Connect-io-n™
RS9110-N-11-22/24/28

Software Programming Reference Manual

Version 4.15

January 2013

Redpine Signals, Inc.
2107 N. First Street, #680
San Jose, CA 95131.
Tel: (408) 748-3385
Fax: (408) 705-2019
Email: info@redpinesignals.com
Website: www.redpinesignals.com

Document Conventions

This document is for the below part numbers. All references to RS9110-N-11-2X denote the below modules only. **This version of document is not applicable for RS9110-N-11-24 modules.**

Device Number	Description	TCP/IP Enabled	Option for TCP/IP Bypass
RS9110-N-11-22-04	Part with UART/SPI as Host Interface	Yes	Yes in SPI mode
RS9110-N-11-22-05	Part with UART/SPI as Host Interface and uFL connector for external antenna	Yes	Yes in SPI mode
RS9110-N-11-28-04	Part with UART/SPI as Host Interface	Yes	Yes in SPI mode

In the sections that follow, the parts may sometimes be referred to as RS9110-N-11-22 or RS9110-N-11-28, without their extensions-.

Table of Contents

List of Figures

List of Tables

1 Introduction

This document describes the commands to operate the RS9110-N-11-2x modules. The parameters in the commands and their valid values; and the expected responses from the modules are also described. It also describes the flow of commands to be used to configure the modules into specific functionality. The document should be used by the developer to write software on the Host MCU to control and operate the module.

Section RS9110-N-11-2X in UART Mode describes commands to operate the module using the UART interface. Section RS9110-N-11-2X in SPI Mode describes commands to operate the module using the SPI interface.

2 Architecture Overview

The architecture of the RS9110-N-11-2X module is shown below.

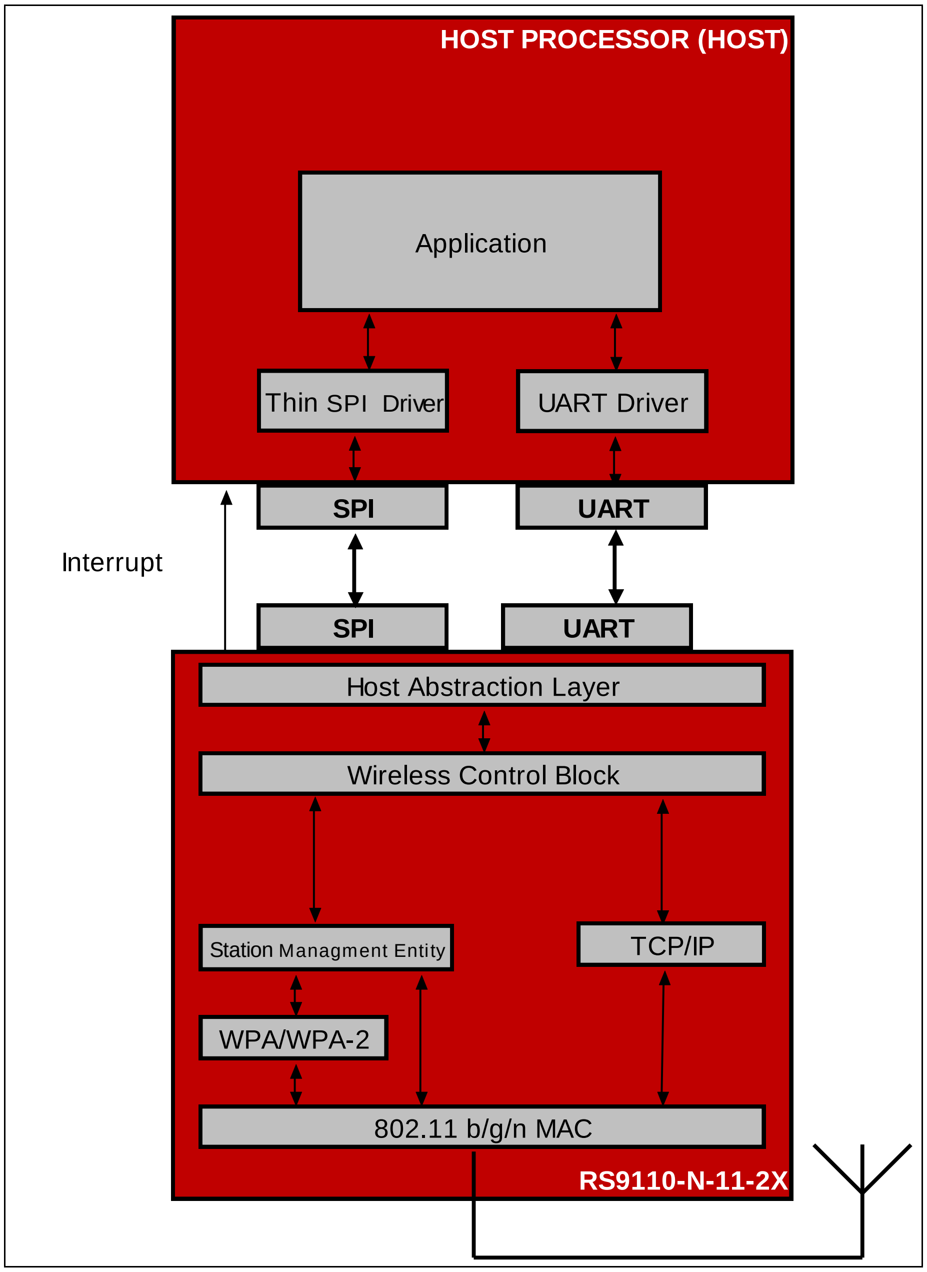


Figure 1:RS9110-N-11-2X Software Architecture Block Diagram

The RS9110-N-11-2X module is integrated with the Host using either UART or SPI interfaces. The transmission and reception of the data to/from the Host depends on the interface used to connect the module as briefed below.

2.1 Host

The Host is any system that has applications being executed and either a UART or SPI interface to connect to the RS9110-N-11-2X module.

2.1.1 UART

The UART on the Host side provides an interface for the Host to access the Wi-Fi module. UART is used to configure various parameters of the RS9110-N-11-2X module and also to send and receive data over the network by accessing the TCP/IP stack in the RS9110-N-11-2X module.

2.1.2 SPI

The SPI on the Host side provides an interface for the Host to access the Wi-Fi module. SPI on the Host acts as the master.

The SPI driver on the Host is a thin driver through which the applications interact with the Wi-Fi module. The SPI driver uses the SPI Host controller driver on the Host to send/receive the data to/from the RS9110-N-11-2X module and also to configure the same over the SPI interface.

2.2 RS9110-N-11-2X

The RS9110-N-11-2X module incorporates Wi-Fi, TCP/IP, DHCP, ARP, etc. It handles all the network connectivity functions. The following sections describe the software components of the RS9110-N-11-2X module in brief.

2.2.1 SPI

The SPI interface can be used to operate the module and transfer data. SPI frames are used to operate the module in SPI mode. The RS9110-N-11-2X acts the SPI slave, while the Host acts as the master. It is a standard 4-wire SPI interface.

2.2.2 UART

The UART interface can be used to operate the module and transfer data. AT command set is used to operate the module in UART mode.

2.2.3 Hardware Abstraction Layer (HAL)

The HAL abstracts the lower layers in the Host interface with which the RS9110-N-11-2X module is connected. The HAL interacts with the Wireless Control Block layer for the processing of the frames obtained from or destined to the Host.

2.2.4 Wireless Control Block (WCB)

The data from/to the Host is classified as Wi-Fi specific frames and TCP/IP specific frames. The WCB layer processes the frame obtained and acts accordingly. The functionality of the WCB module depends on the type of the frame and the direction of the frame (in case of TCP/IP) as described below.

2.2.5 Wi-Fi Control frames

The WCB interprets the Wi-Fi control information from the Host and interacts with the SME (Station Management Entity). Configuration of the RS9110-N-11-2X module from the Host for Wi-Fi access is through AT commands. The Wi-Fi AT commands set or get the values of various configurable parameters for providing the Wi-Fi access.

2.2.6 TCP/IP Control frames

If the frames from the Host are interpreted as TCP/IP specific frames then the WCB interacts with the TCP/IP stack.

2.2.7 Station Management Entity (SME)

The SME is the core layer which manages the Wi-Fi connectivity. The SME maintains the state machine to detect the activity on the Wi-Fi network and indicates to the user accordingly. It also performs re-association to the configured access point in Infrastructure mode. It interacts with the WPA supplicant if Security is enabled in the Wi-Fi network.

2.2.8 WPA Supplicant

The WPA supplicant is used to initiate the 802.1x/E Access Point authentication if WPA/WPA2-PSK is used as the security parameter. It also plays a major part in performing the 4-way handshake to derive the PTK in WPA/WPA2-PSK modes.

2.2.9 Interrupt

The interrupt pin in the module is an output pin, used by SPI based modules in two conditions:

1.When the module has data in its output buffers, that needs to be sent to the host through the SPI interface.
2.When the module wakes up from sleep in Power Save Mode 1.

The interrupt is active high and level triggered.

3 RS9110-N-11-2X in UART Mode

The following sections show how to operate the module using the UART interface.

3.1 Auto Baud Rate Detection (ABRD)

The RS9110-N-11-2X module automatically detects the baud rate of the Host's UART interface by exchanging some bytes. The Host should configure the UART interface for the following parameters for ABRD detection.

Stop bits – 2

Data bits – 8

Parity – None

Flow Control – None

The baud rates supported by the RS9110-N-11-2X module are as follows:

9600 bps

19200 bps

38400 bps

57600 bps

115200 bps

200000 bps

230400 bps

460800 bps

921600 bps

1843200 bps

3686400 bps

The following is the procedure to be followed by the Host for ABRD by the RS9110-N-11-2X module.

1. Configure the UART interface of the Host at the desired baud rate.
2. Power on the RS9110-N-11-2X module.
3. The Host, on power up, should wait for 110 ms for initial boot-up of the module to complete and then transmit 0x1C at the baud rate with which its UART interface is configured. After transmitting '0x1C' to the module, the Host should wait for the module to transmit 0x55 at the same baud rate.
4. If the '0x55' response is not received from the module, the Host has to retransmit 0x1C, after a delay of 200ms.
5. On finally receiving '0x55', the host should transmit '0x55' to the module.

The module is now configured with the intended baud rate. The module waits for a maximum of 18 seconds for the ABRD message handshakes to get completed.

In case of failure of this exchange, the module gets configured with the default baud rate of 115200 bps after a maximum of 18 secs.

3.2 Module Operation in UART Mode

The Wi-Fi AT command set represents the control frames that are sent from the Host to the RS9110-N-11-2X module to set or get the values of various control parameters. The command set resembles the standard AT command interface used for modems.

AT commands start with "AT" and are terminated with a carriage return and a new line character.

The AT command set for the RS9110-N-11-2X module starts with "at+rsi_" followed by the name of the control parameter. The control parameter is followed by either '?' or '=' or NULL. If the character following the control parameter is a '?' the value of the control parameter is retrieved. If the character following the control parameter is '=' the subsequent value is set to the control parameter. If a command requires no parameters then the control parameter is NULL (no parameters).

The following is an example command for sending the INIT command:

```
at+rsi_init\r\n
```

```
OK\r\n
```

"OK" is the success response to the command. The following are the RS9110-N-11-2X module's Wi-Fi configuration AT commands.

NOTE: All commands are issued from Host to module as a sequence of ASCII characters. All return messages from module to Host consist of OK or ERROR strings, returned in ASCII. ERROR is accompanied by <Error code> returned in two's complement format.

3.2.1 Band

Description

This is the first command that needs to be sent from the Host. This command configures the band in which the module has to be configured.

Command

at+rsi_band

Usage

at+rsi_band=band_val[1]\r\n

[1] All AT commands are sent as a sequence of ASCII characters. The termination characters after every command and response are '\r\n' which is 0x0D, 0x0A

Parameters

band_val:

0– 2.4 GHz

1– 5 GHz

RS9110-N-11-22/24(-xx) are single band modules (2.4 GHz only), while RS9110-N-11-28(-xx) is a dual band module (2.4 GHz and 5 GHz).

For Example, band command is given as

at+rsi_band=1\r\n

……………………………..

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x62 0x61 0x6E 0x64 0x3D 0x31 0x0D 0x0A

NOTE: Note: This format of command and the corresponding byte stream under the dotted line is followed in all examples.

Response

Result Code	Description
OK[1]	Successful execution of the command
ERROR<Error code>	Failure

The ERROR message comes with an associated error code. The string ERROR is transmitted in its ASCII form and the error code in its two's complement form.

For Example, error message with code -8 will be returned as

ERROR -8\r\n
………………………..
0x45 0x52 0x52 0x4F 0x52 0xF8 0x0D 0x0A

In this example, the two's complement of -8 is 0xF8 is sent out, while ASCII values of "ERROR" are sent

3.2.2 Init

Description

[1] OK messages with a return value are represented as OK<retvalue>. OK messages without a return value are represented as 'OK'

This is command to be executed right after the at+rsi_band command. This command programs the RS9110-N-11-2X module's Baseband and RF components.

Command

at+rsi_init

Usage

at+rsi_init\r\n

Parameters

No parameters

Response

Result Code	**Description**
OK	Successful execution of the command
ERROR<Error code>	Failure

For example, the init command is given as

at+rsi_init\r\n

……………………

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x69 0x6E 0x69 0x74 0x0D 0x0A

3.2.3 Number of Scan Results

Description

This command is used to query the total number of Access Points scanned by the module after the Scan command is issued or to configure the number of scan results that are sent by the module to the Host as part of the response to the "Scan" or "Next Scan" commands. This command is provided for Hosts which do not have enough memory to receive the complete set of scan results in one go.

This command is optional for Hosts which do not have a limitation with respect to the amount of memory available. If this command is not issued, the module returns the maximum number of scan results that it supports.

Command

at+rsi_numscan

Usage

at+rsi_numscan?\r\n or at+rsi_numscan=value\r\n

Parameters

?: To get the number of networks already scanned by the "Scan" or Next Scan" command.

value: This parameter configures the number of scan results the module returns for the Scan and NextScan commands.

Example 1 : To configure for 10 scan results, the command is

at+rsi_numscan=10\r\n

……………………………….

```
0x61    0x74  0x2B  0x72  0x73  0x69  0x5F  0x6E  0x75  0x6D 0x73
   0x63  0x61  0x6E  0x0A  0x31  0x30  0x0D  0x0A
```

Response

Result Code	Description
OK<retvalue>	*retvalue* (1 byte, hex): Number of nodes scanned. For example if there are 9 nodes, return value is 0x09. If there are 10 nodes, return value is 0x0A. This parameter is returned if "?" is used.
ERROR<Error code>	Failure.

Example 2: If there are 10 wireless networks scanned by at+rsi_scan and at+rsi_numscan? is issued after at+rsi_scan command, the return value is

OK 10 \r\n

………………………………………………

0x4F 0x4B 0x0A 0x0D 0x0A

3.2.4 Passive Scan

Description

This command enables the module to scan for networks without sending a probe request from the module. In this mode, the module gets the network information from the beacons sent by the Access Point. This command is optional and should be given before issuing *at+rsi_scan* command if passive scanning is required.

Command

at+rsi_passscan

Usage

at+rsi_passscan=bit_map\r\n

Parameters

bit_map: Parameter to configure for which channels passive scan is to be done.

For example, if only channel 1 and channel 4 are required to be scanned passively, then the value for bit_map is calculated as

Channel[n]	Channel[n-1] ……..	Channel[4]	Channel[3]	Channel[2]	Channel[1]
0	0	1	0	0	1

Decimal for <000..001001> is 9. Hence, the command is

at+rsi_passscan=9\r\n

………………………………………………

```
0x61     0x74  0x2B  0x72  0x73  0x69  0x5F  0x70  0x61   0x73   0x73
0x73  0x63  0x61  0x6E  0x0A  0x39  0x0D  0x0A
```

After this, the command at+rsi_scan=0\r\n can be given to scan all channels, out of which Channel 4 and 1 would be scanned passively.

Response

Result Code	Description
OK	Successful execution of the command. “OK” is returned in ASCII
ERROR<Error code>	Failure

3.2.5 Scan

Description

This command scans in all the channels or the channel specified in the command. If the user wants to join a hidden Access Point (SSID not being broadcast), the SSID of the Access Point is passed as the parameter also. Otherwise the SSID field is NULL(‘\0’).

Command

at+rsi_scan

Usage

at+rsi_scan=chan_num,SSID\r\n

Parameters

chan_num: Channel Number on which the scan has to be done.

Parameters for 2.4 GHz

Channel Number	chan_num parameter
All channels	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

Table 1: Channel Parameters for 2.4 Ghz

Parameters for 5 GHz

Channel Number	chan_num parameter
All channels	0
36	1
40	2
44	3
48	4
52	5
56	6
60	7
64	8
100	9
104	10
108	11
112	12
116	13
120	14
124	15
128	16
132	17
136	18

140	19
149	20
153	21
157	22
161	23
165	24

Table 2: Channel Parameters for 5 Ghz

If this value is 0, the module scans in all the channels in the band that is selected through the at+rsi_band command.

SSID: Optional Input. To scan a specific AP with a known SSID, or a hidden Access Point, this parameter is used.

NOTE: In 5 GHz, DFS Client (802.11h) is supported. DFS client implementation is internal to the module. The module, by default, does only passive scan in DFS channels. If the Access Point detects radar signals, it indicates to the module (client) to switch to a different channel by using the "channel switch frame". The module performs channel switch as per the AP's channel switch parameters. There is no command required to enable this feature, it is enabled by default.

Response

The response frame for the scan command is as shown below.

Result Code	Description
OK<uSSID1uMode1uRSSIval1> <uSSID2uMode2uRSSIval2>…<upto to the number of APs scanned>	*uSSID* (32 bytes, ASCII):SSID of the Access Point. Filler bytes (0x00) are put to complete 32 bytes, if actual SSID length is not 32 bytes. *uMode*(1 byte, hex):Security Mode of the scanned Access Point. 0x00 – Open (No Security) 0x01 – WPA 0x02 – WPA2 0x03 – WEP uRssival(1 byte, hex): Absolute value of the RSSI information, returned in hexadecimal, 1 byte. RSSI information indicates the signal strength of the Access Point The number of results is

Result Code	Description
	according to the value configured by the *at+rsi_numscan* command. If the *at+rsi_numscan* command is not issued, then the module returns the maximum number of APs it can scan.
ERROR<Error code>	Failure

The maximum number of Access Points that can be scanned by the module is 14 for RS9110-N-11-22(-xx) and RS9110-N-11-24(-xx) and 10 for RS9110-N-11-28(-xx). The SSIDs are returned in decreasing order of their RSSI.

Example1: If channel 6 is to be scanned, the command is

at+rsi_scan=6\r\n

…………………………..

```
0x61   0x74   0x2B   0x72   0x73   0x69   0x5F   0x73   0x63   0x61   0x6E
0x3D   0x36   0x0D   0x0A
```

Example2: To scan a hidden SSID "Redpine_net" in channel 12, the command is

at+rsi_scan=12,Redpine_net\r\n

…………………………..

```
0x61   0x74   0x2B   0x72   0x73   0x69   0x5F   0x73   0x63   0x61   0x6E
0x3D   0x31   0x32   0x2C   0x52   0x65   0x64   0x70   0x69   0x6E
0x65   0x5F   0x6E   0x65   0x74   0x0D   0x0A
```

Example3: If there are two networks with the SSID "Redpine_net1" and "Redpine_net2", both in WPA2 and measured RSSI of -20 dBm and -14 dBm respectively, the return value is

O K R e d p i n e _ n e t1 <uMode1> <uRSSI1> R e d p i ne _ n e t2 <uMode2> <uRSSI2> \r\n

……………………………………………

```
0x61   0x74   0x52   0x65   0x64   0x70   0x69   0x6E   0x65   0x5F   0x6E
0x65   0x74   0x31   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00
0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00
0x00   0x02   0x14   0x52   0x65   0x64   0x70   0x69   0x6E   0x65   0x5F
0x6E   0x65   0x74   0x32   0x00   0x00   0x00   0x00   0x00   0x00   0x00
0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00
0x00   0x00   0x02   0x0E   0x0D   0x0A
```

3.2.6 Next Scan Results

Description

This command is used in conjunction with the *at+rsi_numscan* and *at+rsi_scan* commands. It is valid only if the *at+rsi_numscan* command has been used to configure the number of scan results to be returned by the module. The module responds by returning the scanned Access Points. For example if *at+rsi_numscan* command is set to 5, then the *at+rsi_scan* command issued and there are 12 access points scanned by the module, then the *at+rsi_scan* command will return to the host the information about the first 5 access points out of the 12 scanned nodes, in order of decreasing RSSI . If an *at+rsi_nextscan* command is given after the scan command, it will return to the host the information about the next 5 access points in order of decreasing RSSI. If another *at+rsi_nextscan* command is given again, it will return to the host the information about the remaining 2 access points. Finally, if an *at+rsi_nextscan* command is given again, OK\r\n would be returned. In summary, the module will return the maximum number of access points that it can scan, but in multiple installments. This command should be issued only after the *at+rsi_scan* command.

Command

at+rsi_nextscan

Usage

at+rsi_nextscan\r\n

Parameters

No parameters

Response

Result Code	**Description**
OK<uSSID1uMode1uRSSIval1> <uSSID2uMode2uRSSIval2>…	*uSSID* (32 bytes, ASCII):SSID of the Access Point. Filler bytes (0x00) are put to complete 32 bytes, if actual SSID length is not 32 bytes. *uMode*(1 byte, hex):Security Mode of the scanned Access Point. 0x00 – Open (No Security) 0x01 – WPA 0x02 – WPA2 0x03 – WEP uRssival(1 byte, hex): Absolute value of the RSSI

Result Code	Description
	information, returned in hexadecimal, 1 byte. RSSI information indicates the signal strength of the Access Point
ERROR<Error code>	Failure

3.2.7 Query MAC Address of Scanned APs

Description

This command returns the MAC addresses of the Access Points returned by the "Scan" command. This command should be issued after "Scan" command to retrieve MAC addresses of the scanned networks.Response of this command will be affected only when at+rsi_scan command is given.

Command

at+rsi_bssid

Usage

at+rsi_bssid?

Parameters

None

Response

Result Code	Description
OK<SSID1BSSID1><SSID2BSSID2>…	*SSID* (32 bytes, ASCII): SSID of the scanned AP. Filler bytes (0x00) are put to complete 32 bytes, if SSID length is not 32 bytes. *BSSID*(6 bytes, hex): MAC ID of the scanned AP
ERROR<Error code>	Failure

For example, if there is an infrastructure network "Redpine_net", that is already scanned by the "Scan" command, has a MAC ID of 0x00 0x23 0xA7 0x1B 0x8D 0x31 then

at+rsi_bssid?\r\n

………………………………….

```
0x61    0x74   0x2B   0x72    0x73   0x69   0x5F  0x62  0x73  0x73  0x69  0x64
0x3F   0x0D   0x0A
```

OK Redpine_net <MAC ID> \r\n

……………………………………………………

```
0x4F      0x4B   0x52   0x65   0x64   0x70   0x69   0x6E   0x65   0x5F   0x6E
0x65 0x74    0x00  0x00  0x00  0x00   0x00   0x00   0x00   0x00   0x00   0x00
0x00 0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00   0X00   0x00
0x23  0xA7  0x1B  0x8D  0x31  0x0D   0x0A
```

Similarly, if there are two networks returned by the at+rsi_scan command, then at+bssid? command will return the two SSIDs with the corresponding MAC IDs.

3.2.8 Query Network Type

Description

This command returns the network type of the scanned AP. This command can be issued after the "Scan" command. Response of this command will be affected only when at+rsi_scan command is given.

Command

at+rsi_nwtype

Usage

at+rsi_nwtype?

Parameters

None

Response

Result Code	**Description**
OK<SSID1><NwType1><SSID2><NwType2>…<upto the number of scanned networks>	*SSID*(32 bytes, ASCII): SSID of the scanned AP. Filler bytes (0x00) are put to complete 32 bytes, if SSID length is not 32 bytes. *NwType*(1 byte, hex): Network type indication. 0x01 - Infrastructure

Result Code	Description
	0x00 - Ad-hoc
ERROR<Error code>	Failure

For example, if there is an infrastructure network "Redpine_net", that is already scanned by the at+rsi_scan command, then

at+rsi_nwtype?r\n

…………………………………..

```
0x61   0x74  0x2B  0x72   0x73  0x69  0x5F  0x6E  0x77  0x74  0x79
0x70  0x65  0x3F  0x0D  0x0A
```

OK Redpine_net <Network Type> \r\n

………………………………………………

```
0x4F     0x4B  0x52  0x65  0x64  0x70  0x69  0x6E  0x65  0x5F  0x6E
0x65 0x74   0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x00  0x00 0x00  0x00   0x00   0x00   0x00   0x00   0x00   0X00   0x00   0x01
0x0D  0x0A
```

Similarly, if there are two networks returned by the at+rsi_scan command, then at+rsi_nwtype? command will return the two SSIDs with the corresponding network type.

3.2.9 Set Network Type

Description

This command configures the type of network the module wishes to join. This command is also given before creating an ad-hoc (IBSS)[1] network in creator mode.

Command

at+rsi_network

Usage

at+rsi_network=network_type \r\n

Parameters

network_type: Type of the network to be created/joined

INFRASTRUCTURE – For infrastructure mode

IBSS,type,chan_num - Ad-Hoc mode, security disabled

IBSS_SEC,type,chan_num - Ad-Hoc mode security enabled. Only WEP security is supported

[1] Ad-hoc feature does not support DHCP

INFRASTRUCTURE/IBSS is sent as an ASCII string.

type: 0 indicates IBSS Joiner and 1 indicates IBSS Creator

chan_num: Indicates the channel in which the IBSS network has to be created. If the module is intended to be an ad-hoc joiner, this field can be set to '0' or to the specific channel in which the creator is present.

For 2.4 GHz

Actual Channel Number	chan_num parameter
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

For 5 Ghz

Channel Number	chan_num parameter
36	1
40	2
44	3
48	4
52	5
56	6
60	7
64	8
100	9
104	10
108	11
112	12
116	13
120	14
124	15
128	16

132	17
136	18
140	19
149	20
153	21
157	22
161	23
165	24

Example 1: In order to create an open network in channel 11 in Ad-hoc (IBSS) mode, the command (to be issued before the join command) is

at+rsi_network=IBSS,1,11\r\n

…………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6E 0x74 0x74 0x77 0x6F 0x72 0x6B 0x3D 0x49 0x42 0x53 0x53 0x2C 0x31 0x2C 0x31 0x31 0x0D 0x0A

Example 2: In order to join an open network in channel 11 in Ad-hoc (IBSS) mode, the command (to be issued before the join command) is

at+rsi_network=IBSS,0,11\r\n

…………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6E 0x74 0x74 0x77 0x6F 0x72 0x6B 0x3D 0x49 0x42 0x53 0x53 0x2C 0x30 0x2C 0x31 0x31 0x0D 0x0A

Example 3: In order to create a WEP enabled network in channel 11 in Ad-hoc (IBSS) mode, the command (to be issued before the join command) is

at+rsi_network=IBSS_SEC,1,11\r\n

…………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6E 0x74 0x74 0x77 0x6F 0x72 0x6B 0x3D 0x49 0x42 0x53 0x53 0x5F 0x53 0x45 0x43 0x2C 0x31 0x2C 0x31 0x31 0x0D 0x0A

Example 4: In order to join a WEP enabled network in channel 11 in Ad-hoc (IBSS) mode, the command (to be issued before the "join" command) is

at+rsi_network=IBSS_SEC,0,11\r\n

…………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6E 0x74 0x74 0x77 0x6F 0x72 0x6B 0x3D 0x49 0x42 0x53 0x53 0x5F 0x53 0x45 0x43 0x2C 0x30 0x2C 0x31 0x31 0x0D 0x0A

Example 5: In infrastructure mode, the command is at+rsi_network=INFRASTRUCTURE\r\n

…………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6E 0x74 0x74 0x77 0x6F 0x72 0x6B 0x3D 0X49 0X4E 0X46 0X52 0X41 0X53 0X54 0X52 0X55 0X43 0X54 0X55 0X52 0X45 0x0D 0x0A

Response

Result Code	Description
OK	Successful execution of the command
ERROR<Error code>	Failure

3.2.10 Pre Shared Key

Description

This command configures the PSK (Pre Shared Key) that is used for creating secured access. The PSK is also used as the WEP key in addition to WPA/WPA2-PSK (AES and TKIP). This command should be issued to the module before the Join command if the Access Point is in secure mode. This command can be ignored if the Access Point is in Open mode.

Command

at+rsi_psk

Usage

at+rsi_psk=pre_shared_key\r\n

Parameters

This command is used to configure the pre shared key.

In the case of WPA/WPA2 modes, the Pre-shared key can be any combination of ASCII characters up to a length of 63 characters[1] (refer to Bit[3] of Feature Select command). In the case of WEP, there are two modes in which the Access Point can be set:

WEP (Hex Mode): In this mode, the key to be supplied to the AP should be 10 digits (for 64 bit WEP mode) or 26 digits (for 128 bit WEP mode), and only the following digits are allowed for the key: A,B,C,D,E,F,a,b,c,d,e,f,0,1,2,3,4,5,67,8,9.

In this mode the exact key can be supplied to the module. For example, if the key is ABCDE12345, then the command is

[1] A maximum of 63 bytes is supported in firmware version 4.5.0 and upwards. For earlier versions, a maximum of 32 bytes is supported.

at+rsi_psk=ABCDE12345\r\n

WEP (ASCII Mode): In this mode, the key to be supplied to the AP should be 5 characters (for 64 bit WEP mode) or 13 characters (for 128 bit WEP mode). All ASCII characters are allowed.

In this mode the ASCII code of the key should be supplied to the module. For example, if the key is AB12#, then the ASCII representation is 0x41 0x42 0x31 0x32 0x23, and the command is

at+rsi_psk=4142313223\r\n

Response

Result Code	Description
OK	Successful execution of the command
ERROR<Error code>	Failure

Example 1: To join a WPA2 security enabled network with key "12345ABCDE", the command is

at+rsi_psk=12345ABCDE\r\n

………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x70 0x73 0x6B 0x3D 0x31 0x32 0x33 0x34 0x35 0x41 0x42 0x43 0x44 0x45 0x0D 0x0A

Example 2: To join a WEP-64 security (Hex) enabled network with key "0123456789", the command is

at+rsi_psk=0123456789\r\n

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x70 0x73 0x6B 0x3D 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x0D 0x0A

Example 3: To join a WEP-128 security (Hex) enabled network with key "0123456789ABCDEFGHIJ012345", the command is

at+rsi_psk=0123456789ABCDEFGHIJ012345\r\n

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x70 0x73 0x6B 0x3D 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x30 0x31 0x32 0x33 0x34 0x35 0x0D 0x0A

3.2.11Set WEP Key

Description

This command is issued to supply WEP keys to the module, if the AP is in WEP mode. In some APs, Four WEP keys (with indices 1, 2, 3, 4) can be configured. This command can be used to supply the right key to the module in such a case. It is an optional command and need not be issued if the AP is configured with the WEP key of index 1.

Command

at+rsi_wep_keys

Usage

at+rsi_wep_keys=key_index,key2,key3,key4\r\n

Parameters

key_index: Key index to be used to select the key. Allowable values are 0,1, 2, and 3. If this value is 0, the key supplied in the *at+rsi_psk* command is used. If the value is 1, key2 in this command is used; if the value is 2, key3 in this command is used; if the value is 3, the key4 in this case is used.

Key2, key3, key4 – 2nd, 3rd, & 4th keys of AP respectively

Response

Result Code	**Description**
OK	Successful execution of the command
ERROR<Error code>	Failure

Example: at+rsi_wep_keys=2,ABCDE12345,ABCDE12346,ABCDE12347\r\n

3.2.12Authentication Mode

Description

This command configures the mode of security during the WLAN connection setup. It should be issued before the “Join” command.

Command

at+rsi_authmode

Usage

at+rsi_authmode=authvalue\r\n

Parameters

Authvalue:

0– WEP Open Key Authentication

1– WEP Shared Key Authentication

2– WPA

3- WPA2

4- No security (open)

The relevance of this command changes with Bit[11] of the "Feature Select" command.

Bit[11]=0:

The command is not a mandatory command if the AP is in WPA/WPA2 or open mode. If the AP is in WEP mode, it must be issued to set the mode to either WEP-open or WEP-shared.

Bit[11]=1:

The command is a mandatory command, and should be issued before the "Join" command, irrespective of the security mode of the AP. If the security mode of the AP to connect to, does not match that supplied to the module, Join would fail.

Response

Result Code	Description
OK	Successful execution of the command
ERROR<Error code>	Failure

Example: at+rsi_authmode=1\r\n

………………………………………….

```
0x61  0x74  0x2B  0x72  0x73  0x69  0x5F  0x61  0x75  0x74
0x68  0x6D  0x6F  0x6F  0x64  0x65  0x3D  0x31  0x0D  0x0A
```

NOTE: If the module and AP are configured in WEP-open mode and a wrong PSK is supplied to the module before Joining to the AP, no error messages indicating this will be given. The module will join to the AP but data transfer will fail. If the module and AP are configured in WEP-shared mode and a wrong PSK is supplied to the module, the join operation will fail and an ERROR code -12 will be returned.

3.2.13 Join

Description

This command is used to join a network. The SSID is selected from the list of SSID's returned after performing scan operation. It is also used to create an IBSS network in creator mode.

Prerequisite

The developer is expected to execute "Set Network Type" and Pre Shared Key (if Access Point is in Secure mode) commands prior to the execution of this command

Command

at+rsi_join

Usage

at+rsi_join=SSID,TxRate,TxPower\r\n

Parameters

SSID: The SSID of the network. The maximum length of the SSID is 32 characters. This can be the SSID of the Access Point or Client to which the module has to connect to in Infrastructure or IBSS Joiner modes respectively. It can also be the SSID of the ad-hoc network that is to be created by the module.

TxPower: This fixes the Transmit Power level of the module. This value can be set as follows:

0 – Low power (7dBm)

1 – Medium power (10dBm)

2 – High power (16 to 17dBm)

TxRate: Rate at which the data has to be transmitted. Refer to the table below for the various data rates and the corresponding values.

TxPower and TxRate are optional parameters. If they are not supplied, the default values of '2' and '0' are taken for these parameters respectively.

Data Rate (Mbps)	**Value of uTxDataRate**
Auto-rate	0
1	1
2	2
5.5	3
11	4
6	5
9	6
12	7
18	8
24	9
36	10
48	11
54	12
MCS0	13
MCS1	14
MCS2	15
MCS3	16
MCS4	17
MCS5	18
MCS6	19
MCS7	20

Table 3: Data Rate Parameter

For Example, to join a network called “Redp”, at TX rate of 48 Mbps, and TX power of 10dBm, the command is

at+rsi_join=Redp,11,1\r\n

……………………………………………

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6A 0x6F 0x69 0x6E 0x3D 0x52 0x65 0x64 0x70 0x2C 0x31 0x31 0x2C 0x31 0x0D 0x0A

Response

Result Code	Description
OK	Successful execution of the command
ERROR<Error code>	Failure

3.2.14 Re-join

Description

The module tries to re-join to a network if it loses connection to the network it was associated with. If the re-join is successful, then the link is re-established. The module aborts the re-join after a fixed number of re-tries. If this happens, an asynchronous response "ERROR-22\r\n" is sent from the module to the Host. This is not a response to a command, but an asynchronous message sent to the Host if re-tries fail.

Message

ERROR-22\r\n

………………………

0x45 0x52 0x52 0x4F 0x52 0xEA 0x0D 0x0A

3.2.15 Disassociate[1]

Description

This command is issued to request the module to disassociate (disconnect) from an Access Point. The Host can then issue a fresh set of Scan, Join commands to connect to a different Access Point or the same Access Point with a different set of connection parameters. The Power Mode (0, 1 or 2) is retained from before the Disassociate command to the next association, i.e. if the Power Mode is 1 before the Disassociate command is issued, then, after the next Join command is successful, the module is in Power Mode 1 automatically.

Command

at+rsi_disassoc

Usage

at+rsi_disassoc\r\n

Parameters

[1] This command is not supported in ad-hoc mode. For disconnecting in IBSS mode, the module can be soft-reseted by at+rsi_reset, and then the command sequence from at+rsi_band should be given.

No Parameters

Response

Result Code	Description
OK	Successful execution of the command
ERROR<Error code>	Failure

3.3 Power Modes and commands[1]

The RS9110-N-11-2X module supports three power modes with the UART interface. The Host can switch among the power modes using the AT commands depending on the Wi-Fi connection status as defined in this section.

The power modes supported by the RS9110-N-11-2X module for UART interface are classified based on the Host's capability to negotiate with RS9110-N-11-2X and the Wi-Fi connection status. Please refer to the Wi-Fi commands section for details on the AT commands required to configure the module in different power modes.

3.3.1 Power mode 0

In this mode, power save is disabled. The module will be in Power Mode 0 by default.

3.3.2 Power mode 1

The RS9110-N-11-2X module can put the Baseband, RF and also the Core Control block to sleep in this mode.

The average amount of power consumed in this mode would depend on the sleep period and also on the time taken to exchange the SLEEP and ACK messages.

The functioning of the module in this mode depends on the connection status as explained below

Before Wi-Fi connection

In this state, the module is configured with a sleep timer through the at+rsi_sleeptimer command. The input to this command indicates the amount of time the module puts the Core Control Block to sleep. Once the timer expires, the module wakes up the Core Control block and sends the "SLEEP\r\n" , in upper case ASCII (0x53 0x4C 0x45 0x45 0x50 0x0D 0x0A) message to the Host. If the Host sends "ACK\r\n", in upper case ASCII (0x41 0x43 0x4B 0x0D 0x0A), the module will put to sleep the Core Control block for another interval of the sleep period. But, if the host wants to perform any Wi-Fi related activity viz., SCAN, JOIN, etc., it has to issue these commands accordingly to the module.

