

# **AN1290: RS9116W Firmware Update Application Note**

Version 2.0

11/09/2020

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>3</b>
<b>2</b>	<b>Block Diagram .....</b>	<b>4</b>
<b>3</b>	<b>RS9116W Firmware Load and Update Process .....</b>	<b>5</b>
<b>4</b>	<b>RS9116W Firmware Update Mechanisms.....</b>	<b>6</b>
4.1	Load Firmware .....	6
4.2	Update Firmware .....	7
<b>5</b>	<b>Guidelines and Recommendations.....</b>	<b>9</b>
<b>6</b>	<b>Load Firmware.....</b>	<b>10</b>
6.1	Firmware Update via Bootloader (SPI) .....	10
6.1.1	Introduction .....	10
6.1.2	Prerequisites .....	10
6.1.3	Setup Diagram .....	10
6.1.4	Flow Diagram .....	11
6.1.5	Configuration and Steps for Execution .....	12
6.2	Firmware Update via Bootloader (UART/USB-CDC) .....	13
6.2.1	Introduction .....	13
6.2.2	Prerequisites .....	13
6.2.3	Setup Diagram .....	13
6.2.4	Steps to Load RS9116W Firmware .....	13
<b>7</b>	<b>Update Firmware .....</b>	<b>19</b>
7.1	Host-Based .....	19
7.1.1	Firmware Update from Host (SD Memory) .....	19
7.1.2	Firmware Update from Host (Remote TCP Server).....	24
7.2	Wireless-Based.....	30
7.2.1	OTAF TCP .....	30
7.2.2	OTAF HTTP/S.....	39
<b>8</b>	<b>Appendix A - Firmware Update Considerations .....</b>	<b>51</b>
<b>9</b>	<b>Appendix B - Host Interfaces .....</b>	<b>52</b>

# 1 Introduction

This document provides details on the various firmware update mechanisms supported by RS9116W and provides steps to execute relevant examples.

Firmware is the software that is embedded into a hardware device. It contains a set of commands that control the behavior of a network device. Whenever a new firmware version is available, it is recommended that users update their devices to the latest version. Complete details about latest firmware will be available in the Release Notes (shared as part of release package), which will help the users decide whether to update to the new firmware or not.

RS9116W releases consists of two main components:

- Firmware
- SAPI Library

Both components have the same version number.

The latest releases have bug-fixes, enhancements, and new features in both 'SAPIs' and 'Firmware' together. Most of the new features have associated APIs, which are part of the latest SAPIs. Hence, it is recommended to update 'SAPIs' and 'Firmware' together.

## Note:

Firmware update feature using AT CMDs or SAPI works for any combination of SAPI and Firmware. For example, SAPI with 1.2.X work with Firmware with 2.X versions for firmware update.

Firmware in the RS9116W device can be updated using the following mechanisms:

### 1. Firmware update via Bootloader

In this mechanism firmware in the device can be updated by the following methods

- Firmware Update via (PC/Laptop) as Host:** Firmware is updated via PC/Laptop connected to the device through UART/USB-CDC interface using Kermit protocol.
- Firmware Update via (MCU) as Host:** Firmware is updated via MCU connected to the device through UART/USB-CDC/SPI/SDIO interface.

### 2. Firmware update via Firmware (Existing Firmware on RS9116W device)

In this mechanism firmware in the device can be updated by the following methods

#### a. Wireless Firmware Update:

In Wireless firmware update, host (PC/MCU) sends request to the device for firmware update. The device then downloads new firmware from a remote server (TCP/HTTP server) or from the cloud and stores it in flash. Upon reboot, the device is updated to the new firmware.

- Wireless Firmware Update (PC as Host): In this mechanism firmware in the device is updated by wireless method (TCP/ HTTP/s) through UART/USB-CDC interface.
- Wireless Firmware Update (MCU as Host): In this mechanism firmware in the device is updated by wireless method (TCP/ HTTP/s) through UART/USB-CDC/SPI/SDIO interface.

#### b. Wired Firmware Update:

In Wired firmware update, application on host (PC/MCU) reads the firmware file stored in the local memory (filesystem in the case of PC and SD card/Flash etc. in the case of MCU) and writes it to the device in chunks. Upon reboot, the device is updated to the new firmware.

- Wired Firmware Update (PC as Host): Firmware in the device is updated by PC through UART/USB-CDC interface.
- Wired Firmware Update (MCU as Host): Firmware in the device is updated by MCU through UART/USB-CDC/SPI/SDIO interface.

## 2 Block Diagram

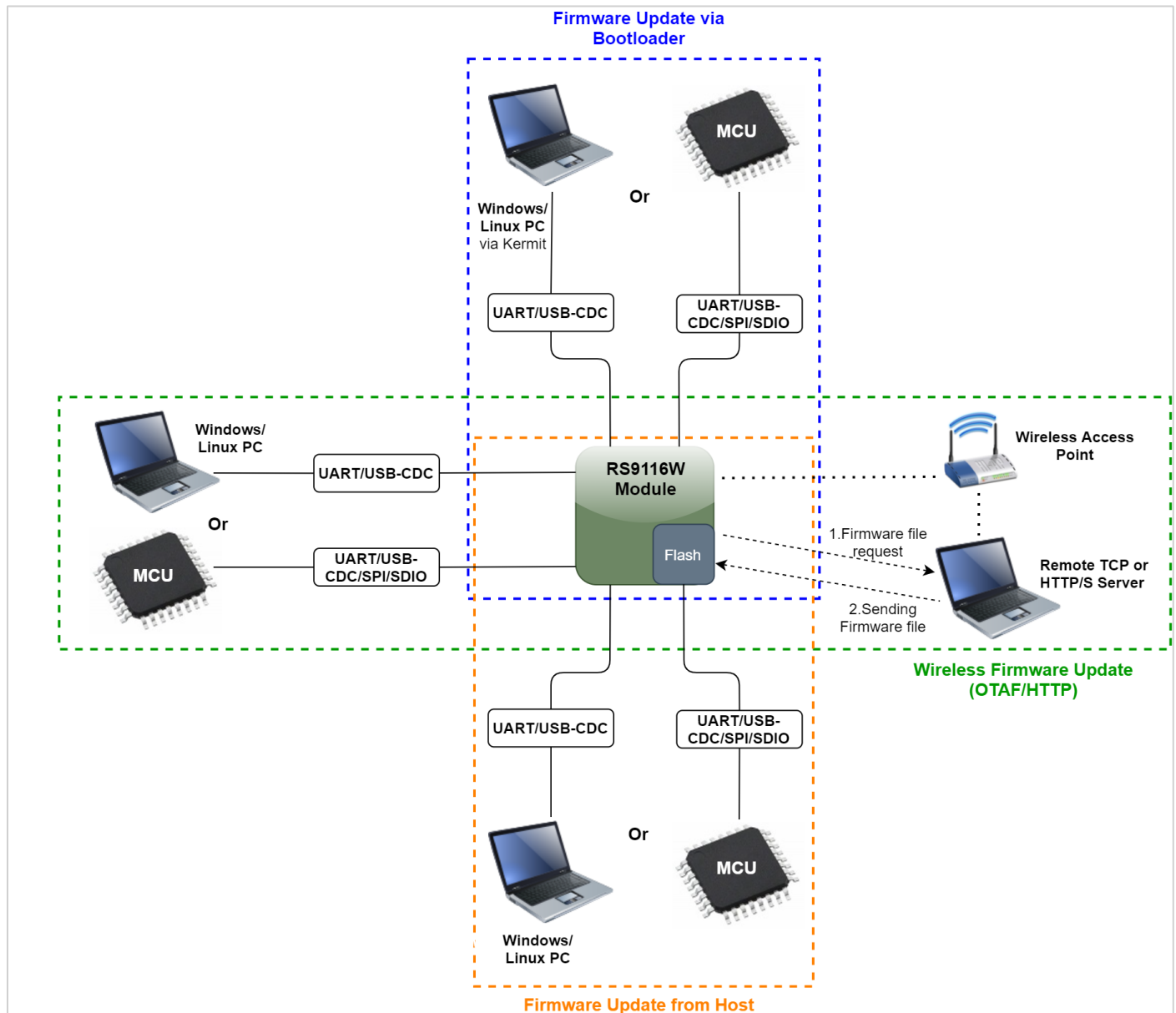


Figure 1: Firmware Update Mechanisms

### Firmware File Format (RPS)

The RPS Format is a binary executable format understood by the Bootloader to perform the required integrity and authenticity checks and load and execute the application.

The Firmware Image in RPS format includes an RPS header, Boot descriptors, Application's binary image and an optional trailer (digital signature).

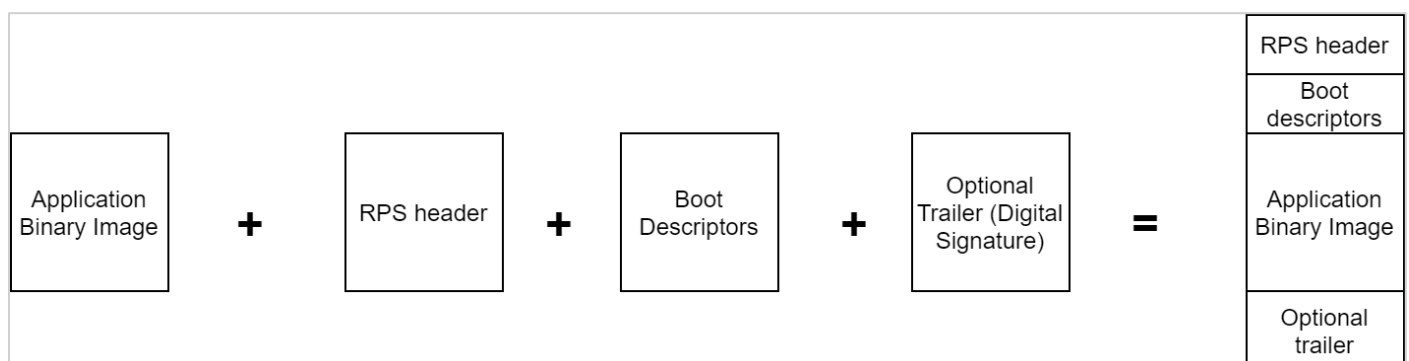


Figure 2: RPS Format

### 3 RS9116W Firmware Load and Update Process

The following figures show basic firmware load and update processes for RS9116W modules. The firmware load process loads firmware onto RS9116W modules for the first time in the case of new modules. Figure 3 shows firmware load process. The Firmware update process updates RS9116W modules with the latest firmware by replacing the firmware already existing in the module. Figure 4 shows the firmware update process.

The steps are as follows:

1. Load initial firmware image into the module via bootloader (for new modules, modules without firmware or modules with default firmware).
2. To update current firmware, download the new firmware file to the module's flash memory from host (MCU/PC) through any host interface (USB, SPI, UART, SDIO), or through OTA process.
3. After reboot, the current firmware is replaced by the new firmware in flash memory, and the module is updated with the new firmware.

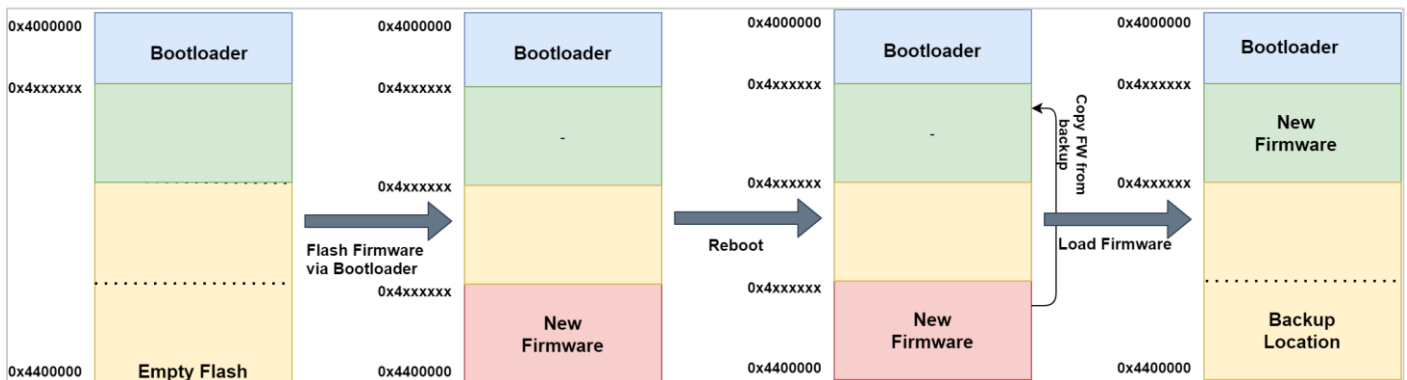


Figure 3: Firmware Load Process (on empty flash)

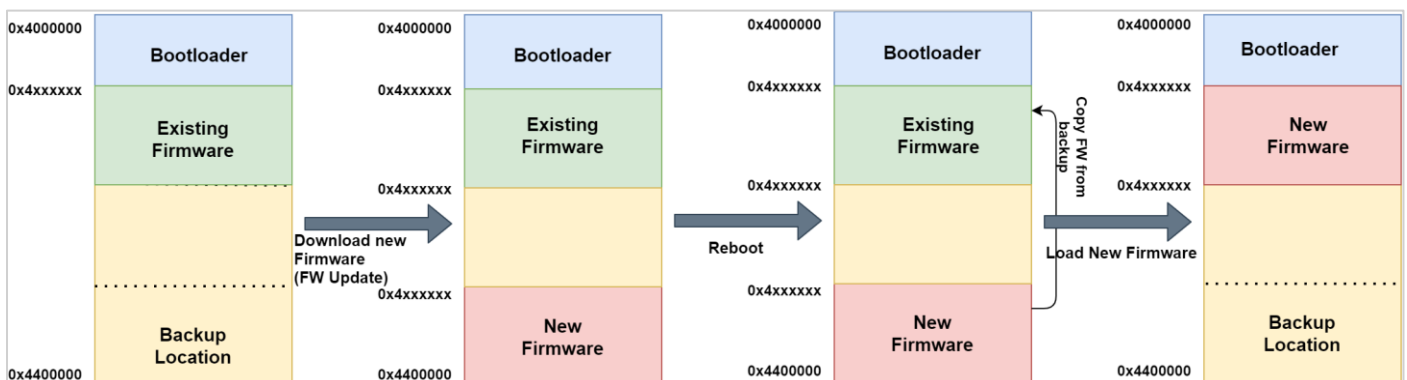


Figure 4: Firmware Update Process

**Note:**

The memory addresses shown in the above diagrams are only for reference purposes.

## 4 RS9116W Firmware Update Mechanisms

Firmware update mechanisms are categorized into two:

- Load Firmware
- Update Firmware

The following subsections provide descriptions for both.

### 4.1 Load Firmware

To load desired the firmware on new modules (with empty flash or default firmware), the user should use the following mechanism. This is mainly followed during manufacturing, design and development phases. In this mechanism, host issues commands to bootloader for loading firmware on to RS9116W module. The following sections provide examples describing this mechanism with UART/USB-CDC and SPI interfaces.

#### 1. [Firmware Update via Bootloader \(SPI\)](#)

This mechanism allows the user to update the firmware in a RS9116W device via bootloader from host using SD card through SPI interface.

#### 2. [Firmware Update via Bootloader \(UART/USB-CDC\)](#)

This mechanism allows the user to update the firmware in a RS9116W device via bootloader using a serial terminal like Tera term which acts like a host and issues bootloader commands to load firmware.

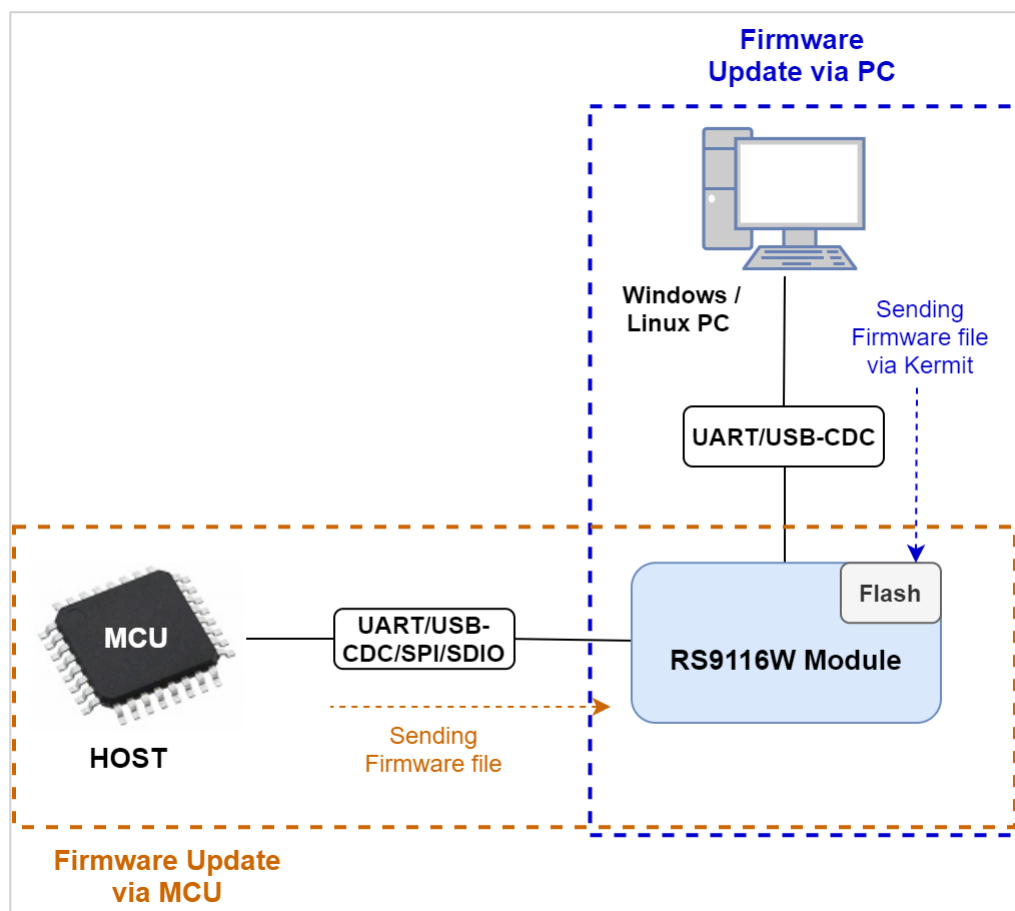


Figure 5: Firmware Update via Bootloader

#### Note:

If the firmware update is via host through UART/USB CDC, then Kermit has to be ported on to host MCU.

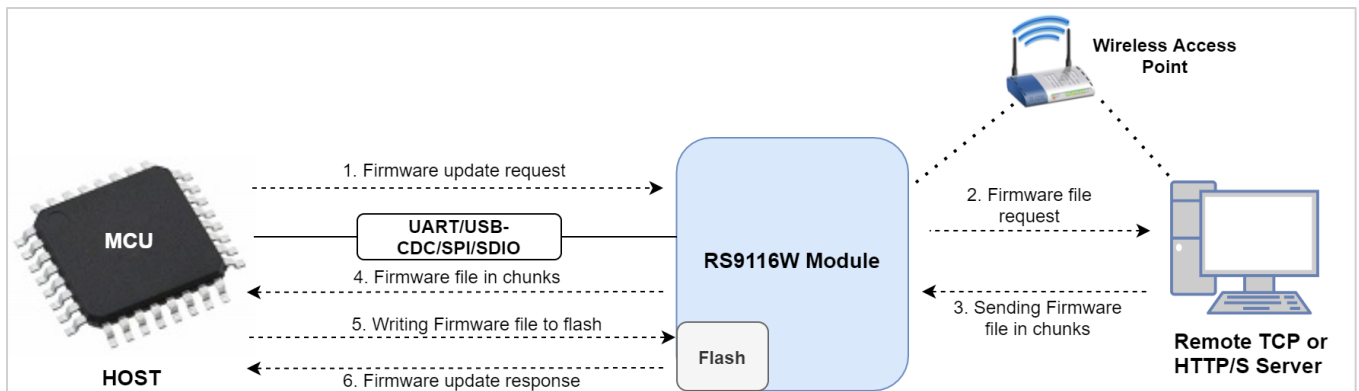
## 4.2 Update Firmware

To update existing firmware with new/latest firmware, users can use any of the following mechanisms

### Host based

In this category, new firmware to be updated will be stored on host and host will update this firmware by sending it chunk-by-chunk to the module. The following examples demonstrate this mechanism.

