

  The 3 structures are defined in the `sf_wifi_instance_t` struct.

  SSP defined APIs and structures are defined in `sf_wifi_api.h` file.

  In addition to these structures, the Wi-Fi module can include extended cfg and ctrl structures that are specific to the functionality of that module. These structures are defined in the `sf_wifi_partnumber.h` and `sf_wifi_partnumber_private.h` files respectively.

  All the API instances are defined in `sf_wifi_partnumber.c` file. These API instances make calls to the Wi-Fi module APIs.

4.2.3 API Implementation

1. `SF_WIFI_PARTNUMBER_Open()`

   This function initializes the module driver (calls the module open() function). It also configures the parameters defined in the cfg structures and creates any parameters required throughout the process.

2. `SF_WIFI_PARTNUMBER_Close()`

   This function de-initializes the Wi-Fi driver and network interface and deletes the threads and other objects associated with the module drivers. It makes calls to the corresponding module APIs.

3. `SF_WIFI_PARTNUMBER_MulticastListAdd()`

   This function adds the MAC address to the Multicast filter list.

4. `SF_WIFI_PARTNUMBER_MulticastListDelete()`

   This function deletes the MAC address from the multicast list.

5. `SF_WIFI_PARTNUMBER_StatisticsGet()`

   This function gets the statistics of Wi-Fi interface. Statistics include number of packets received successfully, number of packets transmitted successfully and number of transmit errors.

6. `SF_WIFI_PARTNUMBER_Transmit()`

   This function sends packets from the Wi-Fi driver to a transmit queue.

7. `SF_WIFI_PARTNUMBER_ProvisioningSet ()`

   This function configures the module in either client or AP mode. This function needs SSID and other provisioning information.

8. `SF_WIFI_PARTNUMBER_ProvisioningGet()`

This function gets the information required for provisioning like SSID, encryption type, security type, key and channel type.

9. `SF_WIFI_PARTNUMBER_InfoGet()`

   This function gets the module and network information. Information includes chipset/driver information, RSSI value, Signal noise level and link quality.

10. `SF_WIFI_PARTNUMBER_Scan()`

    This function scans for Access Points

11. `SF_WIFI_PARTNUMBER_ACLAdd()`

    This function adds a mac address to the access control list

12. `SF_WIFI_PARTNUMBER_ACLDelete()`

    This function deletes the MAC address from the access control list

13. `SF_WIFI_PARTNUMBER_MacAddressSet()`

    This function configures the MAC address of the Wi-Fi module

14. `SF_WIFI_PARTNUMBER_MacAddressGet()`

    This function reads the configured MAC address of the IE-Fi module

15. `SF_WIFI_PARTNUMBER_VersionGet()`

    This function sets the version of the module

4.2.4 Process Flow

Wi-Fi module Initialization Process:

1. NetX IP instance calls `nx_ip_create`.
2. `nx_ip_create` calls the NetX NSAL driver entry point.
3. The NSAL driver entry point calls the Wi-Fi framework `open()` function.
4. Wi-Fi framework `open()` function calls the Wi-Fi device driver `open()` function to initialize and enable the Wi-Fi module.
5. `nx_ip_interface_status_check` API is called and waits for the `NX_IP_LINK_ENABLED` status to be set.
6. On successful completion, the Wi-Fi module is ready for scan and provision.

Wi-Fi Packet Transmission:

1. User application code provisions the Wi-Fi module.
2. User application code calls the NetXs TCP/UDP Socket Send API.
3. NetX Send API calls the NSAL driver entry point for package transmission.
4. NSAL driver entry point calls the Synergy Wi-Fi Framework Transmit API function.
5. Synergy Wi-Fi framework Transmit API function calls the Wi-Fi device driver Transmit API function.
6. Wi-Fi device driver transmits the user data.

Wi-Fi Packet Reception:

1. Wi-Fi packet reception starts from the Wi-Fi device driver interrupt service routine.
2. Once the packet is received, the receive callback function of the Wi-Fi device driver transfers the receive data to buffer and initiates the Wi-Fi framework receive callback function.
3. The Wi-Fi framework receive callback function calls the NSAL receive callback function.
4. The NSAL receive callback function calls the NetX deferred receive processing callback.