[1] Not supported in ad-hoc mode

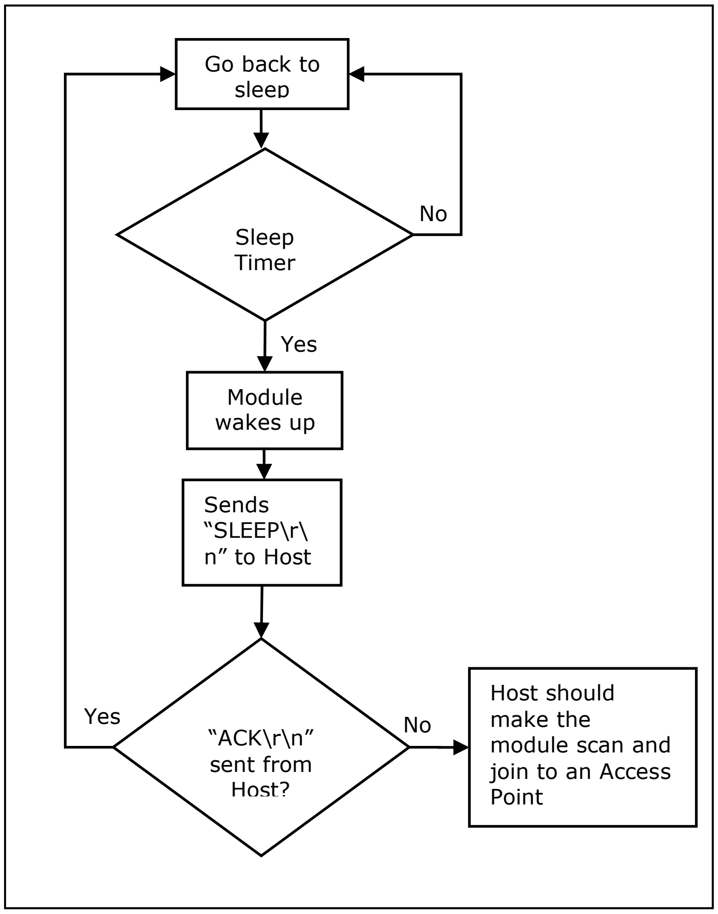

Go back to sleep
Sleep Timer
No
Yes
Module wakes up
Sends "SLEEP\r\n" to Host
"ACK\r\n" sent from Host?
Yes
No
Host should make the module scan and join to an Access Point

Figure 2: Power Save Mode1 before Connection to an AP

After Wi-Fi Connection

In this state, the RS9110-N-11-2X module periodically wakes up to receive DTIM (Delivery Traffic Indication Message) from the Access Point. Once it wakes up, it sends "SLEEP\r\n", in upper case ASCII (0x53 0x4C 0x45 0x45 0x50 0x0D 0x0A) message to host. If the host has data to be transmitted, it sends the data packet instead of "ACK\r\n" to the module through the UART interface. Once host finishes sending all packets, it can send "ACK\r\n" in upper case ASCII (0x41 0x43 0x4B 0x0D 0x0A) to allow the module to go to sleep. If the AP sends data to the module, all these packets will be given to the host before the module checks "ACK\r\n" message to go to sleep. Hence, even if "ACK\r\n" has already come from host, the module will not go to sleep if there are packets to be sent to Host.

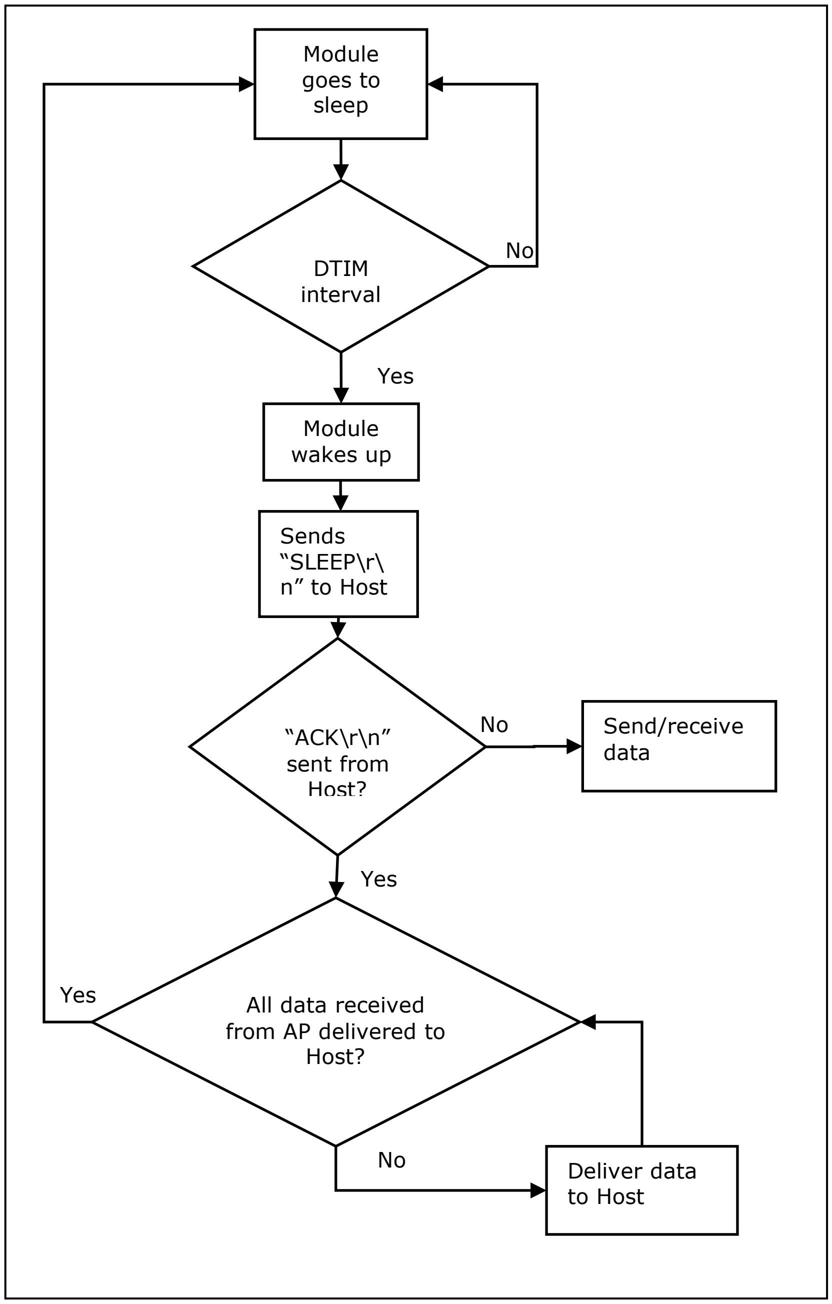


Figure 3: Power Save Mode1 after Connection to an AP

3.3.3 Power mode 2

The RS9110-N-11-2X module puts to sleep the Baseband and RF components in this mode. The Core Control block interacting with the Host is always functional. Hence, the module can receive commands from the host at any time and there is no SLEEP-ACK message exchange with the Host.

NOTE: The average current consumption in Power Mode 2 is higher than Power Mode 1.

The functioning of the module in this mode depends on the connection status as explained below

Before Wi-Fi connection

If this power mode is enabled before the Wi-Fi connection is established, the module powers off the Baseband and RF components until the Host reconfigures the module to Power Mode 0.

If the host wants to perform any Wi-Fi related activity viz., SCAN, JOIN, etc., it has to switch to Power Mode 0. This can be done at any time. The Host can switch back to Power Modes 1 or 2 after the Wi-Fi connection is established.

After Wi-Fi Connection

In this state, the RS9110-N-11-2X module gets information from the Access Point to which it is connected for any buffered data at every beacon. If there is no data from the host to be transmitted or received from the remote terminal or AP, it puts to sleep the Baseband and RF components of the module.

Power Mode Command

Description

This command configures the power mode, viz., Power Mode 0, Power Mode 1 or Power Mode 2 to be used by the RS9110-N-11-2X module.

Command

at+rsi_pwmode

Usage

at+rsi_pwmode=power_mode\r\n

Parameters

The valid values for the parameter of this command are as follows

0– Power Mode 0

1– Power Mode 1

2– Power Mode 2

Response

Result Code	**Description**
OK	Successful execution of the command
ERROR<Error code>	Failure

Example: To configure in power mode 1

at+rsi_pwmode=1\r\n

…………………………………………

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x70 0x77 0x6D 0x6F 0x64 0x65 0x3D 0x31 0x0D 0x0A

Sleep Timer

Description

This command configures the sleep timer which is used in Power Mode 1 when the module has not established the Wi-Fi connection.

Command

at+rsi_sleeptimer

Usage

at+rsi_sleeptimer=timer_value\r\n

Parameters

timer_value: The value of the timer in milliseconds. The maximum value is 10000 milliseconds.

For example to set the timer to 168 milliseconds, the command is

at+rsi_sleeptimer=168\r\n

…………………………………

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x73 0x6C 0x65 0x65 0x70 0x74 0x69 0x6D 0x65 0x72 0x3D 0x31 0x36 0x38 0x0D 0x0A

Response

Result Code	**Description**
OK	Successful execution of the command
ERROR<Error code>	Failure

3.3.4 Feature Select

Description

This command is used to enable some configurable features.

Usage

at+rsi_feat_sel=<bitval>\r\n

Parameters

bitval - Note that the default values of the bits described below is ‘0’.

Bit[0] – ‘1’ - Add DNS Server Address to the Response for Network Param Query command. This command is not used for DNS configuration, it is used to query DNS parameters using the *at+rsi_nwparams* command

‘0’ – Do not add DNS Server Address to the Response for Network Param Query.

Bit[1:2] – Always set to all ‘0’

Bit[3] –

‘1’- 63 byte mode enabled for WPA/WPA2-PSK. Keys that are equal to or less than 63 bytes are accepted in the *at+rsi_psk* command in this mode.

‘0’ – Only 32 byte mode enabled for WPA/WPA2-PSK. Keys that are equal to or less than 32 bytes are accepted in the at+rsi_psk command in this mode. Note: Even if this bit is not set to ‘1’, a PSK of less than or equal to 63 bytes will work successfully while associating to an access point, but there will be a wrong length of PSK returned in the command *at+rsi_nwparams*. It is advisable to set the bit to ‘1’ if the *at+rsi_nwparams* is planned to be used.

Bit[4:6] – Always set to all ‘0’

Bit[7] – If this bit is set to ‘1’, the WEP authentication mode,WEP index,WEP keys(2,3,4) are returned in *at+rsi_cfgget?* command.

Bit[8] – Always set to ‘0’

Bit[9] – Always set to ‘0’

Bit[10]- LED in RS9110-N-11-22-04/05 modules is disabled if this bit is set to ‘1’. The LED will stop glowing after the *at+rsi_init* command is issued.

Bit[11]- Controls the relevance of the command *at+rsi_authmode*.

This command can be issued after the *at+rsi_band* command. The decimal value of the *bitval* word is supplied as the input as shown in the example below. If the command is not issued, the default value of this parameter in the system would be ‘0’. The command is not applicable in IBSS mode.

Example 1: To turn off the LED to save current consumption, the word for *bitval* is “1 00000 00000’, the decimal value of which is 1024.

at+rsi_feat_sel=1024\r\n

……………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x66 0x65 0x61 0x74 0x5F 0x73 0x65 0x6C 0x3D 0x31 0x30 0x32 0x34 0x0D 0x0A

Response

Result Code	Description
OK	Successful execution of the command. “OK” is returned in ASCII
ERROR<Error code>	Failure

3.4 TCP/IP Commands

The TCP/IP AT command set represents the control frames that are sent from the Host to the RS9110-N-11-2X to access the TCP/IP stack.

The following are the RS9110-N-11-2X TCP/IP configuration AT commands.

3.4.1 Set IP Parameters[1]

Description

This command configures the IP address, subnet mask and default gateway of the TCP/IP stack in the RS9110-N-11-2X module.

Command

at+rsi_ipconf

Usage

at+rsi_ipconf=*dhcp_mode*,*IPaddr*,*subnet*,*gateway*\r\n

Parameters

dhcp_mode: Used to configure IP address in manual or DHCP modes.

0– Manual

1– DHCP

IPaddr: IP address in dotted decimal format. This can be 0’s in the case of DHCP.

subnet: Subnet mask in dotted decimal format. This can be 0’s in the case of DHCP.

gateway: Gateway in the dotted decimal format. This can be 0’s in the case of DHCP.

Example 1: To configure in DHCP disabled mode, with 192.168.1.3, 255.255.255.0 and 192.168.1.1 as the IP address, subnet mask and gateway the command is

at+rsi_ipconf=0,192.168.1.3,255.255.255.0,192.168.1.1\r\n

………

[1] DHCP is not supported in ad-hoc mode

```
  0x61  0x74  0x2B  0x72  0x73  0x69  0x5F  0x69  0x70  0x63  0x6F
  0x6E  0x66  0x3D  0x30  0x2C  0x31  0x39  0x32  0x2E  0x31  0x36
  0x38  0x2E  0x31  0x2E  0x33  0x2C  0x32  0x35  0x35  0x2E  0x32
  0x35  0x35  0x2E  0x32  0x35  0x35  0x2E  0x30  0x2C  0x31  0x39
  0x31  0x2E  0x31  0x36  0x38  0x2E  0x31  0x2E  0x31  0x0D  0x0A
```

Example 2: To configure in DHCP mode, the command is

at+rsi_ipconf=1,0,0\r\n

………

```
0x61    0x74  0x2B  0x72  0x73  0x69  0x5F  0x69  0x70  0x63  0x6F
  0x6E 0x66   0x3D  0x31  0x2C  0x30  0x2C   0x30  0x0D  0x0A
```

Response

Result Code	Description
OK<MAC_Address><IPAddr><subnet><gateway>	*MAC_Address* (6 Bytes, hex): MAC address of the module *IPAddr* (4 Bytes, hex): IP address of the module *subnet*(4 Bytes, hex): Subnet mask *gateway*(4 Bytes, hex): Gateway address
ERROR<Error code>	Failure

3.4.2 Open and Connect to a TCP Socket

Description

This command opens a client TCP socket and attempts to connect it to the specified "port" on a server defined by "host". A listening socket should be created in the server before issuing this command, for connection to go through successfully.

Command

at+rsi_tcp

Usage

at+rsi_tcp=destIP,port,lport\r\n

Parameters

destIP – Destination IP Address of the target server

port – The target port

lport – Local Port on the RS9110-N-11-2X module (allowed range 1024 to 49151)

Response

Result Code	**Description**
OK<sock_handle>	Upon successfully opening and connecting the TCP socket to the Host port, a socket handle is returned. OK is returned as ASCII, while <sock_handle> in hexadecimal. The socket handle would be followed after OK. Socket handle is a 1 byte value, ranging from 1 onwards. Hence, the first socket opened will have a socket handle of 1.
ERROR<Error code>	Failure

Example 1

at+rsi_tcp=192.168.40.10,8000,1234\r\n

…………………………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x74 0x63 0x70 0x3D 0x31 0x39 0x32 0x2E 0x31 0x36 0x38 0x2E 0x34 0x30 0x2E 0x31 0x30 0x2C 0x38 0x30 0x30 0x30 0x2C 0x31 0x32 0x33 0x34 0x0D 0x0A

For a socket handle of 1 the response,
OK 1 \r\n
…………………….
0x4F 0x4B 0x01 0x0D 0x0A

3.4.3 Open a Listening UDP Socket

Description

This command opens a User Datagram Protocol (UDP) socket and binds to a specified port. The UDP socket waits for the data from the peer. This socket is not connected to any peer and is used if the user wants to receive/send data from/to any peer.

Command

at+rsi_ludp

Usage

at+rsi_ludp=lport\r\n

Parameters

lport – Local port on the RS9110-N-11-2X module

Response

Result Code	Description
OK<sock_handle>	Upon successfully opening and connecting the UDP socket to the host port, a socket handle is returned. The socket handle is followed after OK. OK is returned as ASCII, while <sock_handle> as a 1 byte value in hexadecimal.
ERROR<Error code>	Failure

Example:

at+rsi_ludp=8000\r\n

…………………………

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6C 0x75 0x64 0x70 0x3D 0x38 0x30 0x30 0x30 0x0D 0x0A

For a socket handle of 1 the response,
OK 1 \r\n
…………………….
0x4F 0x4B 0x01 0x0D 0x0A

3.4.4 Open a UDP Socket

Description

This command opens a User Datagram Protocol (UDP) socket and links to the remote system's specific host and port address. The UDP socket is virtually connected to the peer specified by the IP and the port.

Command

at+rsi_udp

Usage

at+rsi_udp=Iphost,rport,lport\r\n

Parameters

Iphost – IP Address of the Target server

rport – Target port (0 to 65535)

lport – Local port on the RS9110-N-11-2X module

Response

Result Code	**Description**
OK<sock_handle>	Upon successfully opening and connecting the UDP socket to the host:port, a socket handle is returned. The socket handle is followed after OK. OK is returned as ASCII, while <sock_handle> as a 1 byte value in hexadecimal
ERROR<Error code>	Failure

Input parameter delimiter

The input parameters are delimited by commas (0x2C in ASCII format) in this command.

Example:

at+rsi_udp=192.168.40.10,8000,1234\r\n

…………………………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x75 0x64 0x70 0x3D 0x31 0x39 0x32 0x2E 0x31 0x36 0x38 0x2E 0x34 0x30 0x2E 0x31 0x30 0x2C 0x38 0x30 0x30 0x30 0x2C 0x31 0x32 0x33 0x34 0x0D 0x0A

For a socket handle of 1 the response,
OK 1 \r\n
…………………….
0x4F 0x4B 0x01 0x0D 0x0A

3.4.5 Open a Listening TCP Socket

Description

This commands opens a TCP listening socket on the local IP address and the specified "port". Once the listening socket is open, it automatically accepts remote *connect* requests. The status of the connection on a listen socket can be queried by the AT+RSI_CTCP command. Only one connection can be established on a single invocation of this command.

If multiple connections on a port have to be established, then the same command has to be invoked another time.

For multiple connections on the same port:

a. Open the first LTCP socket in module (for example port no. 8001)
b. Socket handle returned for this socket would be 1.
c. Connect this socket to the remote peer socket

d. You can now open the second socket in module with the same port no. 8001
e. Socket handle returned for the new socket would be 2
f. Connect this socket to another remote peer socket

Command

at+rsi_ltcp

Usage

at+rsi_ltcp=port\r\n

Parameters

port – Listening port to be used by a remote system when connecting to the RS9110-N-11-2X module.

Response

Result Code	**Description**
OK<sock_handle>	Upon successfully opening and connecting a TCP listening socket, a socket handle is returned. The socket handle is followed after OK. OK is returned as ASCII, while <sock_handle> as a 1 byte value in hexadecimal.
ERROR<Error code>	Failure

at+rsi_ltcp=8000\r\n

…………………………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6C 0x74 0x63 0x70 0x3D 0x38 0x30 0x30 0x30 0x0D 0x0A

For a socket handle of 1 the response,
OK 1 \r\n
…………………….
0x4F 0x4B 0x01 0x0D 0x0A

NOTE: Redpine module has total of 7 sockets for UART modules and 6 sockets for SPI modules. User must use the socket handle returned from the socket creation commands.

Apart from these user provided sockets there are internal sockets reserved for HTTP client,DNS client, Wireless firmware upgrade/configuration, ARP and DHCP client.

3.4.6 Open a Multicast socket[1]

Description

This command opens a multicast socket.

Command

at+rsi_multicast

Usage

at+rsi_ multicast=Iphost,rport,lport\r\n

Parameters

Iphost – Multicast IP address

Rport – Target port (0 to 65535)[2]

Lport – Local port on the RS9110-N-11-2X module

Response

Result Code	**Description**
OK<sock_handle>	Upon successfully opening a multicast socket, a socket handle (1 byte, hex) is returned.
ERROR<Error code>	Failure

at+rsi_multicast=239.0.0.0,8000,8001\r\n

…………………………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6D 0x75 0x6C 0x74 0x69 0x63 0x61 0x73 0x74 0x3D 0x32 0x33 0x39 0x2E 0x30 0x2E 0x30 0x2E 0x30 0x2C 0x38 0x30 0x30 0x30 0x2C 0x38 0x30 0x30 0x31 0x0D 0x0A

For a socket handle of 1 the response,
OK 1 \r\n

[1] Not supported in RS9110-N-11-28-04
[2] To Receive Multicast packets from any peer in multicast group the Rport of the module should be always 00

…………………….
0x4F 0x4B 0x01 0x0D 0x0A

3.4.7 Query a Listening Socket's Active Connection Status

Description

This command retrieves handles of active socket connections established through the listening socket. This command can be issued when a listening/server TCP socket has been opened in the module, to get information about the **remote peer's client socket**.

Command

```
at+rsi_ctcp
```

Usage

```
at+rsi_ctcp=hn\r\n
```

Parameters

hn – TCP listening socket handle for an already open listening socket in the module.

Response

Result Code	Description
OK<hn><IP_Address><Port>	An active socket handle. <hn> is returned as two's complement. *hn* >=0: A handle to an active connected socket. Returned as hex. For example, if hn = 7, it is returned as 0x07 *hn* =-1: No connection has been established yet between module's listening socket and remote peer. Returned as two's complement (0xFF). IP_Address is the client/remote peer's IP address. Returned in hex, 4 bytes. Port is the port number on which the client/peer is exchanging data. Returned in hex, 2 bytes . For example, if the port number is 8001 (0x1F41), then first 0x41

Result Code	Description
	is returned, then 0x1F
ERROR<Error code>	Failure

Example 1:

at+rsi_ctcp=1\r\n

…………………………………………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x63 0x74 0x63 0x70 0x3D 0x31 0x0D 0x0A

Example 2:

OK<hn=7><IP_Address=192.168.40.10><Port=8001> \r\n

…………………….

0x4F 0x4B 0x07 0xC0 0xA8 0x28 0x0A 0x41 0x1F 0x0D 0x0A

Example 3:

OK<hn=-1> \r\n

…………………….

0x4F 0x4B 0xFF 0x0D 0x0A

3.4.8 Close a Socket

Description

This command closes a TCP/UDP socket in the module.

Command

at+rsi_cls

Usage

at+rsi_cls=hn\r\n

Parameters

hn: TCP/UDP socket handle of an already open socket.

For example, to close the socket with handle 1, the command is

at+rsi_cls=1\r\n

……………………..

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x62 0x6C 0x73 0x3D 0x31 0x0D 0x0A

Response

Result Code	Description

Result Code	**Description**
OK	Upon successfully closing the socket "OK" would be returned.
ERROR<Error code>	Failure

NOTE: In the case of TCP socket, when a remote peer closes the socket connection, the RS9110-N-11-2X module sends the "AT+RSI_CLOSE<hn>\r\n" response (hn is the socket handle) to the Host. This is an asynchronous message sent from module to host and not the response of a specific command. <hn> is sent in 1 byte hex. AT+RSI_CLOSE is returned in uppercase and ASCII format.

3.4.9 Send data to a Socket

Description

This command sends a byte stream to the socket specified by the socket handle. The maximum number of bytes that can be sent as a data stream in this command at a time is 1460 in case of a TCP socket and 1472 in case of a UDP socket.

Command

at+rsi_snd

Usage

at+rsi_snd=hn,reserved,Dip,Dport, data stream\r\n

Parameters

hn – TCP/UDP socket handle of an already open socket.

reserved – 1 byte. Set this to '0' always.

Dip – Destination IP Address. Should be '0' if transacting on a TCP socket

Dport – Destination Port. Should be '0' if transacting on a TCP socket

Data stream – A byte stream of size <*sz*> to be sent to the specified socket.

Response

Result Code	**Description**
OK<reserved>	1 bytes, hex. The Host should always ignore this value.

Result Code	**Description**
ERROR<Error code>	Failure. On a failure while sending the data on the TCP socket, if the error code indicates "TCP connection closed", then the RS9110-N-11-2X module closes the socket.

Input parameter delimiter

The input parameters are delimited by commas (0x2C in ASCII format) in this command.

For example to send a stream "abcdefghij" over a TCP socket

at+rsi_snd=1,0,0,0,abcdefghij\r\n

……………………………………………..

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x73 0x6E 0x64 0x3D 0x31 0x2C 0x30 0x2C 0x30 0x2C 0x30 0x2C 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x0D 0x0A

For example, for 250 bytes sent, the response is

OK 250\r\n

………………….

0x4F 0x4B 0xFA 0x00 0x0D 0x0A

For example to send a stream "abcdefghij" over a Multicast socket

at+rsi_snd=1,239.0.0.0,1900,2100,abcdefghij\r\n

……………………………………………..

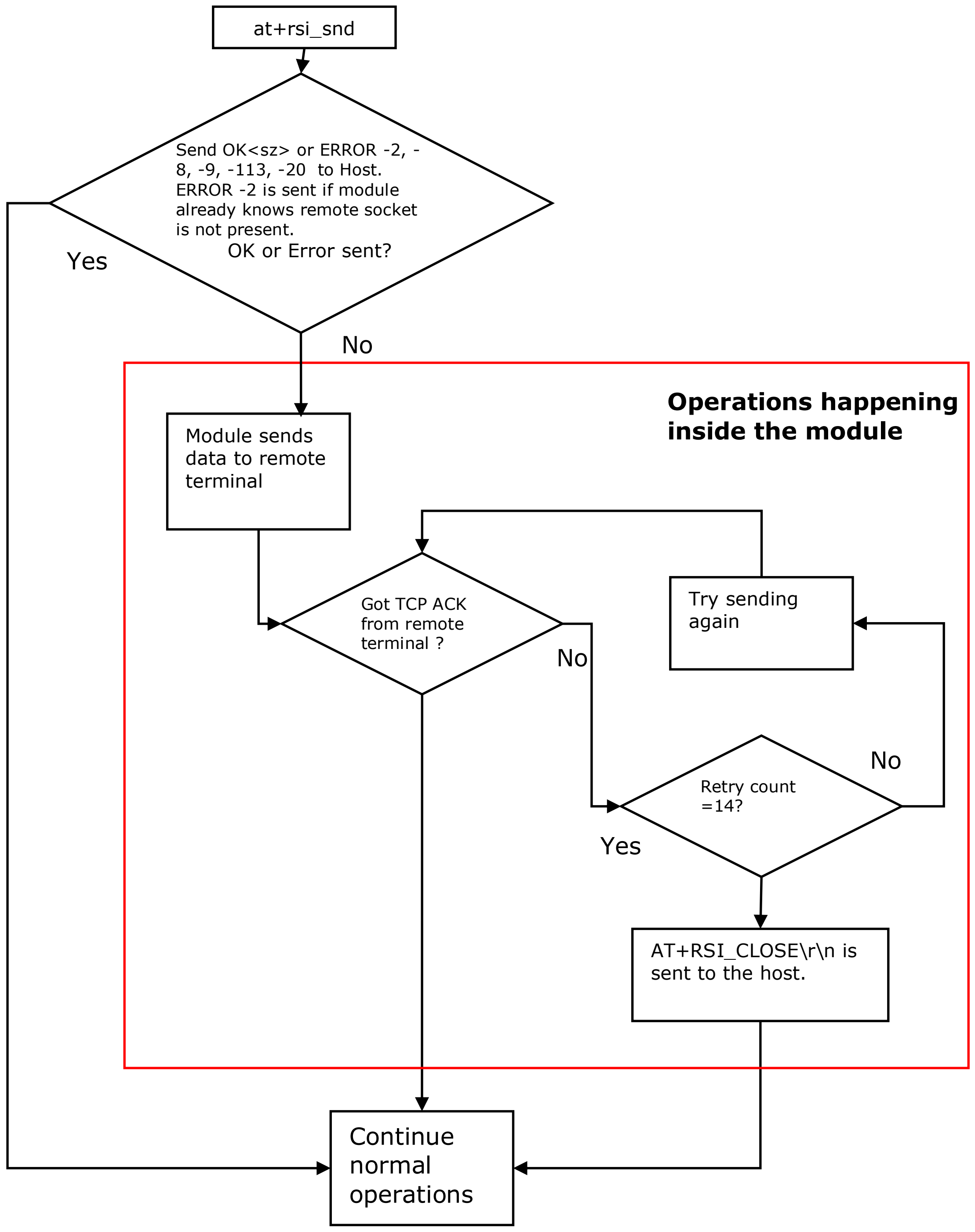
0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x73 0x6E 0x64 0x3D 0x31 0x2C 0x32 0x33 0x39 0x2E 0x30 0x2E 0x30 0x2E 0x30 0x2C 0x31 0x39 0x30 0x30 0x2C 0x32 0x31 0x30 0x30 0x2C 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x0D 0x0A

For example, for 250 bytes sent, the response is

OK 250\r\n

………………….

0x4F 0x4B 0xFA 0x00 0x0D 0x0A


at+rsi_snd
Send OK<sz> or ERROR -2, -8, -9, -113, -20 to Host. ERROR -2 is sent if module already knows remote socket is not present.
OK or Error sent?
Yes
No
Operations happening inside the module
Module sends data to remote terminal
Got TCP ACK from remote terminal ?
No
Try sending again
Retry count =14?
No
Yes
AT+RSI_CLOSE\r\n is sent to the host.
Continue normal operations

Some common reasons for not getting a TCP ACK from the remote terminal:

- Abnormal destruction of the remote network, plugging out network cables etc.

Figure 4: Send Operation

NOTE on Byte Stuffing: The '\r\n' character sequence (0x0D, 0x0A in hex) is used to indicate the termination of an AT command. If the actual data to be sent from Host comprises of \r\n characters in sequence, the host should replace this set of characters with (0xDB) and (0xDC). If (0xDB) itself is part of the data then (0xDB 0xDD) has to be sent. If (0xDB 0xDC) itself is part of the data then (0xDB 0xDD 0xDC) has to be sent. If either 0xDD or 0xDC is not sent after 0xDB, then an error (-9) is sent.

Example 1 : If **0x41 0x42 0x43 0x0D 0x0A** is the actual data stream that needs to be sent then the command is

at+rsi_snd <hn> <sz=5> <Dip> <Dport> < 0x41> <0x42> <0x43> <0xDB> <0xDC> <0x0D> <0x0A>

Example 2 : If **0x41 0x42 0x43 0x0D 0x0A 0x31 0x32** is the actual data stream that needs to be sent then the command is

at+rsi_snd <hn> <sz=7> <Dip> <Dport> < 0x41> <0x42> <0x43> <0xDB> <0xDC> <0x31> <0x32> <0x0D> <0x0A>

Example 3 : If **0x41 0x42 0x43 0xDB 0x31 0x32** is the actual data stream that needs to be sent then the command is

at+rsi_snd <hn> <sz=7> <Dip> <Dport> < 0x41> <0x42> <0x43> <0xDB> <0xDD> <0x31> <0x32> <0x0D> <0x0A>

Example 4: If **0x41 0x42 0x43 0xDB 0xDC 0x31 0x32** is the actual data that needs to be transmitted, then the command is

at+rsi_snd <hn> <sz=8> <Dip> <Dport> <0x41> <0x42> <0x43> <0xDB><0xDD><0xDC> <0x31><0x32> <0x0D> <0x0A>

Example 5: If **0x41 0x42 0x43 0x0D 0x0A 0xDB 0x31 0x32** is the actual data that needs to be transmitted, then the command is

at+rsi_snd <hn> <sz=9> <Dip> <Dport> <0x41> <0x42> <0x43> <0xDB><0xDC> <0xDB> <0xDD> <0x31><0x32> <0x0D> <0x0A>

Example 6: If **0x41 0x42 0x43 0x0D 0x0A 0xDB 0xDC 0x31 0x32** is the actual data that needs to be transmitted, then the command is

at+rsi_snd <hn> <sz=10> <Dip> <Dport> <0x41> <0x42> <0x43> <0xDB><0xDC> <0xDB> <0xDD> <0xDC> <0x31><0x32> <0x0D> <0x0A>

at+rsi_snd is the only command that requires byte stuffing to be done by the Host before sending to the module. There are NO other commands (from Host to module) that require byte stuffing. There are NO responses (from module to Host) that are byte stuffed by module before giving to Host.

3.4.10 Receive data on a Socket

Description

The RS9110-N-11-2X module delivers the data obtained on a socket to the Host with this message. This is an asynchronous operation and NOT one that the host can invoke in the module. It is sent from the module to the host when the module receives data from a remote terminal. The data received by the module is sent to the host through this command.

Message

AT+RSI_READ (returned in upper case)

Usage

AT+RSI_READ<hn><sz><Sip><Sport><stream>\r\n

Parameters

hn (1 byte) – TCP/UDP socket handle of an already open socket. Returned in Hex.

Sz (2 bytes) – Number of bytes received. Sz = 0 indicates remote termination for a TCP socket. Returned in hex. For example, 900 bytes (0x0384) would be sent as <0x84> <0x03>

Sip (4 bytes) – Source IP address. Returned in hex. For example, if the IP address is 192.168.1.1, then <0xC0><0xA8><0x01><0x01> is returned

Sport (2 bytes) – Source port. Returned in hex. For example, if the port number is 8001 (0x1F41) , it is returned as <0x41> <0x1F>

Stream – data stream, returned in ASCII. Example, "abcd" would be sent as <0x61><0x62><0x63><0x64>. A maximum of 1460 bytes can be received through a TCP socket and 1472 bytes through a UDP socket, in one AT+RSI_READ response. When the module sends data to the Host, byte stuffing is NOT done by the module, and the data is sent as it was received by the module from the wireless media. The <sz> field should be used to know how many bytes of valid data is expected.

Example 1, if 'abcd' is sent from remote terminal to module, on an UDP socket with handle 1, from source ip 192.168.1.20 and source port 8001, the module sends the following response to the host.

AT+RSI_READ 1 4 192 168 1 1 8001 abcd \r\n

………………………..

0x41 0x54 0x2B 0x52 0x53 0x49 0x5F 0x52 0x45 0x41 0x44 0x01 0x04 0x00 0xC0 0xA8 0x01 0x01 0x41 0x1F 0x61 0x62 0x63 0x64 0x0D 0x0A

Example 2, if 'abcd' is sent from remote terminal to module, on a TCP socket with handle 1, the module sends the following response to the host.

AT+RSI_READ 1 4 abcd \r\n

………………………..

0x41 0x54 0x2B 0x52 0x53 0x49 0x5F 0x52 0x45 0x41 0x44 0x01 0x04 0x00 0x61 0x62 0x63 0x64 0x0D 0x0A

NOTE: The data delivered to the Host on receiving data on a TCP socket does not include the source IP address and source port (Sip and Sport).

3.4.11 DNS Commands[1]

DNS Query

Description

This command is used to configure the DNS server's IP address when the DNS is used in DHCP disable mode. This command should be used before DNS query command. This command should be sent before sending at+rsi_ipconf command.

Command

at+rsi_dnsserver

Usage

at+rsi_dnsserver=<ipaddress_of_the_DNS_server>\r\n

Parameters

ipaddress_of_the_DNS_server - This is the IP address of the DNS server to which the query is sent.

Response

Result Code	**Description**
OK	DNS server IP address configured successfully
ERROR<Error code>	Failure

For example to send a query to a DNS server with IP address 192.168.10.20

[1] Not supported in RS9110-N-11-28-04

at+rsi_dnsserver=192.168.10.20\r\n

……………………………………………..

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x64 0x6E 0x73 0x73 0x65 0x72 0x76 0x65 0x72 0x3D 0x31 0x39 0x32 0x2E 0x31 0x36 0x38 0x2E 0x31 0x30 0x2E 0x32 0x30 0x0D 0x0A

For success, the response is

OK\r\n

………………….

0x4F 0x4B 0x0D 0x0A

DNS Resolution

Description

This command is used to send the domain name of a website to the module to resolve the IP address. The domain name should be maximum of 134 bytes.

Command

at+rsi_dnsget

Usage

at+rsi_dnsget=<domain_name>\r\n

Parameters

domain_name - This is the domain name of the target website.

Response

Result Code	**Description**
OK<no. of IP addresses(1 byte)><IP address1(4 bytes)><IP address2(4 bytes)>…<IP address10(4 bytes)>	Returns the number of IP addresses (in hex), and the individual IP addresses (in hex), up to a maximum of 10.
ERROR<Error code>	Failure

For example to resolve the IP address of www.google.com, send the command as below

at+rsi_dnsget=www.google.com\r\n

……………………………………………..

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x64 0x6E 0x73 0x67 0x65 0x74 0x3D 0x77 0x77 0x77 0x2E 0x67 0x6F 0x6F 0x67 0x6C 0x65 0x3E 0x63 0x6F 0x6D 0x0D 0x0A

For example, for 3 IP addresses, the response is

OK 3 192 168 10 10 192 168 10 20 192 168 10 30 \r\n

………………….

0x4F 0x4B 0x03 0xC0 0xA8 0x0A 0x0A 0xC0 0xA8 0x0A 0x14 0xC0 0xA8 0x0A 0x1E 0x0D 0x0A

DNS Command Sequence

Case 1: If manual IP configuration is used, then the following commands should be given in sequence.

at+rsi_dnsserver

at+rsi_ipconfig (with DHCP disabled)

at+rsi_dsnget

Case 2: If DHCP enabled mode for IP configuration is used, then the following commands should be given in sequence.

at+rsi_ipconfig(DHCP enabled)

at+rsi_dnsget

3.5 Other Commands

These commands are in addition to the Wi-Fi and TCP commands.