1. [Firmware Update from Host \(Flash or SD Memory\)](#)  
In this example, new firmware to be updated is stored on SD card mounted on to the Host which is interfaced to RS9116W over SPI. Application on host reads the firmware on SD card and writes to the module chunk by chunk.
2. [Firmware Update from Server](#)  
In this example, new firmware to be updated is received to host from a remote TCP server. Application on host sends the received firmware chunks to the module.



**Figure 6: Firmware Update via Host from Remote Server**

### Wireless-based

In this category, new firmware will be downloaded from a remote server (TCP/HTTP server) or from the cloud. Here, module will take request from host for update, and the remaining processes will be executed by the module. The module downloads the new firmware to a temporary location in module flash, and the current firmware will be updated with this new version once the host resets the module. The following examples demonstrate this mechanism.

1. [OTAF TCP](#)  
In this example, new firmware is downloaded to a temporary location in module flash from a remote TCP server. This process is initiated when module receives OTA update request from host. Once specified firmware is downloaded and verified, success response is indicated to the host.
2. [OTAF HTTP/S](#)  
In this example, new firmware is downloaded to a temporary location in module flash from a remote HTTP/S server. This process is initiated when module receives OTA update request from host. Once specified firmware is downloaded and verified, success response is indicated to the host.

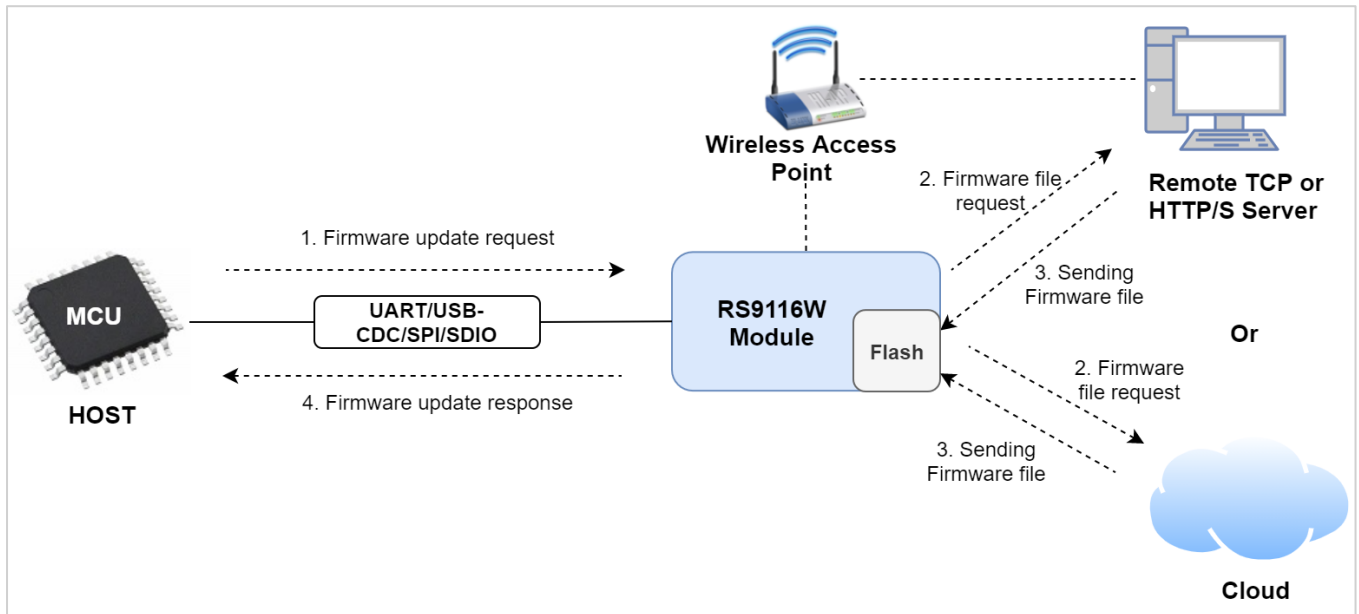


Figure 7: Wireless Firmware Update

**Note:**

To update existing firmware with new/latest firmware, user can also use Load Firmware (Firmware update via bootloader) mechanism.

The following table shows the various available firmware update mechanisms and supported interfaces,

S. No	Interface	Firmware Update Mechanisms		
		via Bootloader	Host based (FW stored on Host)	Wireless based (FW stored on Cloud/Remote Sever)
1	UART	Y	Y	Y
2	USB-CDC	Y	Y	Y
3	USB	Y	Y	Y
4	SPI	Y*	Y*	Y*
5	SDIO	Y*	Y*	Y

**Note:**

Reference examples are provided only for cases marked with \*.



## 5 Guidelines and Recommendations

1. It is strongly recommended to use the same version of firmware and SAPIs, especially for all MAJOR releases, as this have many changes compared to MINOR releases.
2. The user should first update the firmware and then SAPI. By doing so the existing firmware should be capable of performing the update.
3. The RS9116W module should be kept powered on during the complete firmware update process.
4. The module should not be placed in power save mode during any portion of the firmware update process.
5. Post successful firmware update, the user should update SAPI Library and related application changes if any exist on his MCU code.

## 6 Load Firmware

The examples below showcase the loading of firmware into RS9116W devices via bootloader using the different interfaces available for this purpose.

S. No	Example	Description	Example Source Path
1	Firmware Update via Bootloader (SPI)	This application demonstrates how to update the firmware of a RS9116W device via bootloader from the host using SD card through SPI interface	RS9116.NB0.WC.GENR.OSI.x.x.x\host\sapis\examples\firmware_update_via_bootloader
2	Firmware Update via Bootloader (UART/USB-CDC)	This application demonstrates how to update the firmware of a RS9116W device via bootloader through UART/USB-CDC interface	NA

### 6.1 Firmware Update via Bootloader (SPI)

#### 6.1.1 Introduction

This application demonstrates how to load firmware onto a RS9116W device via bootloader from host. In this example NXP FRDM-K28F is used as host and is interfaced to RS9116W EVK via SPI. Firmware to be loaded is stored on SD card mounted on host.

The application on host issues bootloader commands to load the new firmware and then fetches firmware chunk from SD card and sends it to RS9116W over SPI. The host keeps sending FW chunks until the last chunk is sent. At that point the module verifies the received data and returns a success message.

#### 6.1.2 Prerequisites

- RS9116W module
- NXP FRDM-K28F board
- SD card
- SPI connector

#### Example Setup

Connect the RS9116W with host (NXP\_FRDM\_K28F) using SPI interface. For connections, refer to 'Pin Connections' section in **RS9116W\_NXP\_FRDM-K28F User Guide.pdf**.

#### 6.1.3 Setup Diagram

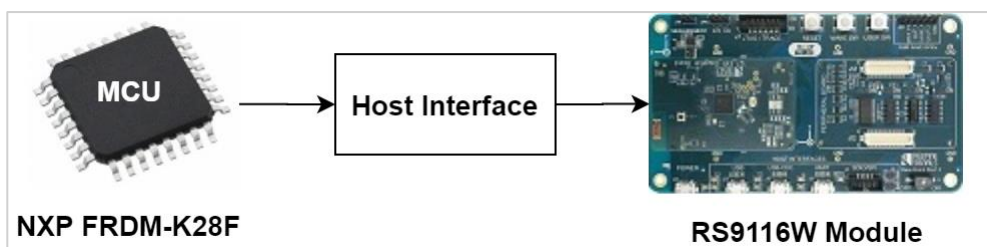
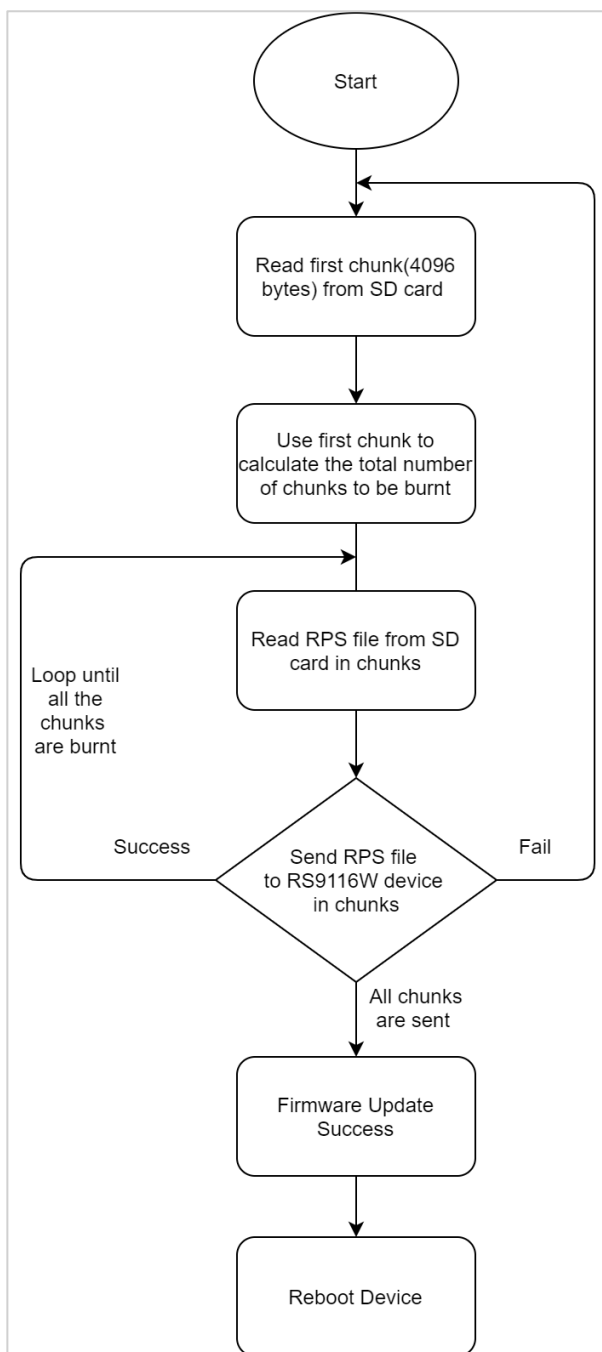


Figure 8: Setup diagram for Firmware Update Via Bootloader

## 6.1.4 Flow Diagram



**Figure 9: Flow Diagram**

## 6.1.5 Configuration and Steps for Execution

### 6.1.5.1 Application Configuration

Open **rsi\_wlan\_config.h** file and update/modify the following macros:

```
#define RSI_FEATURE_BIT_MAP                (FEAT_ULP_GPIO_BASED_HANDSHAKE |
FEAT_DEV_TO_HOST_ULP_GPIO_1 | FEAT_SECURITY_OPEN)
#define RSI_TCP_IP_BYPASS                  RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP        (TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_FEAT_SSL |
TCP_IP_FEAT_DNS_CLIENT | TCP_IP_FEAT_EXTENSION_VALID)
#define RSI_CUSTOM_FEATURE_BIT_MAP        FEAT_CUSTOM_FEAT_EXTENSION_VALID
#if ENABLE_1P8V
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP    (EXT_FEAT_LOW_POWER_MODE |
EXT_FEAT_XTAL_CLK_ENABLE | EXT_FEAT_384K_MODE | EXT_FEAT_1P8V_SUPPORT)
#else
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP    (EXT_FEAT_LOW_POWER_MODE |
EXT_FEAT_XTAL_CLK_ENABLE | EXT_FEAT_384K_MODE)
#endif
#define RSI_EXT_TCPIP_FEATURE_BITMAP      (EXT_TCP_IP_WAIT_FOR_SOCKET_CLOSE |
EXT_DYNAMIC_COEX_MEMORY | EXT_TCP_IP_WINDOW_DIV | EXT_TCP_IP_TOTAL_SELECTS_4 |
EXT_TCP_IP_BI_DIR_ACK_UPDATE )
```

#### Note:

**rsi\_wlan\_config.h** file is already set with desired configuration in its respective example folders. The user does not need to change it for each example.

### Application Execution

1. Copy new firmware file (**RS9116.NB0.WC.GENR.OSI.x.x.x.rps**) to SD card and rename it as RS9116.rps. Firmware file is available in release package in the following path:  
RS9116.NB0.WC.GENR.OSI.x.x.x\firmware\RS9116.NB0.WC.GENR.OSI.x.x.x.rps.
2. Insert SD card into host's (FRDM-K28F) SD card slot.
3. Connect the host to RS9116 EVK using SPI interface.
4. Import application settings from the file mentioned in the path below  
RS9116.NB0.WC.GENR.OSI.x.x.x\host\platforms\NXP\_FRDM\_K28\RSI\_FWUP\_VIA\_BOOTLOADER
5. Build and run the application. For details on how to Import, build and run the application, refer to RS9116W\_NXP\_FRDM-K28F User Guide.pdf.
6. Observe the output on console as shown below

```
DEMO STARTED
.....
.....
.....
.....
.....fw_update Success
fw version after update is: 1610.2.0.0.0022
DEMO DONE
```

Firmware version number shown above is just for reference purpose.

## 6.2 Firmware Update via Bootloader (UART/USB-CDC)

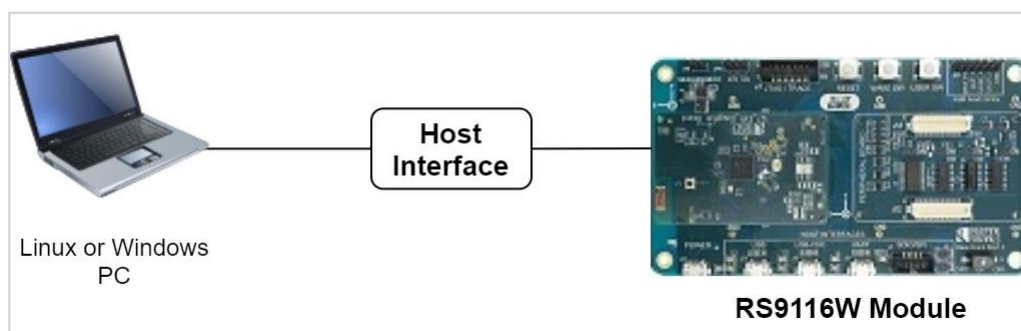
### 6.2.1 Introduction

This section briefs about Firmware loading process in RS9116W module via bootloader. In this process, a serial terminal like Tera Term or CuteCom acts like a host to RS9116W and issues bootloader commands to load firmware. Here, RS9116W EVK is connected to PC (Windows/Linux) via USB-CDC interface.

### 6.2.2 Prerequisites

- RS9116W EVK
- USB Cable
- Windows PC with Tera Term (Download Tera Term from <http://gensho.ftp.acc.umu.se/mirror/osdn.net/ttssh2/68719/teraterm-4.97.exe>) (or)
- Linux PC with Kermit/CuteCom

### 6.2.3 Setup Diagram

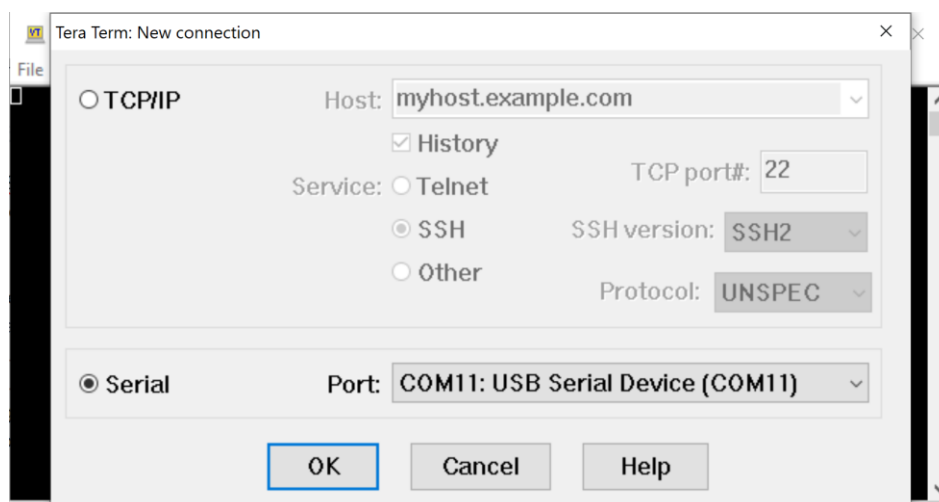


**Figure 10 Setup Diagram**

### 6.2.4 Steps to Load RS9116W Firmware

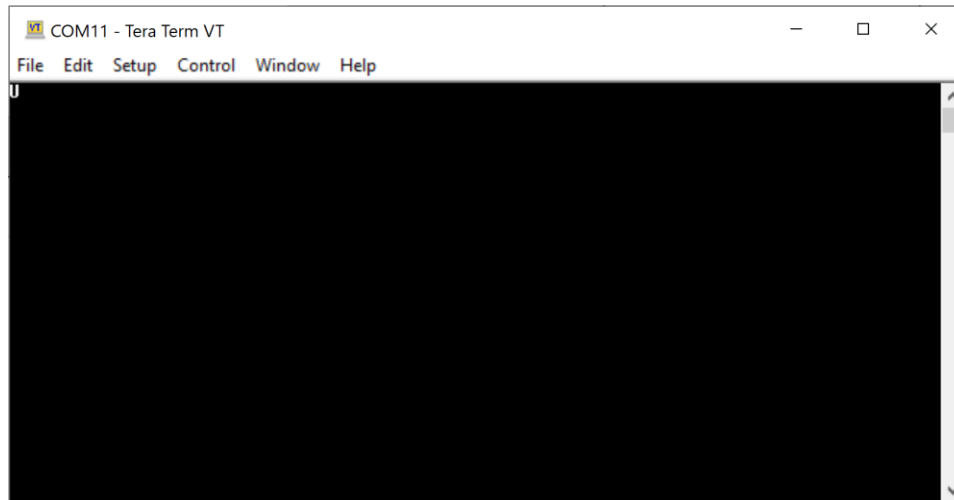
#### 6.2.4.1 Windows

1. Connect USB cable to on board USB-CDC connector (install windows driver available in **RS9116.NB0.WC.GENR.OSI.X.X.X\Utils\usb\_cdc** folder to enumerate RSI WSC Virtual Com Port).
2. Open Tera Term or any Kermit application on Windows.
3. Select respective COM port as shown in below figure