4.3 SSP Wi-Fi Framework Limitations

Wi-Fi framework does not support S1 series of Synergy MCUs due to memory constraints.

5. Testing Guidelines

- Pack file and application should have no build errors or warnings
- Minimum acceptance and testing criterion: Code should comply with the minimum acceptance and testing criterion agreed upon at the inception of the project.
- An Application should be included with the pack file to show the functionality of the module
- C-stat report should be included with the Application Project.
- Application should be optimized with the -O2 option in e^2 studio
- Optimization in IAR should be set to High (Balanced)
- Parameter checking enabled
- Code should comply with SSP Best Practices Guide

6. Known Issues & Limitations

- Limitation with Export Pack Tool in e^2 studio: The tool does not allow spaces in the Class category. So, creating a class of type 'Framework Services' would not be possible. Therefore, it is recommended to use the templates.
- Once the pack file is created using the Export Pack tool, pack should be unzipped and the `.module_description` folder and XML file should be added manually.
- To have a version number that is independent of SSP, the Vendor name in the pack and XML files should not be Renesas. If the vendor field is Renesas and the version does not match SSP versions, the pack file does not show up in the configurator.
- If a new pack file is installed, the existing project will not support it. For a newly installed pack file to be included in an already opened project
  - The `supportfilepacks.xml` file should be deleted from the `/packs` folder and e^2 studio or IAR workbench should be restarted.
  - The `.module_descriptions` folder should be deleted from the project and the project content should be generated once again.
  - If none of the above work, the project needs to be recreated.

7. Example Wi-Fi Application using Reloc On-MCU Pack File

The ATWINC1500_OnMCU Application Project provides an example of using a Wi-Fi On-MCU custom pack file in an application. This project configures the Reloc PMOD.WM1A Wi-Fi module in AP mode.

7.1 Required Resources

Hardware

- Renesas Synergy™ SK-S7G2
- Reloc PMOD.WM1A (Microchip® ATWINC15x0-MR210 Wi-Fi™ radio module)
- Two micro USB connector cables

Software

- Synergy Software Package (SSP) v1.5.0-rc.1
- Renesas Synergy™ e^2 studio Integrated Solution Development Environment (ISDE) v6.2.1
- IAR Embedded Workbench® for Renesas Synergy™ (IAR EW for Synergy) v8.23.1
- Microsoft® Windows® 7 or 10

7.2 Application Setup

1. Download Synergy USB CDC Driver files from https://www.renesas.com/en-us/products/synergy/software/add-ons/usb-cdc-drivers.html

   Follow the instructions in Application Note – Installing Synergy signed USB CDC Drivers to install Synergy USB CDC driver.