3.5.1 Query Firmware Version

Description

This command is used to retrieve the firmware version in the module.

Command

at+rsi_fwversion

Usage

at+rsi_fwversion?\r\n

Parameters

None

Response

Result Code	Description
OKMajor.Minor1.Minor2	The firmware version follows after OK. The version is indicated by 3 bytes. Each byte is separated by a dot. Major number (1 Byte), Minor number1 (1 Byte), Minor number2 (1 Byte). All are returned in ASCII.
ERROR<Error code>	Failure

Example, if the firmware version is 1.2.3, then the response is

OK 1.2.3\r\n

………………….

0x4F 0x4B 0x31 0x2E 0x32 0x2E 0x33 0x0D 0x0A

3.5.2 Query Network Parameters

Description

This command is used to retrieve the WLAN connection and IP parameters. This command can be sent only after the connection to the Access Point is successful.

Command

at+rsi_nwparams

Usage

at+rsi_nwparams?\r\n

Parameters

None

Response

Result Code	Description

OK<SSID><SEC_TYPE><PSK><Chn_No><MAC_ADDR><DHCP_MODE/ERROR_CODE><IP_Addresss><SUBNET><GATEWAY><NO_OF_OPEN_SOCKET> **[**<S_ID1><S_TYPE1><Sport1><Dport1><Dip1>**]** **[**<S_ID2><S_TYPE2><Sport2><Dport2><Dip2>**]** … up to the number indicated by <NO_OF_OPEN_SOCKET><DNS_Server_Address>	String of response parameters follows after OK. Please refer to the next table for description of each parameter.
ERROR<Error code>	Failure

Response Parameter	**Description**
SSID	This is 32 bytes and is the SSID of the Access Point to which the module is connected. Sent in ASCII. If the AP does not have a 32 character SSID, then filler bytes 0x00 are sent to complete 32 bytes of return value. This parameter contains dummy bytes in IBSS mode (with module as creator) if no Wi-Fi nodes have joined the network
SEC_TYPE	This is 1 byte, sent in hex, and indicates the mode of WLAN connection (Secure or Open). 0x00 – Open (No security) 0x01 – WPA 0x02 – WPA2 0x03 – WEP This parameter contains dummy bytes in IBSS mode (with module as creator) if no Wi-Fi nodes have joined the network
PSK	This is 32 bytes if Bit[3] = '0' ; 63 bytes if Bit[3] = '1' in the command. Refer to Bit[3] in **Feature Select** command. It is sent in ASCII and indicates the Pre-Shared Key that is configured from the Host to

Response Parameter	Description
	connect to the Access Point. If the connection to the Access Point is in Open mode, then these bytes will be zeros. If the actual key is less than 32/64 characters long, then filler bytes 0x00 are added to complete the length of the return value. This parameter contains dummy bytes in IBSS mode (with module as creator) if no Wi-Fi nodes have joined the network.
Chn_No	This is 1 byte, returned in hex and indicates the channel number in which the module is connected to the Access Point. This parameter contains dummy bytes in IBSS mode (with module as creator) if no Wi-Fi nodes have joined the network
MAC_ADDR	This is 6 bytes and is the MAC Address of the module. Returned in hex.
DHCP_MODE	This is 1 byte, returned in hex. It indicates the module is connected to the Access Point through DHCP or Manual IP configuration. 0x00 – Manual 0x01 – DHCP
IP_Address	This is 4 bytes and indicates the IP Address of the module. Returned in hex. For example, if IP address is 192.168.50.2, the value returned is 0xC0 0xA8 0x32 0x02
SUBNET	This is 4 bytes and indicates the Subnet Mask address of the module. Returned in hex. For example, if subnet address is 192.168.50.1, the value returned is

Response Parameter	Description
	0xC0 0xA8 0x32 0x01
GATEWAY	This is 4 bytes and indicates the Gateway address of the module. Returned in hex. For example, if gateway is 255.255.255.0, the value returned is 0xFF 0xFF 0xFF 0x00
NO_OF_OPEN_SOCKETS[1]	This is 1 byte and indicates the number of open sockets in the module. Returned in hex. This value ranges from 0x00 to 0x08. Default value is 0x01 since wireless firmware upgrade uses a Socket always.
Socket details	The socket details are mentioned below as a separate section
DNS_Server_Address	IP address of the DNS server. Please refer to the command **Feature Select** . Returned in 4 bytes hex. For example, if IP address of the DNS server is 192.168.50.2, the value returned is 0xC0 0xA8 0x32 0x02. Note that this response is optional, it will not be present if Bit[0] =' 0' Refer to Bit[0] in **Feature Select** command . Also, it is not present in case of RS9110-N-11-28-04 modules

The rest of the parameters sent in the response are repeated for each open socket. The number of such sets is equal to the NO_OF_OPEN_SOCKETS parameter.

These parameters are explained below.

Response Parameter	Description
S_ID	This is 1 byte and indicates the socket handle. Returned in hex.
S_TYPE	This is 1 byte and indicates the type of socket. 0– TCP 1– UDP

Response Parameter	Description
	2- LTCP 3– Multicast 4-LUDP Returned in hex.
Sport	This is 2 bytes and indicates the source port of the open socket. Returned in hex
Dport	This is 2 bytes and indicates the destination port for the open socket. Returned in hex.
Dip	This is 4 bytes and indicates the destination IP of the open socket. Returned in hex.

[1]The First set of socket details are the defaut values associated with wireless firmware upgrade

3.5.3 Soft Reset

Description

This command acts as a software reset to the module. The module will reset all information regarding the WLAN connection and IP configuration after receiving this command. The baud rate is the only setting which is not changed. The Host has to start from the AT+RSI_BAND command after issuing this command.

Command

```
at+rsi_reset
```

Usage

```
at+rsi_reset\r\n
```

Parameters

None

Response

Result Code	Description
OK	Success
ERROR<Error code>	Failure

3.5.4 Query MAC Address of Module

Description

This command is used to retrieve the MAC address of the module.

Command

at+rsi_mac?

Usage

at+rsi_mac?\r\n

Parameters

NA

Response

Result Code	**Description**
OK<MAC Address>	The MAC address follows after OK. The MAC address is 6 bytes long, returned in hex.
ERROR<Error code>	Failure

Example:

at+rsi_mac?\r\n

…………………………………….

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6D 0x61 0x63 0x3F 0x0D 0x0A

If the MAC ID is 0x00 0x23 0xA7 0x1B 0x8D 0x31 0x0D 0x0A, then the response is

OK <MAC ID=0x00 0x23 0xA7 0x1B 0x8D 0x31 0x0D 0x0A > \r\n

…………………………………………………………

0x4F 0x4B 0x00 0x23 0xA7 0x1B 0x8D 0x31 0x0D 0x0A

3.5.5 Query RSSI value

Description

This command is used to get the signal strength of the Access Point or network that the module is connected to.

Command

```
at+rsi_rssi?
```

Usage

```
at+rsi_rssi?\r\n
```

Parameters

NA

Response

Result Code	**Description**
OK<RSSI>	1 Byte, hexadecimal, return value for absolute value of RSSI. For example, if RSSI is -20dBm, then the return value is 0x14 (20 in decimal)
ERROR<Error code>	Failure

For example, for a RSSI of -20dBm, the return string is

```
OK <RSSI=20> \r\n
………………….
0x4F 0x4B 0x14 0x0D 0x0A
```

3.6 Connecting to a Preconfigured Access Point[1]

The RS9110-N-11-2X module can connect to a pre-configured access point after the module boots up. This feature facilitates fast connection to a known network. The corresponding commands are described below.

[1] The Configuration GUI Application is not compatible with this feature. Hyperterminal should be used to evaluate this feature.

3.6.1 Store Configuration in Flash Memory

Description

This command is used to save the parameters of an access point to internal memory. This command should be given after at+rsi_scan, at+rsi_psk, at+rsi_join and at+rsi_ipconf commands , to save the configuration of the access point to which the module joined. This will enable the module to join immediately to this access point the next time the module powers up (within 200 ms of power up), provided the automatic join feature is enabled by the command *at+rsi_cfgenable*.

Command

at+rsi_cfgsave

Usage

at+rsi_cfgsave\r\n

Parameters

None

Response

Result Code	**Description**
OK	Successful execution
ERROR<Error Code>	Failure

3.6.2 Enable automatic joining to pre-configured Access Point

Description

This command is used to enable or disable the feature of automatic joining to a pre-configured Access Point on power up.

Command

at+rsi_cfgenable

Usage

at+rsi_cfgenable=0\r\n disables the feature

at+rsi_cfgenable=1\r\n enables the feature

Response

Result Code	Description
OK	Successful execution
ERROR<Error code>	Failure

3.6.3 Get Information about Stored Configuration

Description

This command is used to get the configuration values that have been stored in the module's memory. The parameters returned in this command are the parameters used in automatically joining to a pre-configured network when the module boots up in the auto-join mode.

Command

at+rsi_cfgget

Usage

at+rsi_cfgget?

Response

Result code	Description
OK <ValidFlag><ChannelNumber><NetworkType><SecMode><DataRate><PowerLevel><PSK><SSID><Reserved><DHCP><IP address><SubnetMask><Default Gateway><Reserved><Authmode>	Successful execution of the function.
ERROR<Error code>	Failure

Structure of Response

Structure Member Name	Description
ValidFlag	2 bytes, hex. 0x0000 Invalid configuration 0x0001 Valid configuration
ChannelNumber	1byte, hex. It returns the channel number of

Structure Member Name	**Description**
	the stored access point or network. For example, if channel number is 6, then 0x06 is returned. If channel number is 12, then 0x0C is returned.
Network Type	1 byte, hex. It returns the 0x01 : Infrastructure 0x00 : Ad-hoc
Sec Mode	1 byte, hex. Security Mode of the scanned Access Point, returned in hexadecimal, 1 byte. 0x00 – Open (No Security) 0x01 – WPA 0x02 – WPA2 0x03 – WEP
Data Rate	1 byte, hex. Data Rate of the module
Power Level	1 byte, hex. Power level of the module. 0 – Low power (7dBm) 1 – Medium power (10dBm) 2 – High power (16 to 17dBm)
PSK	64 bytes ASCII if the PSK is smaller than 64 bytes then filler bytes (0x00) are appended to the PSK to make it 64 bytes. For example, if the key is ‘12345678’, the return value is 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x00 0x00 …..0x00 (to make 64 bytes)
SSID	32 Bytes, ASCII. SSID of the network to which the module should get connected. If the actual length is not 32 bytes, then filler bytes (0x00) is appended to the SSID to make it 32 bytes. For example, if the SSID is ‘abcd’, the return value is 0x61 0x62 0x63 0x64 0x00 0x00 …..0x00 (to make 32 bytes)
Reserved	3 Bytes
DHCP	1 byte, hex.

Structure Member Name	Description
	0x00 DHCP Disable 0x01 DHCP Enable
IP Address	4 bytes, hex. IP Address of the module
Subnet Mask	4 bytes, hex. Subnet mask
Default Gateway	4 bytes, hex. Default gateway
Feature Select bit map	4 bytes hex, please refer the Feature Select section for more information.
Authmode	1 byte, hex. 0x01- WEP shared authentication 0x00- WEP open authentication Returned only when Bit[7] of "Feature Select" command is set to '1'
WEP index	1 byte,hex. 0x00- WEP key0 0x01- WEP key1 0x02- WEP key2 0x03- WEP key3 Returned only when Bit[7] of "Feature Select" command is set to '1'
WEP keys	96 bytes,ASCII. Three WEP keys each of 32 bytes. . If the actual length is not 32 bytes, then filler bytes (0x00) are appended to the PSK to make it 32 bytes. For example, if the wep key1 is '12345678', the return value is 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x00 0x00 …..0x00 (to make 32 bytes) Similarly for wep key2 and wep key3 Returned only when Bit[7] of "Feature Select" command is set to '1'

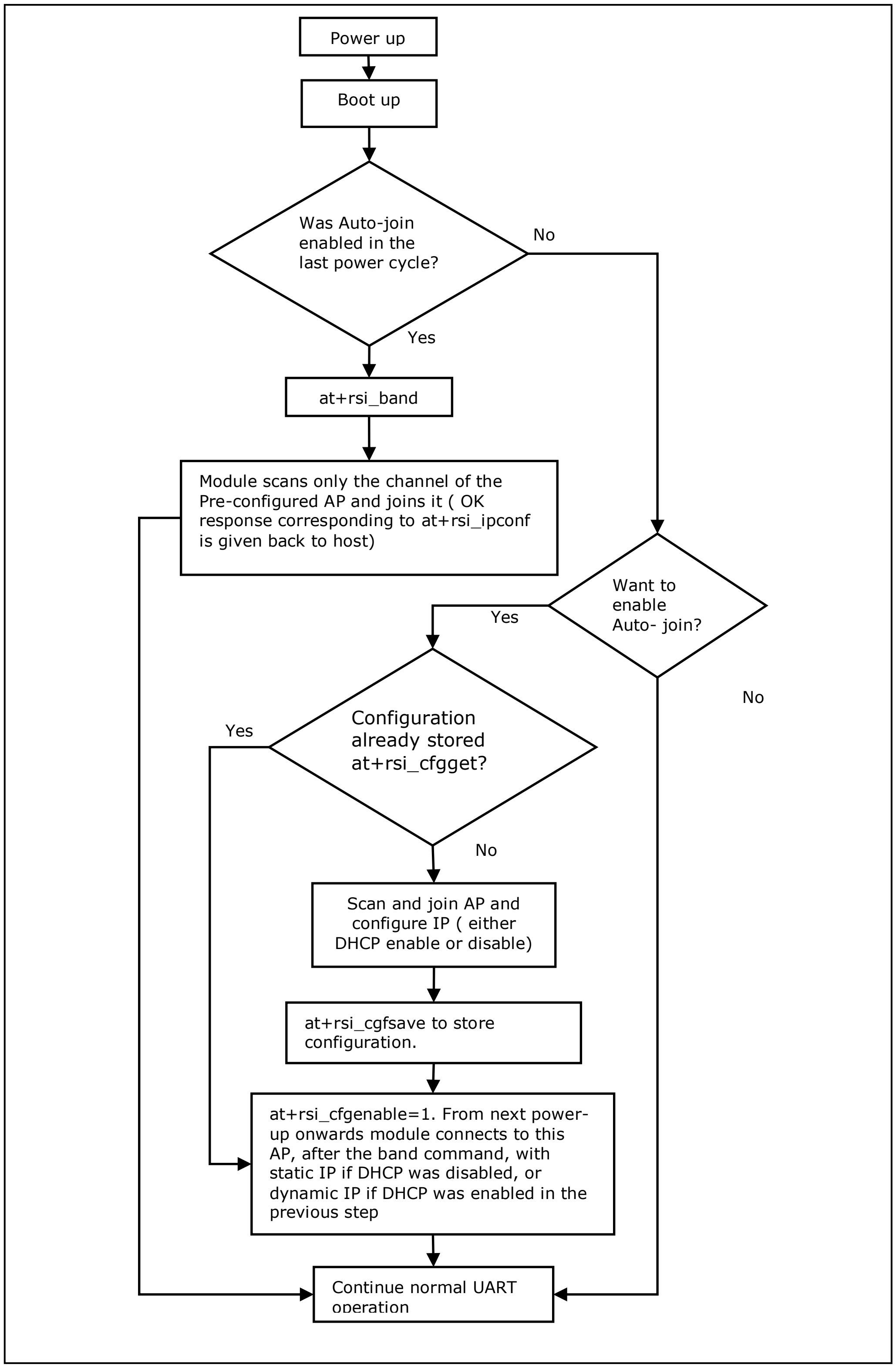

Power up
Boot up
Was Auto-join enabled in the last power cycle?
No
Yes
at+rsi_band
Module scans only the channel of the Pre-configured AP and joins it (OK response corresponding to at+rsi_ipconf is given back to host)
Want to enable Auto- join?
Yes
No
Configuration already stored at+rsi_cfgget?
Yes
No
Scan and join AP and configure IP (either DHCP enable or disable)
at+rsi_cgfsave to store configuration.
at+rsi_cfgenable=1. From next power-up onwards module connects to this AP, after the band command, with static IP if DHCP was disabled, or dynamic IP if DHCP was enabled in the previous step
Continue normal UART operation

Figure 5: Connecting to Pre-configured AP

3.7 Upgrading Firmware

There are two options for firmware upgrade:

- Firmware upgrade without wireless firmware upgrade feature
- Firmware upgrade with wireless firmware upgrade feature.

The detailed procedure for firmware upgrade is mentioned in the below sections.

3.7.1 Firmware Upgrade Using UART Interface without wireless firmware upgrade feature

When the module is powered up, the following sequence is executed

1. Auto Baud Rate Detection

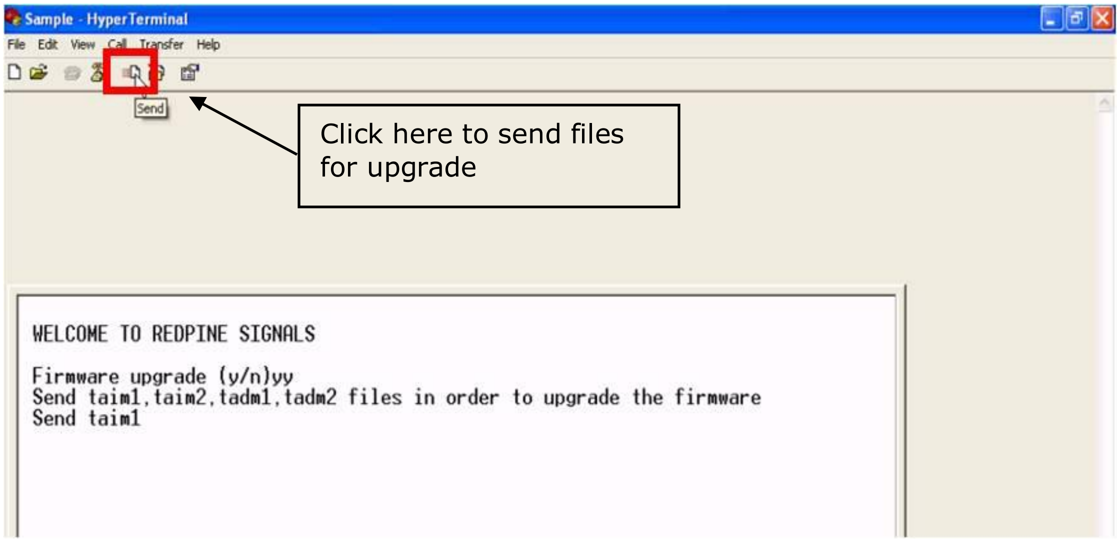
2. The module sends the message:

<space>WELCOME TO REDPINE SIGNALS\r\n<space>\r\n<space>
Byte sequence 0x20 0x57 0x45 0x4C 0x43 0x4F 0x4D 0x45 0x20 0x54 0x4F 0x20 0x52 0x45 0x44 0x50 0x49 0x4E 0x45 0x20 0x53 0x49 0x47 0x4E 0x41 0x4C 0x53 0x0D 0x0A 0x20 0x0D 0x0A 0x20

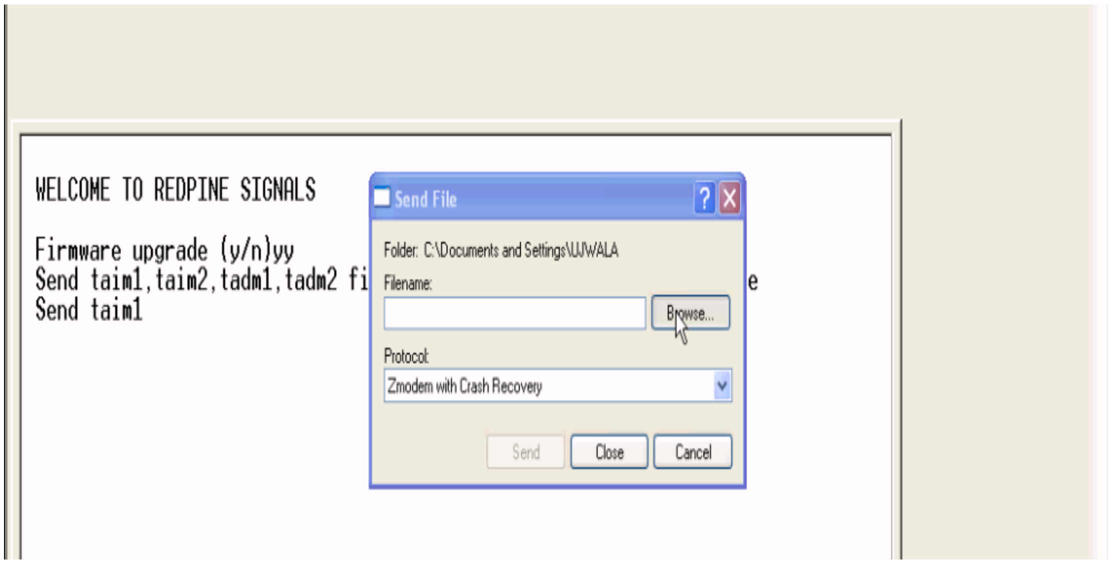
3. The module sends the message:

<space>Firmware upgrade (y/n)
Byte sequence 0x20 0x46 0x69 0x72 0x6D 0x77 0x61 0x72 0x65 0x20 0x75 0x70 0x67 0x72 0x61 0x64 0x65 0x20 0x28 0x79 0x2F 0x6E 0x29

4. If the user wishes to upgrade firmware, 'y' (0x79) is sent to the module.


Sample - HyperTerminal
File Edit View Call Transfer Help
Send
Click here to send files for upgrade

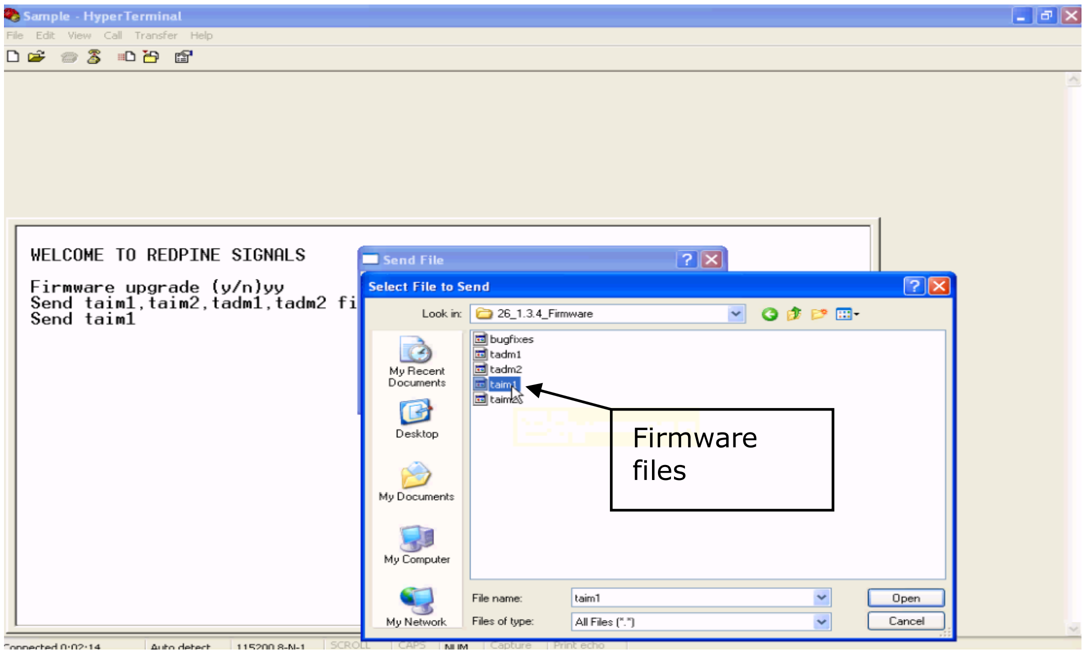
```
WELCOME TO REDPINE SIGNALS

Firmware upgrade (y/n)yy
Send taim1,taim2,tadm1,tadm2 files in order to upgrade the firmware
Send taim1
```



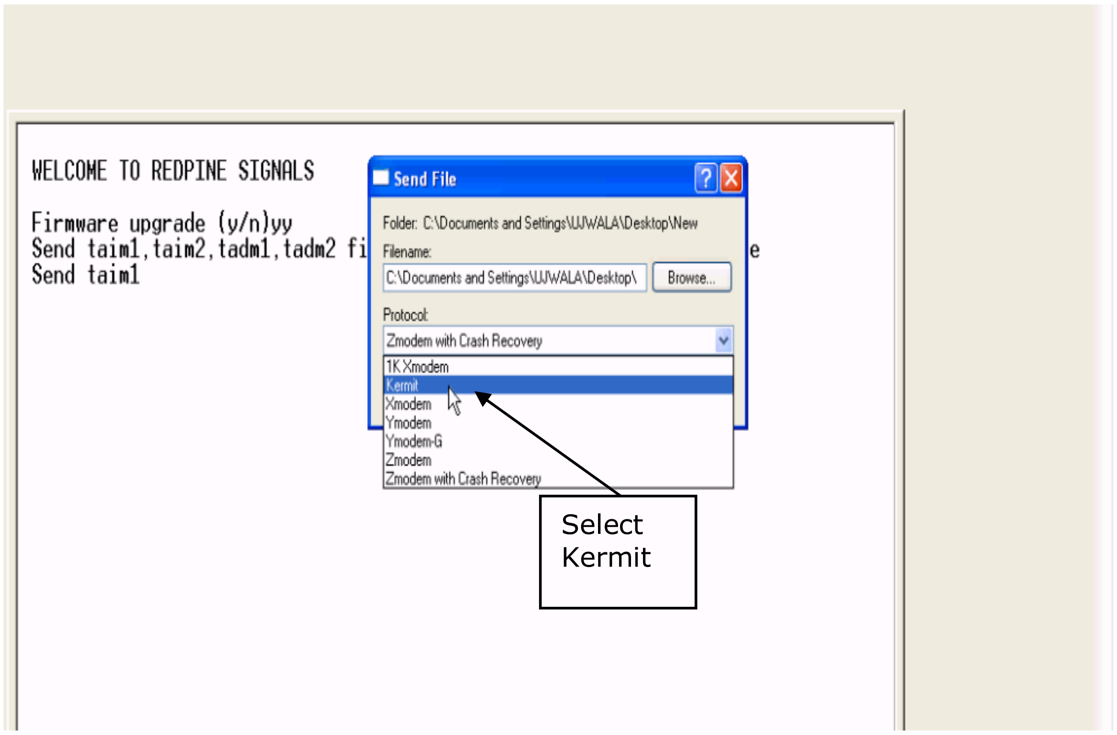
```
WELCOME TO REDPINE SIGNALS

Firmware upgrade (y/n)yy
Send taim1,taim2,tadm1,tadm2 fi
Send taim1
```

Send File
Folder: C:\Documents and Settings\UJWALA
Filename:
Browse...
Protocol:
Zmodem with Crash Recovery
Send
Close
Cancel

Select the firmware files from RS.CN.xx.GENR.UT.x.x.x\Software\Firmware\

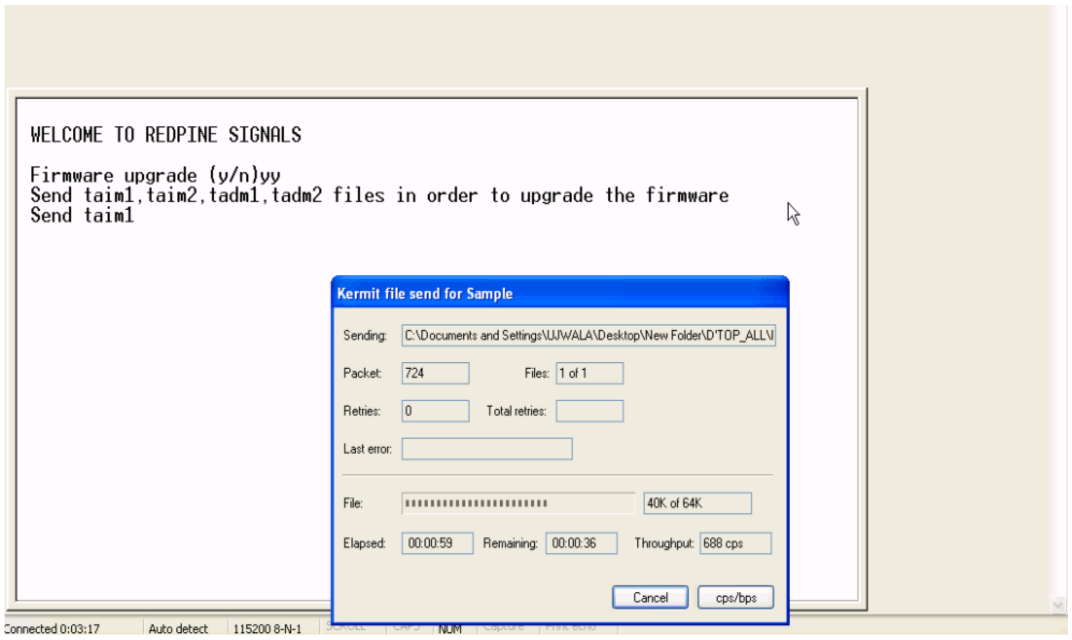


```
WELCOME TO REDPINE SIGNALS

Firmware upgrade (y/n)yy
Send taim1,taim2,tadm1,tadm2 fi
Send taim1
```



```
WELCOME TO REDPINE SIGNALS

Firmware upgrade (y/n)yy
Send taim1,taim2,tadm1,tadm2 fi
Send taim1
```

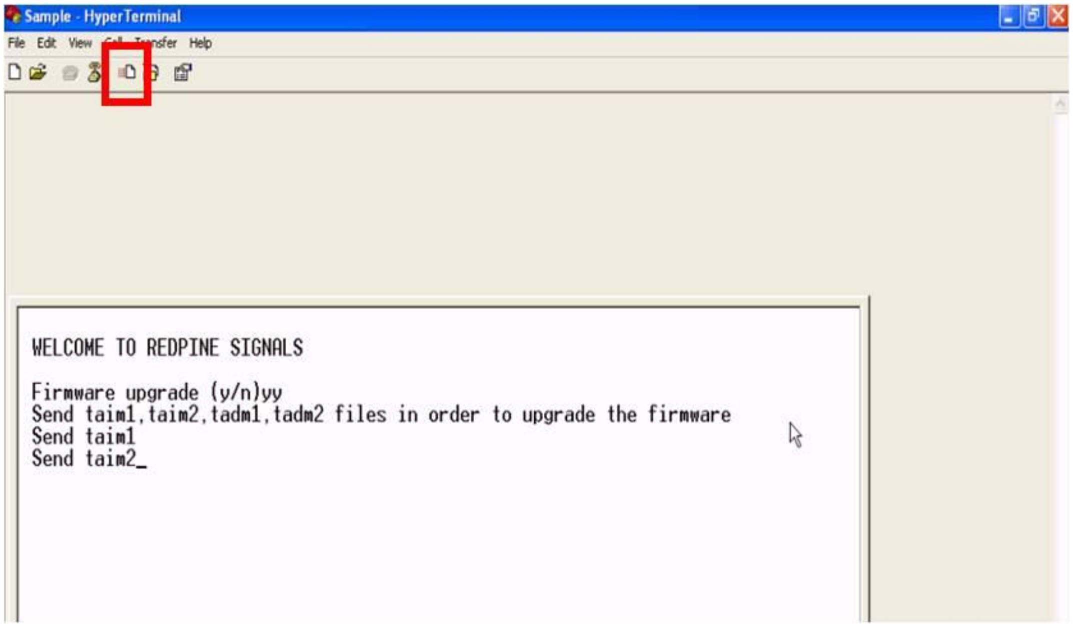
Send File
Folder: C:\Documents and Settings\UJWALA\Desktop\New
Filename: C:\Documents and Settings\UJWALA\Desktop\
Protocol: Zmodem with Crash Recovery



```
WELCOME TO REDPINE SIGNALS

Firmware upgrade (y/n)yy
Send taim1,taim2,tadm1,tadm2 files in order to upgrade the firmware
Send taim1
```

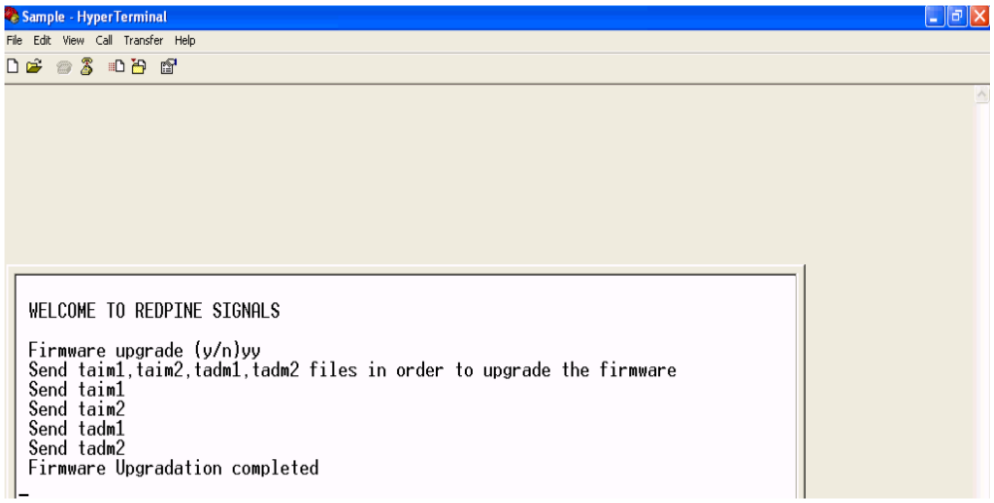
Kermit file send for Sample
Sending: C:\Documents and Settings\UJWALA\Desktop\New Folder\D'TOP_ALL\I
Packet: 724 Files: 1 of 1
Retries: 0 Total retries:
Last error:
File: 40K of 64K
Elapsed: 00:00:59 Remaining: 00:00:36 Throughput: 688 cps

Connected 0:03:17 Auto detect 115200 8-N-1 NUM



```
WELCOME TO REDPINE SIGNALS

Firmware upgrade (y/n)yy
Send taim1,taim2,tadm1,tadm2 files in order to upgrade the firmware
Send taim1
Send taim2_
```

Follow the same sequence to load all the files from RS.CN.xx.GENR.UT.4.7.x\Software\Firmware\ta*.



```
WELCOME TO REDPINE SIGNALS

Firmware upgrade (y/n)yy
Send taim1,taim2,tadm1,tadm2 files in order to upgrade the firmware
Send taim1
Send taim2
Send tadm1
Send tadm2
Firmware Upgradation completed
_
```

After all the files are loaded, the message “Firmware Upgradation completed” is sent by the module.

<space>Firmware Upgradation completed\r\n
Byte sequence 0x20 0x46 0x69 0x72 0x6D 0x77 0x61 0x72 0x65 0x20 0x55 0x70 0x67 0x72 0x61 0x64 0x61 0x74 0x69 0x6F 0x6E 0x20 0x63 0x6F 0x6D 0x70 0x6C 0x65 0x74 0x65 0x64 0x0D 0x0A

NOTE: Firmware upgrade need not be done every time the module is powered up. It is only done when a new firmware is released for the product.

5. The user can now power cycle or hard reset the module and not opt for firmware upgrade when "Firmware upgrade (y/n)" comes in the screen, by sending 'n\n' (0x6E 0x0A) to the module immediately after the message comes. In such a case, the following messages are sent from the module

   <space>Loading…<space>\r\n
   Byte sequence 0x20 0x4C 0x6F 0x61 0x64 0x69 0x6E 0x67 0x2E 0x2E 0x2E 0x20 0x0D 0x0A

   <space>Loading Done\r\n
   Byte sequence 0x20 0x4C 0x6F 0x61 0x64 0x69 0x6E 0x67 0x20 0x44 0x6F 0x6E 0x65 0x0D 0x0A

6. After this message, AT commands can be given as described

3.7.2 Firmware Upgrade Using UART Interface with wireless firmware upgrade feature[1]

1. Open Hyperterminal and power up the module. The module sends the message – "Firmware upgrade (y/n)". Send 'y' to the module.
2. Send the files taim1, taim2, tadm1 and tadm2 from the location RS.CN.xx.GENR.UT.4.7.x\Software\Wireless_Upgrade\Firstlevel_bootloader\ using the Kermit protocol. Follow the same process as described in section Firmware Upgrade from a Lower Version to 4.7.x Using UART Interface without wireless firmware upgrade feature to load the file from the location *RS.CN.xx.GENR.UT.4.7.x\Software\Wireless_Upgrade\Firstlevel_bootloader\*
3. Power cycle or hard reset the module. The module sends the message – "Firmware upgrade (y/n)". Send 'n\n' to the module immediately. The module displays the messages "Loading…" and then

   "1st level boot-loader upgradation completed"

   Hex stream:

   20 31 73 74 20 4C 65 76 65 6C 20 42 6F 6F 74 4C 6F 61 64 65 72 20 55 70 67 72 61 64 61 74 69 6F 6E 20 43 6F 6D 70 6C 65 74 65 64 0D 0A

[1] Wireless Firmware Upgrade Feature is available from 4.7.X onwards only.