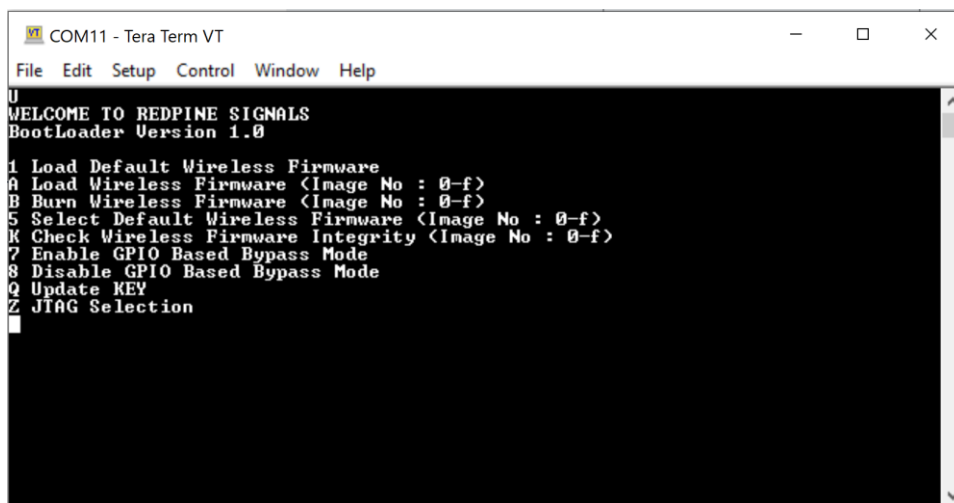
**USB CDC Serial Selection**

4. In open terminal send "|", the board responds with "U".



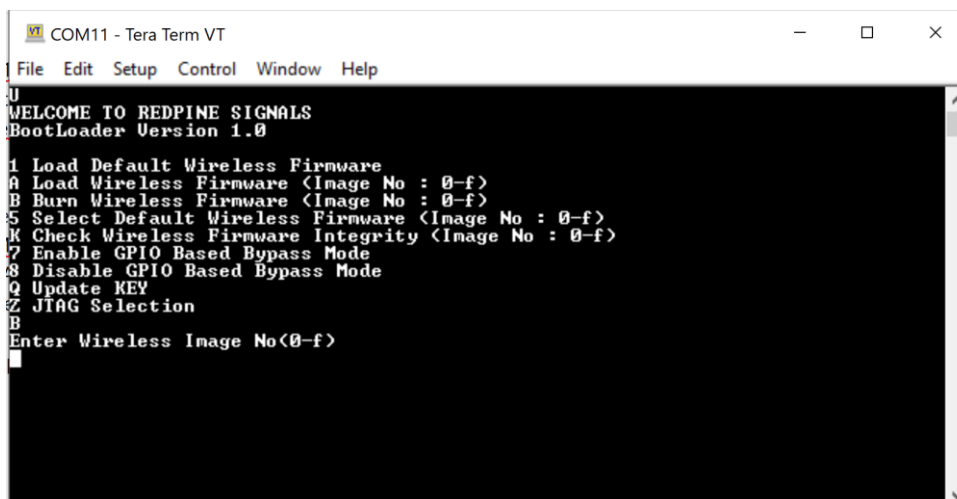
**Response for '|'**

5. Enter "U". After doing so, the bootloader option menu will be shown as illustrated in the following figure:



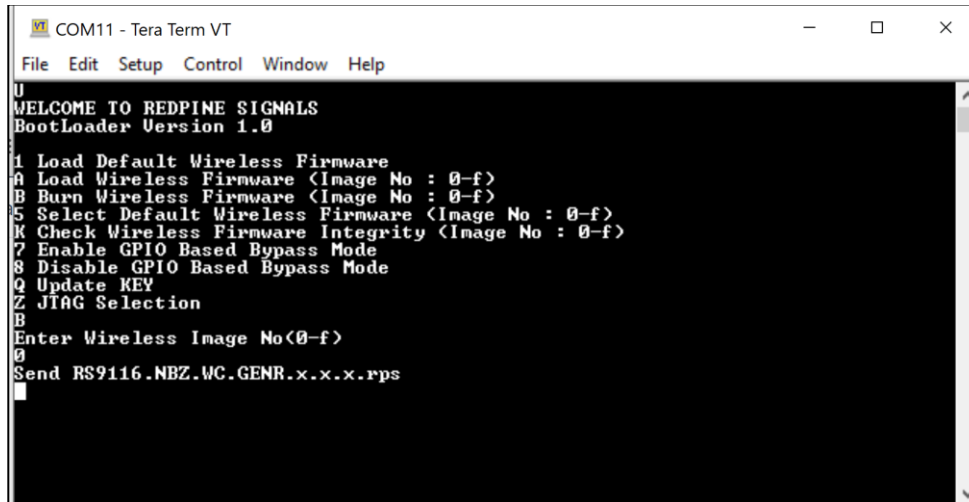
**Bootloader Options**

6. Enter "B" to burn new firmware.



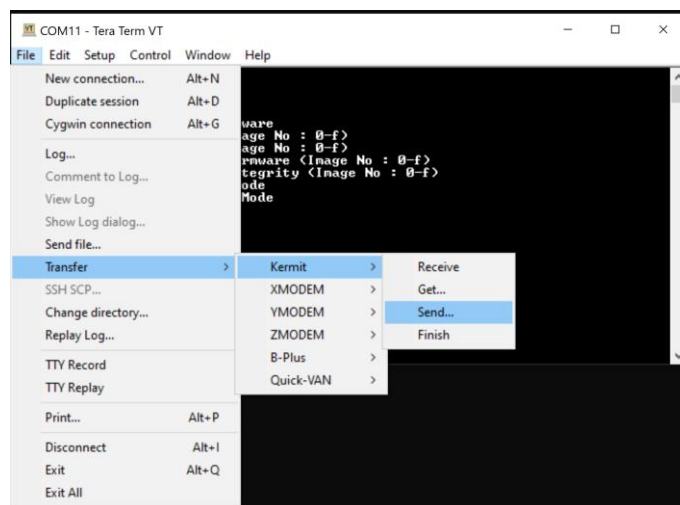
**Burn New Firmware**

- Then enter "0" to select section in flash to update image.



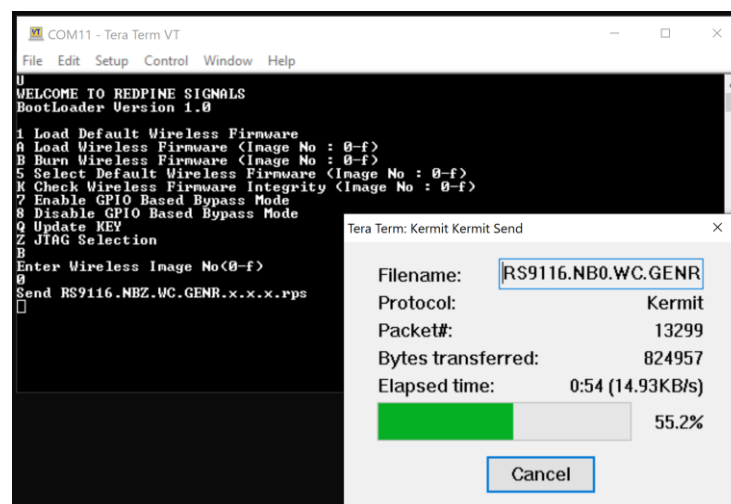
Firmware Image Number Selection

- Navigate to **File->Transfer->Kermit->Send** and select image to update.



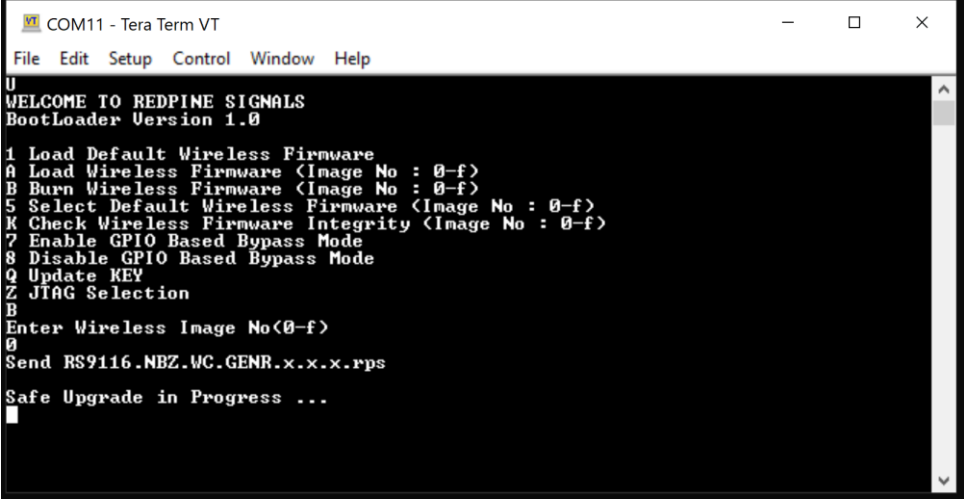
Send Firmware using Kermit

- Process takes a few minutes to update the firmware.



Sending Firmware

10. Once sending firmware is complete, bootloader starts updating the firmware.

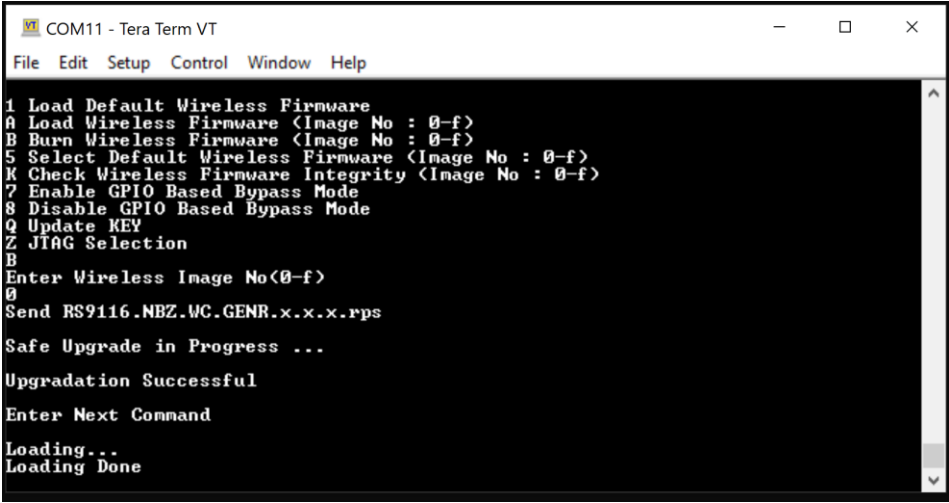


```

COM11 - Tera Term VT
File Edit Setup Control Window Help
U
WELCOME TO REDPINE SIGNALS
BootLoader Version 1.0
1 Load Default Wireless Firmware
A Load Wireless Firmware <Image No : 0-f>
B Burn Wireless Firmware <Image No : 0-f>
5 Select Default Wireless Firmware <Image No : 0-f>
K Check Wireless Firmware Integrity <Image No : 0-f>
7 Enable GPIO Based Bypass Mode
8 Disable GPIO Based Bypass Mode
Q Update KEY
Z JTAG Selection
B
Enter Wireless Image No<0-f>
0
Send RS9116.NBZ.WC.GENR.x.x.x.rps
Safe Upgrade in Progress ...
  
```

**Firmware Update in Progress**

11. Once the process is completed message appears saying "**Upgradation Successful**". Enter "1" to load updated firmware.



```

COM11 - Tera Term VT
File Edit Setup Control Window Help
1 Load Default Wireless Firmware
A Load Wireless Firmware <Image No : 0-f>
B Burn Wireless Firmware <Image No : 0-f>
5 Select Default Wireless Firmware <Image No : 0-f>
K Check Wireless Firmware Integrity <Image No : 0-f>
7 Enable GPIO Based Bypass Mode
8 Disable GPIO Based Bypass Mode
Q Update KEY
Z JTAG Selection
B
Enter Wireless Image No<0-f>
0
Send RS9116.NBZ.WC.GENR.x.x.x.rps
Safe Upgrade in Progress ...
Upgradation Successful
Enter Next Command
Loading...
Loading Done
  
```

**Firmware Loaded Successful**

12. Remove USB-CDC power supply to board then switch off the ISP.

13. Power up the board and continue.



#### 6.2.4.2 Linux

1. Connect the USB cable to on board USB-CDC connector. In PC terminal type "**dmesg -c**" to find the enumerated serial port.

```
[root@lapt224 ~]# dmesg -c
[ 5011.460331] usb 2-1: USB disconnect, device number 16
[ 5011.905322] usb 2-1: new high-speed USB device number 17 using xhci_hcd
[ 5012.073852] usb 2-1: New USB device found, idVendor=1618, idProduct=9116
[ 5012.073857] usb 2-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 5012.073861] usb 2-1: Product: Wireless USB-CDC Network Module
[ 5012.073863] usb 2-1: Manufacturer: Redpine Signals, Inc.
[ 5012.074604] cdc_acm 2-1:1.0: ttyACM0: USB ACM device
[root@lapt224 ~]# gvim ~/.kermrc
```

2. Open the ".kermrc" file from root directory (**gvim ~/.kermrc**)
  - a. Edit set line **/dev/tty** with enumerated serial port in step 1 after dmesg -c, as shown in below figure.
  - b. Edit "**set speed**" to 921600 baud rate
  - c. Save and exit the file (esp → colon → wq)

```
1 set modem type none
2 #set line /dev/ttyS0
3 set line /dev/ttyACM0
4 #set line /dev/ttyUSB0
5 #set speed 460800
6 set speed 921600
7 set carrier-watch off
8 set handshake none
9 set flow-control none
10 #set stop-bit 2
11 robust
12 set receive packet-length 100
13 set send packet-length 4000
14 set window 10
15 set file type binary
16 set file names litera
```

3. From terminal give command "**kermi**" and then "**Ctrl + \ + c**" to continue. Type "**|**" for boot message "**U**". Then type "**U**" for boot menu. Type "**B**" to select 'burn new firmware' option and then type "**0**" to select firmware image number.

```
[root@lapt224 test]# kermi
C-Kermit 9.0.302 OPEN SOURCE:., 20 Aug 2011, for Linux
Copyright (C) 1985, 2011,
Trustees of Columbia University in the City of New York.
Type ? or HELP for help.
(/work/test/) C-Kermit>c
Connecting to /dev/ttyACM0, speed 921600
Escape character: Ctrl-\ (ASCII 28, FS): enabled
Type the escape character followed by C to get back,
or followed by ? to see other options.
-----
U
WELCOME TO REDPINE SIGNALS
BootLoader Version 1.0

1 Load Default Wireless Firmware
A Load Wireless Firmware (Image No : 0-f)
B Burn Wireless Firmware (Image No : 0-f)
5 Select Default Wireless Firmware (Image No : 0-f)
K Check Wireless Firmware Integrity (Image No : 0-f)
7 Enable GPIO Based Bypass Mode
8 Disable GPIO Based Bypass Mode
Q Update KEY
Z JTAG Selection
B
Enter Wireless Image No(0-f)
0
Send RS9116.NBZ.WC.GENR.x.x.x.rps
```

4. Give "**Ctrl + \ + c**" to exit from kermit. From terminal give "**send firmware/RS9116.NB0.WC.GENR.OSI.x.x.x.rps**" which starts to burn the image into RS9116W.

```
C-Kermit 9.0.302 OPEN SOURCE:, 20 Aug 2011, for Linux
Copyright (C) 1985, 2011,
Trustees of Columbia University in the City of New York.
Type ? or HELP for help.
(/work/test/) C-Kermit>send RS9116.NB0.WC.GENR.OSI.x.x.x.rps
```

```
C-Kermit 9.0.302 OPEN SOURCE:, 20 Aug 2011, lapt224 [192.168.1.109]

Current Directory: /work/test
Communication Device: /dev/ttyACM0
Communication Speed: 921600
Parity: none
RTT/Timeout: 01 / 02
SENDING: => RS9116.NB0.WC.GENR.OSI.x.x.x.rps
File Type: BINARY
File Size: 1496960
Percent Done: 16  //////////
...10...20...30...40...50...60...70...80...90...100
Estimated Time Left: 00:00:27
Transfer Rate, CPS: 47898
Window Slots: 1 of 1
Packet Type: D
Packet Count: 234
Packet Length: 1520
Error Count: 0
Last Error:
Last Message:
```

X to cancel file, Z to cancel group, <CR> to resend last packet,  
E to send Error packet, ^C to quit immediately, ^L to refresh screen.

5. Once firmware file sending is completed, type "**Ctrl + \ + c**" to continue to check update sequence. Once update is successful give "**Ctrl + \ + c**" to exit from kermit and "**q**" to continue on to terminal.

```
Safe Upgrade in Progress ...
```

```
Upgradation Successful
```

```
Enter Next Command
```

```
Loading...
```

```
Loading Done
```

```
(Back at lapt224)
```

```
-----
(/work/test/) C-Kermit>q
```

## 7 Update Firmware

This section provides examples for updating new firmware for RS9116W using SAPIs. Here there are two different ways of updating new firmware,

### Host-based

In this mechanism, host sends new firmware to RS9116W over supported interfaces like SPI/UART/USB-CDC/SDIO to RS9116W using firmware update APIs provided in SAPI library.

### Wireless-based

In this mechanism, host calls OTA APIs, which directly download (from remote server) the new firmware into RS9116W. Once this downloaded firmware is verified, RS9116W responds the relevant status like SUCCESS or FAILURE to host.

### 7.1 Host-Based

Following examples are described in this section.

S. No	Example	Description	Example Source Path
1	Firmware Update from Host	This application demonstrates how to update the firmware to RS9116W device from external SD card connected to host. New firmware is stored on SD card which is mounted on host.	RS9116.NB0.WC.GENR.OSI.x.x.x\host\sapis\examples\firmware_update_from_host
2	Firmware Update from Server	This application demonstrates how to update new firmware to RS9116W device using remote TCP server. New firmware is downloaded to host from remote TCP server.	RS9116.NB0.WC.GENR.OSI.x.x.x\host\sapis\examples\wlan\firmware_update

#### 7.1.1 Firmware Update from Host (SD Memory)

##### 7.1.1.1 Introduction

This section demonstrates how to update firmware into RS9116W module from flash or an external SD card connected to host. Here, new firmware is stored in SD card.

Host reads the new firmware and sends it to RS9116W chunk by chunk using firmware update APIs provided in SAPI.

##### 7.1.1.2 Prerequisites

- RS9116W module
- NXP FRDM-K28F board
- SD card
- SPI connector

### 7.1.1.3 Setup Diagram

Connect the RS9116W with host (NXP\_FRDM-K28F) using SPI interface. For connections, refer to 'Pin Connections' section in **RS9116W\_NXP\_FRDM-K28F User Guide.pdf**.