2. Connect the Reloc PMOD.WM1A Wi-Fi module to PMOD B. Set the jumper J15 to 3V3 position.

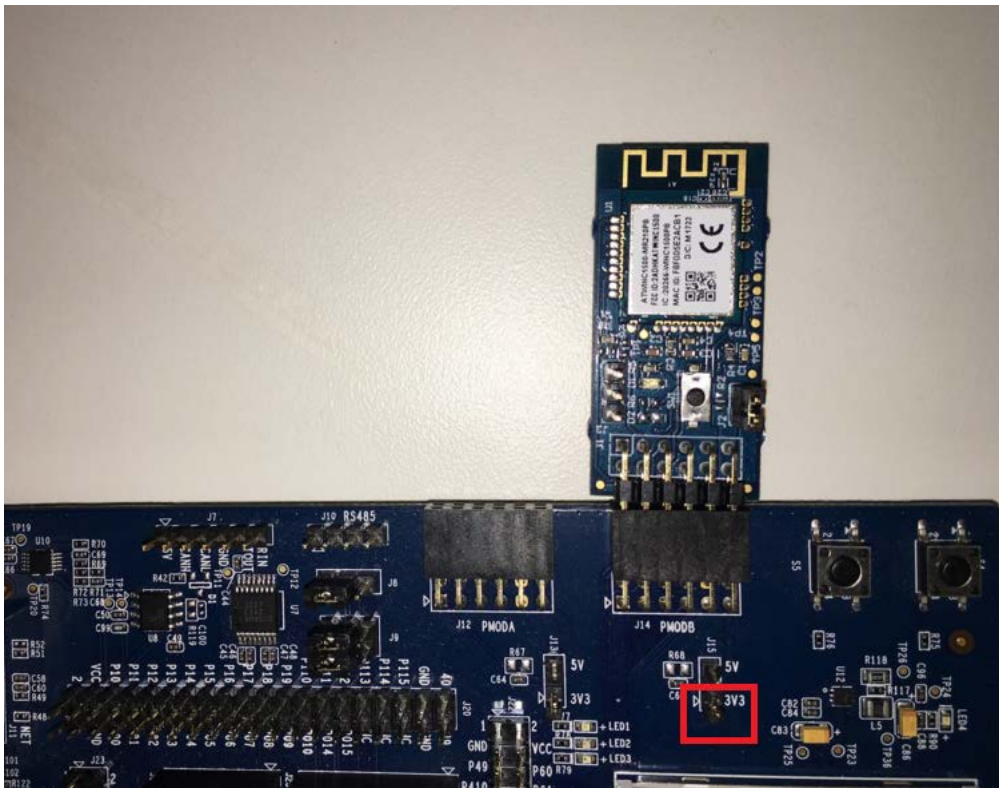

Figure 18 PMOD and Jumper Settings

3. Connect the micro USB cable to J19 port to power up the board
4. Connect the micro USB cable to J5 for serial console

7.3 Importing and Building ATWINC1500_OnMCU Project

Copy file to e^2 studio ver 6.2.1/IAR ver 8.23.1 Packs folder.

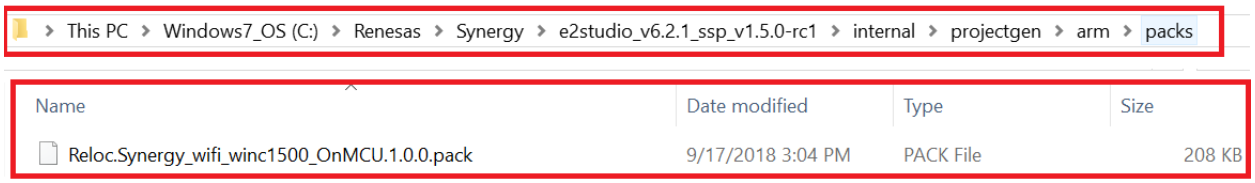

Figure 19 Pack file copied to Packs folder of e^2 studio

Refer to the Synergy SSP Import Guide for instructions on importing, building and running a Synergy project.

7.4 Running the Application

To access the command console from the computer, install TeraTerm Pro (or equivalent serial terminal emulator) and open the terminal with the appropriate COM port configured.