4. Power cycle or hard reset the module. The module sends the message – "Firmware upgrade (y/n)". Send 'y' to the module.
5. On being prompted by the module, send taim1, taim2, tadm1 and tadm2 from RS.CN.xx.GENR.UT.4.7.x\Software\Firmware\ta** to the module. The module sends the message "Firmware Upgradation completed"
6. Again power cycle or hard reset the module. The module sends the message – "Firmware upgrade (y/n)". Send 'u' to the module[1]. The module prompts to send files sbim1, sbim2 and sbdm. Send these files from RS.CN.xx.GENR.UT.4.7.x\Software\Wireless_Upgrade\Secondlevel_bootloader\ in the same order, using the Kermit protocol. For each file being transferred, wait for a display on the HyperTerminal, which says "Send <nextfile>". Note that after a file is transmitted, there is some delay in writing the data to the non-volatile memory. This might take a few seconds and the user has to wait for the prompt on the HyperTerminal to send the next file.
7. After all the files are transferred, the message is sent by the module:

   "2nd Level Bootloader Upgradation completed"

   Hex stream:

   20 32 6E 64 20 4C 65 76 65 6C 20 42 6F 6F 74 4C 6F 61 64 65 72 20 55 70 67 72 61 64 61 74 69 6F 6E 20 63 6F 6D 70 6C 65 74 65 64 20 0D 0A
8. Power cycle or hard reset the module. The module sends the message – "Firmware upgrade (y/n)". Send 's' to the module[2].
9. The module sends the message :

   "Select one of the following option :

   1.Configuration firmware upgrade:

   2.Load configuration firmware :"
10. Enter '1' immediately. Send files cfim1, cfim2 and cfdm1 from RS.CN.xx.GENR.UT.x.x.x\Software\Wireless_Upgrade\Config_firmware in the same order.
11. After the process is over, the module sends the message "Configuration firmware upgrade completed".

    Hex stream:

    20 63 6F 6E 66 69 67 75 72 61 74 69 6F 6E 20 66 69 72 6D 77 61 72 65 20 75 70 67 72 61 64 65 20 63 6F 6D 70 6C 65 74 65 64 20 0D 0A

    This completes the upgrade process.

[1] Note that in the message "Firmware upgrade…", 'u' is not displayed as an option.
[2] Note that in the message "Firmware upgrade..", 's' is not displayed as an option

3.7.3 Firmware Upgrade from 4.7.x to a higher version wirelessly

The modules RS9110-N-11-22(-xx) and RS9110-N-11-28(-xx) can be upgraded wirelessly. Also, they can be configured wirelessly[1] to join specific access points. The features of wireless configuration and wireless firmware upgrade are available from firmware version 4.7.1 onwards.

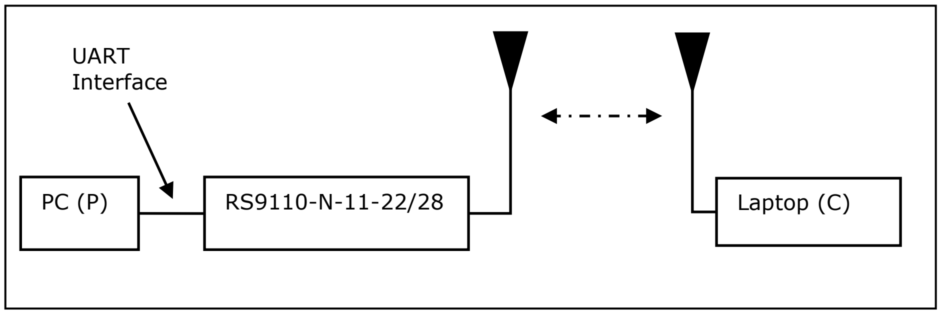


Figure 6: Wireless Firmware Upgrade Set-up

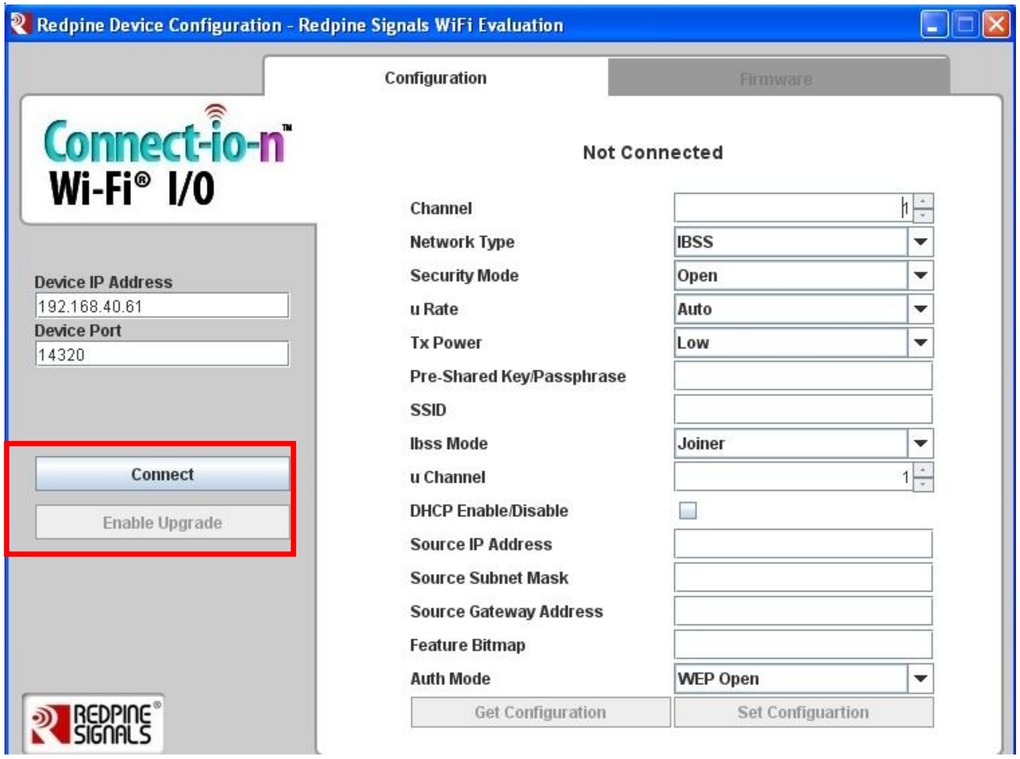
1. Power up the module. The module sends the message – "Firmware upgrade (y/n)". Send 's' to the module[2]. The module sends two options

   1.Configuration Firmware upgrade

   2.Load Configuration Firmware

   Press 2. The module now becomes an Access Point with open mode and SSID as REDPINE_<last 3 bytes of MAC ID of the module>
2. Connect Laptop (C) to the Access Point created by the module. The Laptop will acquire an IP address from the AP.
3. Open the application RS.CN.xx.GENR.UT.x.x.x\Software\Wireless_Upgrade\DeviceConfigGUI.jar in Laptop (C).

[1] Wireless firmware upgrade and wireless configuration are not supported in RS9110-N-11-24
[2] Note that in the message "Firmware upgrade..", 's' is not displayed as an option



Not Connected

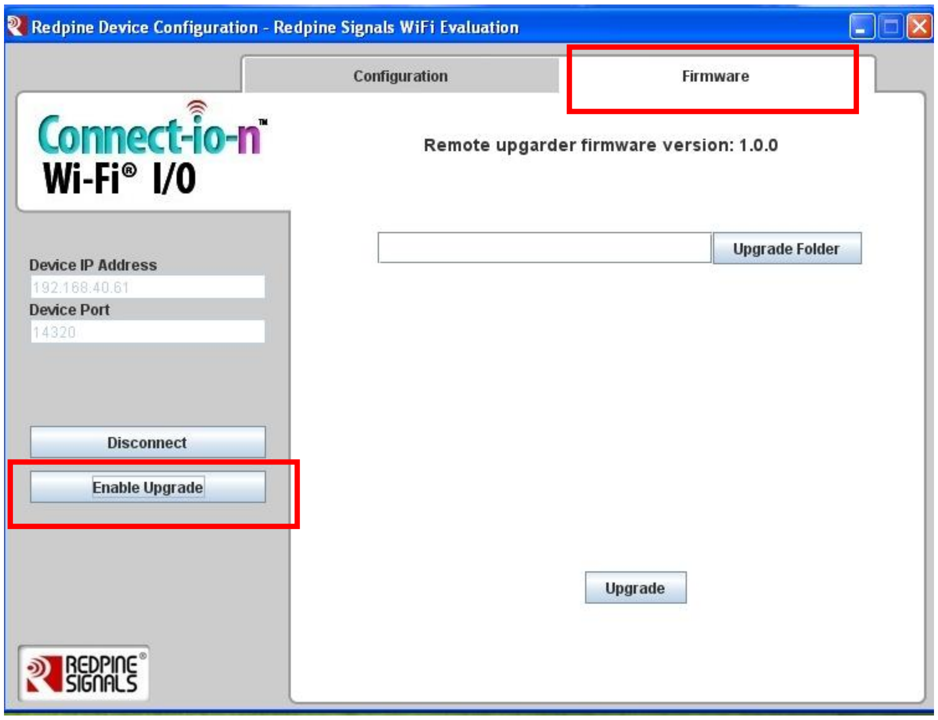
Device IP Address: 192.168.40.61
Device Port: 14320

Channel	1
Network Type	IBSS
Security Mode	Open
u Rate	Auto
Tx Power	Low
Pre-Shared Key/Passphrase	
SSID	
Ibss Mode	Joiner
u Channel	1
DHCP Enable/Disable	☐
Source IP Address	
Source Subnet Mask	
Source Gateway Address	
Feature Bitmap	
Auth Mode	WEP Open

Figure 7: Configuration GUI

4. By default, the "Device IP Address" is 192.168.40.61 and "Device Port" is 14320. Keep them unchanged and click "Connect". This makes the laptop establish a TCP socket connection with the AP. The AP has an IP hardcoded into itself – 192.168.40.61. Ignore the values on the right side of the GUI.

5. Click "Enable Upgrade" button and then click on the "Firmware" tab.



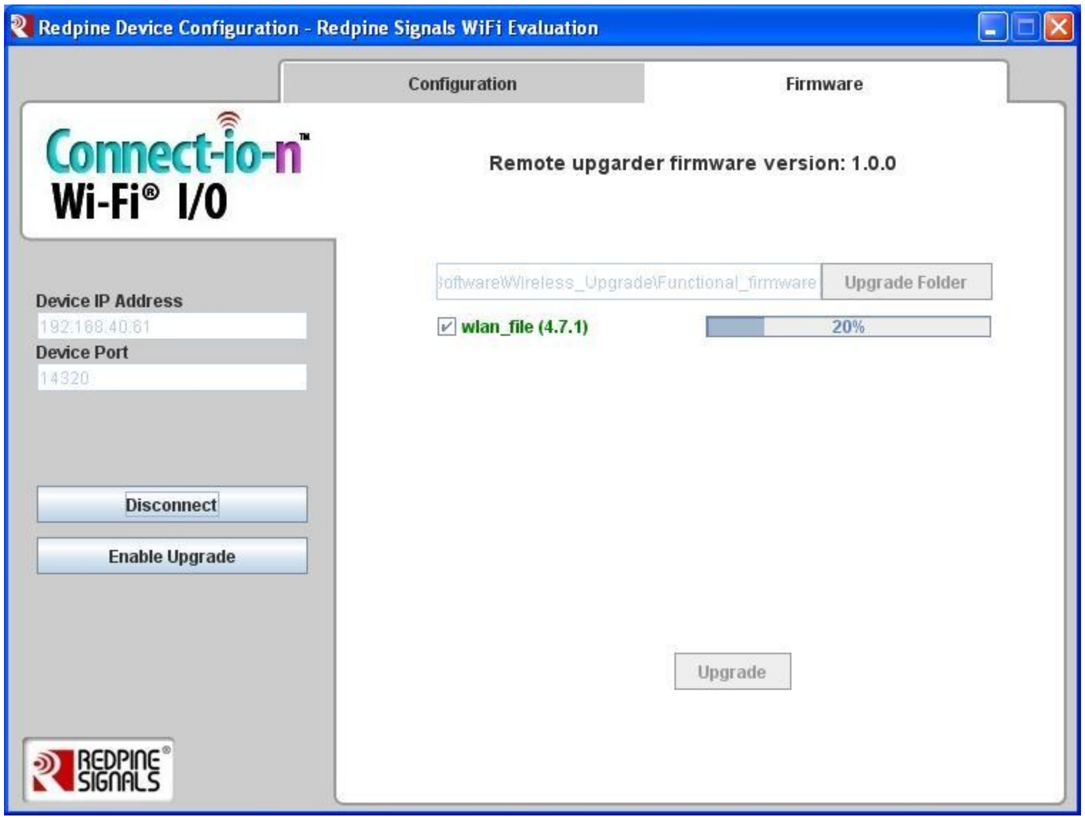
Remote upgarder firmware version: 1.0.0

Device IP Address: 192.168.40.61

Device Port: 14320

6. Click on "Upgrade Folder" and select the FOLDER RS.CN.xx.GENR.UT.x.x.x\Software\Wireless_Upgrade\Functional_firmware\. Then click on the check box for the file wlan_file.rps that appears on the GUI. Click on "Upgrade" button.

NOTE: The file RS.CN.xx.GENR.UT.x.x.x\Software\Wireless_Upgrade\Functional_firmware\wlan_file.rps has the same contents as the files RS.CN.xx.GENR.UT.x.x.x\Software\Firmware\ta**, but is formatted appropriately for wireless upgrade



7. The upgrade process starts. After successful upgrade , a message "Upgrade Complete" is shown in the GUI. At the same time, the message "AT+FWUPGRADE_DONE" is sent to the Host PC (P) from the module through the UART interface. It can be seen on hyperterminal.
8. The module is now upgraded with the latest firmware. It can be now power cycled or hard reset and used in functional mode by providing "n\n" as an answer to the module's message "Firmware Upgrade (y/n)".

3.8 Using the Wireless Configuration Feature

Using this feature, the module can be configured to

1. Join a specific Access Point
2. Join a specific ad-hoc network

This will be referred to as "Auto-connect".

3.8.1 Wireless Configuration in AP mode

This section describes wireless configuration by putting the module initially into Access Point mode. The next section describes wireless configuration when the module is already connected to an AP.

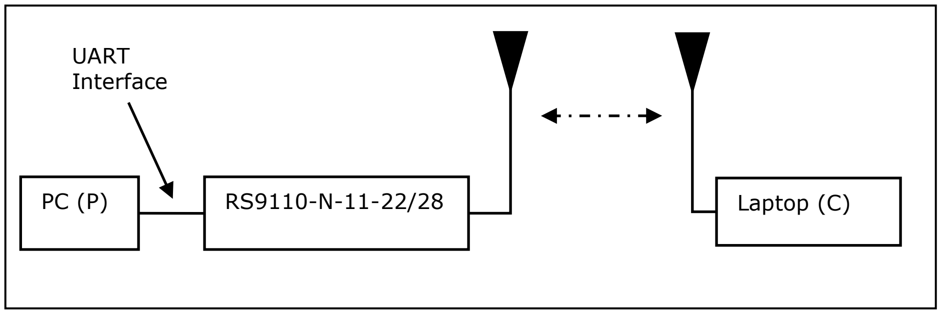
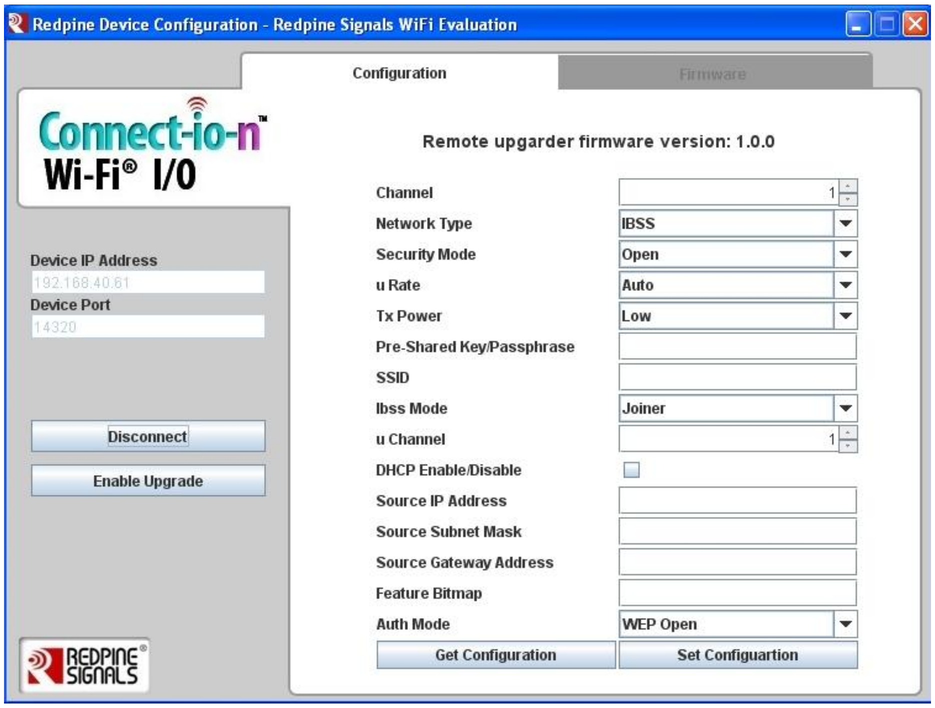


Figure 8: Wireless Configuration Set-up

1. Power up the module. The module sends the message – "Firmware upgrade (y/n)". Send 's' to the module. The module sends two options: 1.Configuration Firmware upgrade

   2.Load Configuration Firmware

2. Press '2'. The module now becomes an Access Point with open mode and SSID REDPINE_<last 3 bytes of MAC ID of the module>
3. Connect a Laptop (C) to the AP created by the module. The Laptop should be configured as a DHCP client. The Laptop will acquire the IP address from the AP.
4. Open the application RS.CN.xx.GENR.UT.x.x.x\Software\Wireless_Upgrade\DeviceConfigGUI.jar in Laptop (C) and click on the connect button. Leave the "Device IP Address" and "Device Port" unchanged to 192.168.40.61 and 14320 respectively. After clicking on "Connect" Button, click on the "Configuration" tab on top.


Redpine Device Configuration - Redpine Signals WiFi Evaluation

Configuration | Firmware

Connect-io-n™ Wi-Fi® I/O

Device IP Address: 192.168.40.61
Device Port: 14320

Disconnect
Enable Upgrade

Remote upgarder firmware version: 1.0.0

Channel	1
Network Type	IBSS
Security Mode	Open
u Rate	Auto
Tx Power	Low
Pre-Shared Key/Passphrase	
SSID	
Ibss Mode	Joiner
u Channel	1
DHCP Enable/Disable	☐
Source IP Address	
Source Subnet Mask	
Source Gateway Address	
Feature Bitmap	
Auth Mode	WEP Open

Get Configuration | Set Configuartion

Figure 9: Configuration GUI

5. Click on the "Get Configuration" button. It displays the currently stored configuration parameters. These parameters can be ignored for now.

6. Fill the values of the parameters

    a. Channel: Channel number of the AP or Adhoc network that the module should connect to. Refer tables Channel Number Parameter (2.4 Ghz) and Channel Number Parameter (5 Ghz).

    b. Network Type

    c. Security Mode- Security mode of the Access Point or Adhoc network

    d. uRate (refer table Data Rate Parameter)

    e. Tx Power

    f. Pre shared key- This can be left unfilled if the security mode is "Open".

    g. SSID- This is the SSID of the Access Point or Adhoc network to which the module should connect

    h. IBSS mode (Adhoc)- Set it to "Joiner" is NetworkType is IBSS

    i. uChannel- This field can be ignored

    j. DHCP Enable/Disable- Used in infrastructure mode

    k. Source IP Address, Source Subnet Mask, Source Gateway Address: If DHCP is disabled, an IP address, subnet mask and gateway address should be entered for "Source IP Address", "Source Subnet Mask" and "Source Gateway" respectively. The module will get configured with these parameters.

l. Feature Bitmap- The decimal value of the parameter "*bitval*" of the command Feature Select can be entered here. It can be set to '0' for most use cases.

m. Auth mode- If "WEP" is selected in "Security Mode", then the options of "WEP Open" or "WEP Shared" should be used here.

7. Click on the button "Set Configuration". The configuration information is stored in non-volatile memory of the module and can be accessed by the Host at any time using the *at+rsi_cfgget* command.
8. The GUI will send the message "Configure Success". The module sends the message "AT+REMOTE_CFG" to the Host[1] PC (P). Once the message is received, the module can be power cycled or hard reset. The module sends the message – "Firmware upgrade (y/n)". Send "n/n" to the module. Now issue the command *at+rsi_band* to configure the correct band of operation. After this command is issed, the module will automatically join to the configured AP or Adhoc network depending on the parameters entered in the above steps.

[1] The message is sent in uppercase alphabets

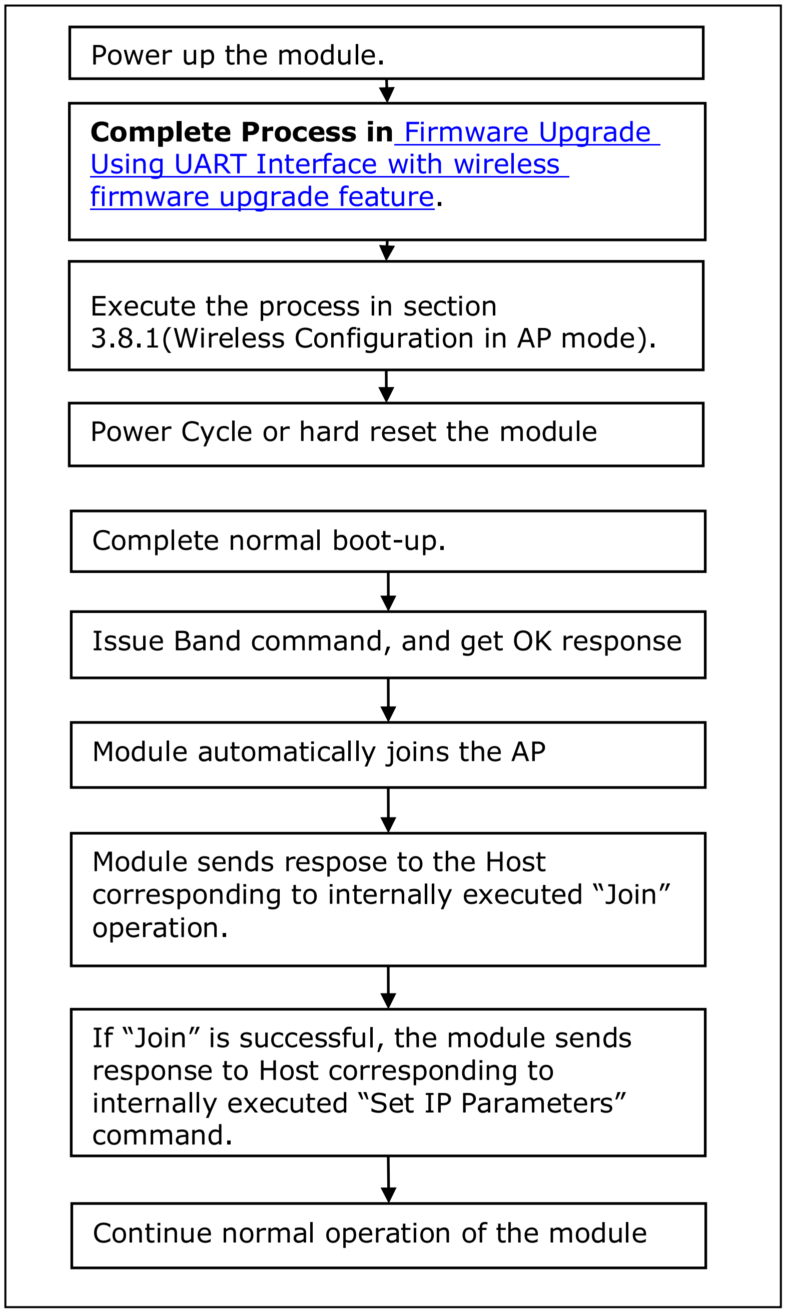

Power up the module.
Complete Process in Firmware Upgrade Using UART Interface with wireless firmware upgrade feature.
Execute the process in section 3.8.1(Wireless Configuration in AP mode).
Power Cycle or hard reset the module
Complete normal boot-up.
Issue Band command, and get OK response
Module automatically joins the AP
Module sends respose to the Host corresponding to internally executed "Join" operation.
If "Join" is successful, the module sends response to Host corresponding to internally executed "Set IP Parameters" command.
Continue normal operation of the module

Figure 10: Wireless Configuration Flow-UART

3.8.2 Wireless Configuration when the module is connected to an AP

This section describes wireless configuration when the module is already connected to an AP.

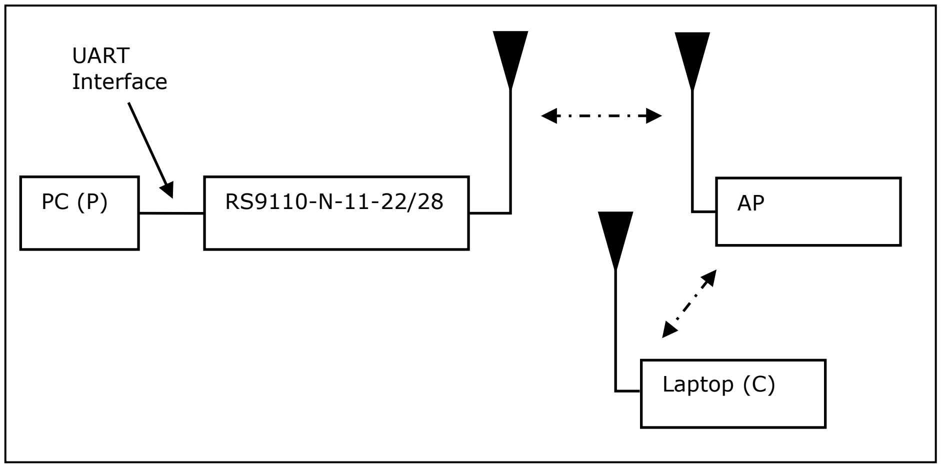
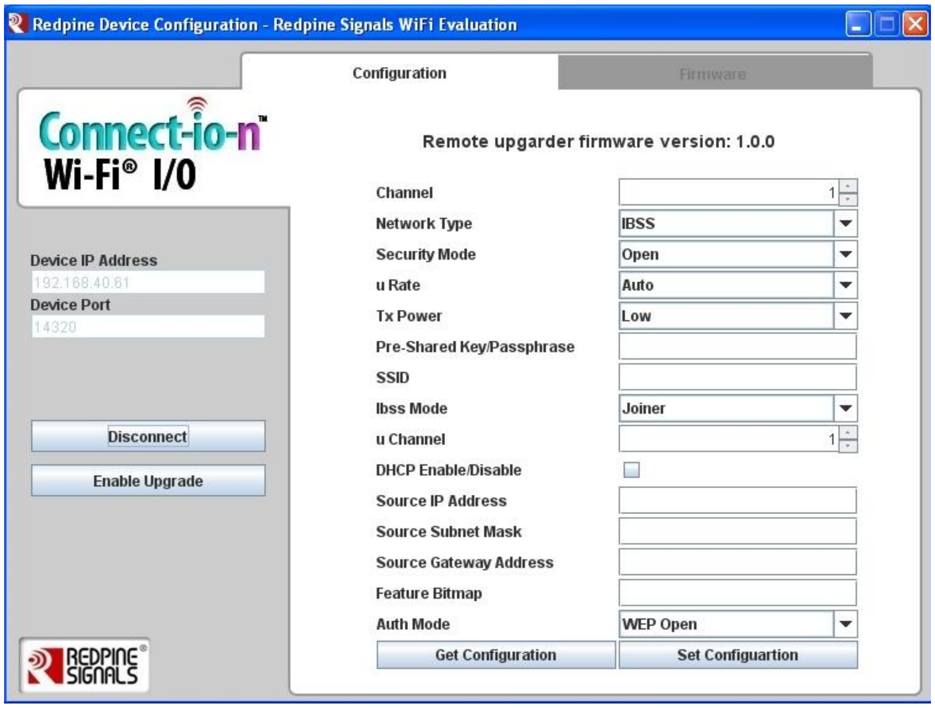


Figure 11: Wireless Configuration Set-up

1. Assuming the module is already connected to an AP, connect a Laptop to the same AP.
2. Open the application RS.CN.xx.GENR.UT.x.x.x\Software\Wireless_Upgrade\DeviceConfigGUI.jar in Laptop (C) and click on the connect button. Enter the IP address of the module in the field "Device IP Address". Keep the "Device Port" unchanged to 14320. After clicking on "Connect" Button, click on the "Configuration" tab on top.



Remote upgarder firmware version: 1.0.0

Device IP Address: 192.168.40.61
Device Port: 14320

Channel	1
Network Type	IBSS
Security Mode	Open
u Rate	Auto
Tx Power	Low
Pre-Shared Key/Passphrase	
SSID	
Ibss Mode	Joiner
u Channel	1
DHCP Enable/Disable	☐
Source IP Address	
Source Subnet Mask	
Source Gateway Address	
Feature Bitmap	
Auth Mode	WEP Open

Figure 12: Configuration GUI

3. Click on the "Get Configuration" button. It displays the currently stored configuration parameters. These parameters can be ignored for now.
4. Fill the values of the parameters
    a. Channel: Channel number of the AP or Adhoc network that the module should connect to. Refer tables Channel Number Parameter (2.4 Ghz) and Channel Number Parameter (5 Ghz).
    b. Network Type
    c. Security Mode- Security mode of the Access Point or Adhoc network
    d. uRate (refer table Data Rate Parameter)
    e. Tx Power
    f. Pre shared key- This can be left unfilled if the security mode is "Open".
    g. SSID- This is the SSID of the Access Point or Adhoc network to which the module should join
    h. IBSS mode (Adhoc)- Set it to "Joiner" is NetworkType is IBSS
    i. uChannel- this field can be ignored
    j. DHCP Enable/Disable- Used in infrastructure mode
    k. Source IP Address, Source Subnet Mask, Source Gateway Address: If DHCP is disabled, an IP address, subnet mask and gateway address should be entered for "Source IP Address", "Source Subnet Mask" and "Source Gateway" respectively. The module will get configured with these parameters.
    l. Feature Bitmap- The decimal value of the parameter *bitval* of the command Feature Select can be entered here. It can be set to '0' for most use cases.
    m. Auth mode- If "WEP" is selected in "Security Mode", then the options of "WEP Open" or "WEP Shared" should be used here.
5. Click on the button "Set Configuration". The configuration information is stored in non-volatile memory of the module and can be accessed by the Host at any time using the *at+rsi_cfgget* command.
6. The GUI will send the message "Configure Success". The module sends the message "AT+REMOTE_CFG" to the Host[1] PC (P). Once the message is received, the module can be power cycled or hard reset. The module sends the message – "Firmware upgrade

[1] The message is sent in uppercase alphabets

(y/n)". Send "n/n" to the module. Now issue the command *at+rsi_band* to configure the correct band of operation. After this command is issed, the module will automatically join to the configured AP or Adhoc network depending on the parameters entered in the above steps.

NOTE: When the module is configured using wireless interface, the "Auto-join" feature is enabled. This makes the module join to the configured AP on the next power cycle or hard reset. If at any point of time, the user wants to disable the "Auto-join" feature, the command should be used.

3.9 General Operation of the Modules

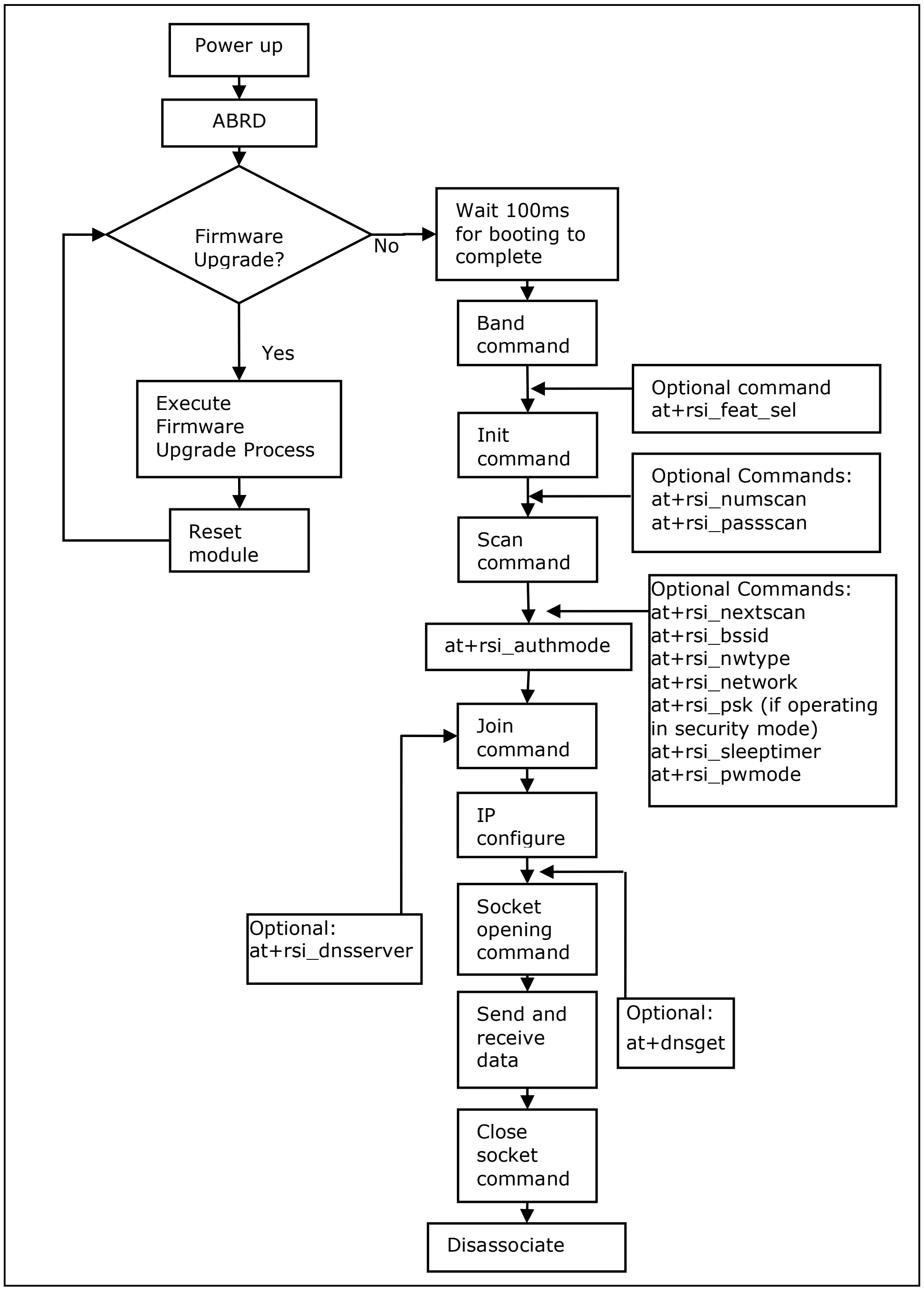

Power up
ABRD
Firmware Upgrade?
No
Yes
Wait 100ms for booting to complete
Band command
Optional command at+rsi_feat_sel
Init command
Optional Commands: at+rsi_numscan at+rsi_passscan
Scan command
Optional Commands: at+rsi_nextscan at+rsi_bssid at+rsi_nwtype at+rsi_network at+rsi_psk (if operating in security mode) at+rsi_sleeptimer at+rsi_pwmode
at+rsi_authmode
Execute Firmware Upgrade Process
Reset module
Join command
IP configure
Optional: at+rsi_dnsserver
Socket opening command
Send and receive data
Optional: at+dnsget
Close socket command
Disassociate

Figure 13: Firmware Upgrade and General Operation in UART modules[1]

[1] An AT command should be given only after receiving a response from the Host to the preceding command.

3.10 Error Codes

The following are the valid error codes, along with their two's complement values in hexadecimal, for the commands.