Setup Diagram

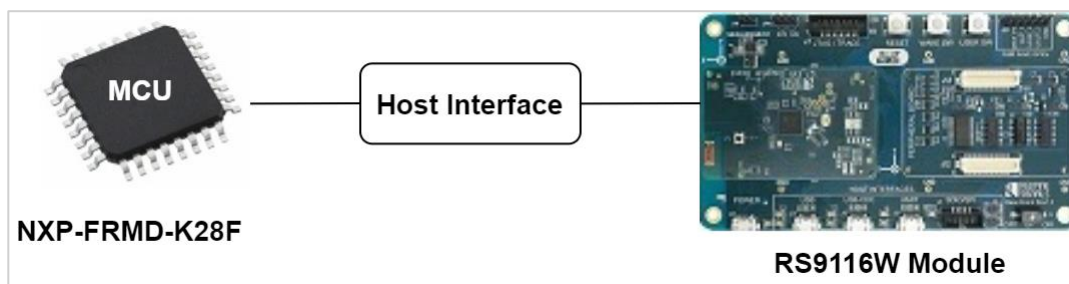
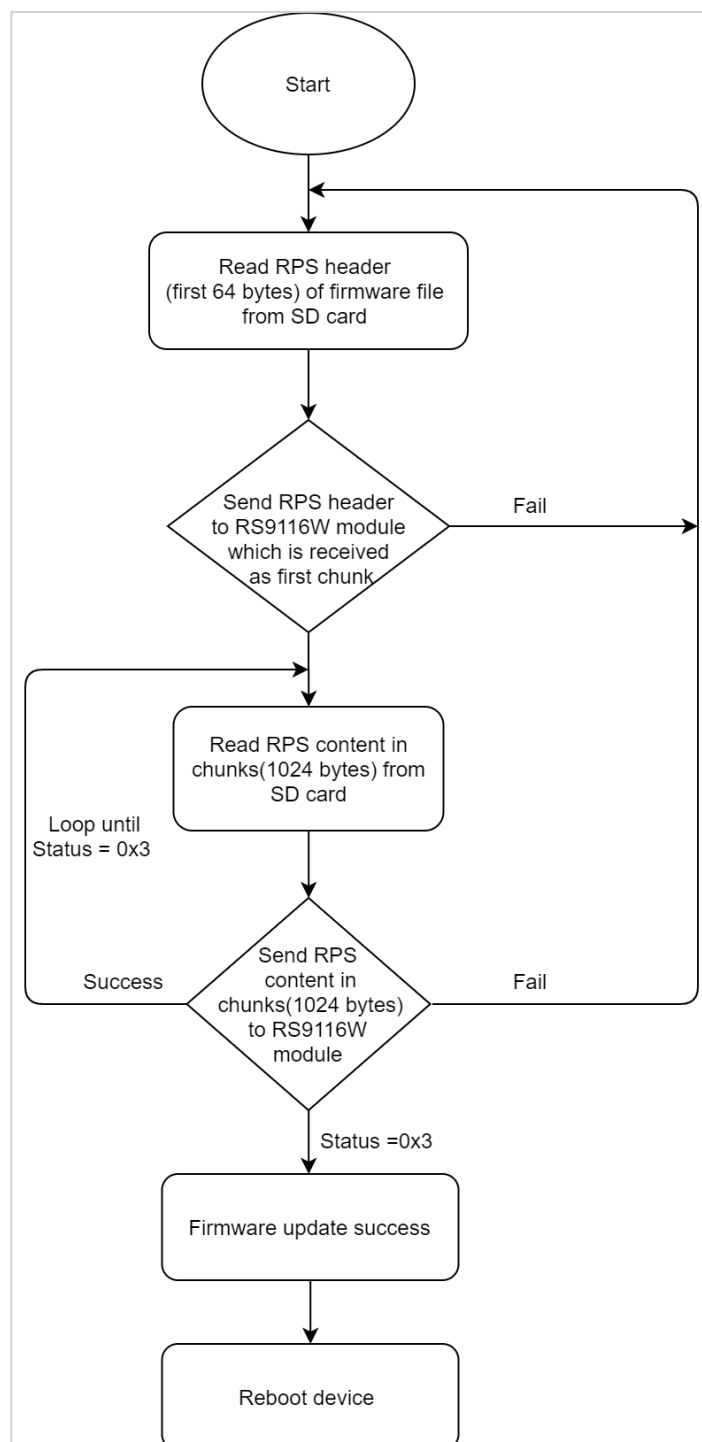


Figure 11: Firmware Update from Host Setup Diagram

#### 7.1.1.4 Flow Diagram

The following flow chart shows the firmware update flow of this application example.



**Figure 12: Flow Diagram**

#### 7.1.1.5 Description

This application demonstrates user how to update the firmware into RS9116W module from an external SD card connected to host. In this application, the device connects to host through SPI interface. Application loads the firmware file from SD card into device using firmware update API. After successful firmware update, firmware update API returns 0x03 response.

### 7.1.1.6 Configuration and Steps for Execution

#### 7.1.1.6.1 Application Configuration

Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define RSI_FEATURE_BIT_MAP (FEAT_ULP_GPIO_BASED_HANDSHAKE |
FEAT_DEV_TO_HOST_ULP_GPIO_1 | FEAT_SECURITY_OPEN)
#define RSI_TCP_IP_BYPASS RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_FEAT_SSL |
TCP_IP_FEAT_DNS_CLIENT | TCP_IP_FEAT_EXTENSION_VALID)
#define RSI_CUSTOM_FEATURE_BIT_MAP FEAT_CUSTOM_FEAT_EXTENSION_VALID
#if ENABLE_1P8V
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP (EXT_FEAT_LOW_POWER_MODE |
EXT_FEAT_XTAL_CLK_ENABLE | EXT_FEAT_384K_MODE | EXT_FEAT_1P8V_SUPPORT)
#else
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP (EXT_FEAT_LOW_POWER_MODE |
EXT_FEAT_XTAL_CLK_ENABLE | EXT_FEAT_384K_MODE)
#endif
#define RSI_EXT_TCPIP_FEATURE_BITMAP (EXT_TCP_IP_WAIT_FOR_SOCKET_CLOSE |
EXT_DYNAMIC_COEX_MEMORY | EXT_TCP_IP_WINDOW_DIV | EXT_TCP_IP_TOTAL_SELECTS_4 |
EXT_TCP_IP_BI_DIR_ACK_UPDATE )
```

#### Note:

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders, user need not change for each example.

#### 7.1.1.6.2 Application Execution

1. Copy new firmware file (**RS9116.NB0.WC.GENR.OSI.x.x.x.rps**) to SD card and rename it as RS9116.rps.  
Firmware file is available in release package in the path  
RS9116.NB0.WC.GENR.OSI.x.x.x\firmware\RS9116.NB0.WC.GENR.OSI.x.x.x.rps
2. Insert SD card into host's (FRDM-K28F) SD card slot.
3. Connect the host to RS9116 EVK using SPI interface.
4. Import application settings from the file mentioned in the path below  
**RS9116.NB0.WC.GENR.OSI.x.x.x\host\platforms\NXP\_FRDM\_K28\RSI\_FWUP\_FROM\_HOST**
5. Build and run the application. For details on how to Import, build and run the application, refer to RS9116W\_NXP\_FRDM-K28F User Guide.pdf.
6. Observe the output on console as shown below

[illegible]

**Note:**

Firmware version number shown above is just for reference purpose.

**Note:**

After successful firmware update, reset RS9116W module to get loaded with new firmware. Device takes about 40 seconds to give CARD READY indication after first reboot, so wait for ~40 seconds after resetting the device.

## Assumptions

S. No	Item	Description
1	HOST should check availability of new firmware	Host should check the availability of new firmware, then proceed with update
2	Firmware version check	Assume Host will check firmware version before and after firmware update
3	Module to download and store	Module is responsible for downloading the firmware image and storing it to backup location
4	HOST to Reset the Module	HOST is responsible for resetting the Module to LOAD downloaded Firmware

## Limitations

S. No	Constraints	Remarks
1	Only RPS format supported	Firmware to be updated should be in RPS format

## 7.1.2 Firmware Update from Host (Remote TCP Server)

### 7.1.2.1 Introduction

This application demonstrates how to update new firmware in RS9116W devices using a remote TCP server. Here, new firmware to be updated is downloaded to host from a remote TCP Server. The host then sends firmware chunks to RS9116W using firmware update APIs.

### 7.1.2.2 Prerequisites

- Windows / Linux PC with Host interface (UART/ USB-CDC/ SPI/ USB/ SDIO) in case of WiSeConnect
- RS9116W Module
- Wireless Access point
- Linux PC with TCP server application (TCP server application is provided as part of release package)

### 7.1.2.3 Setup Diagram

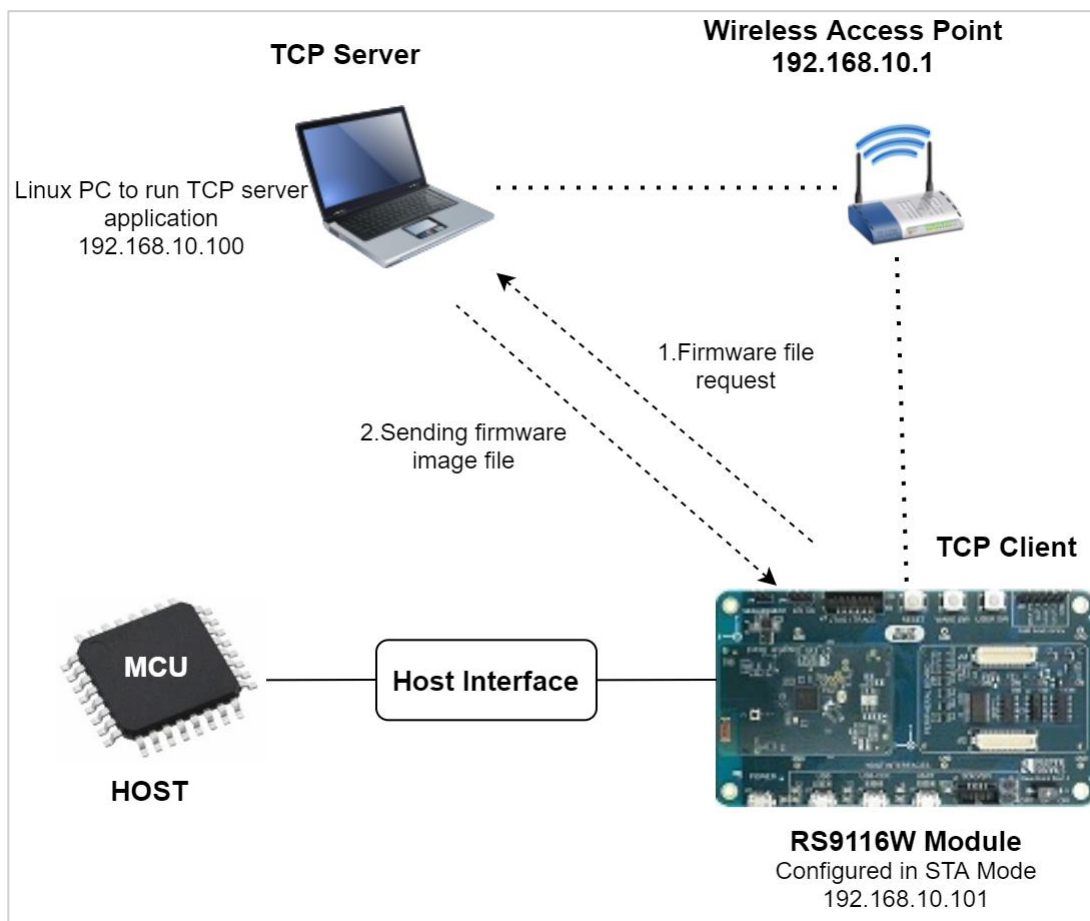
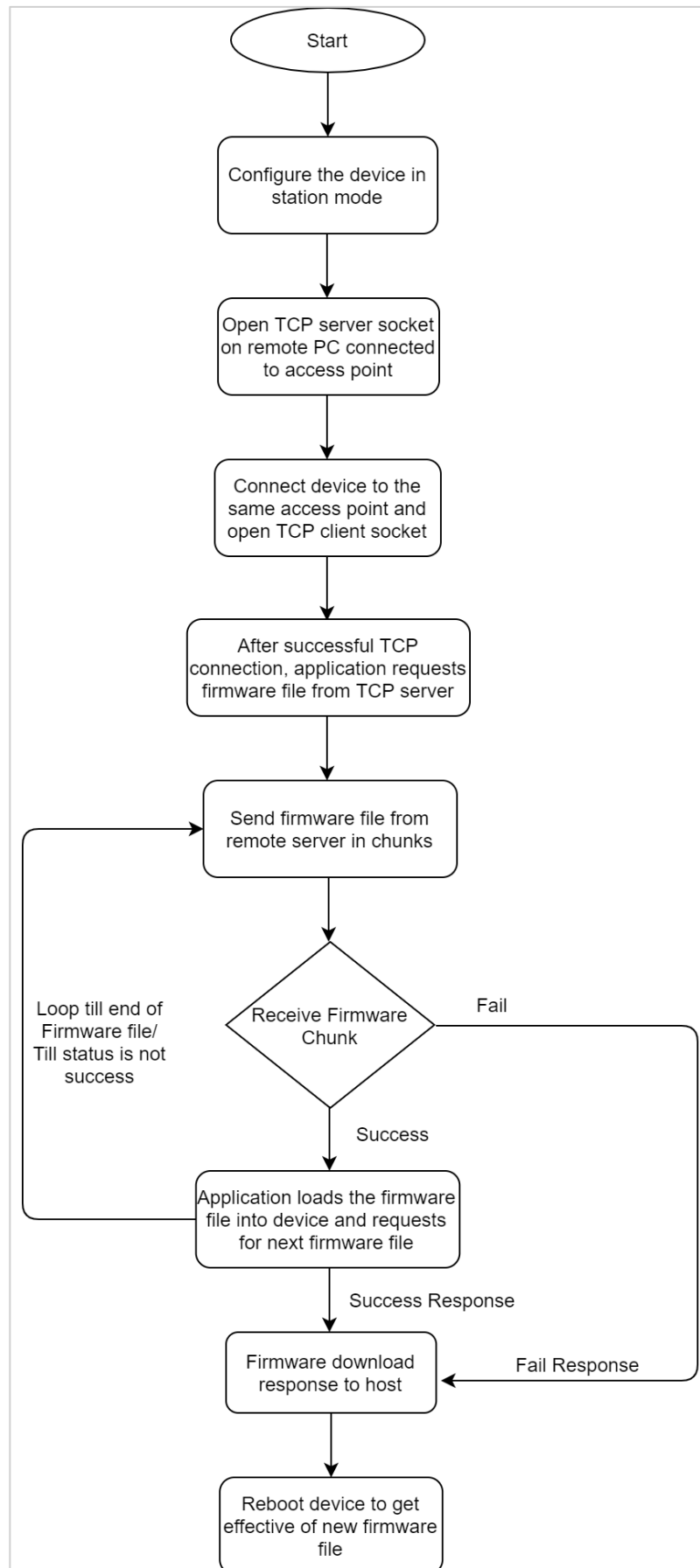


Figure 13: Firmware Update from Server Setup Diagram



#### 7.1.2.4 Flow Diagram

The following flow chart shows the firmware update flow of this application example.



**Figure 14: Flow Diagram**

### 7.1.2.5 Description

In this application, the device connects to access point and establishes TCP client connection with TCP server opened on remote peer. After successful TCP connection, application sends the firmware file request to remote TCP server and server responds with Firmware file and waits for the next firmware file request. Once firmware file is received from the TCP server, application loads the firmware file into device using firmware update API and gets next firmware file from TCP server. After successful firmware update, firmware update API returns 0x03 response.

This Application explains user how to:

1. Configure the device in station mode
2. Open TCP server socket on remote PC connected to access point
3. Connect device to the same access point and open TCP client socket on the device
4. Request Firmware file from remote server
5. Send firmware file from remote server
6. Update the received Firmware into the device

### 7.1.2.6 Configuration and Steps for Execution

#### 7.1.2.6.1 Application Configuration

Open **rsi\_firmware\_update\_app.c** file and update/modify following macros

**SSID** refers to the name of the Access point.

```
#define SSID                "<AP_SSID>"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode

**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE        RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK                  "<psk>"
```

**DEVICE\_PORT** port refers TCP client port number

```
#define DEVICE_PORT          5001
```

**SERVER\_PORT** port refers remote TCP server port number which is opened in Linux PC.

```
#define SERVER_PORT          5001
```

**SERVER\_IP\_ADDRESS** refers remote peer IP (Linux PC) address to connect with TCP server socket.

IP address should be in long format and in little endian byte order.

Example: To configure "192.168.0.100" as remote IP address, update the

macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

```
#define SERVER_IP_ADDRESS    0x6400A8C0
```

**RECV\_BUFFER\_SIZE** refers Memory for receive data

```
#define RECV_BUFFER_SIZE          1027
```

### To configure IP address

**DHCP\_MODE** refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE                  1
```

#### Note:

- To configure STA IP address through DHCP
  - Set DHCP\_MODE to 1
  - Skip configuring DEVICE\_IP, GATEWAY and NETMASK macros.
- To configure STA IP address through STATIC
  - Set DHCP\_MODE macro to "0"
  - Configure DEVICE\_IP, GATEWAY and NETMASK macros.

IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP                  0X0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY                    0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order.

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

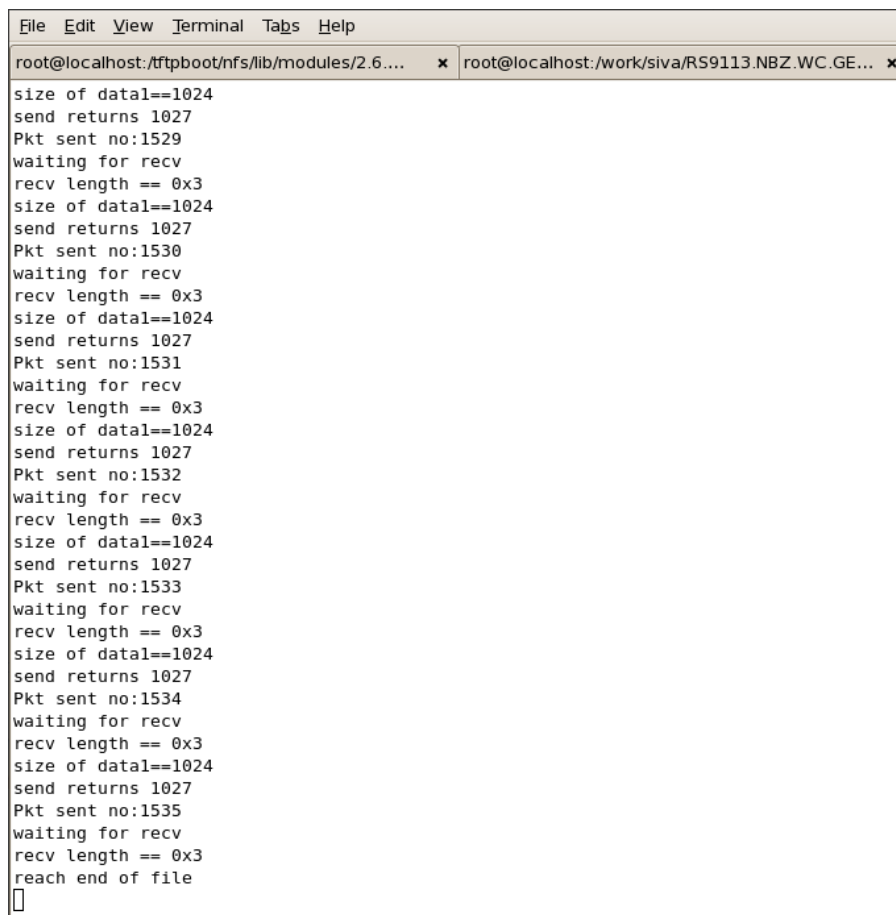
```
#define NETMASK                    0x00FFFFFF
```

open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE            RSI_DISABLE
#define RSI_FEATURE_BIT_MAP        FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS          RSI_DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP TCP_IP_FEAT_DHCPV4_CLIENT
#define RSI_CUSTOM_FEATURE_BIT_MAP FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP EXT_FEAT_384K_MODE
#define RSI_BAND                   RSI_BAND_2P4GHZ
```

#### 7.1.2.6.2 Application Execution

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect RS9116W device in STA mode.
2. Copy TCP server application present in release package "**firmware\_update/firmware\_update\_tcp\_server.c**" into Linux PC which is connected to access point through LAN.  
Compile and run by providing port number and Firmware file path
3. Compile by giving **gcc firmware\_update\_tcp\_server.c** command
4. Run the application **./a.out 5001 RS9116.NB0.WC.GENR.OSI.x.x.x.rps**
5. After the program gets executed, device connects to AP and opens TCP client socket connection.
6. After TCP connection is established with remote server, application sends firmware file request to the server.
7. Server receives request and sends firmware file in chunks.
8. After receiving chunk from remote server, application again sends firmware request to server. Server waits for the firmware request from the device before sending next chunk.
9. Packet is sent to the device in chunks as shown in the following figure. After successful update from TCP server, terminal shows "reach end of file".