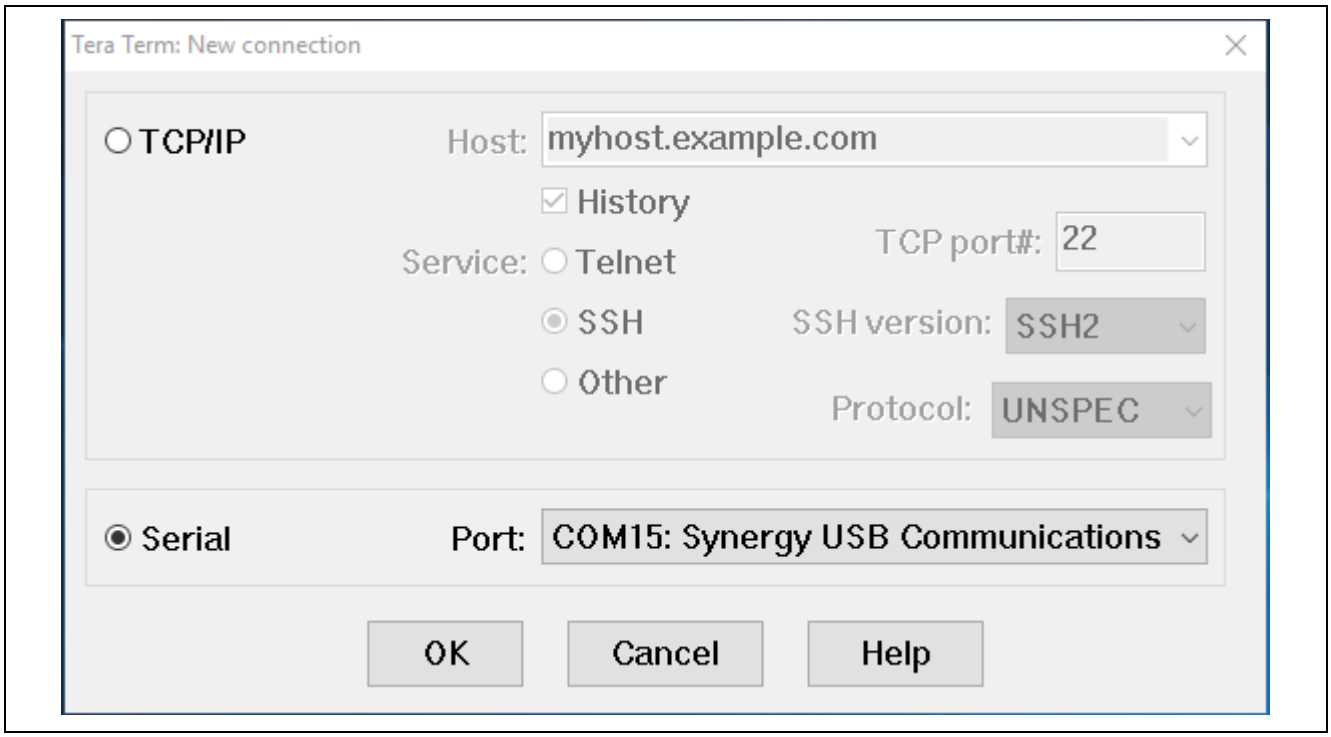

Figure 20 COM Port

Type "?" and press Enter key to display the command menu.

```
Synergy Wi-Fi Framework>?
  Synergy Wi-Fi Framework Help Menu
    1 : Press 1 and then the Enter key to input the Wi-Fi AP SSID
    2 : Press 2 and then the Enter key to input the Wi-Fi AP Password if needed
```

Press "1" to enter the SSID.