Error Code	Description
-1 (0xFF)	Waiting for the connection from peer.
-2 (0xFE)	Socket not available.
-3 (0xFD)	De-authentication from the Access Point.
-4 (0xFC)	Illegal IP/Port parameters.
-5 (0xFB)	TCP/IP configuration failure.
-6 (0xFA)	Invalid socket.
-7 (0xF9)	Association not done.
-8 (0xF8)	Error in command.
-9 (0xF7)	Error with byte stuffing for escape characters.
-10 (0xF6)	IP Lease expired
-11 (0xF5)	TCP Connection Closed
-12 (0xF4)	Pre-Shared Key contains invalid characters (applicable in case of WEP)
-13 (0xF3)	No Access Points present. This error is also issued if the security mode of the AP is different from the one supplied to the module in the "Authentication mode" command.
-14 (0xF2)	1.This error is sent by the module when the INIT command is sent more than once. 2. Flash access error for at_rsi_cfgenable command 3. Returned for at+rsi_cfgget command when the config save feature is not enabled (at+rsi_cfgenable=1 not issued)
-15 (0xF1)	This error is sent by the module when 1. JOIN command is sent when module is already associated with an existing network. 2. Scan command is issued after the module has already associated with a network. 3. When there is a flash write failure for the at+rsi_cfgsave command 4. Module not in connected state (for at+rsi_cfgsave command)

Error Code	Description
-16 (0xF0)	DHCP Failure, DHCP Renewal not done by AP
-17 (0xEF)	Baud Rate Not Supported
-18 (0xEE)	Encryption mode not supported
-19 (0xED)	Invalid channel
-20 (0xEC)	at+rsi_snd command is issued but there is no network in the remote terminal
-21 (0xEB)	Authentication failure
-22 (0xEA)	Re-join failure
-23 (0xE9)	Country information not found in probe response from AP during scan.
-24 (0xE8)	Wrong PSK was supplied to the module
-25 (0xE7)	Invalid network type
-26 (0xE6)	Back ground scan cancelled
-28 (0xE4)	Unsupported bits are set in Feature Select bitmap
-29 (0xE3)	“Scan” command is issued before “Band” and “Init” commands
-30 (0xE2)	Tx data rate configured in module does not match the rates supported by the target AP
-56 (0xC8)	“Query RSSI” command issued in IBSS mode
-55(0xC9)	For RSSI query in unassociated state
-59 (0xC5)	Unable to process command. The Host should stop sending commands to the module for some tens of millisecs
-69 (0xBB)	The reply did not contain an answer with an IP address
-70 (0xBA)	DNS class error
-72 (0xB8)	The number of queries is more than 1 or the number of replies is zero
-73 (0xB7)	The reply contained an error code
-74 (0xB6)	The reply was truncated or not a response to a standard query
-75 (0xB5)	The ID number of the reply did not match the ID number sent
-85 (0xAB)	No buffer was passed or the reply was too short
-91 (0xA5)	Invalid IP address passed or unable to send IGMP report

Error Code	Description
-92 (0xA4)	DNS response timed out
-95 (0xA1)	Unable to send data as ARP is not resolved
-100 (0x9C)	DHCP handshake failure
-113 (0x8F)	Unable to send data because of connectivity disruption (such as AP switched off abruptly)
-121 (0x87)	Error issued when trying to connect to a non-existent TCP server socket
-124 (0x84)	Connection establishment failure.
-127 (0x81)	Socket already exists.
-128 (0x80)	Attempt to open more than 8 sockets
-131 (0x7D)	DHCP_MODE parameter in "Set IP Parameters" command is set to a value other than 0 or 1.
-211(0x2D)	This code is returned in the following scenario: An LTCP socket is opened in the module. The remote peer tries to connect to the socket using a client socket, by sending a SYN packet (according to TCP protocol). The module in turn responds with a SYN+ACK packet. The remote peer is now supposed to send an ACK packet. If this packet is missed or does not reach the module, the module retries sending the SYN+ACK packet 4 times, and if the maximum number of retries does not lead to reception of an ACK from the remote terminal, the module returns this error to the Host and closes the LTCP socket.
1 (0x01)	Wireless configuration failure
25(0x19)	Invalid value supplied as input in *at+rsi_authmode* command
51 (0x33)	Keep alive timeout. On lack of activity over an existing TCP socket connection, the module sends out a Keep alive frame to the remote terminal. If there is no response, the module closes its TCP socket and sends out an asynchronous message ERROR<Error_code=0x33>
64 (0x40)	Tx buffers are not available. The Host should stop sending data to the module for some tens of millisecs to relax the buffers
65 (0x41)	A packet of size more than 1460 bytes (TCP) or more than 1472 bytes (UDP) is sent in *at+rsi_snd* command.
66(0x42)	Zero TCP/UDP payload is sent from the module.

Table 4: Error Codes

NOTE: When the error code received is -10 (IP lease expired), the Host has to issue the TCP/IP configure command to reconfigure the IP address. The Host has to ensure that the Wi-Fi connection is established.

4 RS9110-N-11-2X in SPI Mode

The following figure illustrates the software components of the Host when the RS9110-N-11-2X module is configured in SPI mode.

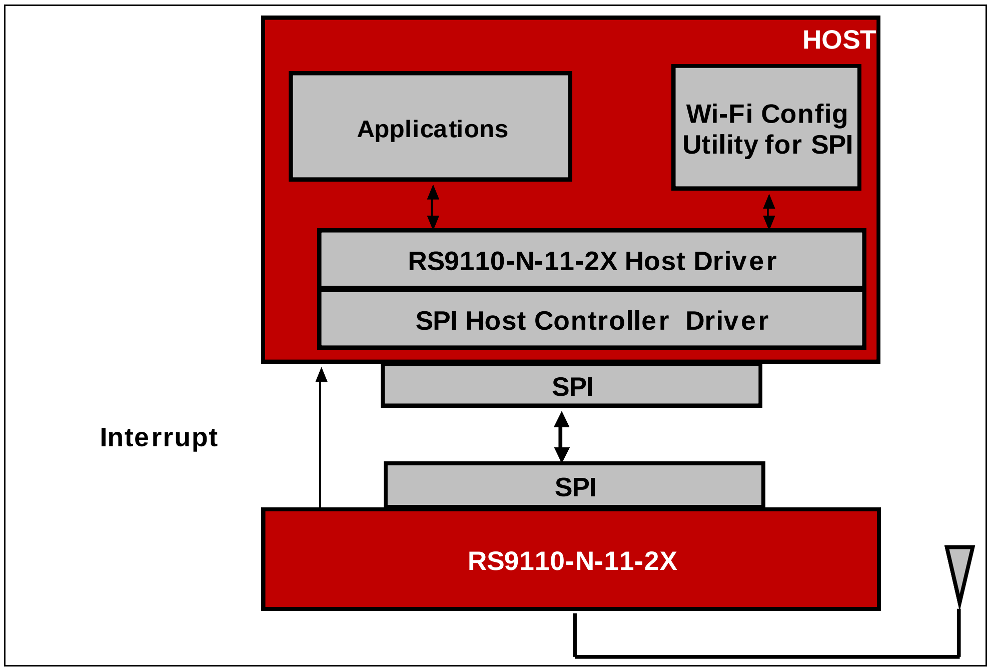

HOST
Applications
Wi-Fi Config Utility for SPI
RS9110-N-11-2X Host Driver
SPI Host Controller Driver
SPI
Interrupt
SPI
RS9110-N-11-2X

Figure 14:RS9110-N-11-2X in SPI Mode

As shown in the figure above, the "Wi-Fi Config Utility for SPI" configures the RS9110-N-11-2X module and the "Applications" use the TCP/IP stack in the RS9110-N-11-2X module to access the network. SPI driver on the Host resolves the SPI interactions with the RS9110-N-11-2X module.

4.1 SPI Interface

4.1.1 Overview

The SPI Interface is a full duplex serial Host interface, which supports 8-bit and 32-bit data mode. The SPI interface of the module consists of the following signals:

SPI_MOSI (Input)- Serial data input for the module.

SPI_MISO (Output)- Serial data output for the module.

SPI_CS (Input)- Active low slave select signal. This should be low when SPI transactions are to be carried out.

SPI_CLK (Input)- SPI clock. Maximum value allowed is 25 MHz.

INTR (Output)- Active high, level interrupt output from the module.

The module acts as a SPI slave while the Host is the SPI master

4.1.2 Features

- Serial interface
- Supports 8-bit and 32-bit data mode
- Supports little-endian format

4.1.3 Interrupt

The interrupt signal, an output pin of the module, is asserted when the module needs to indicate to the host that there is data waiting to be sent from the module to the host through the SPI interface. It is also asserted when the module wakes up from sleep in SPI based modules. This interrupt is active-high, level triggered.

4.1.4 Operation

The SPI interface is invoked by the Host by sending specific commands. The descriptions that follow pertain to a Little Endian Host**.**

The SPI interface in the module acts as a SPI slave. Following parameters should be in **the host SPI interface.**
CPOL (clock polarity) = 0,
CPHA (clock phase)   = 0.

The SPI interface is programmed to perform a certain transfer using commands C1, C2, C3 and C4 and a 32-bit address. For all the Commands and Addresses, the Host is configured to transmit data with 8-bit mode. At the end of all the Commands and Address, the Host is reconfigured to transfer data with 8-bit or 32-bit mode depending on the commands issued. The Slave responds to all the commands with a certain response pattern.

The four commands C1, C2, C3, and C4 indicate to the SPI interface all the aspects of the transfer.

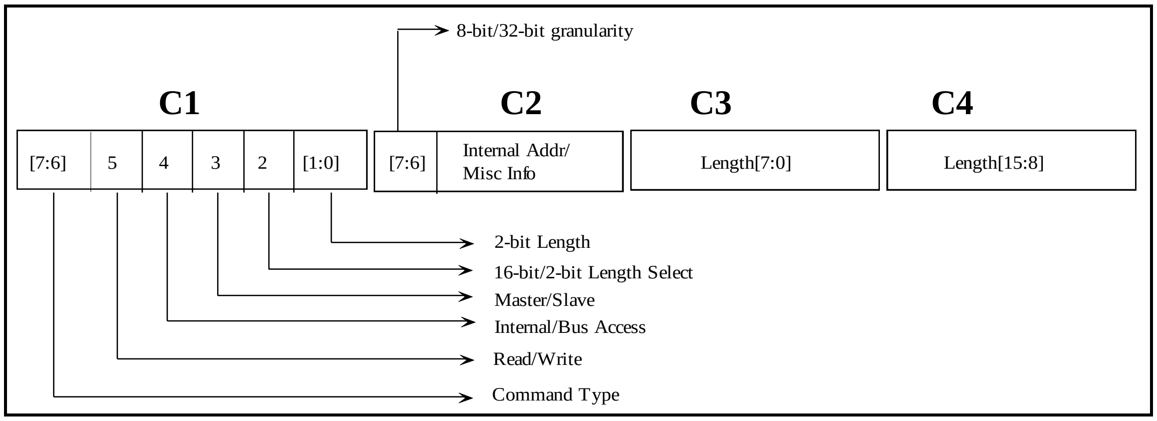


Figure 15: SPI Command Description

The command description is as follows:

Command	Bit Number	Description
C1	[7:6]	Command Type "00"- Initialization Command "01"- Read/Write Command "10", "11"- Reserved for future use
	5	Read/Write '0'- Read Command '1'- Write Command
	4	Internal/Bus Access '0'- SPI Slave Internal Access. Used in register read/write '1'- Bus Access. Used in Master read/write and slave read/write
	3	Master/Slave Access '0'- Master Access. Used in master read/write '1'- Slave Access. Used in slave read/write
	2	2-bit or 16-bit length for the transfer '0'- 2-bit length for the transfer '1'- 16-bit length for the transfer
	1:0	2-bit length (in terms of bytes) for the transfer (valid only if bit 2 is cleared) "00"- 4 Bytes length "01"- 1 Byte length "10"- 2 Bytes length "11"- 3 Bytes length
C2	7:6	8-bit or 32-bit mode. Indicates the granularity of the write/read data. Note: The SPI (C1, C2, C3, C4) commands and addresses (A1, A2, A3, A4) will always be 8-bit irrespective of this value. "00"- 8-bit mode "01"- 32-bit mode "10", "11"- Reserved for future use
	5:0	This carries the SPI Slave's internal address if bit 4 for Command C1 is

Command	Bit Number	Description
		cleared. Otherwise, it these bits are used to indicate type of the packet, flowing from the Host to the RS9110-N-11-2X module. Definition is as follows: 2 – Regular data frame. 4 – Management frame. Other values are reserved for future use.
C3	7:0	Length (7:0) LSB of the transfer's length (which is in terms of bytes) in case bit 2 of C1 is set. This command is skipped if bit 2 of C1 is cleared.
C4	7:0	MSB of the transfer *Length (15:8)* (which is in terms of bytes) in case bit 2 of C1 is set. This command is skipped if bit 2 of C1 is cleared i.e. if 2-bit length is selected.

Table 5: SPI Command Description

To all these commands, the SPI interface responds with a set of unique responses. Refer section Slave Response .

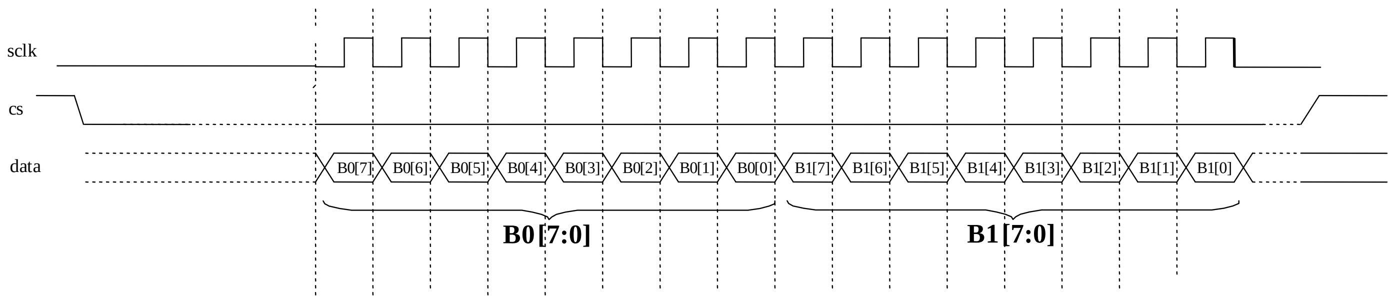
4.2 Bit ordering of SPI transmission/reception

8-bit mode

If a sequence of bytes <B3[7:0]> <B2[7:0]> <B1[7:0]> <B0[7:0]> is to be sent, where B3 is interpreted as the most significant byte, then the sequence of transmission is as follows :

B0[7] ..B0[6] .. B0[0] -> ***B1[7] ..B1[6] ..B1[0]*** -> ***B2[7] ..B2[6] .. B2[0]*** -> ***B3[7] ..B3[6] .. B3[0]***

B0 is sent first, then B1, then B2 and so on.
In each of the bytes, the MSB is sent first. For example, when B0 is sent, B0[7] is sent first, then B0[6], then B0[5] and so on. Same is the case when receiving data. In this example, B0[7] is expected first by the receiver, then B0[6] and so on. Also, as can be seen, the data changes value at the falling edge and is latched at the rising edge.



<u>Figure 16: 8-bit mode</u>

32-bit mode

If a sequence of 32-bit words is <W3[31:0]> <W2[31:0]> <W1[31:0]> <W0[31:0]> is to be sent, where W3 is interpreted as the most significant word, then the sequence of transmission is as follows :

W0[31] ..W0[30] ..W0[0] -> ***W1[31] ..W1[30] ..W1[0]*** -> ***W2[31] ..W2[30] ..W2[0]*** -> ***W3[31] ..W3[30] ..W3[0]***

W0 is sent first, then W1, then W2 and so on.
In each of the 32-bit words, the MSB is sent first. For example, when W0 is sent, W0[31] is sent first, then W0[30], then W0[29] and so on. Same is the case when receiving data. In this example, W0[31] is expected first by the receiver, then W0[30] and so on. Also, as can be seen, the data changes value at the falling edge and is latched at the rising edge.

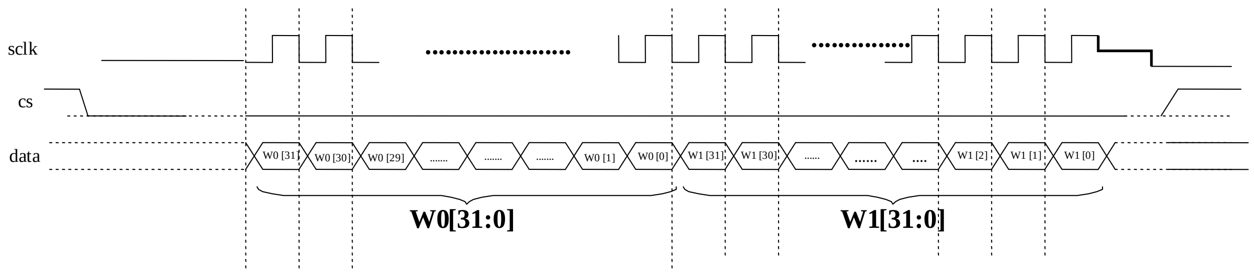


Figure 17: 32-bit mode

4.3 Slave Response

The SPI slave gives responses to the SPI Master's requests. These are as follows:

- A success/failure response at the end of receiving the command. Refer Interactions in the physical interface (this structure is similar for all following read/write operations) . **This response is driven with 8-bit mode during the Command and Address phase and is then switched to 8-bit or 32-bit mode during the Data phase as per the command issued**.
    - Success: 0x58 or0x00000058
    - Failure: 0x52 or 0x00000052
- An 8-bit or 32-bit start token is transmitted once the four commands indicating a read request are received and the slave is ready to transmit data. The start token is immediately followed by the read-data.
    - Start Token: 0x55 or 0x00000055
- An 8-bit or 32-bit busy response in case a new transaction is initiated while the previous transaction is still pending from the slave side.
    - Busy Response: 0x54 or 0x00000054

Bit ordering of slave response

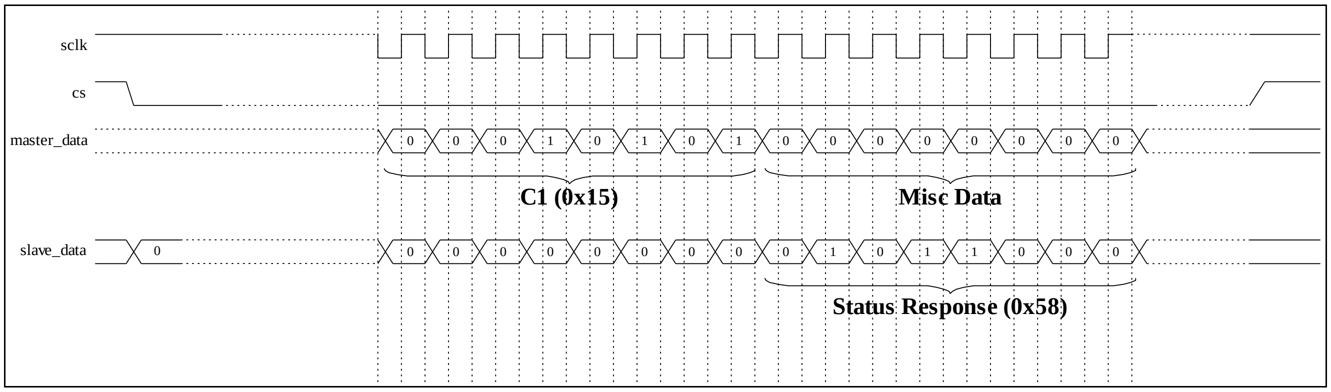
The bit ordering is same as explained in Bit ordering of SPI transmission/reception. For example, 0x58 response for 8-bit success is sent as

0 -> 1 -> 0 -> 1-> 1 -> 0 -> 0 -> 0 . That is 0 is sent first, then 1, then 0, then 1, and so on.

4.4 Initialization

The Initialization Command is given to the Slave to initialize the SPI interface. The SPI interface remains non-functional to any command before initialization and responds only after successful initialization. Initialization should be done only once after the power-on. The SPI Slave treats any subsequent initialization commands before the reset as errors.

For the initialization command, the Host drives C1 command followed by an 8-bit dummy data. Bits [7:6] of C1 are cleared and 0x15 is driven on bits [5:0]. Status response from the SPI Interface is driven during the transmission of the dummy data i.e. after the transfer of 8-bits of command C1.



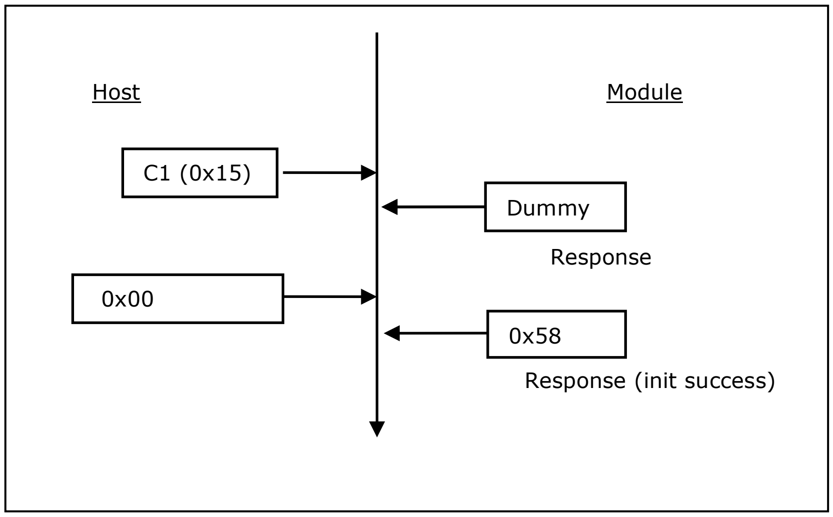


Figure 18: SPI Slave Initialization

4.5 Host Interactions using SPI commands.

This section describes the procedures to be followed by the Host to interact with the RS9110-N-11-2X module using SPI commands.

The Host interactions to the module could be categorized as below

Command	Command Category
Memory write	Master Write
Memory read	Master Read
Frame write	Slave Write
Frame read	Slave Read
Register write	SPI Interface Internal Registers Write/ master write
Register Read	SPI Interface Internal Registers Read/Master read

Table 6: Command Types

4.5.1 Memory Write (Master Write)

Host accesses the memory of the RS9110-N-11-2X module to upgrade the firmware and configure some memory addresses. Memory write is also referred to as master write.

To write into a memory location, **memory write** command has to be framed as described in the figure below. If a "busy" or "failure" response is sent from the slave, the retries need to be done or the slave should be reset.

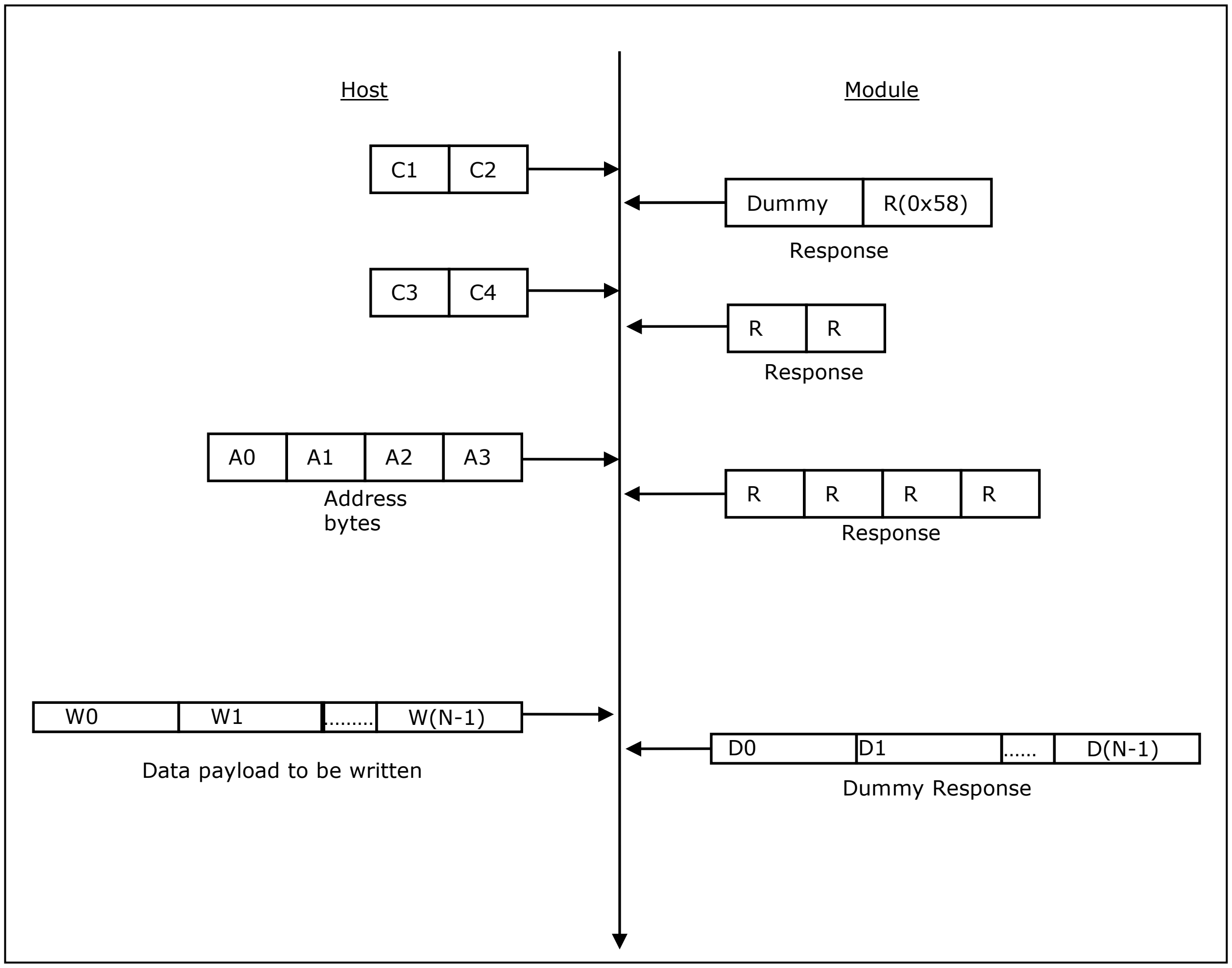


Figure 19: Memory Write

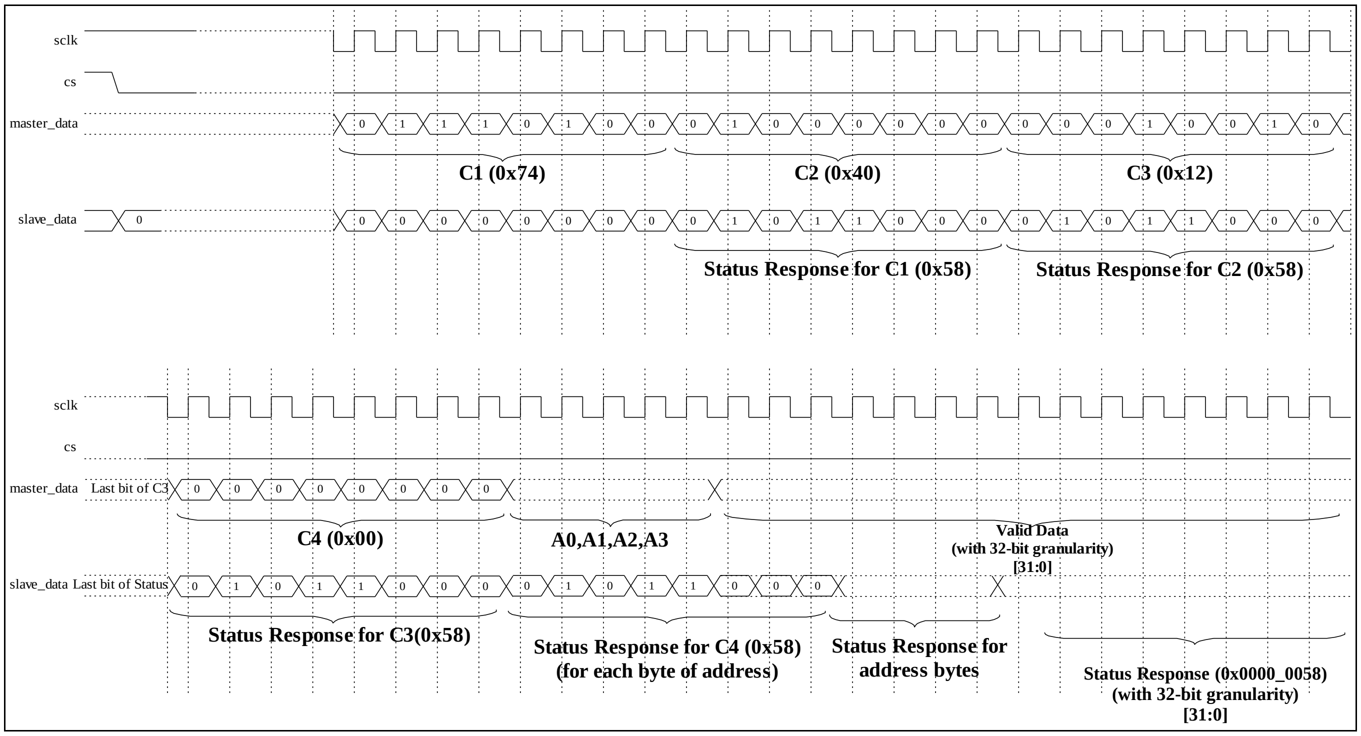

sclk
cs
master_data
C1 (0x74)
C2 (0x40)
C3 (0x12)
slave_data
Status Response for C1 (0x58)
Status Response for C2 (0x58)
Last bit of C3
C4 (0x00)
A0,A1,A2,A3
Valid Data (with 32-bit granularity) [31:0]
Last bit of Status
Status Response for C3(0x58)
Status Response for C4 (0x58) (for each byte of address)
Status Response for address bytes
Status Response (0x0000_0058) (with 32-bit granularity) [31:0]

Figure 20: Interactions in the physical interface (this structure is similar for all following read/write operations)

The following is the procedure to be followed by the Host.

1.Send the commands C1, C2.

2.Read the response (R) from the SPI slave (RS9110-N-11-2X module). Response will as described in Slave Response. Status 0x58 indicates that the slave is ready.

3.Host should send the commands C3 and C4, followed by the 4 byte address (corresponding to the memory location) and data. Status 0x54 indicates that the device is busy. Host has to retry. Status 0x52 indicates a failure response.

The commands C1, C2, C3 and C4 are sent in the following sequence (explained in

Bit ordering of SPI transmission/reception) : C1 first, then C2, C3 and finally C4. The bit ordering is

C1[7] -> C1[6] … C1[0] -> C2[7] -> C2[6] … C2[0] -> C3[7] -> C3[6] … C3[0] -> C4[7] -> C4[6] … C4[0]. That is, C1[7] bit is sent first, then C1[6] and so on.

Total data payload size should be 1 byte, or 2 bytes or multiples of 4 bytes in this operation. For example, if 5 bytes need to be sent, it should be padded with 3 more dummy bytes to make it 8 bytes (multiple of 4) before sending

Original data <W0[7:0]> <W1[7:0]> <W2[7:0]> <W3[7:0]> <W4[7:0]>

Padded data <W0[7:0]> <W1[7:0]> <W2[7:0]> <W3[7:0]> <W4[7:0]> <D5[7:0]><D6[7:0]><D7[7:0]> , where D[7:0] is the dummy data.

The first byte sent is W0, the second byte sent is W1 and so on (refer Bit ordering of SPI transmission/reception).

4.5.2 Frame Write (Slave Write)

The sequence of command transactions (with no failure from the slave) for a Frame Write (also called slave write) is as described in the figure below. The operations are similar to the memory write – except that bit 3 of C1 is set and the Address phase (A0,A1, A2, A3) is skipped.

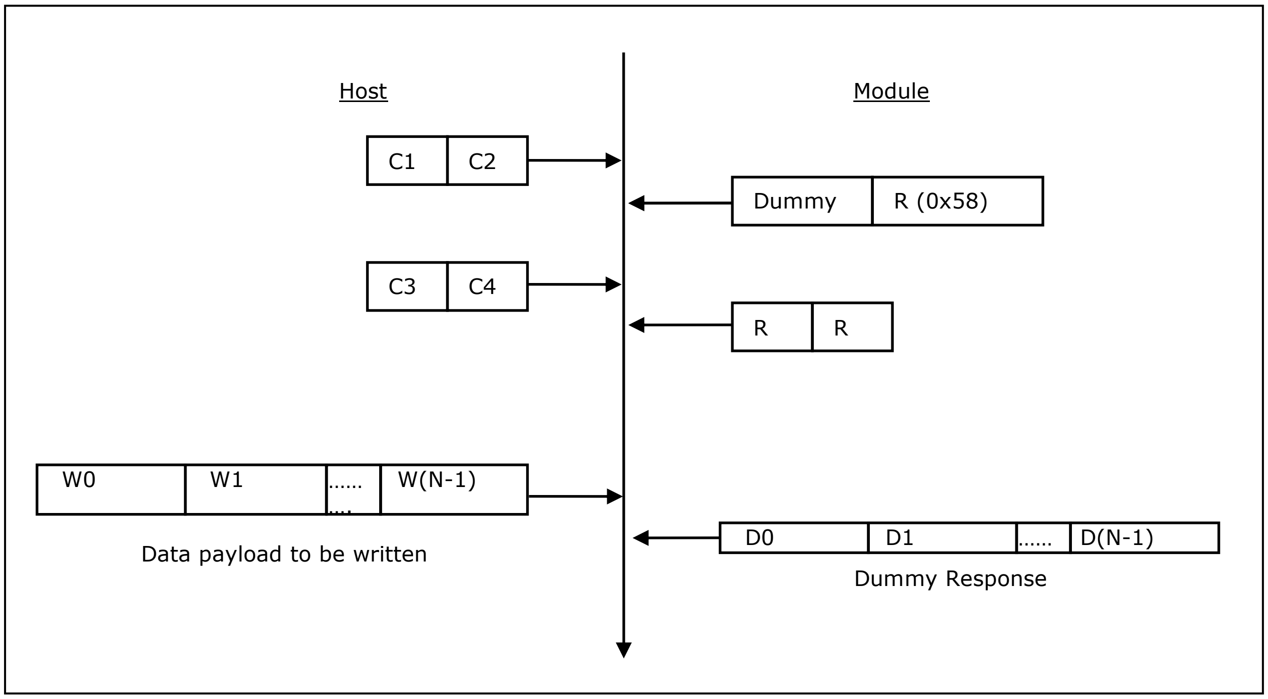


Figure 21: Frame Write using SPI Commands

The following is the procedure to be followed by the Host.

1. Prepare and send the commands C1, C2 together as described for Slave write.

2. Read the response (R) from the SPI slave (RS9110-N-11-2X module).

3. Status 0x58 indicates that the slave is ready. Host can send the commands C3 and C4, followed by the data. Status 0x54 indicates that the device is busy. Host has to retry. Status equal to 0x 52 indicates a failure response.

Data payload size should be in multiples of 4 bytes in this operation. For example, if 5 bytes of data is to be written, it should be padded with 3 more dummy bytes to make it 8 bytes (multiple of 4) before sending.

Original data <W4[7:0]> <W3[7:0]> <W2[7:0]> <W1[7:0]> <W0[7:0]>

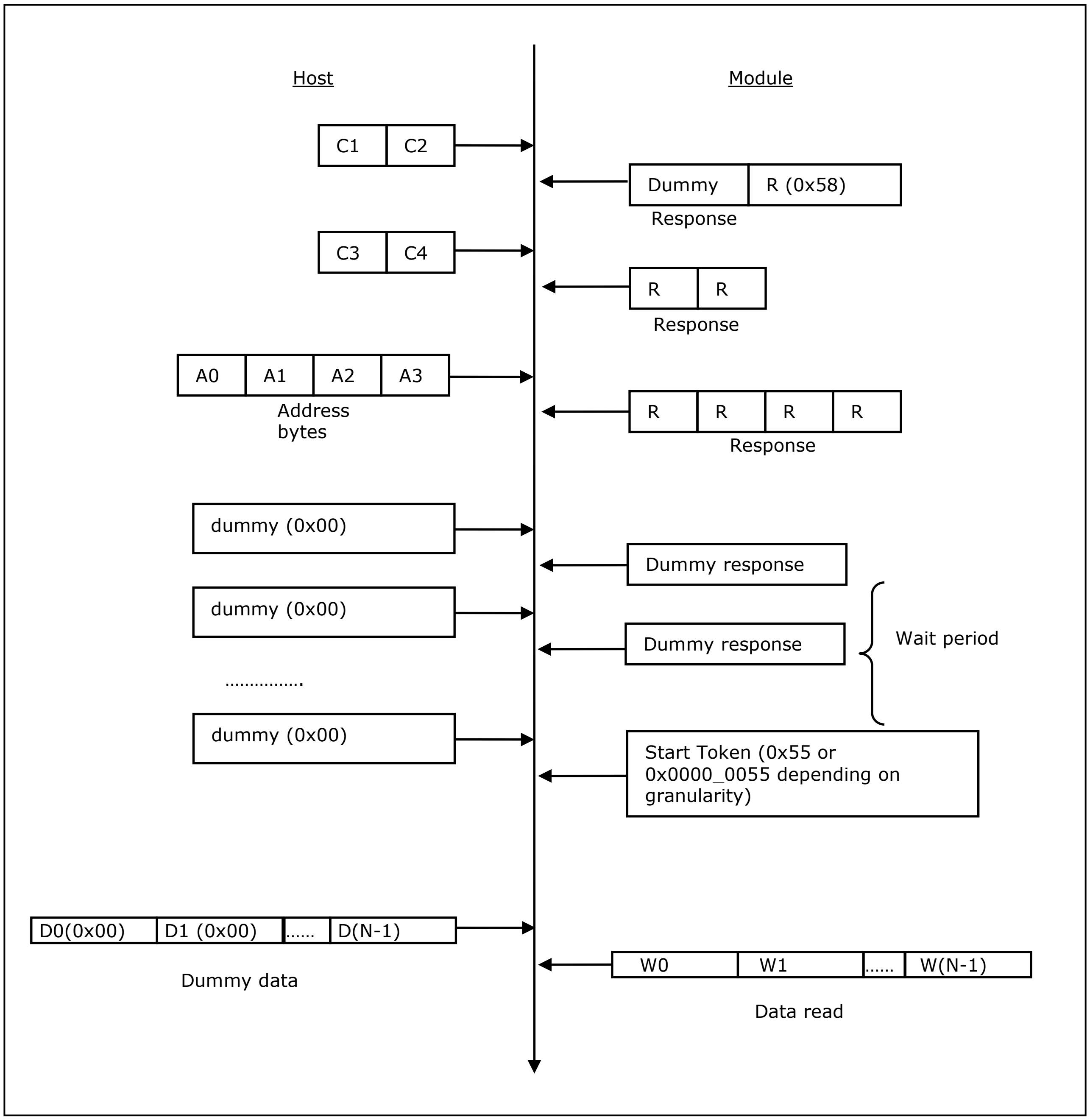
Padded data <D7[7:0]> <D6[7:0]> <D5[7:0]> <W4[7:0]> <W3[7:0]> <W2[7:0]> <W1[7:0]> <W0[7:0]> , where D[7:0] is the dummy data.

The first byte sent is W0 , the second byte sent is W1 and so on.(refer Bit ordering of SPI transmission/reception.)