```

File Edit View Terminal Tabs Help
root@localhost:/ftpboot/nfs/lib/modules/2.6... x root@localhost:/work/siva/RS9113.NBZ.WC.GE... x
size of data==1024
send returns 1027
Pkt sent no:1529
waiting for rcv
rcv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1530
waiting for rcv
rcv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1531
waiting for rcv
rcv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1532
waiting for rcv
rcv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1533
waiting for rcv
rcv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1534
waiting for rcv
rcv length == 0x3
size of data==1024
send returns 1027
Pkt sent no:1535
waiting for rcv
rcv length == 0x3
reach end of file

```

**Figure 15: Packets Sent to Device**

10. In Application **rsi\_firmware\_update\_app.c**, **rsi\_fwup\_load** API returns 0x03 response after successful firmware update and closes TCP client socket.

**Note:**

After successful firmware update, reset RS9116W module to get loaded with new firmware. Device takes about 40 seconds to give CARD READY indication after first reboot, so wait for ~40 seconds after resetting the device.

#### 7.1.2.7 API Description

Server sends firmware file to the device in chunks. The first chunk received is the RPS header followed by RPS content.

The following code shows how the corresponding firmware update APIs are called based on the chunk type received. The size of the received buffer (recv\_buffer) is configured in the macro RECV\_BUFFER\_SIZE.

rsi\_fwup\_load() API returns 0x03 response after successful firmware update

```
if(fwup_chunk_type == RSI_FWUP_RPS_HEADER)
{
    status = rsi_fwup_start(recv_buffer);
}
else
{
    status = rsi_fwup_load(recv_buffer, fwup_chunk_length);
}
```

#### 7.1.2.8 Assumptions

S. No	Item	Description
1	HOST should check availability of new firmware	Host should check the availability of new firmware, then proceed with update
2	Firmware version check	Assume Host will check firmware version before and after firmware update
3	Module to download and store	Module is responsible for downloading the firmware image and storing it to backup location
4	HOST to Reset the Module	HOST is responsible for resetting the module to LOAD downloaded Firmware

#### 7.1.2.9 Limitations

S. No	Constraints	Remarks
1	Only RPS format supported	Firmware to be updated should be in RPS format

## 7.2 Wireless-Based

The following examples are described in this section.

S. No	Example	Description	Example Source Path
1	OTAF TCP	This application demonstrates how to update new firmware to RS9116W device using remote TCP server	RS9116.NB0.WC.GENR.OSI.xxxxx\host\sapis\examples\wlan\otaf
2	OTAF HTTP/S	This application demonstrates how to update new firmware to RS9116W device using remote HTTP/S server	NA

### 7.2.1 OTAF TCP

#### 7.2.1.1 Introduction

This section demonstrates how to update new firmware to RS9116W device using remote TCP server. Here, the host uses OTA APIs for firmware update. Here host just issues firmware update request to RS9116W and new firmware is downloaded directly into RS9116W.

Both APIs and AT Commands are described in this section.

#### 7.2.1.2 Prerequisites

- Windows / Linux PC with Host interface (UART/ USB-CDC/ SPI/ USB/SDIO)
- Wireless Access point
- RS9116W module
- Linux PC with TCP server application (TCP server application is provided as part of release package)

#### 7.2.1.3 Setup Diagram

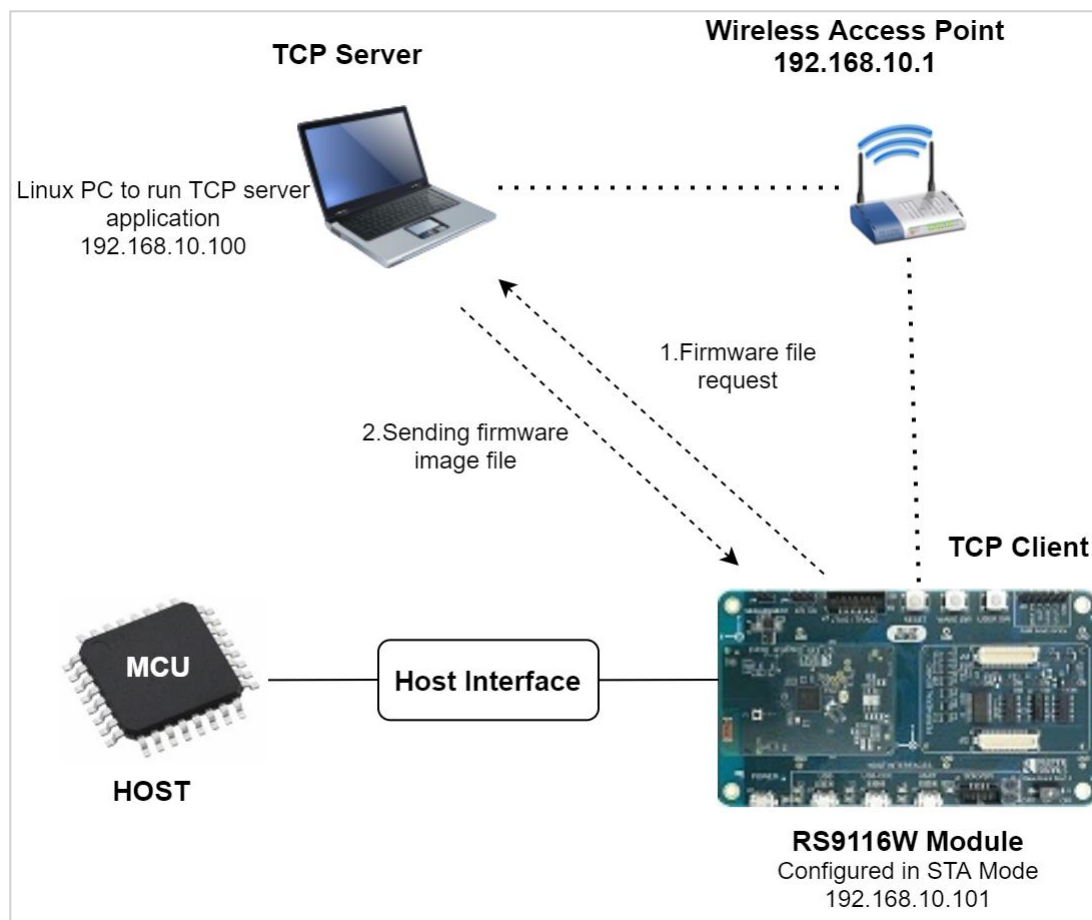
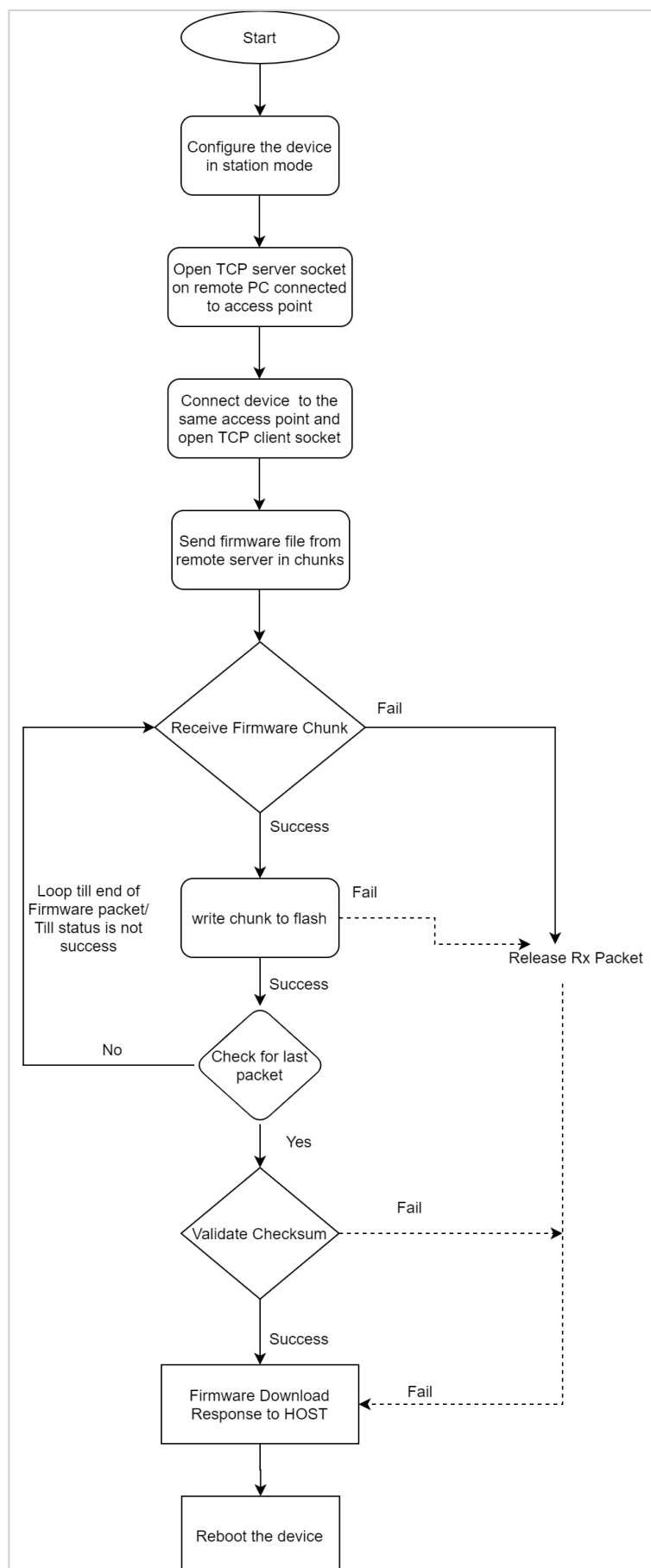


Figure 16: Setup Diagram for Over the Air Firmware Update from Server

#### 7.2.1.4 Flow Diagram

The following flow chart shows the firmware update flow of this application example.



**Figure 17: Flow Diagram**

### 7.2.1.5 Description

In this application RS9116W device connects to Access Point and using OTAF command establishes TCP client connection with remote TCP server. After successful TCP connection, module sends the firmware chunk request to remote TCP server and server responds with Firmware chunk and waits for the next firmware chunk request. Once module receives firmware chunk from the TCP server, it is copied to RS9116W flash at a pre-defined location. After successful firmware update, OTAF API returns success response to host.

This Application explains user how to:

- Configure the device in station mode
- Open TCP server socket on remote PC connected to access point
- Connect device to the same access point and open TCP client socket on the device
- Call OTA firmware update API to request Firmware file from remote server
- Send firmware file from remote server

#### Note:

The TCP server application is provided in release package in the following path:  
**sapis/examples/wlan/otaf/firmware\_update\_ota\_server.c**

### 7.2.1.6 Configuration and Steps for Execution

#### 7.2.1.6.1 Application Configuration

1. Open **rsi\_ota\_firmware\_update\_app.c** file and update/modify following macros:

**SSID** refers to the name of the Access point.

```
#define SSID "<REDPINE_AP>"
```

**SECURITY\_TYPE** refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK securities.

Valid configuration is:

**RSI\_OPEN** - For OPEN security mode

**RSI\_WPA** - For WPA security mode

**RSI\_WPA2** - For WPA2 security mode

```
#define SECURITY_TYPE RSI_OPEN
```

**PSK** refers to the secret key if the Access point configured in WPA-PSK/WPA2-PSK security modes.

```
#define PSK "<psk>"
```

**DEVICE\_PORT** port refers to TCP client port number

```
#define DEVICE_PORT 5001
```

**SERVER\_PORT** port refers remote to TCP server port number which is opened in Linux PC.

```
#define SERVER_PORT 5001
```



**SERVER\_IP\_ADDRESS** refers to remote peer IP (Linux PC) address to connect with TCP server socket. IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.0.100" as remote IP address, update the macro **SERVER\_IP\_ADDRESS** as **0x6400A8C0**.

```
#define SERVER_IP_ADDRESS 0x6400A8C0
```

**RECV\_BUFFER\_SIZE** refers to the memory to be used to receive data

```
#define RECV_BUFFER_SIZE 1027
```

### To configure IP address

**DHCP\_MODE** refers to whether the module's IP address will be STATIC or configured through DHCP

```
#define DHCP_MODE 1
```

#### Note:

If user wants to configure STA IP address through DHCP then set **DHCP\_MODE** to 1 and skip configuring the following **DEVICE\_IP**, **GATEWAY** and **NETMASK** macros.

(Or)

If user wants to configure the module to use a STATIC STA IP address then set DHCP\_MODE macro to "0" and configure the DEVICE\_IP, GATEWAY and NETMASK macros.

IP address should be in long format and in little endian byte order.

**Example:** To configure "192.168.10.10" as IP address, update the macro **DEVICE\_IP** as **0x0A0AA8C0**.

```
#define DEVICE_IP 0x0A0AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

**Example:** To configure "192.168.10.1" as Gateway, update the macro **GATEWAY** as **0x010AA8C0**

```
#define GATEWAY 0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order.

**Example:** To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define NETMASK 0x00FFFFFF
```

**OTAF\_SERVER\_PORT** refers to remote TCP server port number which is opened in Linux PC.

```
#define OTAF_SERVER_PORT 5001
```

**OTAF\_RX\_TIMEOUT** refers to remote TCP RX packet receive timeout.

```
#define OTAF_RX_TIMEOUT 200
```

**OTAF\_TCP\_RETRY\_COUNT** refers to TCP maximum retransmissions count.

```
#define OTAF_TCP_RETRY_COUNT 20
```

**OTAF\_RETRY\_COUNT** refers to OTAF update retry count.

```
#define OTAF_RETRY_COUNT 10
```

2. Open **rsi\_wlan\_config.h** file and update/modify following macros:

```
#define CONCURRENT_MODE          DISABLE
#define RSI_FEATURE_BIT_MAP      FEAT_SECURITY_PSK
#define RSI_TCP_IP_BYPASS        DISABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP (TCP_IP_FEAT_OTAF |
TCP_IP_FEAT_DHCPV4_CLIENT)
#define RSI_CUSTOM_FEATURE_BIT_MAP FEAT_CUSTOM_FEAT_EXTENTION_VALID
#define RSI_EXT_CUSTOM_FEATURE_BIT_MAP EXT_FEAT_256k_MODE
#define RSI_BAND                 RSI_BAND_2P4GHZ
```

**Note:**

**rsi\_wlan\_config.h** file is already set with desired configuration in respective example folders, user need not change for each example.

#### 7.2.1.6.2 Application Execution

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect RS9116W device in STA mode.
2. Compile and run by providing port number and Firmware file path from the server application from the path:  
**"/otaf/firmware\_update\_ota\_server.c "**  
**gcc firmware\_update\_ota\_server.c**  
**./a.out 5001 RS9116.NBZ.WC.GEN.OSI.x.x.x.rps**
3. After program gets executed, device connects to AP and opens TCP client socket connection.
4. After TCP connection established with remote server, application sends firmware file request to the server.
5. Server receives request and sends Firmware file in chunks.
6. After receiving chunk from remote server, application again sends firmware request to server. Server will wait for the firmware request from WiSeConnect device before sending next chunk.
7. Packet is sent to the device in chunks as shown in the following figure. After successful update from TCP server, terminal shows "reach end of file".

```

File Edit View Terminal Tabs Help
root@localhost:ftpboot/nfs/lib/modules/2.6.... x root@localhost:/work/siva/RS9113.NBZ.WC.GE... x
size of datal==1024
send returns 1027
Pkt sent no:1529
waiting for rcv
rcv length == 0x3
size of datal==1024
send returns 1027
Pkt sent no:1530
waiting for rcv
rcv length == 0x3
size of datal==1024
send returns 1027
Pkt sent no:1531
waiting for rcv
rcv length == 0x3
size of datal==1024
send returns 1027
Pkt sent no:1532
waiting for rcv
rcv length == 0x3
size of datal==1024
send returns 1027
Pkt sent no:1533
waiting for rcv
rcv length == 0x3
size of datal==1024
send returns 1027
Pkt sent no:1534
waiting for rcv
rcv length == 0x3
size of datal==1024
send returns 1027
Pkt sent no:1535
waiting for rcv
rcv length == 0x3
reach end of file

```

#### Note:

After successful firmware update, reset RS9116W module to get loaded with new firmware. Device takes about 40 seconds to give CARD READY indication after first reboot, so wait for ~40 seconds after resetting the device.

#### 7.2.1.7 APIs

Below API is used to update firmware over the air.

```

int32_t rsi_ota_firmware_upgradation(uint8_t flags,
                                     uint8_t *server_ip,
                                     uint32_t server_port,
                                     uint16_t chunk_number,
                                     uint16_t timeout,
                                     uint16_t tcp_retry_count,
                                     void(*ota_fw_up_response_handler)(uint16_t status,
                                     uint16_t chunk_number))

```

#### Input parameters

Parameter	Description
uint8_t flags	input flags
uint8_t *server_ip	TCP server IP (IP Address of the target server)
uint32_t server_port	TCP server port (destination port.)
uint16_t chunk_number	chunk number of the firmware to be updated. Initially keep it as 1.
uint16_t timeout	RX timeout
uint16_t tcp_retry_count	TCP retry count number
void(*ota_fw_up_response_handler)(uint16_t status, uint16_t chunk_number)	Firmware update call back response handler

## Return values

0 = success

3 = Firmware update completed successfully

<0 = failure

```
do
{
status = rsi_ota_firmware_upgradation(RSI_IP_VERSION_4, (uint8_t *)&otaf_server_addr,
OTAF_SERVER_PORT, chunk_number, OTAF_RX_TIMEOUT, OTAF_TCP_RETRY_COUNT,
rsi_ota_fw_up_response_handler);

if(status != RSI_SUCCESS)
{
LOG_PRINT("\r\nOTA Firmware Update Failed, Error Code : 0x%X\r\n", status);
return status;
}
//! wait for the success response
do
{
//! event loop
#ifdef RSI_WITH_OS
rsi_wireless_driver_task();
#endif
}while(!rsp_received);

//! Check for FWUP success
if(rsp_received == 1)
{
break;
}
//! Check for FWUP failure
if((rsp_received == 2) && (otaf_retry_cnt))
{
otaf_retry_cnt--;
}
//! Check for FWUP failure with retry count
if((rsp_received == 2) && (otaf_retry_cnt == 0))
{
return rsi_wlan_get_status();
}
//! reset rsp received
rsp_received = 0;
} while(otaf_retry_cnt);
```

### 7.2.1.8 AT Commands

```
at+rsi_ota=<ip_version>, <destIPAddr>, <server_port>,<chunk_number>, <rx_timeout>
,<tcp_retry_count>\r\n
```

## Command Parameters:

### ip\_version:

IP version used, either 4 or 6.

### destIPAddr:

IP Address of the target server.

### server\_port:

Destination port. Value ranges from 1024 to 49151.

### Note:

User can give any Port number from the above range except for 30000, which is reserved for specific feature.

**chunk\_number:**

Chunk number of the firmware to be updated. Initially keep it as 1.

**rx\_timeout:**

To configure timeout.

**tcp\_retry\_count:**

To configure TCP retransmissions count.

**Response:**

After successful update, firmware gives a success indication with an asynchronous message as "AT+RSI\_FWUPSUCCESS\r\n".

In addition, "reach end of file" message comes at server side.

On firmware update failure, firmware gives error message as "ERROR<Error\_Code><Number of chunk where up gradation stopped>".

User needs to give same command again from the chunk number specified here.

**Possible error codes:**

0xBB01, 0xBB38

**Example:**

at+rsi\_otaf = 4, 192.168.0.100, 5001, 1, 100, 10\r\n

**Response:**

AT+RSI\_FWUPSUCCESS on successful update.