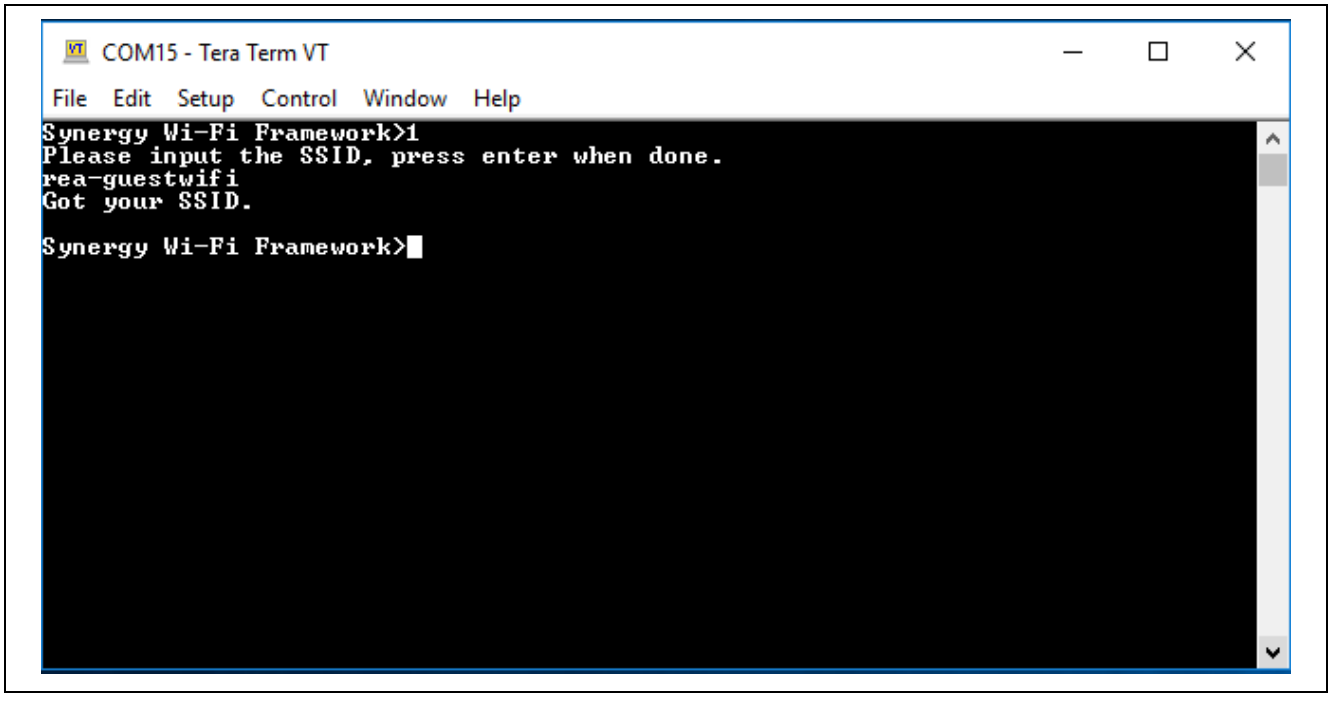

Figure 21 Input SSID

Press "2" to enter the Password.

Figure 22 Input Password

When the provisioning is successful, an IP address is assigned.

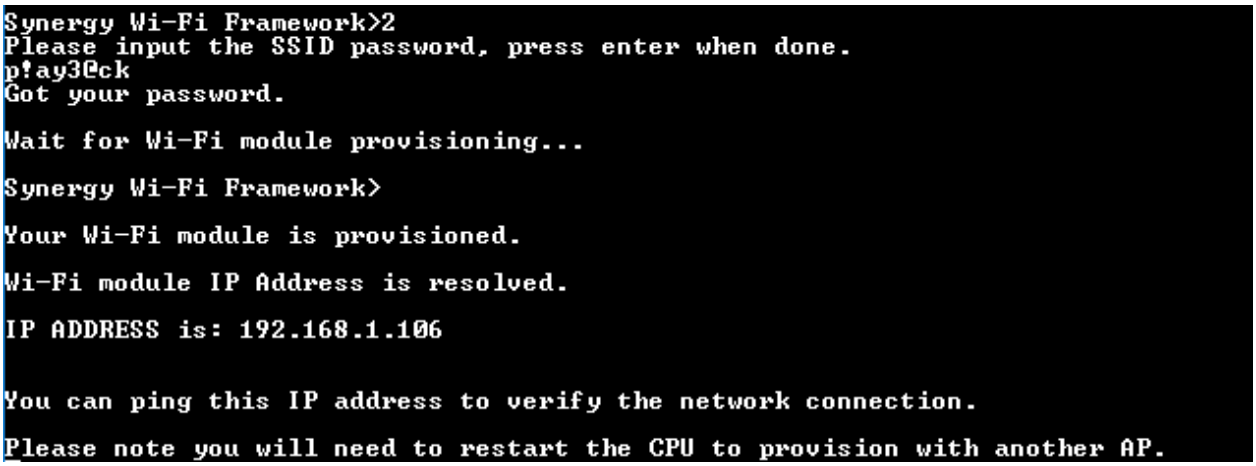

Figure 23 Successful Provisioning

If the provisioning fails, below error message is displayed. Restart the device and enter the correct credentials.