4.5.3 Memory Read (Master Read)

To read from a memory location, Master Read command has to be formed.

The following figure gives the flow for memory reads between the master and the slave.



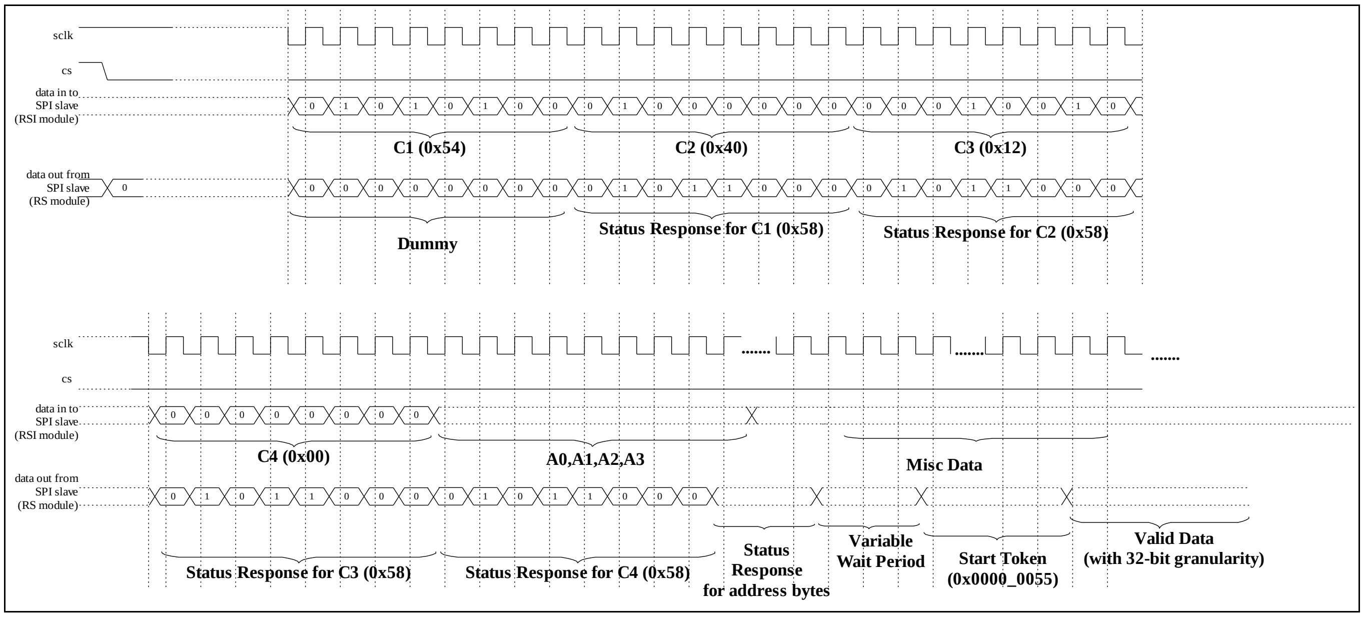

sclk
cs
data in to SPI slave (RSI module)
C1 (0x54)
C2 (0x40)
C3 (0x12)
data out from SPI slave (RS module)
Dummy
Status Response for C1 (0x58)
Status Response for C2 (0x58)
C4 (0x00)
A0,A1,A2,A3
Misc Data
Status Response for C3 (0x58)
Status Response for C4 (0x58)
Status Response for address bytes
Variable Wait Period
Start Token (0x0000_0055)
Valid Data (with 32-bit granularity)

Figure 22: Memory Read

The following is the procedure to be followed by the Host to do a memory read.

1. Prepare and send the commands C1, C2 as described for Master read.
2. Read the response from the SPI slave (RS9110-N-11-2X module).
3. Status 0x58 indicates that the slave is ready. Host should next send the commands C3, C4, which indicate the length of the data to be read, followed by the address of the memory location to be read.
4. After sending/receiving C3, C4 commands/response and the addresses, the Host should wait for a start token (0x55). Host writes a stream of dummy bytes to read the data from the module each time. The Host should read this data until the start token is received. The data following the start token should be interpreted as the frame of specified length that is read from the slave.
5. Status 0x54, after C1 and C2 bytes, indicates that the device is busy. Host has to retry.
6. Status 0x52, after C1 and C2 bytes, indicates a failure response from the slave.

There is a variable wait period, during which dummy data is sent. After this period, a start token is transmitted from the slave to the master. The start token is followed by valid data. The start token indicates to the Host the beginning of valid data. To read out the valid data, dummy data [D0, D1, D2, …. D(N-1)] is sent. N is the number of bytes to be read.

The commands C1, C2, C3 and C4 are sent in the following sequence (explained in Bit ordering of SPI transmission/reception) : C1 first, then C2, C3 and finally C4. The bit ordering is

C1[7] -> C1[6] … C1[0] -> C2[7] -> C2[6] … C2[0] -> C3[7] -> C3[6] … C3[0] -> C4[7] -> C4[6] … C4[0]. That is, C1[7] bit is sent first, then C1[6] and so on.

If <D3[7:0]> <D2[7:0]> <D1[7:0]> <D0[7:0]> is the data read, then D0 is sent from module first, then D1 and so on.

D0[7] -> D0[6] … D0[0] -> D1[7] -> D1[6] … D1[0] -> D2[7] -> D2[6] … D2[0] -> D3[7] -> D3[6] … D3[0]

4.5.4 Frame read (Slave Read)

This is same as master read, except that bit 3 of C1 is set and the Address phase is skipped. The sequence of command transactions (with no failure from the slave) for a Frame Read is as described in the figure below.

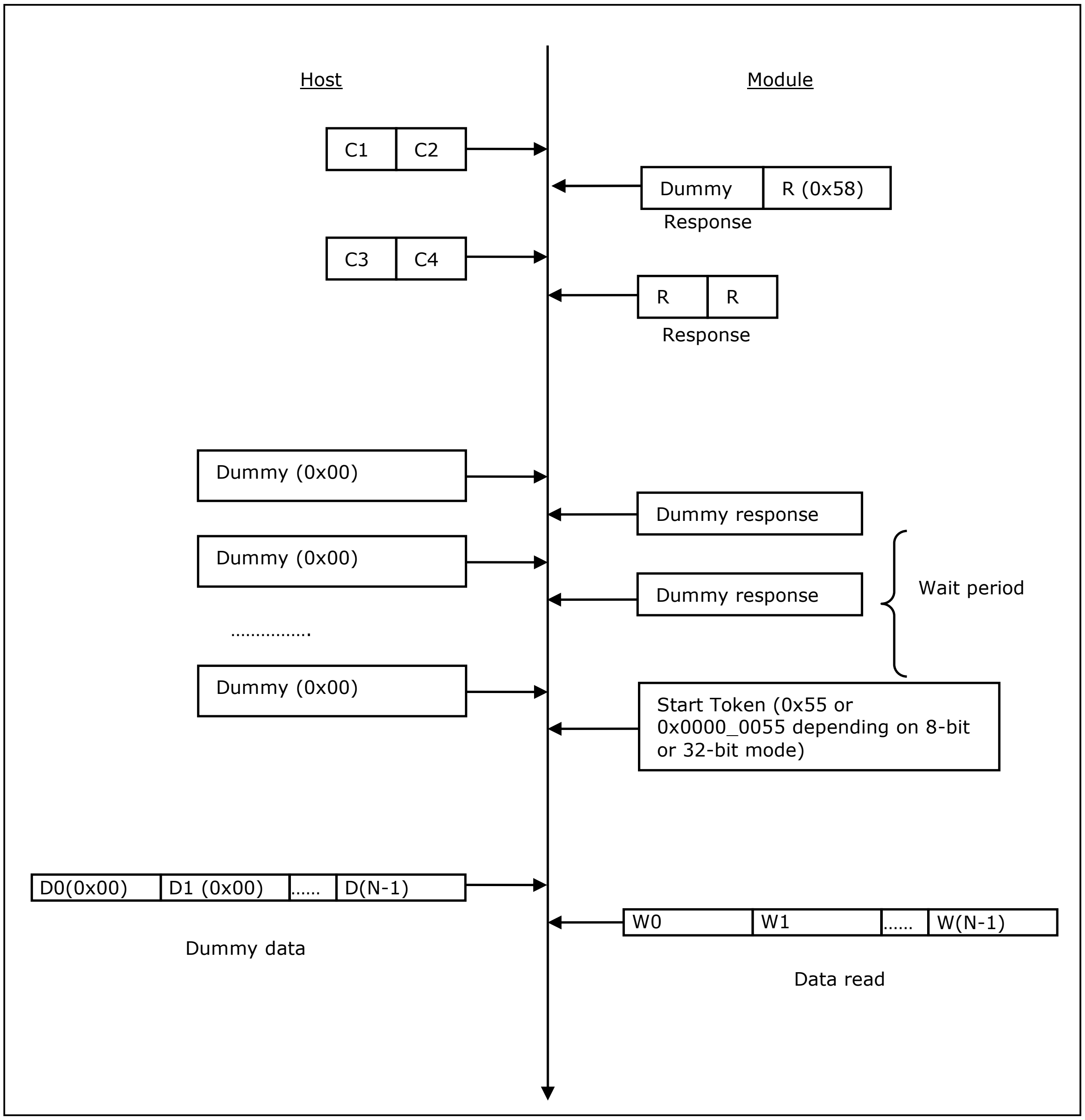


Figure 23: Frame Read using SPI Commands

The following is the procedure to be followed by the Host to do a memory read.

1. Prepare and send the commands C1, C2 as described for Slave Read.
2. Read the response from the SPI slave (RS9110-N-11-2X module).
3. Status 0x58 indicates that the slave is ready. Host should send the commands C3, C4 which indicate the length of the data to be read.

4. After sending/receiving C3, C4 commands/response, Host should wait for a start token (0x55). The data then follows after the start token. Host writes a dummy byte to read the data from the module each time. The Host should read this data until the start token is received. The data following the start token should be interpreted as the frame of specified length that is read from the slave. Status 0x54, after C1 and C2 bytes, indicates that the device is busy. Host has to retry. Status 0x52, after C1 and C2 bytes, indicates a failure response from the slave.

The commands C1, C2, C3 and C4 are sent in the following sequence (explained in Bit ordering of SPI transmission/reception) : C1 first, then C2, C3 and finally C4. The bit ordering is

C1[7] -> C1[6] … C1[0] -> C2[7] -> C2[6] … C2[0] -> C3[7] -> C3[6] … C3[0] -> C4[7] -> C4[6] … C4[0]. That is, C1[7] bit is sent first, then C1[6] and so on.

If <D3[7:0]> <D2[7:0]> <D1[7:0]> <D0[7:0]> is the data read, then D0 is sent from module first, then D1 and so on.

D0[7] -> D0[6] … D0[0] -> D1[7] -> D1[6] … D1[0] -> D2[7] -> D2[6] … D2[0] -> D3[7] -> D3[6] … D3[0]

4.5.5 Register Reads and Writes

Register Summary

Base Address: 0x0800_0000	
Address Offset	**Register Description**
0x00	SPI_HOST_INTR
Base Address: 0x2200_0000	
Address Offset	**Register Description**
0x04	SOFT_RESET
0x08	INTR_MASK
0x10	INTR_CLEAR

Table 7: SPI Register Description

SOFT_RESET

SOFT_RESET				
Base Address: 0x2200_0000, Offset Address: 0x04				
Bit	**Access**	**Function**	**Default Value**	**Description**
[31:0]	Read/Write	Soft Reset	0x0000_0001	Valid values are 0x0000_0001- Puts the module in soft reset 0x0000_0000- Brings the module out of reset. Refer figure Firmware Upgrade and Bootload

Table 8: SOFT_RESET Register

Register Write

This is same as **master write**. The address to be supplied is 0x2200_0004.

Register Read

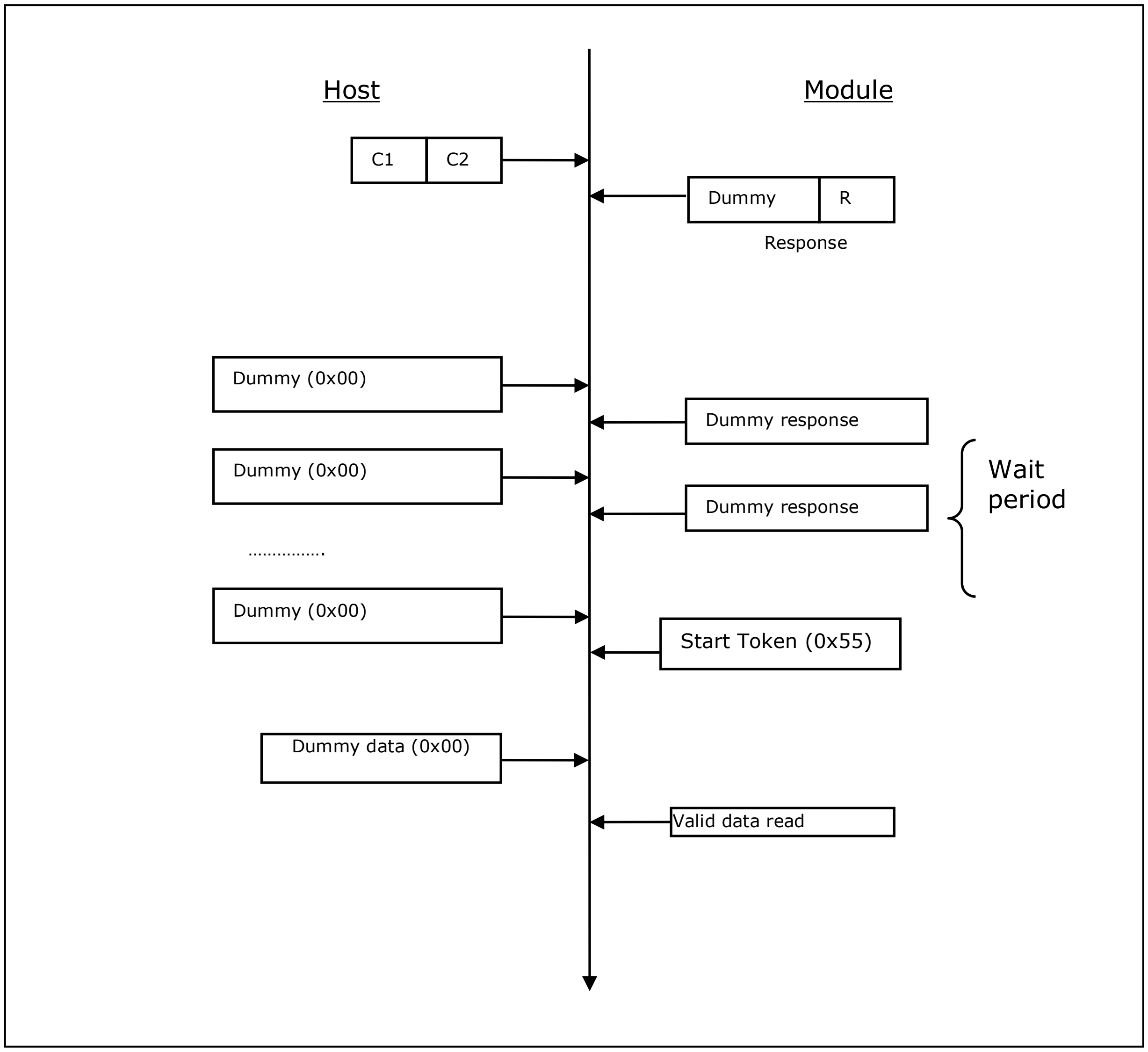
This is same as the **master read**. The address to be supplied is 0x2200_0004.

SPI_HOST_INTR

SPI_HOST_INTR				
Address: 0x00				
Bit	**Access**	**Function**	**Default Value**	**Description**
[7:0]	Read only	SPI_HOST _INTR	0x00	These bits indicate the interrupt vector value coming from module side. Bit 0: If '1', Buffer Full condition reached. This bit has to be polled every time before sending data to the module, to check that the buffer in the module is is full or not. Bit 1: Reserved. Ignore this bit. Bit 2: Reserved Bit 3: If '1', indicates Data packet or response to Management frames is pending. This is a self-clearing interrupt and is cleared after the packet is read by the Host. Bit 4: Reserved. Bit 5: In Power Mode 1, If '1', indicates that the module is awake and requesting to shut down the Core control block. The Host needs to write a '1' to bit 5 of the INTR_CLEAR register to clear the interrupt and also indicate to the module that the Core control block can be shut down. Once the module switches from power mode 1 to power mode 0, this bit is automatically cleared by the module (which indicates that the module successfully switched back to power mode 0)

Table 9: SPI Host Interrupt Register

Register Read
To read this register, C1 should be set to 0x41 and C3, C4 and address phases are skipped. It is a read-only register. It can be read using the following sequence. The C2 command bits [5:0] should be the address mentioned for this register.


Host
Module
C1
C2
Dummy
R
Response
Dummy (0x00)
Dummy response
Dummy (0x00)
Dummy response
Wait period
…………….
Dummy (0x00)
Start Token (0x55)
Dummy data (0x00)
Valid data read

INTR_MASK

INTR_MASK				
Base Address: 0x2200_0000, Offset Address: 0x08				
Bit	**Access**	**Function**	**Default Value**	**Description**
[7:6]	R	Reserved	0x3	Reserved for future use
5	R/W	POWER_MODE1_MASK	0x0	This bit is used to mask the interrupt generated by bit 5 of the Interrupt register. The Host can mask this bit when it does not want to shut down the Core control block after the module raises the interrupt. The Host has to ensure that the default values of the rest of the register's bits are not disturbed. For this, it has to write 0xF3 to this register to enable the mask for this interrupt bit and 0xD3 to disable the mask for this interrupt bit.
[4:0]	R/W	Reserved	0x13	Reserved

Table 10: Interrupt Mask Register

The default values are set during module boot-up process.

Register Write

This is same as **master write**. The address to be supplied is 0x2200_0008. A read- modify-write should be performed to write into this register so that only the bit number 5 is written while the other bits are not changed from their existing values.

Register Read

This is same as the **master read**. The address to be supplied is 0x2200_0008.

INTR_CLEAR

INTR_CLEAR				
Base Address: 0x2200_0000, Offset Address: 0x10				
Bit	**Access**	**Function**	**Default Value**	**Description**
[7:6]	R	Reserved	0x0	Reserved for future use
5	R/W	POWER_MODE1 interrupt clear	0x0	Writing a '1' to this bit clears the interrupt raised by the module due to this bit.
[4:0]	R/W	Reserved	0x0	Reserved for future use

Table 11: Interrupt Clear Register

Register Write

This is same as **master write**. The address to be supplied is 0x2200_0010. Bit numbe5 can be set to '1' or '0' as desired, while the other bits should be set to all '0' while writing.

Register Read

This is same as the **master read**. The address to be supplied is 0x2200_0010.

4.6 Power modes[1]

The RS9110-N-11-2X module supports three power modes with the SPI interface. The Host can switch among the power modes using the commands depending on the Wi-Fi connection status as defined in this section.

The power modes supported by the RS9110-N-11-2X module for SPI interface are classified based on the Host's capability to negotiate with RS9110-N-11-2X and the Wi-Fi connection status.

4.6.1 Power mode 0

In this mode, power save is disabled in the RS9110-N-11-2X module. The module will be in Power Mode 0 by default.

4.6.2 Power mode 1

The module can be configured for Power Mode 1 or Power Mode 2. The module puts to sleep the Baseband and RF components between each DTIM, when there is no traffic. In Power Mode 1, the module also powers off the Core Control Block once it gets the indication from the Host. For this, the module first sends a sleep request (it does this everytime it wakes after a DTIM) to the Host by asserting interrupt line. After the host gets this interrupt, it reads the register **SPI_HOST_INTR** using register read. If the bit 5 of that register is '1', it is

[1] Not supported in ad-hoc mode

confirmed that the interrupt is raised because of sleep request. If there are packets to be sent from the host, then the Host should first mask the received interrupt by writing a '1' to bit 5 of the **INTR_MASK** register **(Base Address: 0x2200_0000, Offset Address: 0x08)** (read back the register, set bit 5 to '1' and write back, so that the other bits are not changed) and then start sending the data. If there are no packets to be sent from Host to module, then the module can be allowed to go to sleep by writing a '1' to bit 5 of **INTR_CLEAR** register **(Base Address: 0x2200_0000, Offset Address: 0x10)**. After this operation on **INTR_CLEAR** register, the Host should unmask the interrupt by writing a '0' to bit 5 of the **INTR_MASK** register(read back the register, set bit 5 to '1' and write back, so that the other bits are not changed). This is to ensure that the host can receive all future interrupts.

The average amount of power consumed in this mode would depend on the DTIM period and data traffic.

If this mode is invoked when the module is not connected to an Access Point, then the module sleeps for the time supplied in command Sleep Timer and then wakes up.

4.6.3 Power mode 2

The RS9110-N-11-2X module powers off the Baseband and RF components in this mode. The Core Control block interacting with the Host is always functional in this mode. Hence, the module can receive commands from the host at any time.

NOTE: The average current consumption in Power Mode 2 is higher than Power Mode 1.

NOTE: While switching back to power mode 0 from power mode 1, module resets bit 5 of Host Interrupt register automatically (to indicate it has successfully switched back to power mode 0).

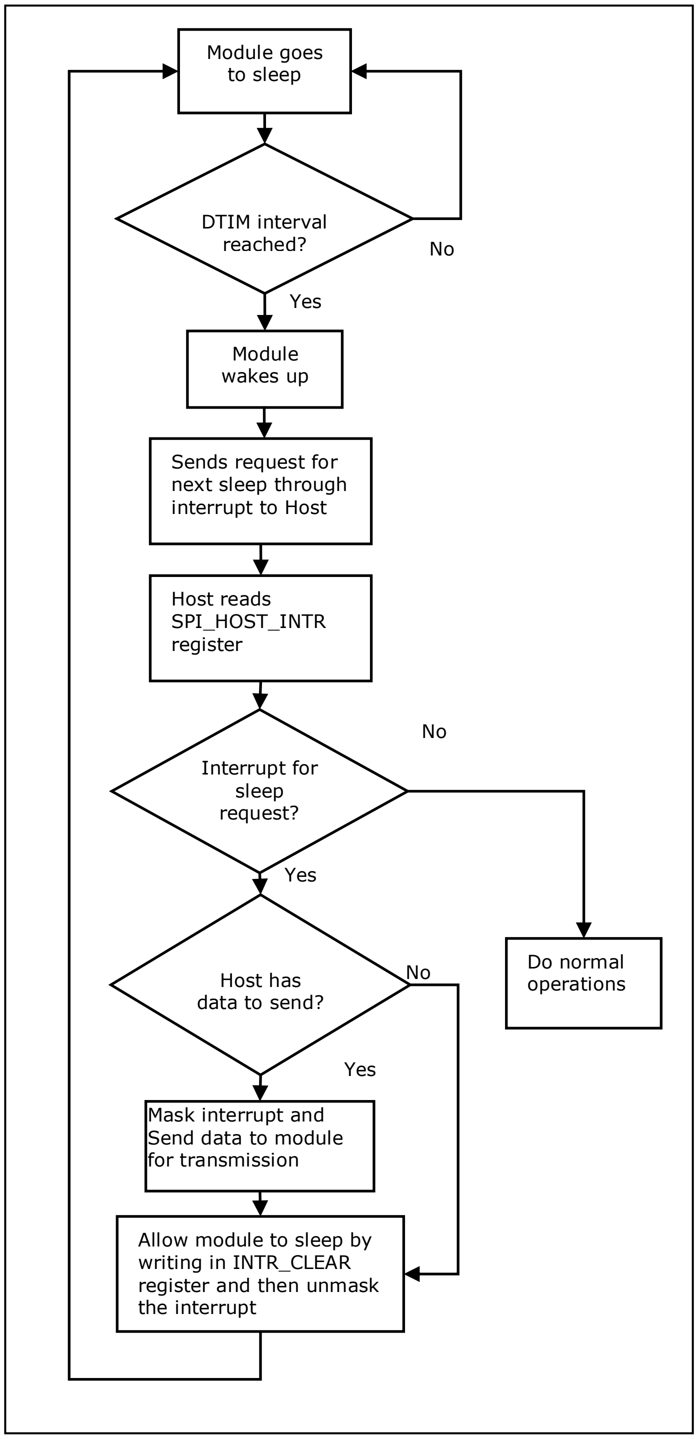

Module goes to sleep
DTIM interval reached?
No
Yes
Module wakes up
Sends request for next sleep through interrupt to Host
Host reads SPI_HOST_INTR register
Interrupt for sleep request?
No
Yes
Do normal operations
Host has data to send?
No
Yes
Mask interrupt and Send data to module for transmission
Allow module to sleep by writing in INTR_CLEAR register and then unmask the interrupt

Figure 24: Activities for Power Save Mode 1

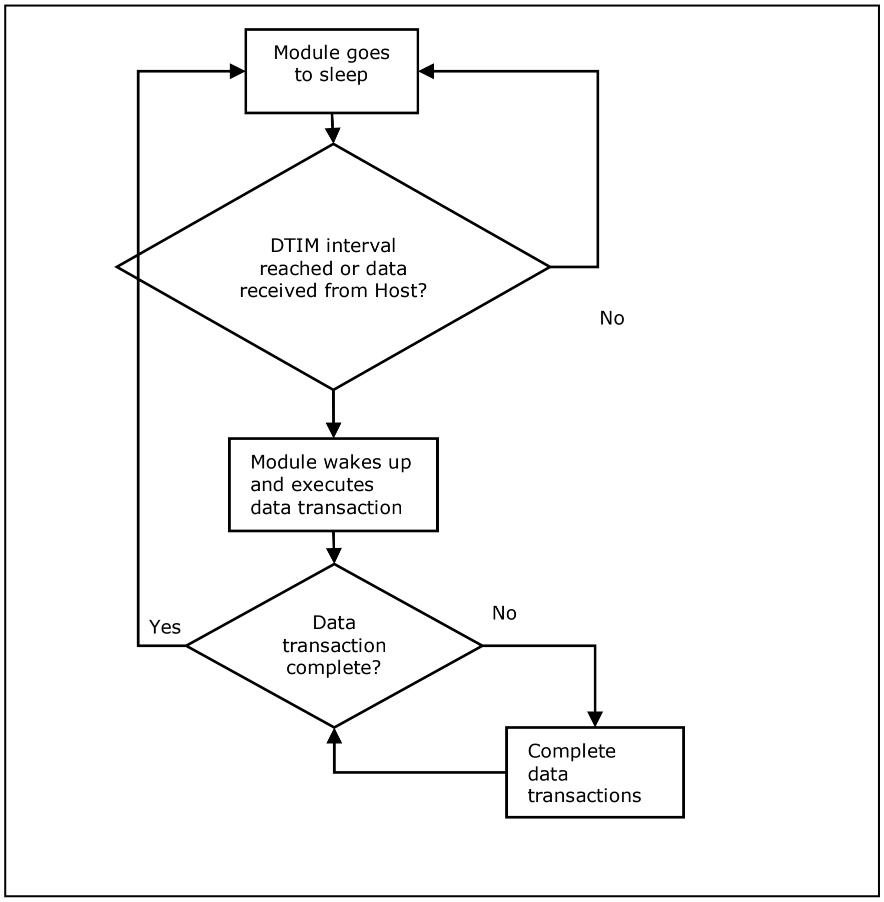

Module goes to sleep
DTIM interval reached or data received from Host?
No
Module wakes up and executes data transaction
Data transaction complete?
Yes
No
Complete data transactions

Figure 25: Activities for Power Save Mode 2

4.7 TCP/IP Stack Usage in RS9110-N-11-22/24/28

RS9110-N-11-22, RS9110-N-11-24 and RS9110-N-11-28 are Wi-Fi modules with integrated TCP/IP stack. This reduces the development effort for designers who don't have the TCP/IP stack on their Host platform. However, there are applications where the designer wants to run his own TCP/IP stack on the Host. The TCP/IP stack inside the module can be bypassed in such scenarios. This feature of bypassing the stack is available from firmware version 4.5.0 and above. For users using a version lower than 4.5.0, this feature is not available, but this PRM would be valid except for this section.

4.7.1 Using the Wi-Fi module with TCP/IP stack enabled

The Wi-Fi module is by default in "TCP/IP stack enabled" mode.

1. The reference driver that the user should use for this mode is placed inside RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP
2. The API manual that should be used is placed in RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP/Documentation/

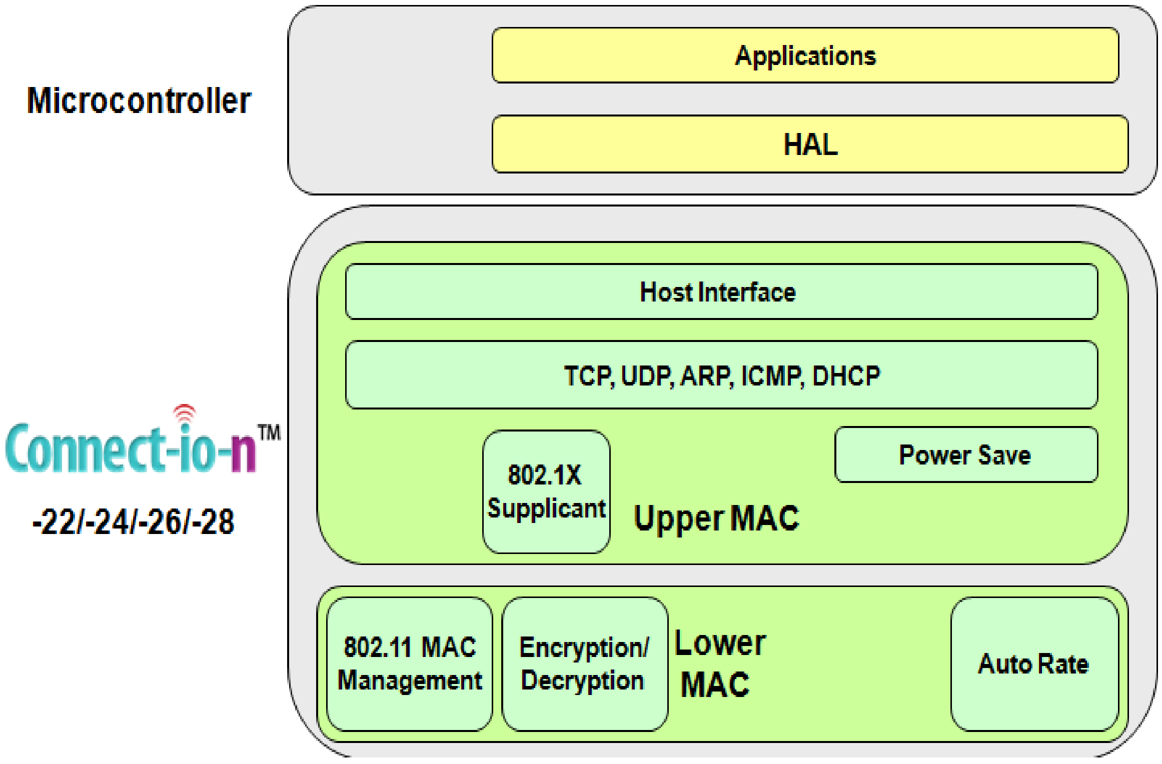


Figure 26: Set-up with TCP/IP Enabled

4.7.2 Using the Wi-Fi module with TCP/IP stack bypassed

The TCP/IP stack in the WiFi module can be bypassed by sending a software command from the Host to the module.

The "TCP bypass command" should be issued to convert the RS9110-N-11-22/24/28 from a TCP/IP enabled module to a TCP/IP bypassed module. The following should be taken care of:

1. The reference driver that the user should use for this mode is placed inside RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP_Bypass
2. The API manual that should be used is placed in RS.CN.GENR.SP.x.x.x/Driver/Driver_TCP_Bypass/Documentation/
3. In this mode, the WiFi module presents itself as a standard network interface, which would talk to the Host TCP/IP stack through the network driver in the Host. For transmitting data to the air interface, the data should be framed by the TCP/IP stack in the Host as an Ethernet frame, and the WiFi module does only the WLAN framing on top of it. The interface is according to standard networking protocols. For example *:* *http://www.makelinux.com/ldd3/chp-17-sect-3* describes the structure for Linux. For successful sending and receiving of data, the MAC ID of the module should be assigned to a field inside the net_device structure dev_addr of the network driver on the Host OS. Once this is stored after association with AP, the user can bring the wlan interface up
   In Linux, this is usually done by:

```
ifconfig wlan0 <ipaddr>

ifconfig wlan0 up
```

The user can query wlan0 interface status using:

```
ifconfig wlan0
```

The Redpine Module's MAC address should be listed in the HW Addr (00:23:A7:xx:xx:xx)

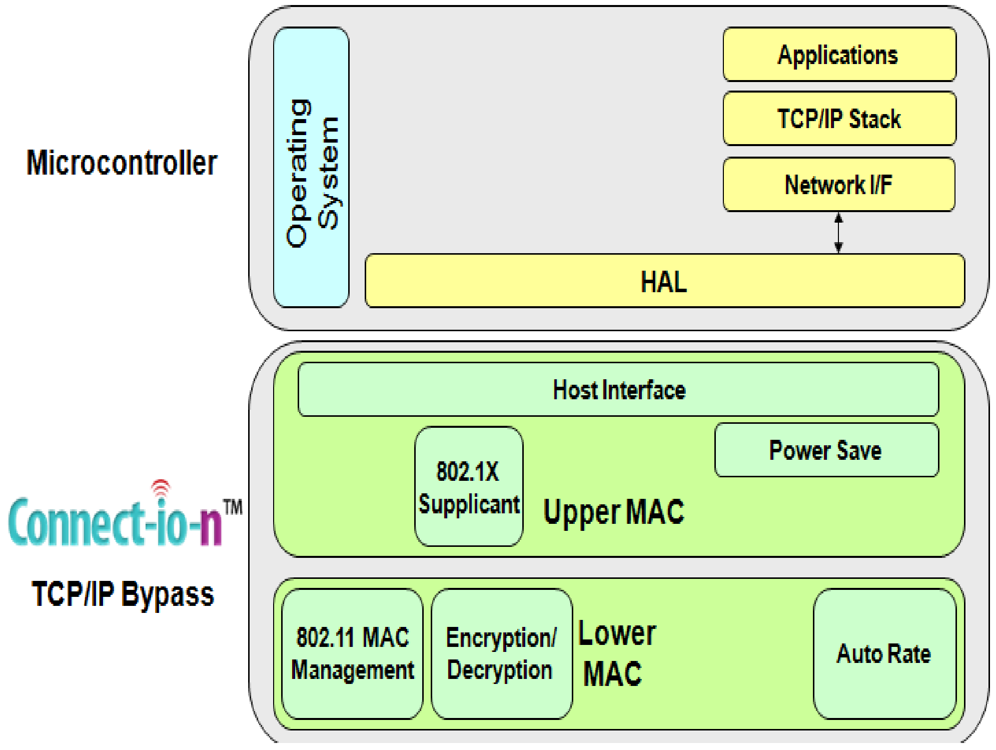


Figure 27: Set-up with TCP/IP Disabled

4.8 Command frames

The module is configured and operated using the SPI interface. The data payload (as mentioned in the preceding sections) for these configuration and operation commands is called a **command frame**.

The **command frame** is categorized **as management or data frames**. The management frames are used to configure the Wi-Fi module to access Wi-Fi connectivity. The data frames are used to interact with the TCP/IP stack and operate the module.

The following is the frame format for management and data frames in SPI mode. Management and data frames are exchanged between host and module. Such a frame is sent from Host to the module to configure the module, and also is sent from module to host to send responses to these commands. The format of the

data payload for commands is shown below. This description is for a Little Endian System.

Descriptor (16 bytes)	Management/Data Frame Body (multiples of 4 bytes)

The following table provides the general description of the frame descriptor for management and data frames.

Word	Management Frame Descriptor	Data Frame Descriptor
Word0 W0[15:0]	Bits [7:0] – Length Bits [15:8] – Type of Management packet (see Management Frames)	Bits [11:0] – Length of the frame Bits [15:12] - Not used, should be set to all '0' while sending from Host to module. While receiving from module, these bits should be ignored
Word1 W1[15:0]	1. (0x0000) when sent from host to module. 2. When sent from module to host (as response frame), it contains the status Status Words for Response to Management Frames. 3. In the case of firmware upgrade, W1[15:0] is used to indicate the length of the firmware file. Please refer to the section Firmware Upgrade and Bootload for more details.	0x0000
Word2 W2[15:0]	0x0000	0x0000 – In TCP enabled mode. 0x0044 – In TCP bypass mode for relevant commands listed in section Other Commands (Data Frames) 0x0000 – In TCP Bypass mode, for IP data send and receive operations
Word3 W3[15:0]	0x0000	0x0000

Word	Management Frame Descriptor	Data Frame Descriptor
Word4 W4[15:0]	If word0[7:0]=0xFF, then W4[15:0] should be used for specifying the length of management frame body.	0x0000
Word5 W5 [15:0]	0x0000	0x0000
Word6 W6 [15:0]	0x0000	0x0000
Word7 W7 [15:0]	0x0004	0x0002

Table 12: Frame Descriptor for Management/Data Frames in SPI mode

NOTE: If the length stored in Word0 is 0xFF, then this is an indication that the actual length is stored in Word4.

4.9 Management Frames (Frame Write)

The management frames represent the command frames that are sent from the Host to the RS9110-N-11-2X module to configure the module for Wi-Fi access. These are frame write commands. The following are the types of management requests and responses and the corresponding codes. The first table is applicable when the Host sends the frames to the module, the second table is applicable when the module sends the frames to the host. The corresponding code is to be filled in W0[15:8] mentioned in the table above.