ERROR<Error\_Code><number of chunk where update stopped>

ERROR<Error\_Code = 0x01 0xBB><number of chunk where update stopped = 0xD2 0x04>

After getting this error, give command

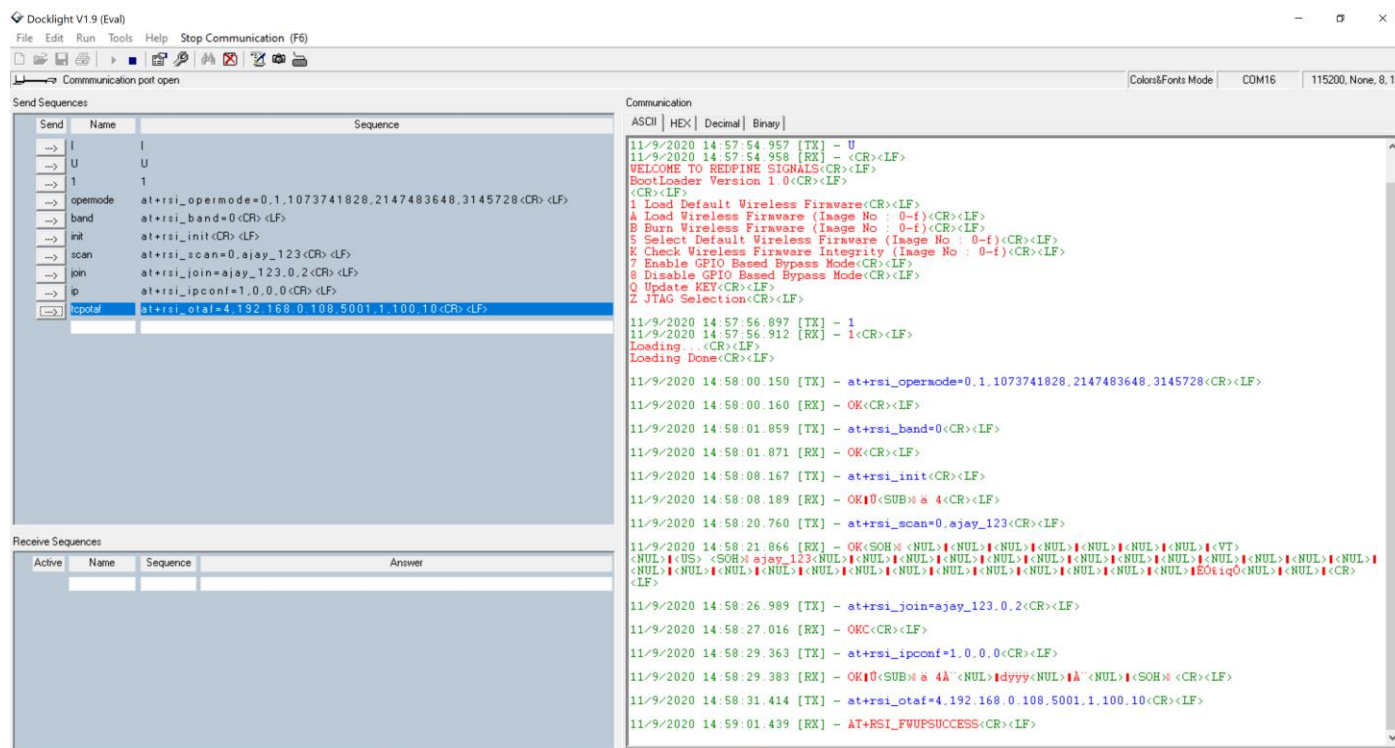
at+rsi\_otaf = 4,192.168.0.100, 5001, 1234, 100, 10\r\n

**Note:**

It is recommended to finish pending transactions (like data transfers and disconnect sockets if any) and disable power save before starting firmware update process.

**Example sequence to issue AT commands:**

- Launch Docklight/Terraterm/cutecom and issue below commands and wait for response from each command
- |
- U
- 1
- at+rsi\_opermode=0,1,1073741828,2147483648,3145728<CR><LF>
- at+rsi\_band=0<CR><LF>
- at+rsi\_init<CR><LF>
- at+rsi\_scan=0,SSID<CR><LF>
- at+rsi\_psk=1,PASSWORD<CR><LF>
- at+rsi\_join=SSID,0,2,<CR><LF>
- at+rsi\_ipconf=1,0,0,0<CR><LF>
- at+rsi\_otaf=4,192.168.0.108,1,100,10<CR><LF>



#### 7.2.1.9 Assumptions

S. No	Item	Description
1	Firmware version check	Host should check firmware version before update and after update
2	HOST should check availability of new firmware	Host should check the availability of new firmware, then issue the update command
3	HOST to Reset the Module	HOST is responsible for resetting the Module to LOAD downloaded Firmware
4	Module download and store FW	Module is responsible for downloading the FW image and storing it to a backup location

#### 7.2.1.10 Limitations

S. No	Constraints	Remarks
1	Only RPS format supported	Firmware to be updated should be in RPS format

## 7.2.2 OTAF HTTP/S

### 7.2.2.1 Introduction

This section details the mechanism to be used to update firmware to a RS9116W device using a remote HTTP/S server. In this process, RS9116W module acts as HTTP Client to download FW from HTTP/S Server and gets updated with downloaded firmware.

#### Note:

1. This feature supports only AT command mode.
2. This feature will be available after 2.0 release.

### 7.2.2.2 Prerequisites

- Wireless Access point
- RS9116W module Wi-Fi STA and HTTP/S Client
- Windows/Linux PC with HTTP server application
- A remote HTTP/S Server (with Firmware File)
- Serial Terminal (like CuteCom or Tera Term)

### 7.2.2.3 Setup Diagram

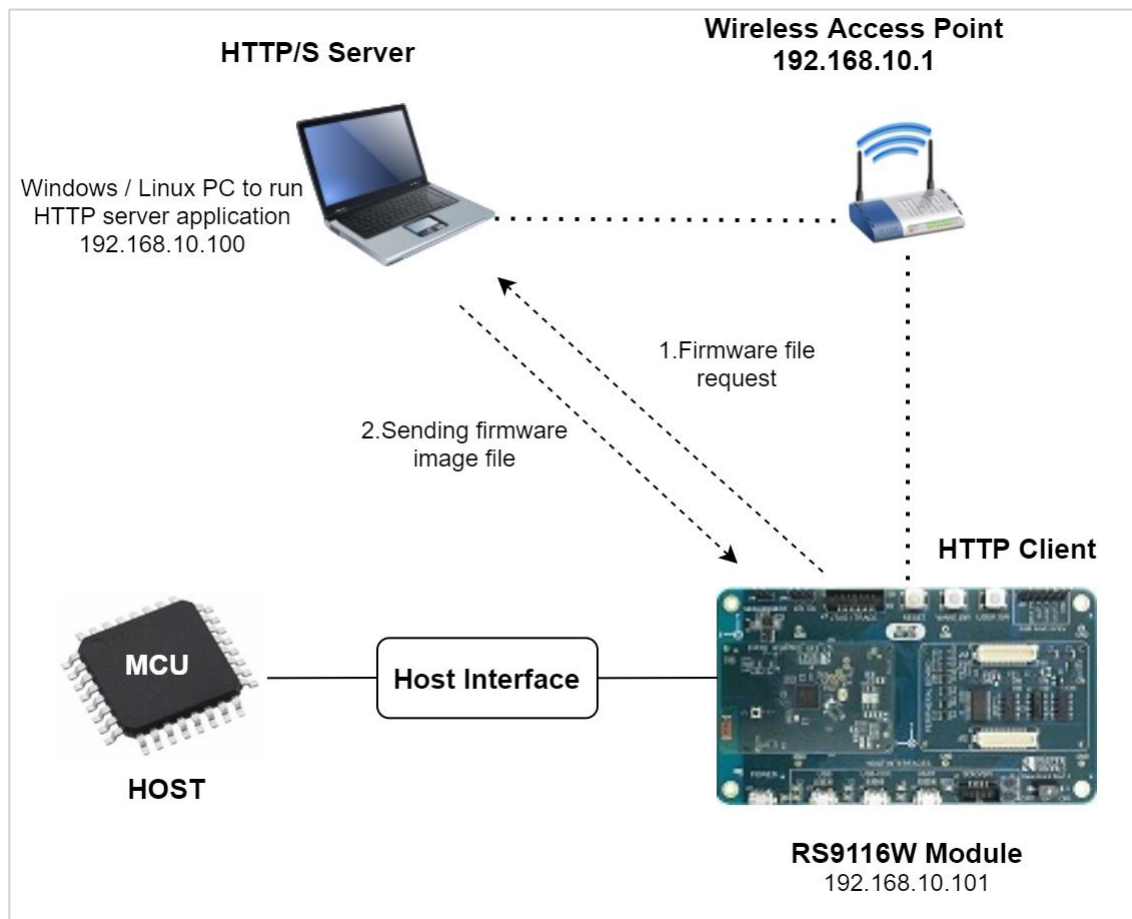


Figure 18: OTA Firmware Update from HTTP Server

#### 7.2.2.4 Block Diagram

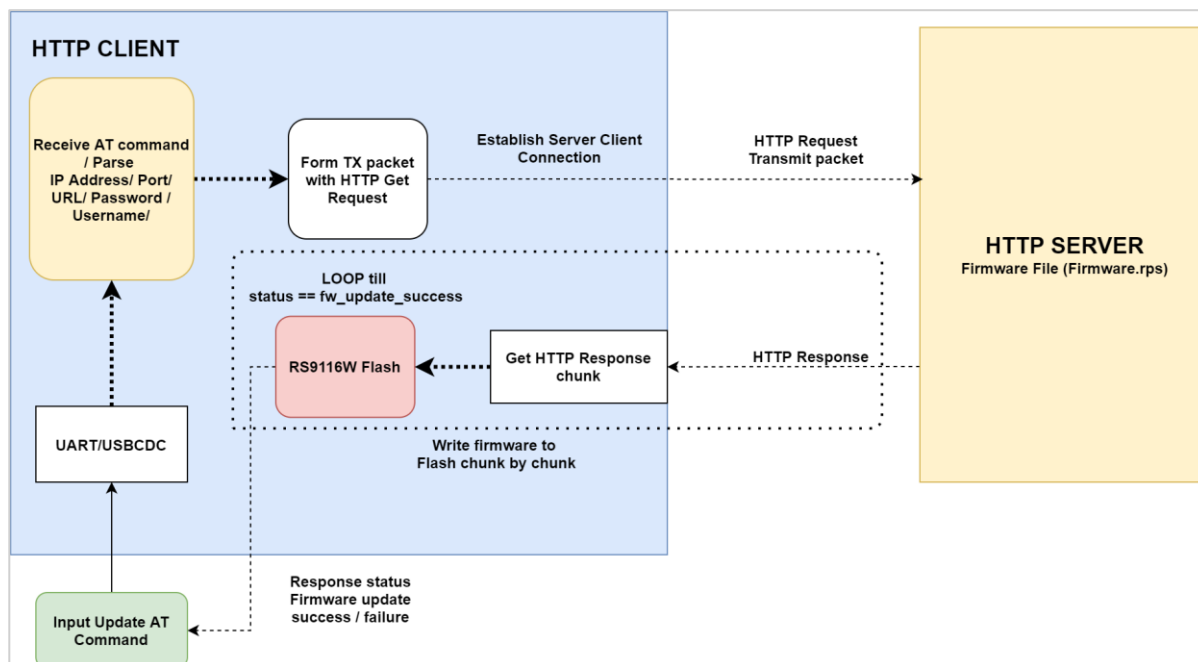


Figure 19: OTA Firmware Update from HTTP Server



### 7.2.2.5 Flow Diagram

The following flow chart shows the firmware update flow of this application example.

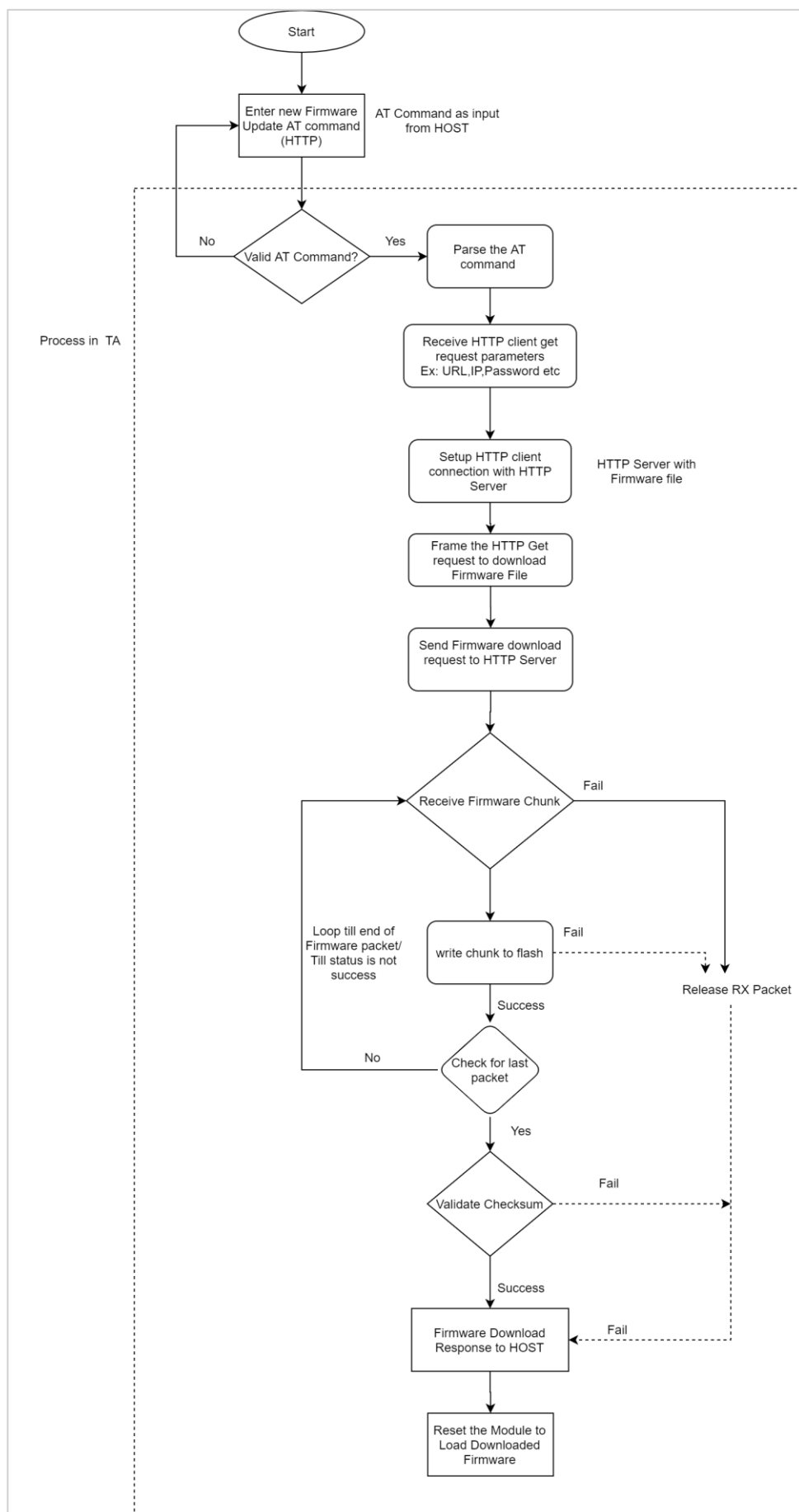


Figure 20: Flow Diagram

#### 7.2.2.6 Description

In this example, the RS9116W device connects to an Access Point and uses HTTP OTAF command. HTTP Client connection gets established with remote HTTP Server. After successful HTTP connection, module sends firmware file request to remote HTTP Server and Server responds with FW file in HTTP chunks. Once module receives firmware chunks from HTTP server, it loads firmware file into the module's flash. After successful firmware update, OTAF API returns success response to host.

This Application explains user how to:

- Configure as station mode
- Connect to Access Point
- Call HTTP OTA firmware update API to request Firmware file from remote server
- Send firmware file from remote server

## Firmware Update Message Exchanges (HTTP)

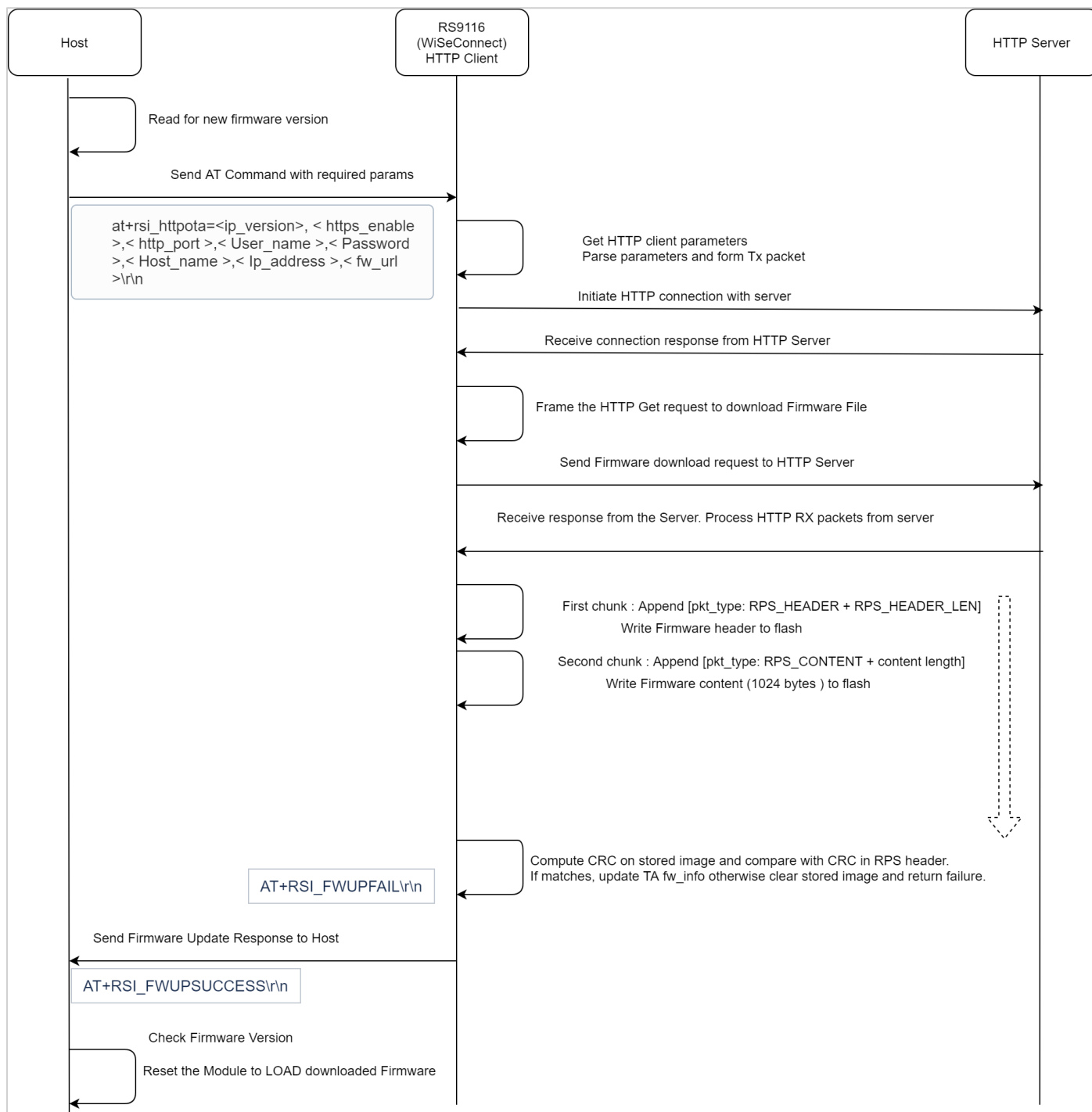
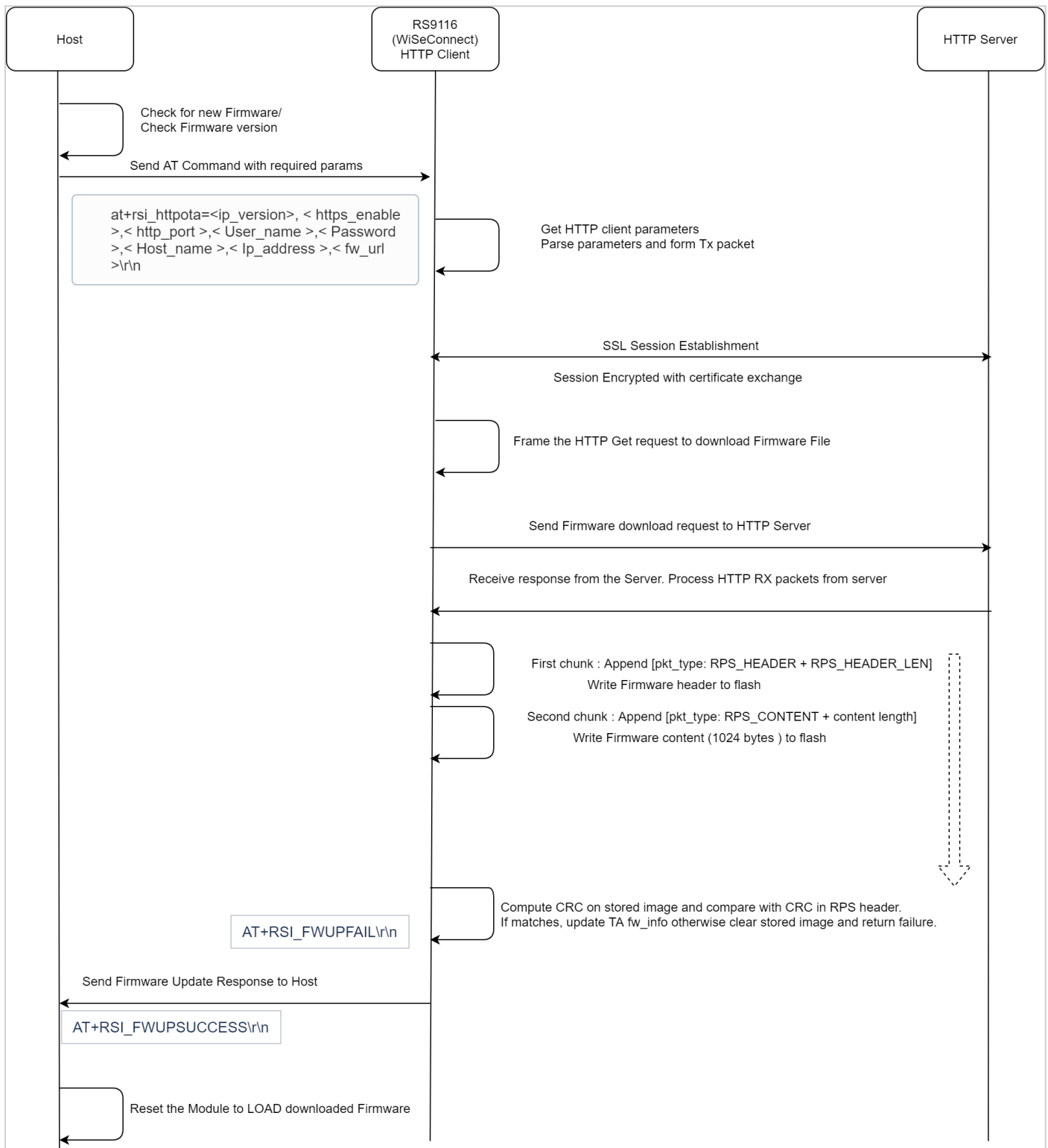