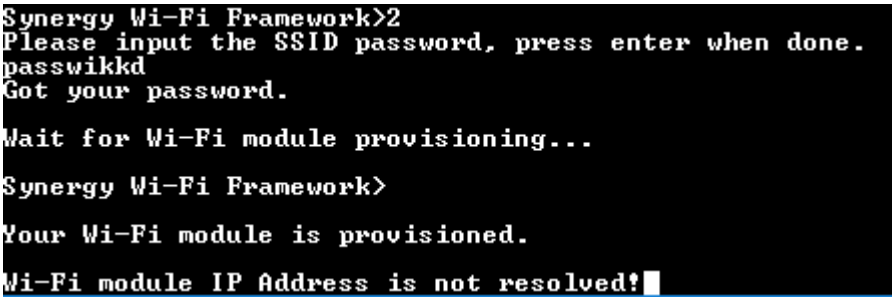

Figure 24 Failed Provisioning

Once the provisioning is successful, ping the IP address displayed for provisioning is successful. In this case, IP address is 192.168.1.106.

To do this, make sure the PC is connected to the same Access Point as the kit.

Open a command window and type the following as shown in Figure below

```
ping <IP address>
```

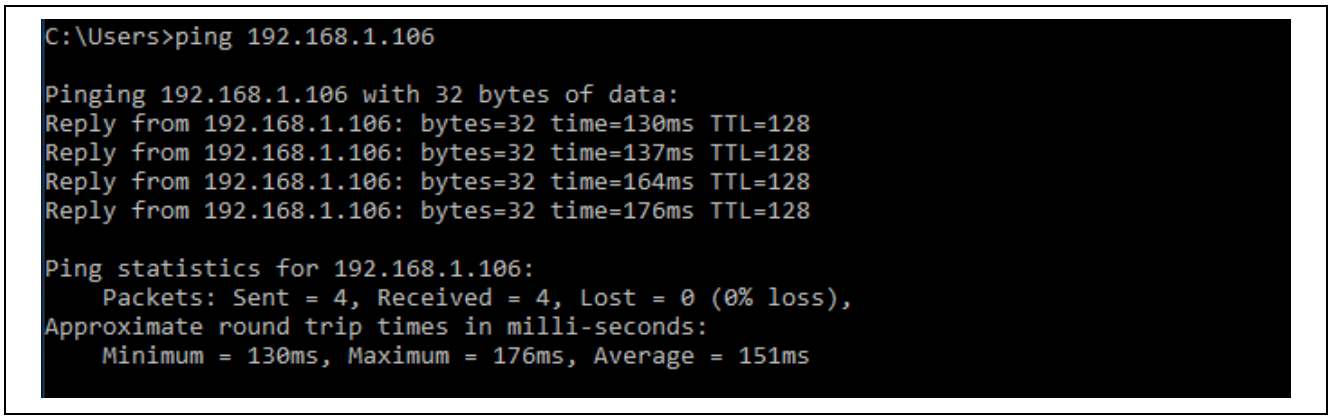

Figure 25 Pinging Wi-Fi Module

Website and Support

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

Synergy Software	renesassynergy.com/software
Synergy Software Package	renesassynergy.com/ssp
Software add-ons	renesassynergy.com/addons
Software glossary	renesassynergy.com/softwareglossary
Development tools	renesassynergy.com/tools
Synergy Hardware	renesassynergy.com/hardware
Microcontrollers	renesassynergy.com/mcus
MCU glossary	renesassynergy.com/mcuglossary
Parametric search	renesassynergy.com/parametric
Kits	renesassynergy.com/kits
Synergy Solutions Gallery	renesassynergy.com/solutionsgallery
Partner projects	renesassynergy.com/partnerprojects
Application projects	renesassynergy.com/applicationprojects
Self-service support resources:	
Documentation	renesassynergy.com/docs
Knowledgebase	renesassynergy.com/knowledgebase
Forums	renesassynergy.com/forum
Training	renesassynergy.com/training
Videos	renesassynergy.com/videos
Chat and web ticket	renesassynergy.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct 1, 2018	—	First release document

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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.2