Request Frame	Code
Band	0x18
Init	0x10
Scan	0x11
Join	0x12
Power mode	0x19
ffinst1 Upgrade	0X13
ffinst2 Upgrade	0X14
ffdata Upgrade	0X15
Sleep Timer	0x22
Query MAC and network type	0x23
Feature Select	0x24
Enable store configuration feature	0x2A

Save configuration	0x2B
Get configuration	0x2C
TCP/IP Bypass	0x2D
Set WEP keys	0x31

Table 13: Management Request and Related Code (Host to Module)

Response Frame	Code
Card Ready	0x89
Band	0x97
Init	0x94
Scan	0x95
Join	0x96
ffinst1 upgrade done	0x91
ffinst2 upgrade done	0x92
ffdata upgrade done	0x93
Sleep timer	0x98
Query MAC and network type	0xA1
Enable store configuration feature	0x9A
Save configuration	0x9B
Get configuration	0x9C
Feature Select Response	0x9D
TCP/IP Bypass	0xA3
Set WEP keys	0xA4

Table 14: Management Response and Related Code(Module to Host)

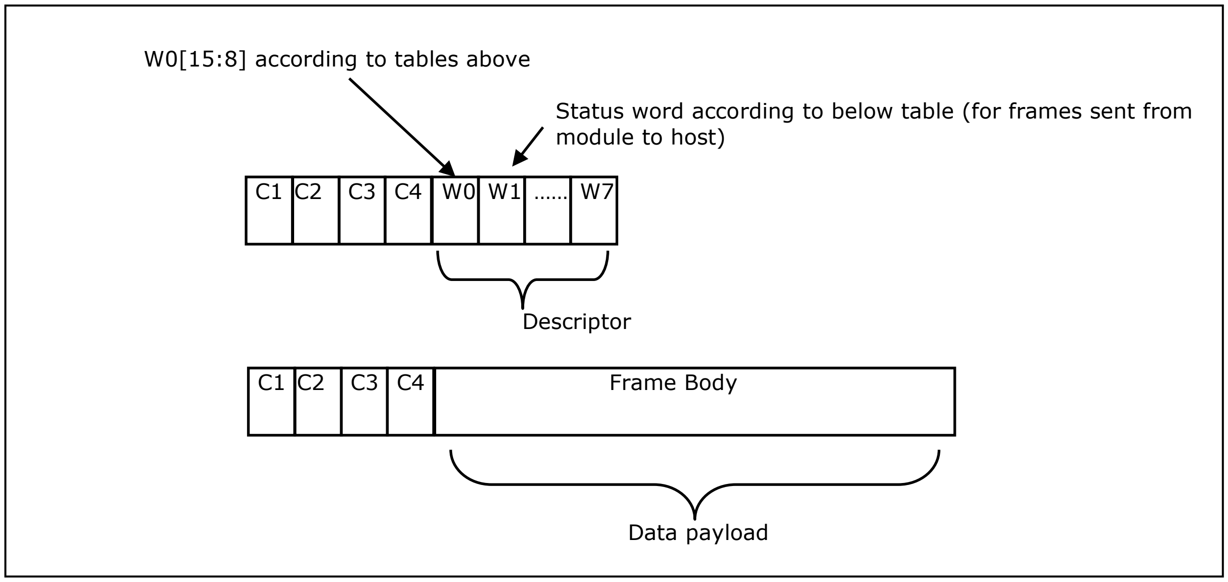

W0[15:8] according to tables above
Status word according to below table (for frames sent from module to host)
C1
C2
C3
C4
W0
W1
……
W7
Descriptor
C1
C2
C3
C4
Frame Body
Data payload

Figure 28: Data payload for management frames

Note: C1, C2, C3, C4 shown in the diagram above are sent from Host to Module. For responses coming from Module to Host, these are not sent by the module. For information about how to read responses of Management Frames please refer to Process for Reading Data .

Request Frame	**Code**
Command Success	0x0000
Wireless configuration failure	0x0001
1.If a scan command is issued after successful association 2.If a join command is issued after successful association	0x0002
No Access Point found. This error is also issued if the security mode of the AP is different from the one supplied to the module.	0x0003
PSK not configured, or PSK configured with invalid characters	0x0004
Roam List Exceeded	0x0005
Roam List Empty	0X0006
SSID not found in roaming list	0X0007
Failed to join in security enabled mode	0X0008
Roam Connect Error	0x0009

Back ground scan cancelled	0x0015
Invalid Channel	0x000A
SSID already in roam list	0x000B
SCAN_NOT_DONE (Upon issue of query MAC address and network type command without having scanned at least once since the bring-up of module)	0x000D
1.802.11 Authentication failure. 2.If a module tries to join to an ad-hoc network while its uNwType flag is set to 1 (infrastructure mode) in the join command	0x000E
3. Re-join limit over. If a module is taken out of coverage of the Access Point, it tries to re-join to the Access Point. It tries a maximum of 12 times to associate to the network. If the maximum number of retries is over and still the module cannot associate, then **this asynchronous response** is given. An interrupt is raised. W0 will be 0x0096 and W1 0x000F	0x000F
Error in TCP/IP Bypass command	0x0011
Channel switch failed	0x0012
Wrong PSK supplied	0x0013
Invalid network type. For example, if the module tries to join to an ad-hoc network with the network type set to infrastructure mode in the Join command and vice-versa.	0x0014
Unsupported bits are set in “Feature Select” command	0x0017
Passive scan is issued without enabling the feature in “Feature Select” command	0x0018
“Scan” command is issued before “Band” and “Init” commands	0x001A
If the data rate supplied to the “Join” command is not supported by the target AP	0x001B

Keep alive timeout. On lack of activity over an existing TCP socket connection, the module sends out a Keep-alive frame to the remote terminal. If there is no response, the module closes its TCP socket and sends out an interrupt to the Host. The Host, on reading the response as it does for any other command, would get 0x0033	0x0033
A packet of size more than 1460 bytes (TCP) or more than 1472 bytes (UDP) is sent.	0x0041
If the parameter uUseDHCP in command "Set IP Parameters" is supplied a value other than 0 or 1	0x007D
Attempt to open more than 8 sockets	0x0080
Unable to send data because of connectivity disruption (such as AP switched off abruptly)	0x008F
DHCP renewal not done by AP	0x009C
If "Query RSSI" command is given while the module is in IBSS mode	0x00C8
For RSSI query in unassociated state	0x00C9
Tx data rate configured in module does not match the rates supported by the target AP	0x00E2
"Scan" command is issued before "Band" and "Init" commands	0x00E3
Tx Buffers Unavailable	0x0064
Tx Data Packet Size exceeded	0x0065
Zero TCP/UDP payload	0x0066

Table 15: Status Words for Response to Management Frames

Bit ordering

The first word in the data payload that is sent or received is W0, then W1 and so on.

Word0[7:0] -> Word0[15:8] -> Word1[7:0] -> Word1[15:8] ->…….

Word0[7] -> Word0[6] ….. Word0[0]

NOTE: Writing code from a scratch to execute the below commands may be time consuming. The user is strongly advised to use the API set, the source code of which is provided with the software package. Reference files are at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/API_Lib . Documentation for the API set is available at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/Documentation/

4.9.1 Band

Description

This command configures the band in which the module has to be configured.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

```
struct WiFi_CNFG_Band
{
UINT8 band_val;
} WiFi_CNFG_Band;
```

The valid values for the parameter for this command (band_val) are as follows:

0 – 2.4 GHz

1 – 5 GHz

Response

The response is found in W0[15:8] as explained in Frame Descriptor for Management/Data Frames in SPI mode. Expected value is 0x97

4.9.2 Init

Description

This command programs the RS9110-N-11-2X module's Baseband and RF components.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

None

Response

The second 16-bit word of the descriptor (W1[15:0]) of the received packet is used to identify the frame's success or failure. A value of '0' for the W1[15:0] is interpreted as a success and a non-zero value is interpreted as a failure. The reason for the init failure is also indicated by the non-zero value of this byte.

4.9.3 Scan

Description

This request makes the module scan in the channel specified. If the user wants to join a hidden Access Point (SSID not being broadcast), the SSID of the Access Point is passed as the parameter. Otherwise the SSID field is NULL ('\0').

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

The structure and description of the scan request structure is as follows.

```
struct WiFi_CNFG_Scan
{
UINT32 uChannelNo;
UINT8 uSSID[32];
UNIT8 pScan_bitmap[1][4];
};
```

Structure Member Name	Description
uChannelNo	Channel Number of the Access Point. This value can be one of many values, as listed.
uSSID	SSID of the Access Point. To be used to scan hidden SSIDs or a specific known SSID. For example, if the user wants to scan an SSID "Test_Network" in

[1] In versions earlier than 4.7.x, this field was not present. Users can use the earlier structure without this field and the original scanning process will not be affected.

Structure Member Name	Description
	channel 10, then uSSID=Test_Network and uChannelNo = 10
pScan_bitmap	Parameter to configure for which channels passive scan is to be done. For example, in 2.4 GHz, if only channel 1 and channel 4 are required to be scanned passively, then the value for *pScan_bitmap* is calculated as Channel[n] Channel[n-1] ……..Channel[4] Channel[3] Channel[2] Channel[1]= 000 …1001 Decimal for <000..001001> is 9. Hence, the value of the parameter in this case is 9 Similarly, in 5 GHz, if only channel 36 and 48 are required to be scanned passively, then the value for bit_map is calculated as 0000….1001. Decimal for <000..001001> is 9. This parameter controls which channels are scanned passively. For example, if pScan_bitmap=9 and the "Scan " command is issued with uChannelNo=0, then all channels except for channel number 1 and 4 will be scanned actively and these two channels will be scanned passively (a probe request will not be sent out). The value of the Bit[9] should be '1' in the command Feature Select for passive scanning to be executed.

uchannelNo parameter for 2.4 Ghz

Actual Channel Number	uChannelNo parameter
All Channels	0
1	1
2	2
3	3
4	4

5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

Table 16: Channel Number Parameter (2.4 Ghz)

uchannel parameter for 5 Ghz

Channel Number	chan_num parameter
All channels	0
36	1
40	2
44	3
48	4
52	5
56	6
60	7
64	8
100	9
104	10
108	11
112	12
116	13
120	14
124	15
128	16
132	17
136	18
140	19
149	20
153	21
157	22
161	23
165	24

Table 17: Channel Number Parameter (5 Ghz)

NOTE: In 5 GHz, DFS Client (802.11h) is supported. DFS client implementation is internal to the module. The module, by default, does only passive scan in DFS channels. If the Access Point detects radar signals, it indicates to the module (client) to switch to a different channel by using the "channel switch frame". The module performs channel switch as per the AP's channel switch parameters. There is no command required to enable this feature, it is enabled by default.

Response

The response frame for the scan response is as shown below. This frame contains the list of SSID's scanned in the network, in decreasing order of RSSI. The maximum number of Access Points that can be scanned by the module is 15 for RS9110-N-11-22(-xx) and RS9110-N-11-24(-xx) and 11 for RS9110-N-11-28(-xx).

```
struct WiFi_CNFG_ScanInfo
{
        UINT8 uChannelNumber;
        UINT8 uSecMode;
        UINT8 rssiVal;
        UINT8 aSSID[32];
};

struct WiFi_CNFG_ScanRsp
{
        UINT32 uScanCount;
        UINT32 Reserved;
        struct WiFi_CNFG_ScanInfo stScanInfo[32];
};
```

Structure Member Name	Description
uChannelNumber	Channel Number of the Access Point. This value can be one of many values, as listed.
uSecMode	Security Mode of the scanned Access Point. 0 – Open (No Security) 1 – WPA 2 – WPA2 3 – WEP
rssiVal	Absolute value of the RSSI information. For example, if the RSSI

Structure Member Name	Description
	is -20dBm, the value returned is 20. RSSI information indicates the signal strength of the Access Point.
aSSID	SSID of the Access Point
uScanCount	Number of scanned Access points

4.9.4 Set WEP Key

Description

This command is issued to supply WEP keys to the module, if the AP is in WEP mode. In some APs, there is an option to provide four WEP keys (with indices 0, 1, 2, 3). This command supplies the keys with index 1, 2 and 3. The key with index 0 is supplied in the "Join" command's parameter uPSK. This is an optional command and need not be issued, if the AP is configured with the WEP key of index 0.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

```
struct wlan_cnfg_WEP_keys_s
{
UINT8 wep_index;
UINT8 wep_psk[3][32];
} wlan_cnfg_WEP_keys_t;
```

Structure Member Name	Description
wep_psk[3][32]	WEP keys of index 1, 2 and 3.
wep_index	Key index to be used to select the key. Allowable values are 0, 1, 2, and 3. If this value is 0, the key supplied in the *uPSK* field of the "Join" command is used. If the value is 1, the first value in the array wep_psk is used; if the value is 2, the second value in array wep_psk is used; if the value is 3, the third value in array wep_psk is used.

Response

The response is found in W0[15:8] as explained in Frame Descriptor for Management/Data Frames in SPI mode. Expected value is 0xA4

4.9.5 Join

Description

This command is issued to associate the RS9110-N-11-2X module (in infrastructure or ad-hoc modes) to an Access Point. The SSID is selected from the list of SSID’s returned after performing scan operation. This command is also used to **create** a network as an ad-hoc(IBSS) creator.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

The structure and description of the join request structure is as follows.

```
struct WiFi_CNFG_Join
{
UINT8 uNwType;
UINT8 uSecType;
UINT8 uDataRate;
UINT8 uPowerValue;
UINT8 uPSK[32 or 64];
UINT8 uSSID[32];
UINT8 uIbss_mode;
UINT8 uchannel;
} WiFi_CNFG_Join;
```

Structure Member Name	Description
uNwType	Network type 0 – IBSS (ad-hoc, open mode)[1] 1 – Infrastructure 2 – IBSS (ad-hoc) with WEP

[1] In the case of IBSS (ad-hoc mode), 4 joinees to a network created by the module, is supported.

Structure Member Name	Description
	security
uDataRate	Transmission data rate. Rate at which the data has to be transmitted.
uPowerValue	This fixes the Transmit Power level of the module. This value can be set as follows: 0 – Low power (6-9 dBm) 1 – Medium power (10-14 dBm) 2 – High power (15-17dBm)
uSecType	Security type[1] 0 – OPEN 1 – WPA 2 – WPA2 3 – WEP
uPSK	Pass-phrase (Only in Security mode). It is ignored in open mode. When the security mode is WPA/WPA2-PSK, then there are two options to supply the key. 32 byte PSK mode – a key of a maximum length of 32 bytes can be supplied if Bit[3] in the Feature Select command is set to '0'. If the actual key is less than 32 bytes, the Host should pad the key with null character termination (\0) to make the parameter 32 bytes long 63 byte key mode[2] - a key of a maximum length of 63 bytes can be supplied if Bit[3] in the Feature Select command is set to '1'. In this mode, the Host should pad the key with null character termination (\0) to make the parameter 64 bytes long before sending the "Join" command to the WiFi module. The system stays in 32-byte PSK mode by default, in which case

[1] Please check the Release Notes of the individual modules' software/firmware releases to check whether this feature is supported.
[2] This mode is added in firmware version 4.5.0

Structure Member Name	Description
	the length of the array of this parameter should be 32. Please refer to the Driver API Library Manual document section “MCU Applications” to get the rsi_global.h file description that enables 63 byte mode.
uSSID	SSID of the access point to join or to create (in ad-hoc mode)
uIbss_mode	IBSS Mode 0 – Joiner 1 – Creator Unused in infrastructure mode.
uchannel	Channel number for IBSS Creator Mode. Should be ‘0’ in IBSS joiner mode. Unused in infrastructure mode.

uchannel parameter for 2.4 Ghz

Actual Channel Number	chan_num parameter
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13

uchannel parameter for 5 Ghz

Channel Number	chan_num parameter

36	1
40	2
44	3
48	4
52	5
56	6
60	7
64	8
100	9
104	10
108	11
112	12
116	13
120	14
124	15
128	16
132	17
136	18
140	19
149	20
153	21
157	22
161	23
165	24

Data Rate (Mbps)	**Value of uTxDataRate**
Auto-rate	0
1	1
2	2
5.5	3
11	4
6	5
9	6
12	7
18	8
24	9
36	10
48	11

Data Rate (Mbps)	Value of uTxDataRate
54	12
MCS0	13
MCS1	14
MCS2	15
MCS3	16
MCS4	17
MCS5	18
MCS6	19
MCS7	20

Table 18: Data Rates and Corresponding Values

Response

W1[15:0] contains the response, as explained in Frame Descriptor for Management/Data Frames in SPI mode , is used to identify the frame's success or failure. A value of '0' is interpreted as a success and a non-zero value is interpreted as a failure. The reason for the join failure is also indicated by the non-zero value of this byte.

4.9.6 Query MAC address and Network Type of Scanned Networks

Description

This command is issued to retrieve from the RS9110-N-11-2X module the Scan Response with BSSID and Network Type included (infrastructure or ad-hoc modes). This command can be given any time after the scan command.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

None

Response

The response frame for the command is as shown below. This frame contains the list of SSID's scanned in the network by the scan command, and their MAC ids and network types.

```
struct SPI_WiFi_CNFG_BssidRsp
```

```
{
  UINT32 uScanCount;
  UINT32 uErrorcode;
  struct SPI_WiFi_CNFG_BssidInfo stBssidInfo[32];
};

struct SPI_WiFi_CNFG_BssidInfo
{
  UINT8 uChannelNumber;
  UINT8 uSecMode;
  UINT8 rssiVal;
  UINT8 aSSID[32];
  UINT8 uNetworkType;
  UINT8 BSSID[6];
};
```

Structure member name	**Description**
uChannelNumber	Channel Number of the Access Point. This value can be one of many values, as listed.
uSecMode	Security Mode of the scanned Access Point. 0 – Open (No Security) 1 – WPA 2 – WPA2 3 – WEP
rssiVal	Absolute value of the RSSI information. RSSI information indicates the signal strength of the Access Point.
aSSID	SSID of the Station
uNetworkType	Whether the Station detected is in 0-IBSS Mode 1-Infrastructure Mode

Structure member name	Description
BSSID	The MAC address corresponding to the Station

uScanCount	Number of scanned Access points
uErrorcode	Error Code

4.9.7 Power Mode

Description

This command configures the power mode, viz., Power Mode 0, Power Mode 1 or Power Mode 2 to be used by the RS9110-N-11-2X module.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

The structure and description of the power mode request structure is as follows.

```
struct WiFi_CNFG_PwSave  {
UINT16 uMode;
} WiFi_CNFG_PwSave;
```

Structure Member Name	Description
uMode	0 – Power mode 0 1 – Power mode 1 2 – Power mode 2

Response

There is no response for the Power Mode command.

4.9.8 Sleep Timer

Description

This command configures the sleep timer which is used in Power Mode 1 when the module has not established the Wi-Fi connection.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

```
typedef struct WiFi_CNFG_SlpTimer_s
{
   UINT32 timerValue;
}WiFi_CNFG_SlpTimer;
```

Structure member name	**Description**
timerValue	Time in milliseconds for which the module sleeps before waking up. The maximum value is 10000 milliseconds.

Response

The response is found in W0[15:8] as explained in Frame Descriptor for Management/Data Frames in SPI mode. Expected value is 0x98

4.9.9 TCP/IP Bypass

Description

This command configures the RS22/24/28 module in TCP/IP bypassed mode. A TCP/IP stack in the Host can be used to interface with the RS22/24/28 module in this mode

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

```
struct ModeSelFrameSnd {
          uint8   ModeSelVal;
   } ModeSelFrameSnd;
```

ModeSelVal : Decimal value of 0 enables the TCP/IP stack inside the module. Decimal value of 1 bypasses the TCP/IP stack inside the module. The command is optional. If it is not issued, the TCP/IP stack inside the module is enabled by default. This command should be issued immediately before or after the band command.

Response

The response is found in W0[15:8] as explained in Frame Descriptor for Management/Data Frames in SPI mode. Expected value is 0xA3

4.9.10Feature Select

Description

This command configures different sub-features in the WiFi module.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

```
typedef struct WiFi_CNFG_Feat_Sel  {
UINT32 feat_sel_bitmap;
} WiFi_CNFG_Feat_Sel;
```

Structure Member Name	**Description**
Feat_Sel_Bitmap	Bit[0] – Set to '0' Bit[1] – Set to '0' Bit[2]– '1' for WEP shared mode, '0' for WEP Open mode (stays at Open mode by default) Bit[3] – Set to '0' if the actual PSK length of the AP is equal to or less than 32 bytes. The length of the array of the parameter uPSK in the command "Join" should be 32 if this bit is '0'. Set to '1' if the actual PSK length of the AP is more than 32 bytes but equal to or less than 63 bytes. The length of the array of the parameter uPSK in the command "Join" should be 64 if this bit is '1'. If this command is not issued, the module accepts PSK length equal to or less than 32 bytes. In such a case, the length of the array of the parameter uPSK in the command "Join" should be 32. Bit[4:6] – Set to '0' Bit[7]- When set to '1', the command "Get Information about Stored Configuration" will return three additional values- one is the

Structure Member Name	Description
	WEP Open or Shared authentication mode, WEP index and WEP keys Bit[8] – Set to '0' Bit[9]- Passive scan is enabled if this bit is set to '1'. If this command is not issued, or is issued with value of this bit as '0', passive scan will remain disabled by default. Bit[10]- LED in RS9110-N-11-22-04/05 modules is disabled if this bit is set to '1'. The LED will stop glowing after the *at+rsi_init* command is issued Bit[11]- When set to '0', module will not obey the security type given by host. When set to '1', module obeys the security type given by host. Under this situation, module will connect with the AP only when "Sec Type" of the module matches with that of the AP's

Response

The response is found in W0[15:8] as explained in Frame Descriptor for Management/Data Frames in SPI mode. Expected value is 0x9D

4.10 Data Frames (TCP/IP Commands) (Frame Write)

The TCP/IP Commands/Responses/Events represent the command frames that are sent/received from/by the Host. These commands are sent to the module through slave writes. There are some general commands also that fall under the category of data frames.

NOTE: Writing code from a scratch to execute the below commands may be time consuming. The user is strongly advised to use the API set, the source code of which is provided with the software package. Reference files are at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/API_Lib . Documentation for the API set is available at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/Documentation/

Unlike management frames the frame body of data frames comprises of a two-byte type field, which identifies the type of the command, followed by the corresponding command data structure.

The following are the types of the TCP/IP commands, responses and events and the corresponding codes.

Each data frame will start with the code as mentioned below.

Request	**Code**
TCP/IP Configure	0x01
Socket Create	0x02
Send	0x03
Socket Close	0x04
RSSI Query	0x05
Network Parameters	0x06
Connection Status Query	0x07
Disconnect	0x08
DHCP Info Query	0x09
HTTP Get Request	0x0A
HTTP Post Request	0x0B
Query Firmware Version	0x0D
Set MAC Address	0x0E
Query MAC Address	0x0F
DNS Request	0x11
Listen Interval Command	0x12

Table 19: Data Frame Based Codes (from Host to Module)

Response	**Code**
TCP/IP Configure	0x01
Socket Create	0x02
Connection Establishment	0x04
Remote Terminate	0x05
Close	0x06
Receive	0x07
RSSI Response Event	0x08
Network Parameter Response Event	0x09

Response	Code
Connection Status Response Event	0x0A
DHCP Info Response Event	0x0B
Disconnect Response Event	0x0C
HTTP Response Event	0x0D
Firmware Version Query Response Event	0x0F
Set MAC Address Response Event	0x10
Query MAC Address	0x12
DNS Response	0x14
Listen Interval Command Response	0x15

Table 20: Data Frame Based Response Codes (sent from module to host)

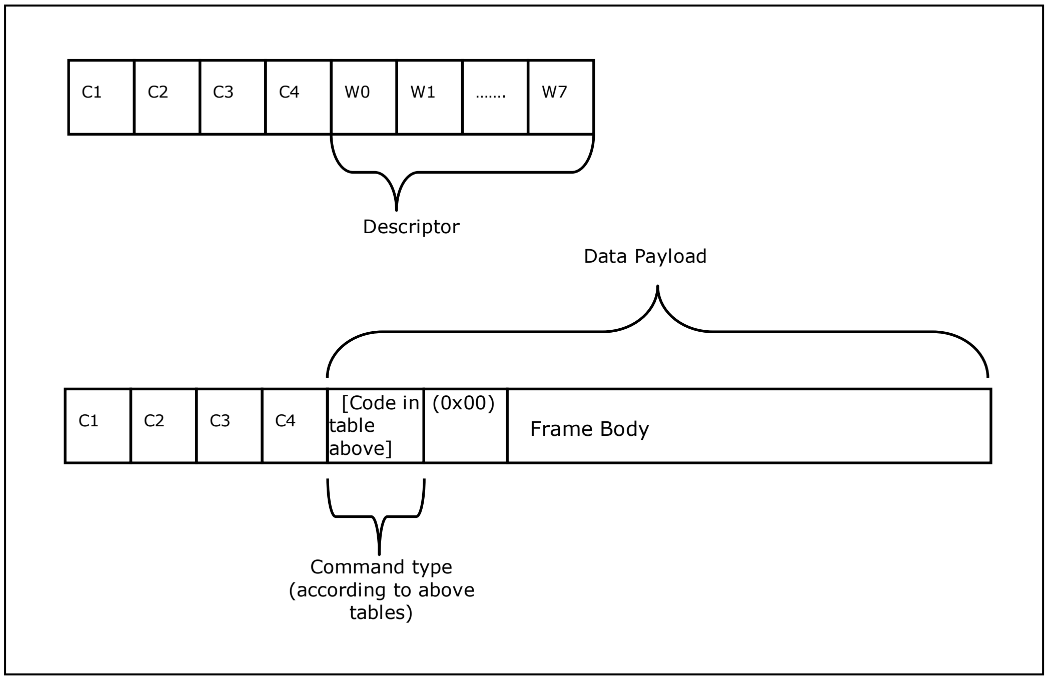


Figure 29: Data payload for data frames (from Host to module)

Note: C1, C2, C3, C4 shown in the diagram above are sent from Host to Module. For responses coming from Module to Host, these are not sent by the module. For information about how to read responses of Management Frames please refer to Process for Reading Data .

The response frames from the module contain error codes where applicable. These error codes and their descriptions are mentioned below. **These are applicable only for data frames (TCP/IP related commands)**.

Error Code	**Description**
Generic Error Codes – WLAN, Sockets, DHCP, ARP	
-1	This error is issued when "Query DHCP Information" is given when DHCP mode is disabled.
-2	Sockets not available. A maximum of 8 sockets can be operational at a time. If creation of more than 8 sockets is attempted, then this error is issued.
-4	IP configuration failed
-7	If WLAN disconnect command is issued when the module is not associated to a network.
-55	If rssi command is issued in un-associated state.
-91	IGMP Error
-95	ARP request failure
-99	DHCP lease time expired
-100	DHCP handshake failure
-121	Error issued when trying to connect to non-existent TCP server socket
-123	Invalid socket parameters
-127	Socket already open
-131	If parameters other than 0 or 1 is given for set IP parameters command.
-191	HTTP socket creation failed
-192	TCP socket close command is issued before getting the response of the previous close command.
TCP Error Codes	
-124	TCP socket open failure
DNS Error Codes	
-69	DNS_NOT_FOUND
-70	DNS_CLASS_ERROR

Error Code	Description
-72	DNS_COUNT_ERROR
-73	DNS_RCODE_ERROR
-74	DNS_OPCODE_ERROR
-75	DNS_ID_ERROR
-85	INVALID_VALUE
-190	DNS_RESPONSE_TIME_OUT

Table 21: Error Codes and Descriptions

NOTE: Writing code from a scratch to execute the below commands may be time consuming. The user is strongly advised to use the API set, the source code of which is provided with the software package. Reference files are at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/API_Lib . Documentation for the API set is available at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/Documentation/

4.10.1 Set IP Parameters

Description

This command configures the IP address, subnet mask and default gateway of the TCP/IP stack in the RS9110-N-11-2X module. The TCP/IP stack can be configured to DHCP mode or Manual mode. In either mode, the final IP parameters are returned as part of the response to the TCP Configure command.

Relevance

This command is relevant when the TCP/IP stack is enabled.

Frame Body

The following is the structure of the TCP/IP Configure command.

The second byte is

```
struct TCP_CNFG_Configure
{
        UINT8 uUseDHCP;
        UINT8 aIPaddr [4];
        UINT8 aNetMask [4];
UINT8 aDefaultGateway [4];
UNIT8 aDNSAddr[4];
};
```

Structure Member Name	Description
uUseDHCP	The mode with which the TCP/IP stack has to be configured. 0 – Manual 1 – DHCP (Not supported in TCP/IP bypass mode)
aIPaddr	IP address of the TCP/IP stack (valid only in Manual mode)
aNetMask	Subnet mask of the TCP/IP stack (valid only in Manual mode)
aDefaultGateway	Default gateway of the TCP/IP stack (valid only in Manual mode)
aDNSAddr	It is the DNS server’s IP address. Optional input. Should be used when DNS feature is used in DHCP disabled mode.

Response

The following is the data structure of the TCP/IP Configure command’s response.

```
struct TCP_EVT_Configure
{
        UINT8 aMacAddr [6];
        UINT8 aIPaddr [4];
        UINT8 aNetMask [4];
        UINT8 aDefaultGateway [4];
        UINT32 uErrorCode;
};
```

Structure Member Name	Description
aMacAddr	MAC address of RS9110-N-11-2X module
aIPaddr	IP address with which the TCP/IP stack is configured

Structure Member Name	**Description**
aMacAddr	MAC address of RS9110-N-11-2X module
aNetMask	Subnet mask with which the TCP/IP stack is configured.
aDefaultGateway	Default gateway with which the TCP/IP stack is configured.
uErrorCode	0 – Success -100: DHCP handshake failure -4 : IP configuration failed -131:If parameters other than 0 or 1 is given for set IP parameters command.

4.10.2Open a Socket

Description

This command opens a TCP/UDP client socket, a Listening TCP/UDP socket or a multicast socket. This command enables the TCP/IP stack to perform the corresponding action on opening a socket.

On opening a TCP client socket, the TCP/IP stack tries to connect to the specified TCP peer and gives the response to the user.

On opening a UDP (client/server) socket, the TCP/IP stack returns the socket ID created immediately since there is no handshake required for UDP.

On opening a TCP server socket, the TCP/IP stack returns the socket ID created immediately and waits for the connections on the specified ports from peers.

Relevance

This command is relevant only when the TCP/IP stack is enabled.

Frame Body

```
struct in_addr1
{
        UINT8 s_addr[4];
};
struct sockaddr_in
{
    UINT16  sin_port;
    struct          in_addr1 sin_addr;
};
struct TCP_CNFG_Socket
{
        UINT16 uSocketType;
     UINT16 stLocalPort;
        struct sockaddr_in stRemoteAddress;

};
```

Structure Member Name	**Description**
uSocketType	Type of the socket 0 – TCP Client 1 – UDP Client 2 – TCP Server (Listening TCP) 3 – Multicast[1] socket 4- Listening UDP. This type of socket is used to receive/send from any remote UDP socket with any remote IP and port number.
stLocalPort	Local port on which the socket has to be bound.
sin_port	The destination's port. This port number is not valid for a listening socket.
sin_addr	The destination's IP address. This IP address is not valid for a listening socket. The IP address is represented by 4 bytes. For example, if the IP address is 192.168.10.9, then 192 (0xC0) is sent first, then 168(0xA8) and so on. This is true for other commands also.

Response

The following is the data structure of the socket command's response.

[1] Multicast not supported in RS9110-N-11-28-04

```
struct TCP_EVT_SocketResp
{
  UINT16 uSocketType;
  UINT16 uSocketDescriptor;
  struct sockaddr_in stLocalAddress;
  UINT32  nErrorCode;
  UINT16  max_segment_size;
};
```

Structure Member Name	**Description**
uSocketDescriptor	Created socket's descriptor or handle . Starts from 1. The first socket opened will have a socket descriptor of 1. This should be used as an input for commands where socket descriptor is an input parameter. Example - **RSI_Send_Socket_Close**, **RSI_Socket_Send**
uSocketType	Type of the created socket. 0 – TCP Client 1 – UDP Client 2 – TCP Server (Listening TCP) 3- Multicast socket 4 – Listening UDP
stLocalAddress	The local IP address and the port number on which the socket is created.
uErrorCode	0 – Success -2: Socket not available. A maximum of 8 sockets can be operational at a time. If creation of more than 8 sockets is attempted, then this error is issued -95: ARP request failed -121 : Error issued when trying to connect to non-existent TCP server socket in remote terminal -123: Invalid socket parameters (if invalid parameters are given like, source port number 0, destination IP starts with 224

Structure Member Name	Description
	etc.) -124: TCP socket open failure -127: Socket already exists
max_segment_size	Indicates the maximum allowable TCP data packet size

NOTE: Redpine module has total of 7 sockets for UART modules and 6 sockets for SPI modules. User must use the socket handle returned from the socket creation commands.

Apart from these user provided sockets there are internal sockets reserved for HTTP client,DNS client, Wireless firmware upgrade/configuration, ARP and DHCP client.

4.10.3Close a Socket

Description

This command closes a TCP/UDP socket. For UDP you will get socket close response immediately from module, but for TCP it will take some time to complete handshake.

Relevance

This command is relevant only when the TCP/IP stack is enabled.

Frame Body

The following is the structure of the Close command.

```
struct TCP_CNFG_Close
{
        UINT16 uSocketDescriptor;
};
```

Structure Member Name	Description
uSocketDescriptor	Socket descriptor to be closed

Response

The following is the data structure of the close response.

```
struct TCP_EVT_Close
```

```
{
  UINT16 uSocketDescriptor;
  UINT32 nErrorCode;
};
```

Structure Member Name	**Description**
uSocketDescriptor	Socket descriptor of the socket to be closed.
uErrorCode	0 – Success -2 : Socket not available -91: IGMP Error -95: ARP request failed

4.10.4 Send Data to a Socket

Description

This command sends a byte stream of a certain size to the socket specified by the socket handle. The maximum size of the data that can be transmitted to the module is limited to 1460 bytes in case of TCP and 1472 in case of UDP.

The Host Driver is also expected to query the status of the buffers in the module for each transfer. The following flowchart gives a brief idea of the protocol to be followed by the driver while doing any transfers.

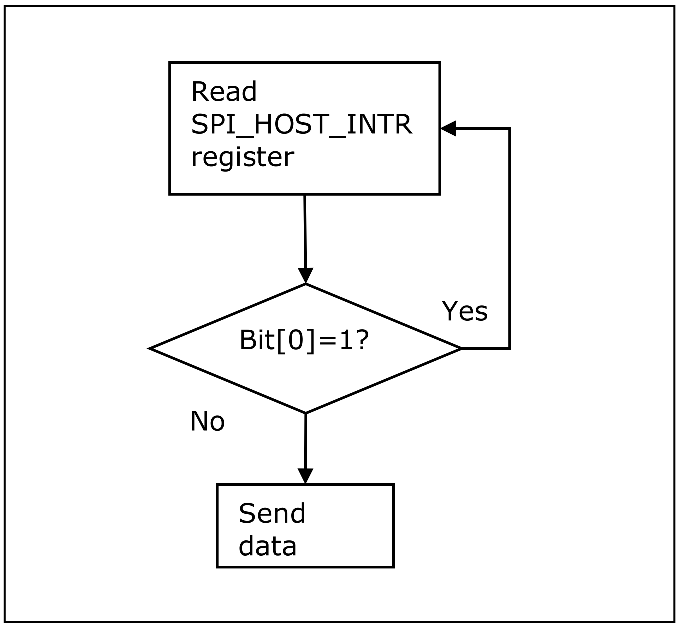


Figure 30:Flowchart for Transmitting Frames

Relevance

This command is relevant only when the TCP/IP stack is enabled.

Frame Body

The following is the structure of the Send command.

```
#define  MAX_PAYLOAD_SIZE 1400

 struct TCP_CNFG_Send
 {
        UINT16 uSocketDescriptor;
        UINT32 uBufferLength;
        UINT16 uDataOffset;
        UINT8 aBuffer[MAX_PAYLOAD_SIZE];

 };
```

Structure Member Name	Description
uSocketDescriptor	Descriptor of the created socket
uBufferLength	Length of the data to be sent
uDataOffset	Offset of the data in the buffer. To avoid redundant "memcpy". The format of the send frame body is shown in the figure below.
aBuffer	Buffer comprising the data payload. The payload starts from the offset as shown in the figure below.