Figure 21: Firmware Update Message Exchanges (HTTP)

## Firmware Update Message Exchanges (HTTPS)



**Figure 22: Firmware Update Message Exchanges (HTTPS)**

### 7.2.2.7 Application Working

- Start HTTP Client and connect to server.
- Receive connection response from HTTP Server.
- Frame HTTP GET request to download Firmware File.
- Send Firmware download request to HTTP Server.
- Receive response from server in chunks (Firmware) as per MTU size (Max MTU size is configured as 1024).
- RPS header will be available in the First chunk (First 64 bytes), all remaining bytes are firmware content.

- Parse RPS Header to fetch image size and CRC.
- Append packet type and packet length (RPS\_HEADER\_TYPE and RPS\_HEADER\_SIZE) for RPS header received and write to flash.
- For remaining bytes of first chunk append packet type and packet length (RPS\_CONTENT\_TYPE and CONTENT\_LENGTH - RPS\_HEADER\_SIZE).
- Process subsequent HTTP chunks of firmware as above by appending packet type and packet length (RPS\_CONTENT\_TYPE and CONTENT\_LENGTH).
- Continue above step till last HTTP packet (which is determined by firmware size present in header).
- After writing complete Firmware, compute CRC and compare with CRC fetched from RPS Header.
- If CRC matches with the value in RPS header then, return success response or else failure.
- After successful download of image, reset the module to load new firmware.

#### 7.2.2.8 AT Commands

This command is used to transmit HTTP GET request from the module to a remote HTTP server for performing OTA firmware update.

A subsequent HTTP GET request can be issued only after receiving the response of the previously issued HTTP GET request. Module acts as a HTTP client when this command is issued.

#### Command Format:

at+rsi\_httpota=< https\_enable >,< http\_port >,< User\_name >,< Password >,< Host\_name >,< Ip\_address >,< url >,< Extended\_header >\r\n

S. No	Parameter	Description
1	https_enable	Set BIT(0) to enable HTTPS. Set BIT(1) to enable NULL delimiter for HTTP buffer instead of comma Set BIT(2) to use SSL TLS 1.0 version if HTTPS is enabled. Set BIT(3) to use SSL TLS 1.2 version if HTTPS is enabled. Set BIT(4) to use SSL TLS 1.1 version if HTTPS is enabled.  <b>Note:</b> <ul style="list-style-type: none"> <li>• If SSL is enabled by default it will use SSL TLS 1.0 and TLS 1.2 version.</li> <li>• BIT(2) and BIT(3) are valid only when HTTPS is enabled.</li> <li>• If SSL is enabled module will use same SSL version until module reboots.</li> </ul>
2	http_port	HTTP server port number. If this is not mentioned default port numbers will be used.
3	User_name	User_name for HTTP/HTTPS server authentication. Default username we can send 'admin'.
4	Password	Password for HTTP/HTTPS server authentication. Default password we can send 'admin'.
5	Host_name	Host name of the HTTP/HTTPS server.
6	Ip_address	IPv4/IPv6 address of HTTP/HTTPS server.
7	fw_url	URL from where Firmware shall be downloaded.
8	Extended_header	The Purpose of this is to append user configurable header fields to the default HTTP/HTTPS header.

## Command Response:

Result	Response	
Success	AT+RSI_HTTPRSP=< more ><status_code>< offset >< data_len >< Update Success>\r\n	<ul style="list-style-type: none"> <li>After the module sends out the HTTP OTAF request to the remote server, it takes some time for response (takes time to download firmware)</li> <li>The response from the remote server is sent out to the Host from the module in the following form: <ul style="list-style-type: none"> <li>AT+RSI_HTTPRSP=&lt;More&gt;&lt;Status_Code&gt;&lt;Data Offset&gt;&lt;Data Length&gt;&lt;Data&gt;</li> <li>The string AT+RSI_HTTPRSP is in uppercase ASCII.</li> </ul> </li> </ul>
Failure	ERROR<Error_code>	Failure case with error code
	AT+RSI_HTTPRSP=< more ><status_code>< offset >< data_len >< Update Failed>\r\n	There would be multiple reasons for Update Failure, Refer to <b>Expected Error Codes</b> Section

## Expected Error Codes:

Error Code	Description
0xFF74	Feature Not Enabled
0xBBF0	HTTP/HTTPS password is too long
0xBB38	Trying to connect non-existing TCP server socket
0xFFF4	Received incomplete packet in HTTP OTAF /Wrong File
0xBB41	Invalid Length

- Refer to **WLAN ERROR CODES** section in **RS9116W Wi-Fi AT Command Programming Reference Manual (PRM).pdf** for more error codes and details
- Refer to **RS9116W Wi-Fi AT Command Programming Reference Manual (PRM).pdf** for more details for API and process to issue AT command and certificate load process.

## Cipher Suites supported by HTTPS:

HTTPS supports TLS 1.2 protocol. The following cipher suites are supported

- TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA256
- TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_DHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA

- TLS\_ECDHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA
- TLS\_ECDHE\_ECDSA\_WITH\_3DES\_EDE\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_128\_CCM\_8
- TLS\_RSA\_WITH\_AES\_256\_CCM\_8

#### 7.2.2.9 Application Execution

##### To install an Apache HTTP/s Server:

###### Step 1:

1. Navigate to [Apache Website](http://httpd.apache.org) - (<http://httpd.apache.org>)
2. Click on "Download" link for the latest stable version
3. After being redirect to the download page, Select: "Files for Microsoft Windows"
4. Select one of the websites that provide binary distribution (for example: **Apache Lounge**)
5. After being redirect to "[Apache Lounge](https://www.apachelounge.com/download/)" website (<https://www.apachelounge.com/download/>), Select: Apache x.x.xx Win64 link
6. After downloaded, unzip the file httpd-x.x.xx-Win64-VC15.zip into C:/

###### Step 2:

1. Open a command prompt: *Run as Administrator*
2. Navigate to directory C:/Apache24/bin
3. Add Apache as a Windows Service: **httpd.exe -k install**

###### Step 4:

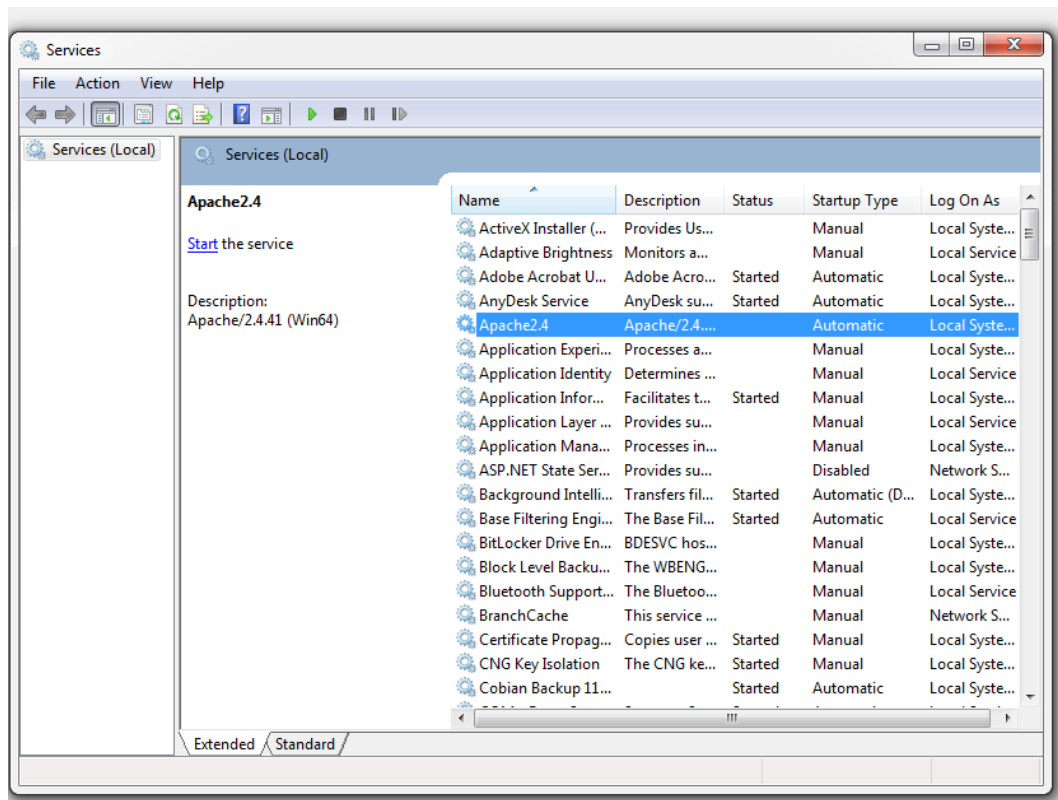
1. Open Windows Services and start Apache HTTP Server
2. Open a Web browser and type the machine IP in the address bar and hit Enter  
The message "*It works!*" should be seen.

##### To Configure an Apache HTTP Server:

- Navigate to the path: C:\Apache24\conf
- Now edit the file httpd.conf: vim httpd.conf
- Change the below lines with your local host  
*IPListen {localhost IP}:80*  
*ServerName {localhost IP}:80*
- Save and exit the file
- Navigate to the path: C:\Apache24\bin
- Right click on ApacheMonitor and Run as Administrator

Apache24 > bin			
Name	Date modified	Type	Size
iconv	6/29/2020 11:44 AM	File folder	
ab	4/21/2020 7:54 PM	Application	96 KB
abs	4/21/2020 7:54 PM	Application	108 KB
ApacheMonitor	4/21/2020 7:54 PM	Application	42 KB
apr_crypto_openssl-1.dll	4/21/2020 7:54 PM	Application extens...	19 KB
apr_dbd_odbc-1.dll	4/21/2020 7:54 PM	Application extens...	31 KB
apr_ldap-1.dll	4/21/2020 7:54 PM	Application extens...	15 KB
certificate	8/10/2020 7:23 PM	Security Certificate	2 KB

- Apache server gets started as shown below



- User can start, stop and restart server with below options



### To Configure an Apache HTTPS Server:

- Path for openssl.exe and openssl.cnf files in Apache: `C:\Apache24\conf\openssl.cnf`
- Copy server-cert.pem -key server-key.pem from `RS9116.NB0.WC.GENR.OSI.x.x.x\host\sapis\examples\utilities\certificates` to `C:\Apache\bin`
- Edit the `httpd_ssl.conf` present in path `C:\Apache24\conf\extra`
- Edit the following lines:  
`ServerName 192.168.0.103:443`

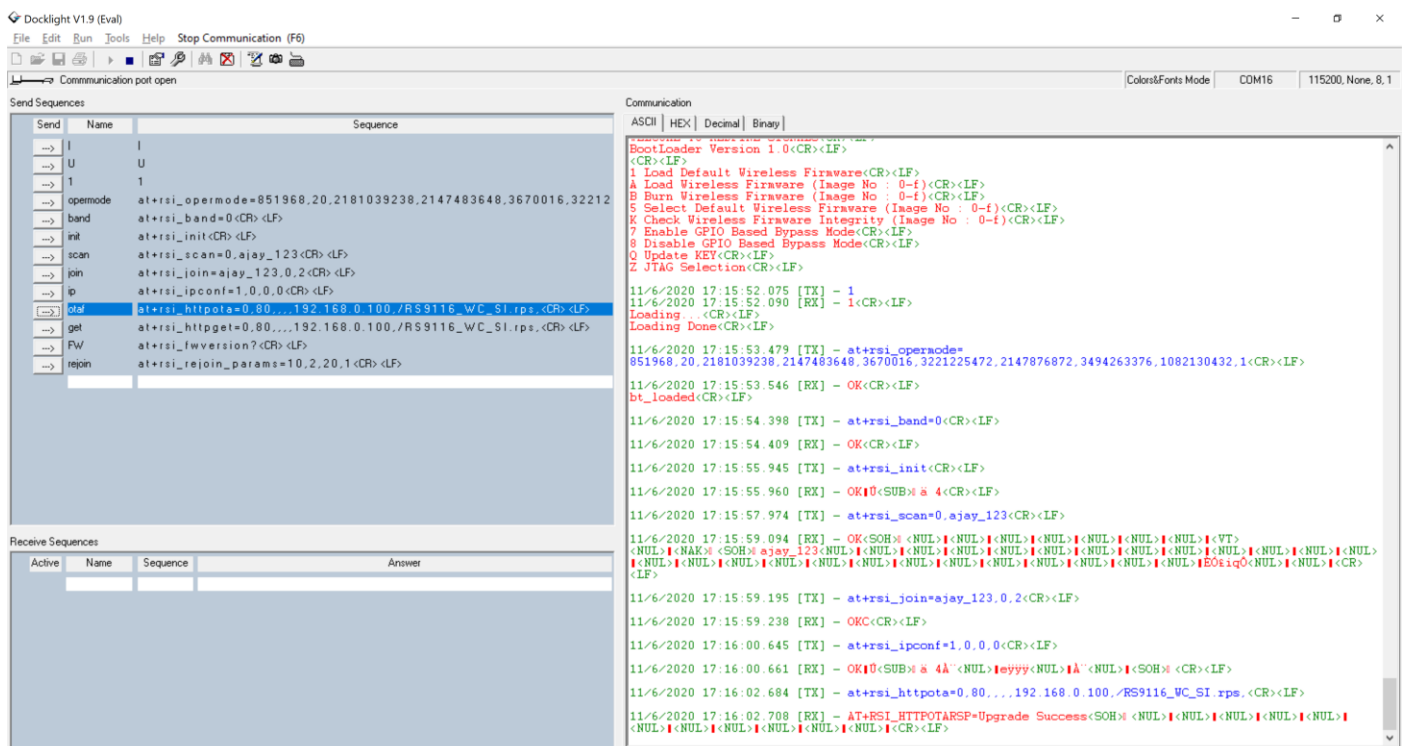


```
SSLCertificateFile "${SRVROOT}/conf/ server-key.pem"
SSLCertificateKeyFile "${SRVROOT}/conf/ server-cert.pem"
```

- Connect to remote server AP, get IP and start the server
- Start the module to connect to AP, get IP and connect to Server
- Remote web server accepts a PUT request and writes the received data to a file. User can find the created new file "index.html" on Windows PC2 in the following path **Apache24/htdocs**
- After successful creation of file using HTTP PUT, Silicon Labs device requests for the file "index.html" using HTTP GET method and waits until complete response is received from Server.
- After receiving complete response for the given HTTP GET, the device posts the given data in HTTP\_DATA macro to HTTP server using HTTP POST method

**Example sequence to issue AT commands:**

- Launch HTTP Server with above steps
- Connect UART/USB-CDC of RS9116W to any host
- Launch Docklight/Terraterm/cutecom and issue below commands and wait for response from each command
- |
- U
- 1
- at+rsi\_opermode=851968,20,2181039238,2147483648,3670016,3221225472,2147876872,3494263376,1082130432,1<CR><LF>
- at+rsi\_band=0<CR><LF>
- at+rsi\_init<CR><LF>
- at+rsi\_scan=0,SSID<CR><LF>
- at+rsi\_psk=1,PASSWORD<CR><LF>
- at+rsi\_join=SSID,0,2,2,2,1000,0,0<CR><LF>
- at+rsi\_ipconf=1<CR><LF>
- at+rsi\_httpota=0,80,,192.168.1.3,/RS9116\_WC.rps,<CR><LF>



#### 7.2.2.10 Assumptions

S. No	Item	Description
1	HOST should check availability of new firmware	Host should check the availability of new firmware, then issue the update command
2	HOST to Reset the Module	HOST is responsible for resetting the module to LOAD downloaded Firmware
3	HOST can Reset Module at any time	HOST can reset the module at any time after firmware download Once Reset gets issued, module validates and Loads the firmware
4	Module to download and store firmware	Module is responsible for downloading the firmware image and storing it to a backup location
5	IPV6 support	IPV6 support is not present.
6	HTTP OTAF timeout	Expected timeout is 30 seconds (30 retries x 1 second)
7	Issue of AT command while download not supported	Expected not to issue another AT CMD while HTTP OTA download is in progress
8	No File name / No File name length limitation	File name and File name length can be user specific, depends on Server compatibility

#### 7.2.2.11 Limitations

S. No	Constraints	Remarks
1	only RPS format supported	Firmware to be updated should be in RPS format
2	HTTP chunk must be <=1024 bytes	The MTU size in HTTP is fixed for 1024
3	No Version Rollback	No scope to Rollback to previous version after Firmware update process
4	No support for Powersave Mode	Powersave mode should be disabled before firmware update
5	No support for Chunk update	Update with chunk number is not possible in HTTP firmware update as HTTP server level changes are not expected
6	Intermediate Download Failure is not supported	If Firmware download fails in between we cannot resume it, need to initiate the download process again

#### Note:

1. OTA\_HTTP feature should be enabled in Opermode
2. HTTP Client should be enabled in Opermode
3. WLAN only mode should be enabled in Opermode
4. For HTTPS Load the certificates to Module
  - Load appropriate CA certificate to the Device to interact with HTTPS Server
  - Once certificate get loaded into device, it will write into device flash. So, user need not load certificate for every boot up unless certificate is changed.

## 8 Appendix A - Firmware Update Considerations

This section explains update considerations for the user updating to higher versions. The table below explains the SAPIs and AT commands considerations for the recent major releases 1.2.24 and 2.0 respectively.

### Recent Major Releases

- 1.2.24
- 2.0

S. No	Update Scenario	SAPIs Considerations	AT CMDs Considerations
1	Update to 1.2.24 from lower versions (or old releases)	TBD	TBD
2	Update from 1.2.24 to 2.0	Refer to section 'Changes/Enhancements in APIs, Configurations and Mechanisms' in RS9116W SAPI Programming Reference Manual.pdf	Refer to section 'Changes/Enhancements in WLAN AT Commands, Configurations and Mechanisms' in RS9116W Wi-Fi AT Command Programming Reference Manual.pdf
3	Update from 2.0 to higher	Refer to same section as mentioned for changes which should be considered when updating to new RS9116W release.	Refer to same section as mentioned for changes which should be considered when updating to new RS9116W release.

## 9 Appendix B - Host Interfaces

This section explains the pin description and schematics of different host interfaces supported by RS9116W device.

### Pin Description of Host Interfaces

Table 1 Host Interface

Pin Name	Pin Number	I/O Supply Domain	Direction	Initial State (Power up, Active Reset)	Description		
UART1_RX	135	VIN_3P3	Inout	HighZ	Host	Default	Sleep
					UART	UART1_RX - UART	HighZ
						Host interface serial input.	
					Non UART	HighZ	HighZ
					The UART interface is supported only in WiSeConnect™.		
UART1_TX	151	VIN_3P3	Inout	HighZ	Host	Default	Sleep
					UART	UART1_TX - UART	HighZ
						Host interface serial output.	
					Non UART	HighZ	HighZ
					The UART interface is supported only in WiSeConnect™.		
UART2_TX	91	VIN_3P3	Inout	HighZ	Default: UART2_TX- Debug UART Interface serial output.  Sleep: HighZ UART2_TX: Debug UART interface serial output.		
SDIO_CLK/SPI_CLK	5	SDIO_IO_VDD	Inout	HighZ	Host	Default	Sleep
					SDIO	SDIO_CLK - SDIO interface clock	HighZ
					SPI	SPI_CLK - SPI Slave interface clock	HighZ
					Non SDIO/SPI	HighZ	HighZ
					The SPI interface is supported only in WiSeConnect™.		
SDIO_CMD/SPI_CSN	62	SDIO_IO_VDD	Inout	HighZ	Host	Default	Sleep
					SDIO	SDIO_CMD - SDIO	HighZ

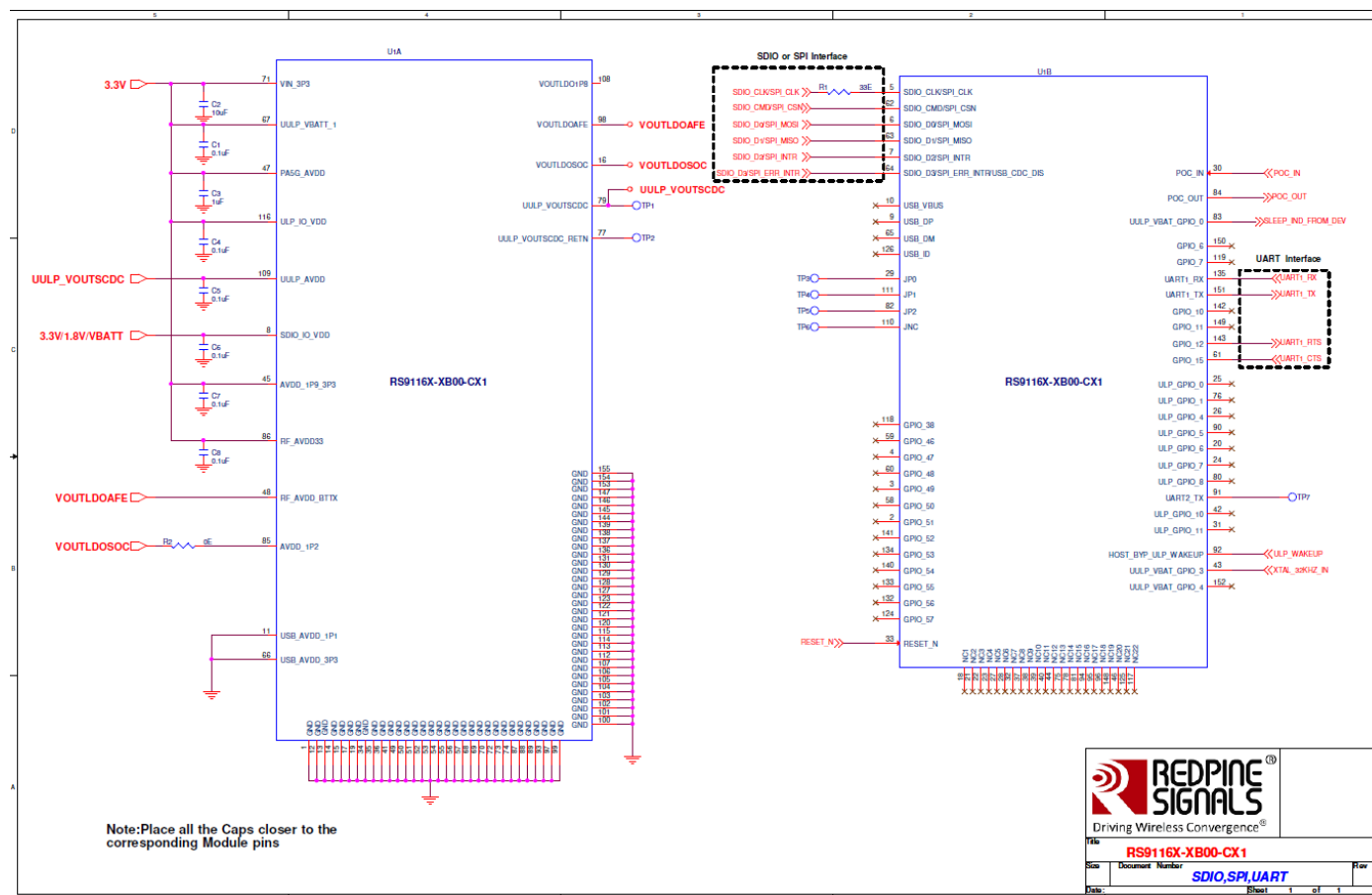
Pin Name	Pin Number	I/O Supply Domain	Direction	Initial State (Power up, Active Reset)	Description
					<div>interface CMD signal</div> <div>SPI <div>SPI_CSN - Active-low Chip Select signal of SPI Slave interface</div> <div>HighZ</div> </div> <div>Non SDIO/SPI <div>HighZ</div> <div>HighZ</div> </div> <div>The SPI interface is supported only in WiSeConnect™.</div>
SDIO_D0/SPI_MOSI	6	SDIO_IO_VDD	Inout	HighZ	<div>HostDefaultSleep</div> <div>SDIO <div>SDIO_D0 - SDIO interface Data0 signal</div> <div>HighZ</div> </div> <div>SPI <div>SPI_MOSI - SPI Slave interface Master-OutSlave-In signal</div> <div>HighZ</div> </div> <div>Non SDIO/SPI <div>HighZ</div> <div>HighZ</div> </div> <div>The SPI interface is supported only in WiSeConnect™.</div>
SDIO_D1/SPI_MISO	63	SDIO_IO_VDD	Inout	HighZ	<div>HostDefaultSleep</div> <div>SDIO <div>SDIO_D1 - SDIO interface Data1 signal</div> <div>HighZ</div> </div> <div>SPI <div>SPI_MISO - SPI Slave interface Master-In-Slave-Out signal</div> <div>HighZ</div> </div> <div>Non SDIO/SPI <div>HighZ</div> <div>HighZ</div> </div> <div>The SPI interface is supported only in WiSeConnect™.</div>
SDIO_D2/SPI_INTR	7	SDIO_IO_VDD	Inout	HighZ	<div>HostDefaultSleep</div> <div>SDIO <div>SDIO_D2 - SDIO interface Data2 signal</div> <div>HighZ</div> </div>

Pin Name	Pin Number	I/O Supply Domain	Direction	Initial State (Power up, Active Reset)	Description		
					SPI	SPI_INTR - SPI Slave interface Interrupt Signal to the Host	HighZ
					Non SDIO/SPI	HighZ	HighZ
					The SPI interface is supported only in WiSeConnect™.		
SDIO_D3/SPI_ERR_INT/USB_CDC_DIS	64	SDIO_IO_VDD	Inout	HighZ	Host	Default	Sleep
					SDIO	SDIO_D3 - SDIO interface Data3 signal	HighZ
					SPI	SPI_ERR_INT - SPI Bus Error Interrupt Signals USB_CDC_DIS - USB CDC ActiveHigh Disable Signal	HighZ
					Non SDIO/SPI	HighZ	HighZ
					The SPI interface is supported only in WiSeConnect™.		
USB_DP	9	USB_AVDD_3P3	Inout	NA	Positive data channel from the USB connector.		
USB_DM	65	USB_AVDD_3P3	Inout	NA	Negative data channel from the USB connector.		
USB_ID	126	USB_AVDD_3P3	Inout	NA	ID signal from the USB connector.		
USB_VBUS	10	USB_AVDD_3P3	Inout	NA	5V USB VBUS signal from the USB connector.		

## Schematics

## SDIO/SPI/UART

The following diagram shows the typical schematic with SDIO/SPI/UART Host Interface.



### Figure 23: SDIO/SPI/UART Interface Schematics

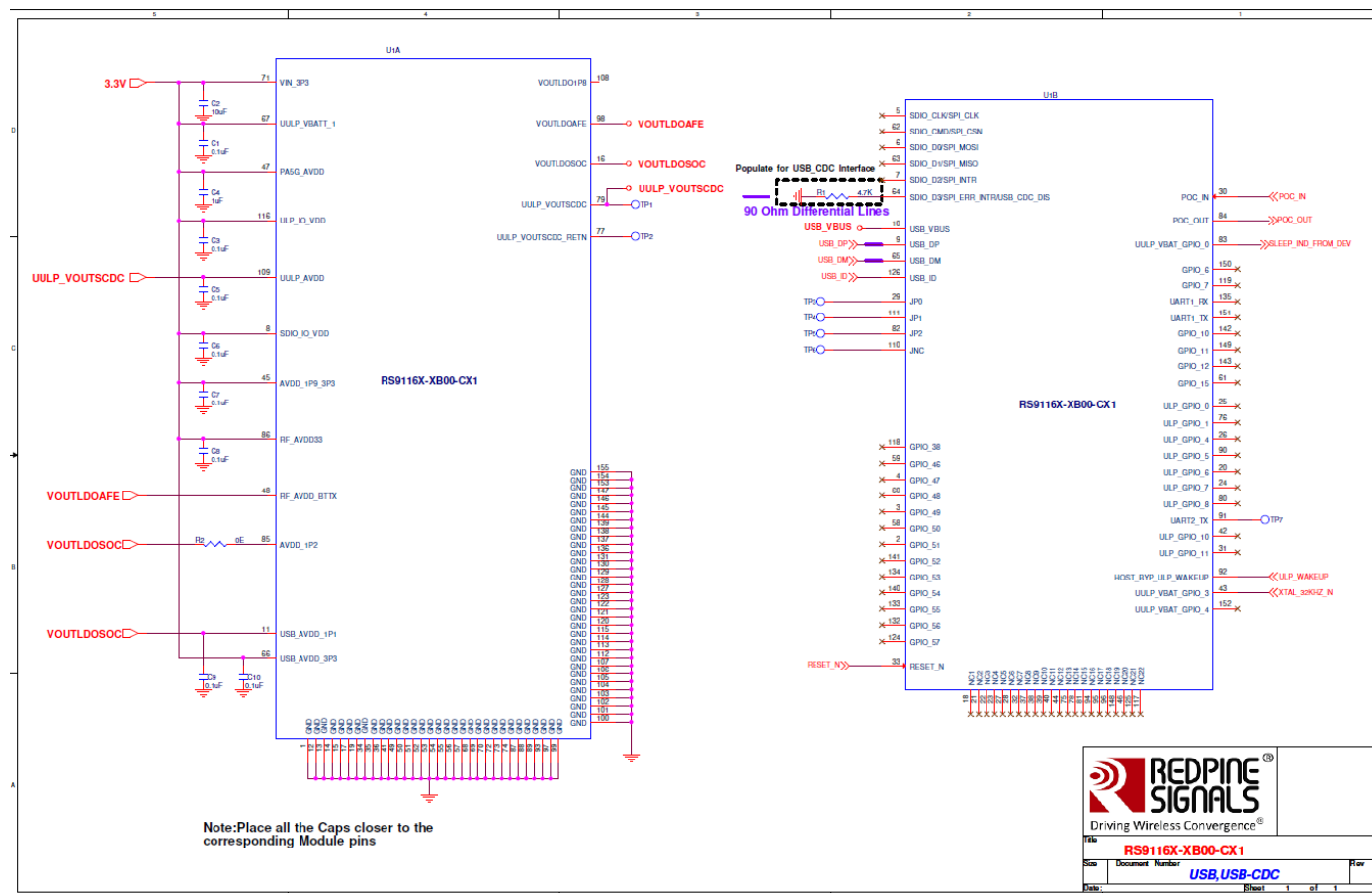
**Note:**

1. The supplies can be driven by different voltage sources within the recommended operating conditions specified in Specifications section.
2. SDIO\_IO\_VDD can be driven by a different source irrespective of other sources to support different interfaces.
3. In the SDIO mode, pull-up resistors should be present on SDIO\_CMD & SDIO Data lines as per the SDIO physical layer specification, version 2.0.
4. In SPI mode, ensure that the input signals, SPI\_CS and SPI\_CLK are not floating when the device is powered up and reset is deasserted. This can be done by ensuring that the host processor configures its signals (outputs) before deasserting the reset. SPI\_INTR is the interrupt signal driven by the slave device. This signal may be configured as Active-high or Active-low. If it is active-high, an external pull-down resistor may be required. If it is active-low, an external pull-up resistor may be required. This resistor can be avoided if the following action needs to be carried out in the host processor.
  - a. To use the signal in the Active-high or Active-low mode, ensure that, during the power up of the device, the Interrupt is disabled in the Host processor before deasserting the reset. After deasserting the reset, the Interrupt needs to be enabled only after the SPI initialization is done and the Interrupt mode is programmed to either Active-high or Active-low mode as required.
  - b. The Host processor needs to disable the interrupt before the ULP Sleep mode is entered and enable it after SPI interface is reinitialized upon wakeup from ULP Sleep.

5. In UART mode, ensure that the input signals, UART\_RX and UART\_CTS are not floating when the device is powered up and reset is deasserted. This can be done by ensuring that the host processor configures its signals (outputs) before deasserting the reset.
6. Resistor "R1" should not be populated if UART is used as Host Interface.

## USB/USB-CDC

The following diagram shows the typical schematic with USB/USB-CDC Host Interface.



### Figure 24: USB/USB-CDC Interface Schematics

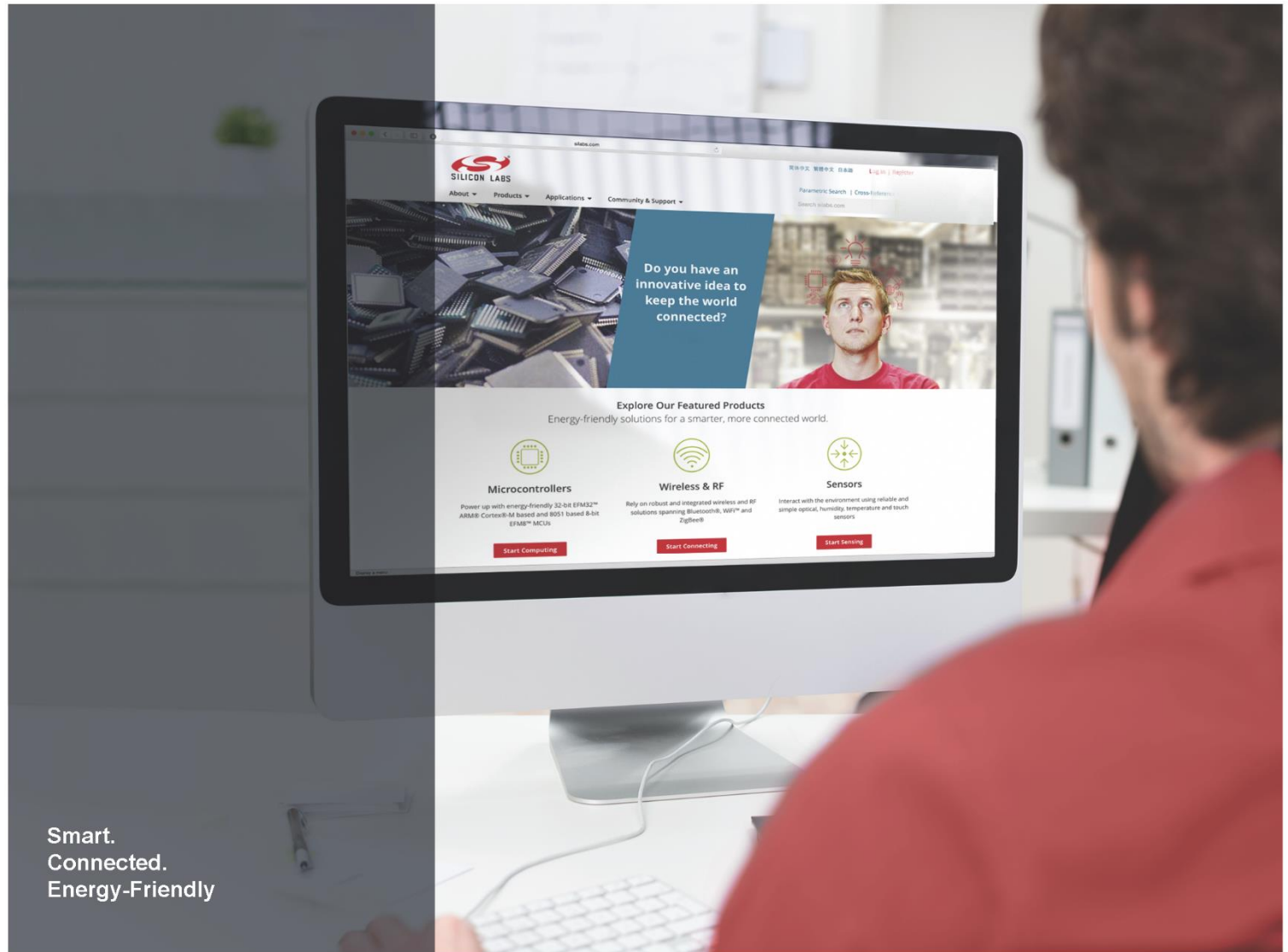
**Note:**

1. The supplies can be driven by different voltage sources within the recommended operating conditions specified in specifications section.
2. Ensure that the pin USB\_CDC\_DIS is left unconnected to ensure normal USB functionality.
3. Resistor "R1" should not be populated if normal USB is used as Host Interface.

**Note:**

For more information refer to <https://www.silabs.com/documents/login/data-sheets/rs9116-cc1-connectivity-module-datasheet.pdf>





Smart.  
Connected.  
Energy-Friendly



**Products**  
[www.silabs.com/products](http://www.silabs.com/products)



**Quality**  
[www.silabs.com/quality](http://www.silabs.com/quality)



**Support and Community**  
[community.silabs.com](http://community.silabs.com)

#### Disclaimer

Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

#### Trademark Information

Silicon Laboratories Inc., Silicon Laboratories, Silicon Labs, SiLabs and the Silicon Labs logo, CMEMS®, EFM, EFM32, EFR, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZMac®, EZRadio®, EZRadioPRO®, DSPLL®, ISOmodem®, Precision32®, ProSLIC®, SiPHY®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.  
400 West Cesar Chavez  
Austin, TX 78701

<http://www.silabs.com>