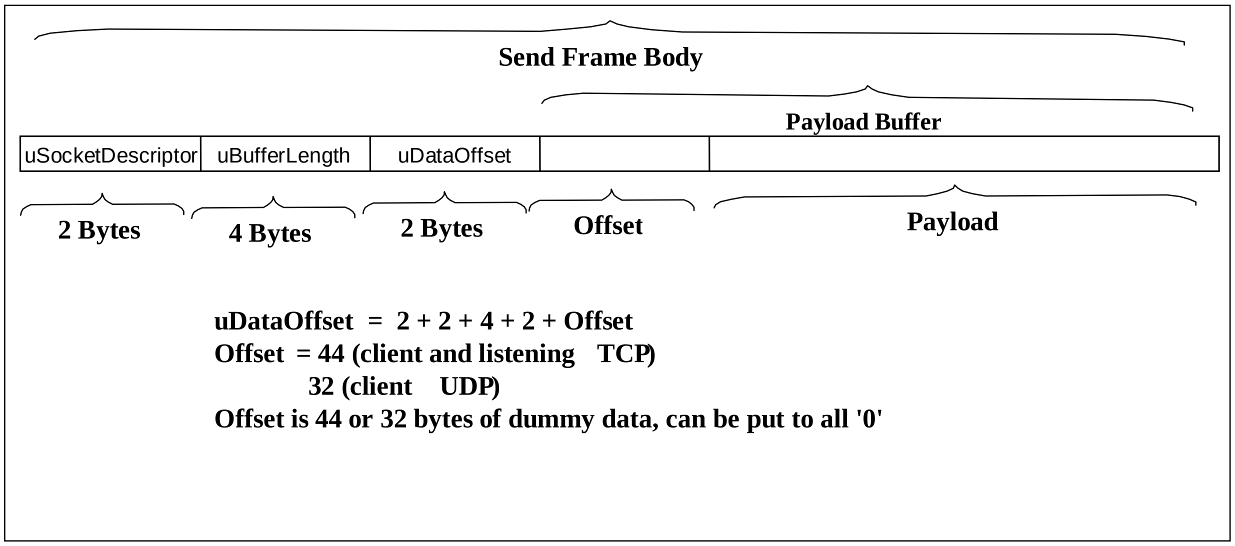

Send Frame Body
Payload Buffer
uSocketDescriptor
uBufferLength
uDataOffset
2 Bytes
4 Bytes
2 Bytes
Offset
Payload
uDataOffset = 2 + 2 + 4 + 2 + Offset
Offset = 44 (client and listening TCP)
32 (client UDP)
Offset is 44 or 32 bytes of dummy data, can be put to all '0'

Figure 31 : Send Frame Body (TCP and client UDP)

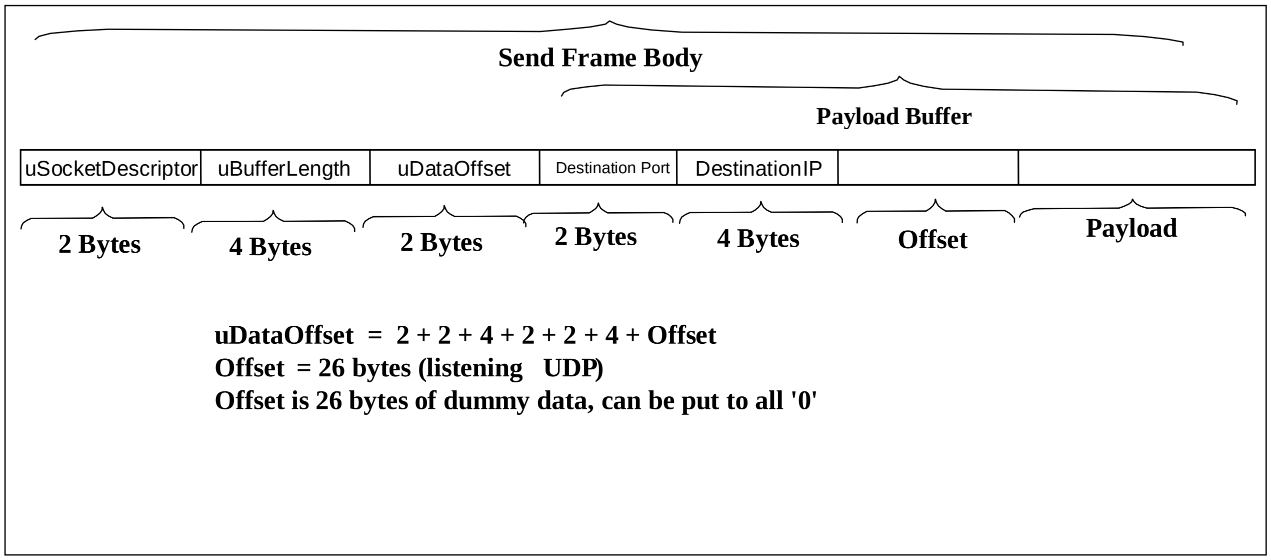


Figure 32 Send Frame Body (Listening UDP)

Note: The extra 2 in the equation for uDataOffset in the diagram above is the two bytes included in the payload. Refer to Data payload for data frames (from Host to module).

The Host driver should take care of buffering the data, thereby isolating the application from the packet size limitation with respect to the module/firmware.

For example, for 3000 bytes of the data the flow will be as follows:

There has to be a wrapper in the driver that buffers the data from the application layer. The maximum size of the buffer that the modules can handle is 1400 bytes. Thus the wrapper takes care of sending multiple frames to the module.

For the 3000 bytes of the data the wrapper has to send 3 frames to the module. The uDataoffset (refer to the figure above) shall depend on the socket type.

Frame1: uBufferLength = 1400 bytes

Frame2: uBufferLength = 1400 bytes

Frame3: uBufferLength = 200 bytes

4.10.5 Receive Data on a Socket

Description

The data received from the network is sent to the Host. This is not a command sent by the Host to module, but the response to a slave read done by the Host to fetch data received by the module. The Host should identify the type of the frame received and interact with the corresponding socket application on the reception of the data.

Relevance

This command is relevant when the TCP/IP stack is enabled.

Frame Body

```
struct TCP_EVT_Recv
{
        UINT16 uSocketDescriptor;
        UINT32 uBufferLength;
        UINT16 uDataOffset;
        struct sockaddr_in stFrom_Address;
        UINT8 *pBuffer;
};
```

Structure Member Name	**Description**
uSocketDescriptor	Socket descriptor created
UBufferLength	Length of the data received
UDataOffset	Offset of the data in the buffer. To avoid redundant “memcpy”. Similar to the send frame, the buffer before the data offset would be used for the UDP/TCP or IP processing. Refer to the figure below.
stFrom_Address	From address of the received data
PBuffer	Pointer to the buffer comprising the

Structure Member Name	Description
	data payload.

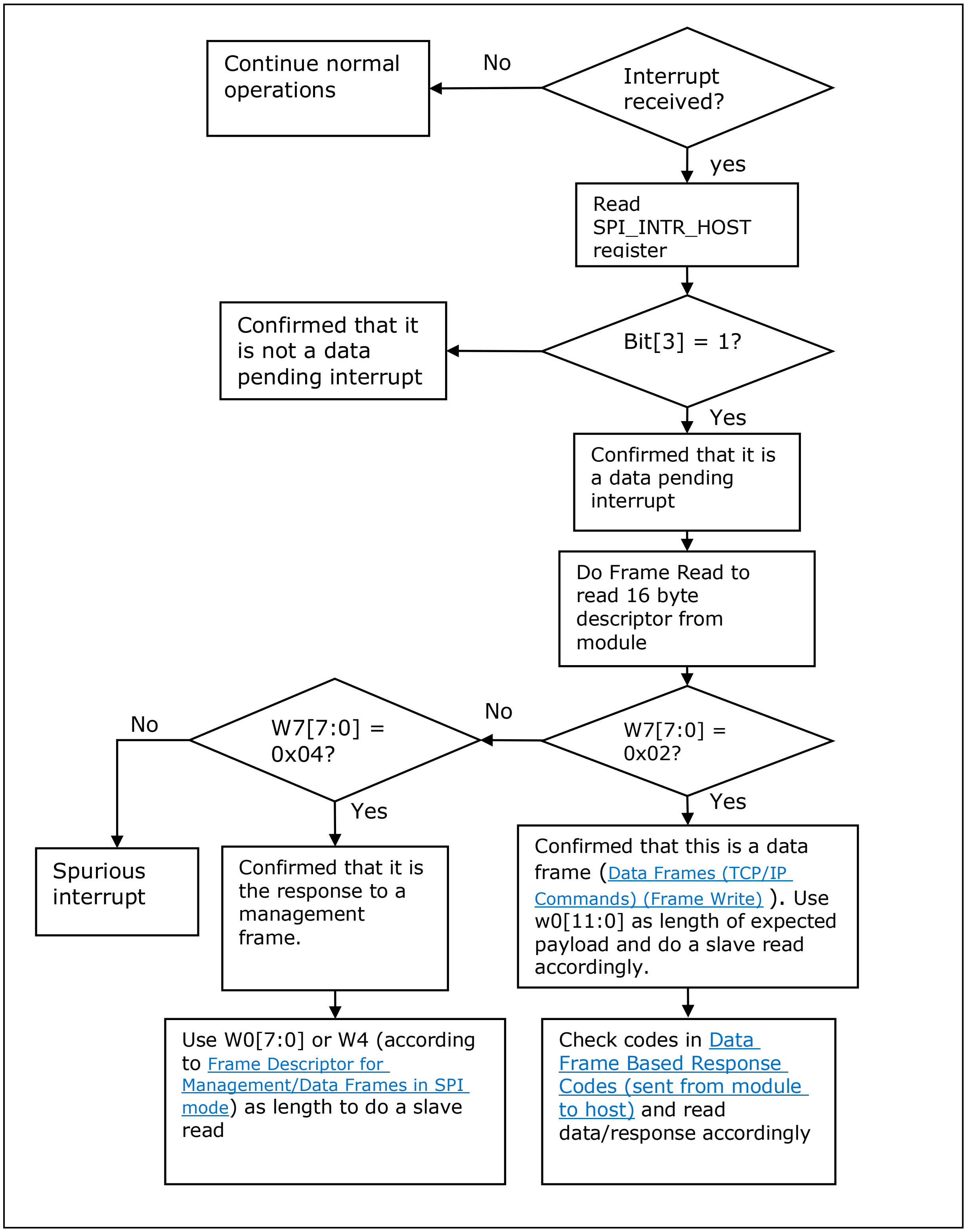

Continue normal operations
No
Interrupt received?
yes
Read SPI_INTR_HOST register
Confirmed that it is not a data pending interrupt
Bit[3] = 1?
Yes
Confirmed that it is a data pending interrupt
Do Frame Read to read 16 byte descriptor from module
No
W7[7:0] = 0x04?
No
W7[7:0] = 0x02?
Yes
Yes
Spurious interrupt
Confirmed that it is the response to a management frame.
Confirmed that this is a data frame (Data Frames (TCP/IP Commands) (Frame Write)). Use w0[11:0] as length of expected payload and do a slave read accordingly.
Use W0[7:0] or W4 (according to Frame Descriptor for Management/Data Frames in SPI mode) as length to do a slave read
Check codes in Data Frame Based Response Codes (sent from module to host) and read data/response accordingly

Figure 33: Process for Reading Data

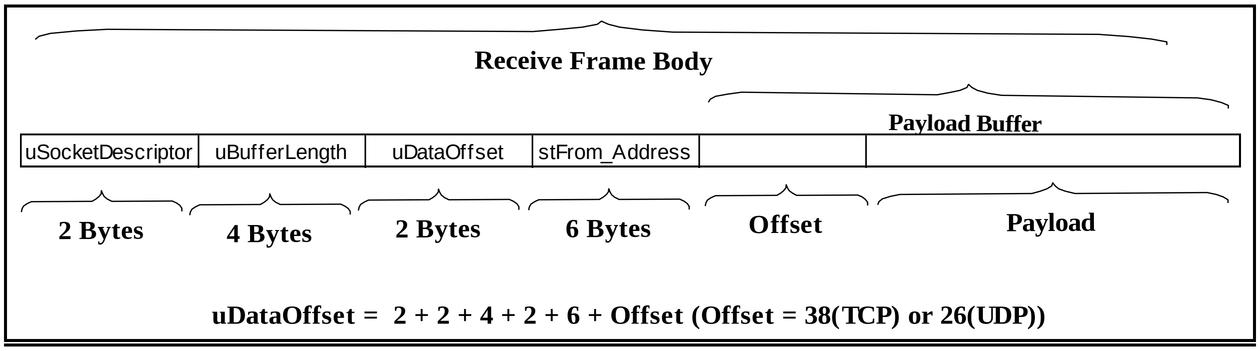


Figure 34:Receive Frame Body

Note: The extra 2 in the equation for uDataOffset in the diagram above is the two bytes included in the payload. Refer to Data payload for data frames (from Host to module).

4.10.6HTTP GET Request[1]

Description

This command is used to transmit an HTTP GET request to an HTTP server. The HTTP GET Request can be issued only in a serial manner meaning only if HTTP response is obtained for the current request, the next Request can be issued.

Relevance

This command is relevant when the TCP/IP stack is enabled.

Frame Body

The following is the structure for the HTTP GET Request:

```
struct TCP_HTTP_Req
{
UINT16 ipaddr_len;
UINT16 url_len;
UINT16 header_len;
UINT16 data_len;
UINT8  buffer[1200];
}__attribute__((packed));
```

Structure Member Name	Description

[1] Please check the Release Notes of the individual modules' software/firmware releases to check whether this feature is supported.

Structure Member Name	**Description**
ipaddr_len	The length of the IP Address. This is required since the IP address is sent as a string in this frame. For example, if the IP address of www.website.com is 192.168.30.6, the ipaddr_len = 12
url_len	The length of the URL. For example, if www.website.com/index.html is the webpage, then url_len = 11 (for "/index.html", www.website.com is not included in the url).
header_len	The length of the header of the HTTP GET request.
data_len	This value has to be assigned 0.
buffer	Buffer contains actual values in the order of IP Address, URL, Header and Data.

Response

The structure for the response to the HTTP GET Request is as follows:

```
struct TCP_EVT_HTTP_Data_t
{
UINT32 more;
UINT32 offset;
INT32 nErrorCode;
}__attribute__((packed));
```

Structure Member Name	**Description**
more	This indicates whether more HTTP data for the HTTP GET request is pending. 1 – More data pending 0 – End of HTTP data
offset	This indicates the byte-offset from which the valid HTTP data is present in the response.
nErrorCode	Error code. 0 :Success

Structure Member Name	Description
	-191 : HTTP socket creation failed

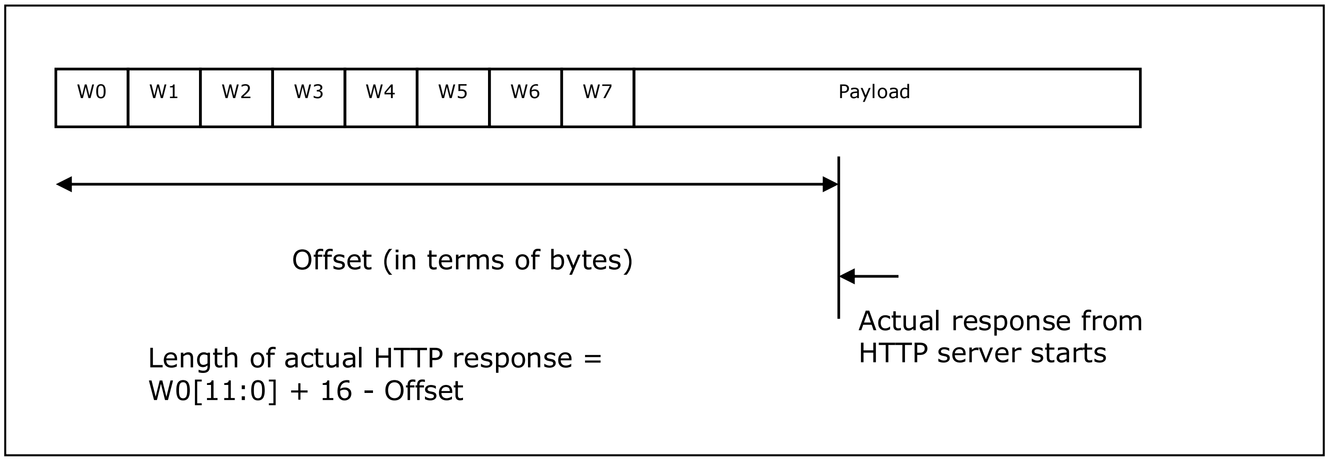

W0
W1
W2
W3
W4
W5
W6
W7
Payload
Offset (in terms of bytes)
Actual response from HTTP server starts
Length of actual HTTP response = W0[11:0] + 16 - Offset

Figure 35: Offset Parameter in HTTP Get Request

4.10.7 HTTP POST Request[1]

Description

This command is used to transmit an HTTP POST request to an HTTP server. The HTTP POST Request can be issued only in a serial manner meaning only if HTTP response is obtained for the current request, the next Request can be issued

Relevance

This command is relevant when the TCP/IP stack is enabled.

Frame Body

The following is the structure for the HTTP POST Request:

```
struct TCP_HTTP_Req
{
UINT16 ipaddr_len;
UINT16 url_len;
```

[1] Please check the Release Notes of the individual modules' software/firmware releases to check whether this feature is supported.

```
UINT16 header_len;
UINT16 data_len;
UINT8  buffer[1200];
}__attribute__((packed));
```

Structure Member Name	Description
ipaddr_len	The length of the IP Address. This is required since the IP address is sent as a string in this frame. For example, if the IP address of www.website.com is 192.168.30.6, the ipaddr_len = 12
url_len	The length of the URL. For example, if www.website.com/welcome.php is the webpage, then url_len = 12 (for "/welcome.php", www.website.com is not included in the url).
header_len	The length of the header of the HTTP GET request.
data_len	This is the length of the data string.
buffer	Buffer contains actual values in the order of IP Address, URL, Header and Data.

Response

The structure for the HTTP POST Request's response is as follows:

```
struct TCP_EVT_HTTP_Data_t
{
UINT32 more;
UINT32 offset;
INT32  nErrorCode;
}__attribute__((packed));
```

Structure Member Name	Description
more	This indicates whether more HTTP data for the HTTP POST request is pending. 1 – More data pending

Structure Member Name	Description
	0 – End of HTTP data
offset	This indicates the byte-offset from which the valid HTTP data is present in the response. This offset is with respect to the first byte of the received frame.
nErrorCode	Error code. 0 :Success -191 : HTTP socket creation failed

NOTE: The actual HTTP server response starts from the value in the "offset"

4.10.8DNS Request[1]

Description

This command is used to send the domain name to the module to resolve the IP address. The DNS Request can be issued only in a serial manner meaning only if DNS response is obtained for the current request, the next Request can be issued

Relevance

This command is relevant when the TCP/IP stack is enabled.

Frame Body

The following is the structure for the DNS Request frame body:

```
#define MAX_NAME_LEN     150

typedef struct TCP_CNFG_DNS_Req
{

UINT8 aDomainName [MAX_NAME_LEN];

}__attribute__ ((packed)) TCP_CNFG_DNS_Req;
```

[1] Please check the Release Notes of the individual modules' software/firmware releases to check whether this feature is supported.

Structure Member Name	Description
aDomainName	Domain name, example: www.website.com . A maximum of 150 characters are allowed.

Response

The structure for the response of DNS Request Command is as follows:

```
#define MAX_DNS_REPLY 10

typedef struct TCP_EVT_DNS_Resp
{
 UINT16 uIPCount;
 UINT8 aIPaddr[MAX_DNS_REPLY][4];
 UINT32 uErrorCode;

}__attribute__((packed))TCP_EVT_DNS_Resp;
```

Structure Member Name	Description
uIPCount	This indicates number of IPs resolved for the given domain name
aIPaddr	This returns the resolved IP addresses. A maximum of 10 IP addresses can be returned. User should read the number of IP addresses indicated by uIPCount.
nErrorCode	Error code. 0 :Success The failure codes for DNS are explained in Table 21: Error Codes and Descriptions

4.10.9 Connection Establishment with LTCP Socket

Description

This packet indicates to the Host, through an interrupt, that a remote client has established a connection on an already open LTCP socket in the module. The Host can read the details of the remote client in the packet contents. This is not a command sent from Host to module. It is an asynchronous message to be read by the module (after an interrupt is sent from the module) when a connection is established with a listening TCP socket.

Relevance

This command is relevant when the TCP/IP stack is enabled.

Frame Body

The following is the data structure for the Connection Establishment frame returned to the Host.

```
struct TCP_EVT_Connect_t
{
UINT16  uSocketDescriptor;
struct sockaddr_in rtAddress;
UINT32 nErrorCode;
}
```

Structure Member Name	**Description**
uSocketDescriptor	Socket descriptor for which the connection has happened.
rtAddress	Port number and IP address of the remote client.
nErrorCode	Always returns '0'

Refer to response code 0x04 in Data Frame Based Response Codes (sent from module to host).

4.10.10 Remote Termination

Description

The TCP peer can remotely terminate the connection asynchronously. The Remote termination frame indicates termination of the connection (preceded by an interrupt, at which the Host should read the frame from the module). This message is sent from module to Host in the following cases:

1.The remote TCP peer closes the socket connection

2.The module closes the socket connection. This could happen if the module tries to send data to a remote peer but fails to do so (because of abrupt closure of remote peer etc.). The module tries to resend the data, but after a number of retries, sends this message to the host and closes the socket.

3.When the module tries to open a client TCP socket, and during the handshake process with the remote peer, a "TCP reset packet" is sent by the remote peer.

Host should act accordingly upon the reception of the remote termination (i.e., close and re-open the socket. This is left to the developer's discretion).

Relevance

This command is relevant when the TCP/IP stack is enabled.

Frame Body

The following is the data structure for the Remote termination frame.

```
    struct TCP_EVT_RemoteTerminate
    {
            UINT16 uSocketDescriptor;
            UNIT32 nErrorCode;
  };
```

Structure Member Name	Description
uSocketDescriptor	Socket descriptor for which the Remote termination has happened.
nErrorCode	0 – Default return value, -121: Socket creation failed.

The identifier for this frame (0x05) can be found in Data Frame Based Response Codes (sent from module to host).

4.11 Send and Receive Data in TCP/IP Bypass mode

In TCP/IP bypass mode, the Host sends Ethernet frames from its own TCP/IP stack to the module for transmission, or receives Ethernet frames from the module as received data.

4.11.1 Send data

Description

This command sends data from the host to the module, to be transmitted over a wireless media.

The description for the frame descriptor for sending data from Host to module is explained below:

Word	**Data Frame Descriptor**
Word0 W0[15:0]	Bits [11:0] – Number of bytes to be sent Bits [15:12] - Not used, should be set to all ‘0’
Word1 W1[15:0]	0x0000
Word2 W2[15:0]	0x0000
Word3 W3[15:0]	0x0000
Word4 W4[15:0]	0x0000
Word5 W5 [15:0]	0x0000
Word6 W6 [15:0]	0x0000
Word7 W7 [15:0]	0x0002

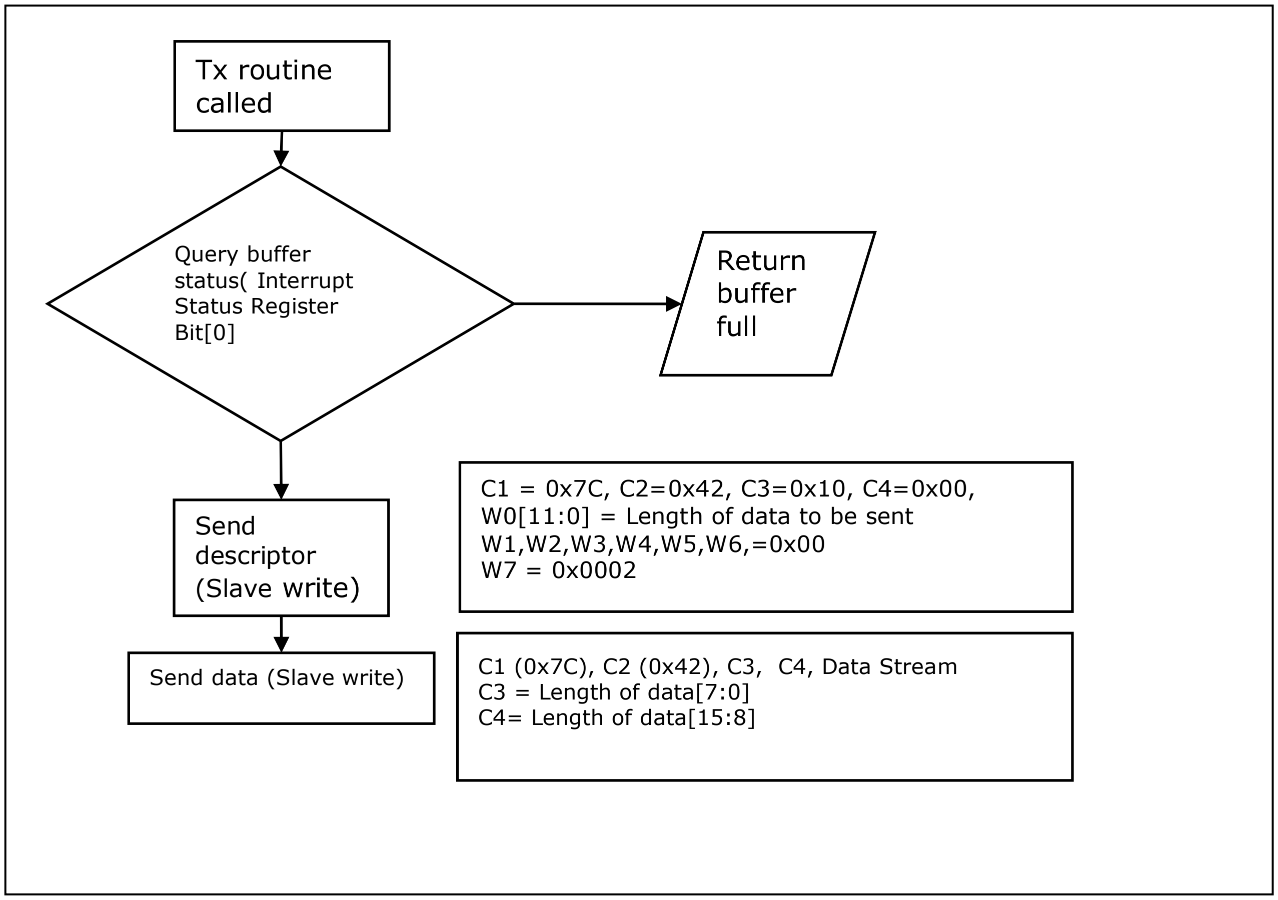

Tx routine called
Query buffer status(Interrupt Status Register Bit[0]
Return buffer full
Send descriptor (Slave write)
Send data (Slave write)
C1 = 0x7C, C2=0x42, C3=0x10, C4=0x00,
W0[11:0] = Length of data to be sent
W1,W2,W3,W4,W5,W6,=0x00
W7 = 0x0002
C1 (0x7C), C2 (0x42), C3, C4, Data Stream
C3 = Length of data[7:0]
C4= Length of data[15:8]

Figure 36: Transmitting Data

4.11.2 Receive data

Description

This process is used to receive data, by the Host, from the module. When data is to be sent by module to the Host, it raises an interrupt to the Host. The actions at the Host side are shown below.

The description for the frame descriptor while receiving data from module to Host is explained below:

Word	Data Frame Descriptor
Word0 W0[15:0]	Bits [11:0] – Number of bytes to be sent Bits [15:12] - Ignore this bits

Word	Data Frame Descriptor
Word1 W1[15:0]	0x0000
Word2 W2[15:0]	0x0000
Word3 W3[15:0]	0x0000
Word4 W4[15:0]	0x0000
Word5 W5 [15:0]	0x0000
Word6 W6 [15:0]	0x0000
Word7 W7 [15:0]	0x0002

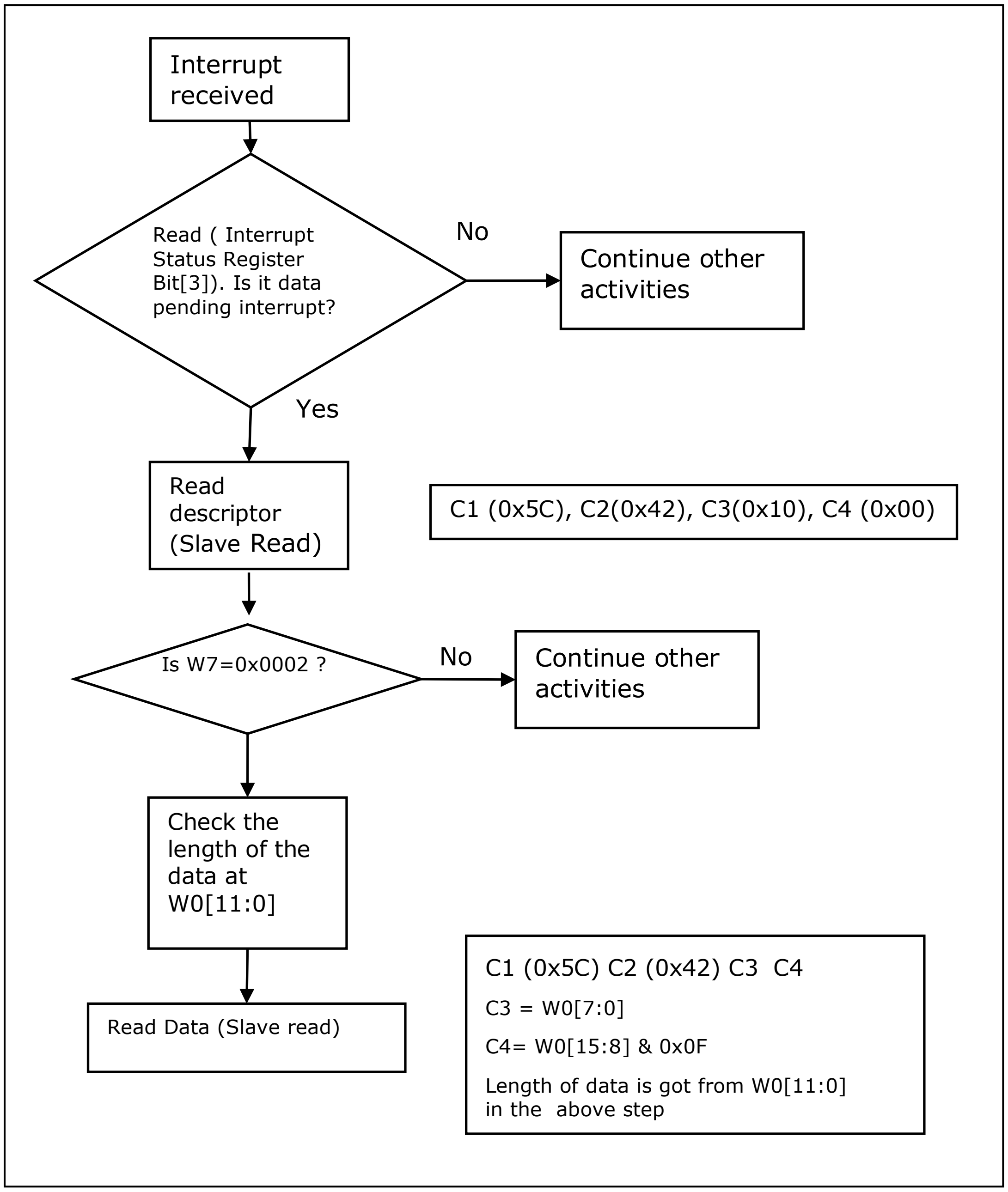

Interrupt received
Read (Interrupt Status Register Bit[3]). Is it data pending interrupt?
No
Continue other activities
Yes
Read descriptor (Slave Read)
C1 (0x5C), C2(0x42), C3(0x10), C4 (0x00)
Is W7=0x0002 ?
No
Continue other activities
Check the length of the data at W0[11:0]
Read Data (Slave read)
C1 (0x5C) C2 (0x42) C3 C4
C3 = W0[7:0]
C4= W0[15:8] & 0x0F
Length of data is got from W0[11:0] in the above step

Figure 37: Receiving Data

4.12 Other Commands (Data Frames)

These commands are in addition to the Wi-Fi and TCP/IP commands. Please note that all the commands listed in this section are based on data frames and not management frames. The data frame is recreated below for reference. Refer Section Command frames to understand the two kinds of frames available, management and data frames.

4.12.1Query RSSI Value[1]

Description

This command is used to retrieve the RSSI value for Access Point to which the module is connected.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

None for the command going from Host to module

Response

The structure for the RSSI query frame's response (returned in the frame body of the response packet) is as follows:

```
struct TCP_EVT_RSSI_t
{
        UINT16 rssi_val;
        UINT32 nErrorCode;
};
```

Structure Member Name	Description
rssi_val	RSSI value for the Access Point to which the module is connected, in dB. This value is the absolute value – for example, -20dbM would be reported as 20.
nErrorCode	'0' is returned for Success, 0xc8 – In IBSS mode, 0xc9 – When unassociated.

4.12.2Query Network Parameters[2]

Description

[1] This command is not supported in IBSS mode.
[2] Please check the Release Notes of the individual modules' software/firmware releases to check whether this feature is supported.

This command is used to query the network parameters, including the WLAN connection and IP configuration details. This command is to be issued only after the WLAN connection is established.

Relevance

This command is relevant when the TCP/IP stack is enabled.

Frame Body

None for the command going from Host to module

Response

The structure for the Network Parameters query frame's response is as follows:

```
struct TCP_EVT_NET_PARAMS
{
UINT16 wlan_state;
UINT8  ssid[32];
UINT8  ipaddr[4];
UINT8  subnet_mask[4];
UINT8  gateway[4];
UINT16 dhcp_mode;
UINT16 conn_type;
UINT32 nErrorCode;
};
```

Structure Member Name	**Description**
wlan_state	This indicates whether the module is connected to an Access Point or not. 0 – Not connected 1 – Connected
Ssid	This value is the SSID of the Access Point to which the module is connected.
ipaddr	This is the IP Address of the module. If the TCP/IP Configure frame has not been sent to the module before Network Parameters query frame is sent, the module returns a default value which should be ignored.
subnet_mask	This is the Subnet Mask of the module. If the TCP/IP Configure frame has not been sent to the module before Network Parameters query frame is sent, the module returns a default value which should be ignored.
gateway	This is the Gateway Address of the module. If the TCP/IP Configure frame has not been sent to the module before Network Parameters query frame is sent, the module returns a default value which should be ignored.
dhcp_type	This value indicates whether the module is configured for DHCP or Manual IP configuration. 0 – Manual IP configuration 1 – DHCP
conn_type	This value indicates whether the module is operational in Infrastructure mode or AdHoc mode. 0 – AdHoc mode 1 – Infrastructure mode
nErrorCode	Always ‘0’ is returned

4.12.3Set a Listen Interval

Description

This feature is used to intimate to the access point the number of beacon intervals for which the AP must buffer the packets destined to our module when in power save mode. This is an optional command and can be issued before Join Command. When the "join" command is given after this command, then "Association Request" frame sent from module to access point will have information about this interval. If this command is not issued (default behavior), the module will wake up at every DTIM interval while in power save mode (refer Activities for Power Save Mode 1 and Activities for Power Save Mode 2). If this command is issued, then the module will wake up according to the Listen Interval that is set.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

```
typedef struct WLAN_CNFG_LISTEN_Interval
{
  UINT16 interval;
} __attribute__((packed))WLAN_CNFG_LISTEN_Interval_t;
```

Structure Member Name	**Description**
Interval	This interval is used to tell the AP the number of beacon intervals it has to buffer the packets when the module is in power save. For example, if this parameter is 20, the module will wake up after every 20 BEACONS.

Response

The structure for the Listen Interval's frame response is as follows:

```
typedef struct WLAN_EVT_SET_LISTEN_INTERVAL
{
        UINT32 uErrorcode;
} WLAN_EVT_SET_Listen_Interval_t;
```

Structure Member Name	**Description**
uErrorCode	Always returns '0'

This feature/command is usually used when the final application requires that the module wakes up less frequently than the DTIM interval, and is not a mandatory requirement for module operation. It can be skipped when such requirements are not there for the target application. Also, this feature can be used only when the target application does not need to receive any broadcast packets.

4.12.4Query Connection Status

Description

This command is used to query status of the existing WLAN connection.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

None for the command going from Host to module

Response

The structure for the Connection Status query frame's response is as follows:

```
struct TCP_EVT_Conection_status
{
UINT16 state;
UINT32 nErrorCode;
};
```

Structure Member Name	Description
state	This indicates whether the module is connected to an Access Point or not. 0 – Not connected 1 – Connected
nErrorCode	Always '0' is returned

4.12.5 Disassociate[1]

Description

This command is used to disconnect from the Access Point to which the module is presently connected.

Relevance

This command is relevant both when the TCP/IP stack is enabled and bypassed.

Frame Body

None for the command going from Host to module

[1] This command is not supported in ad-hoc mode.

Response

The structure for the Disconnect frame's response is as follows:

```
struct TCP_EVT_Disconnect_t
{
UINT32 nErrorCode;
};
```

Structure Member Name	Description
nErrorCode	Always '0' is returned

4.12.6 Query DHCP Information

Description

This command is used to query the DHCP information. This command is valid only if the module is configured to acquire its IP address through DHCP from a DHCP server.

Relevance

This command is relevant when the TCP/IP stack is enabled

Frame Body

None for the command going from Host to module

Response

The structure for the DHCP Information query's response is as follows:

```
struct TCP_EVT_DHCP_Info
{
UINT32 lease_time;
UINT32 lease_time_left;
UINT32 renew_time;
UINT32 rebind_time;
UINT8  server_ip_address[4];
UINT32 nErrorCode;
};
```

Structure Member Name	Description
lease_time	The total lease time for the DHCP connection.

Structure Member Name	Description
lease_time_left	The lease time left for the DHCP connection.
renew_time	The time left for renewal of the IP address acquired through DHCP.
rebind_time	The time left for rebind of the IP address acquired through DHCP.
server_ip_address	The IP address of the DHCP server.
nErrorCode	0: Success Possible error code: -1

4.12.7 Query Firmware Version

Description

This command is used to query the version of the firmware loaded in the module.

Relevance

This command is relevant both when the TCP/IP stack is enabled and bypassed.

Frame Body

None for the command going from Host to module

Response

The structure of the response for this request frame is as follows:

```
struct TCP_EVT_FM_Version_t
{
UINT8 Fm_version[20];
};
```

Structure Member Name	Description
Fm_version	Version of the firmware loaded in the module. This is given in string format. The firmware version format is x.y.z (e.g., 4.7.1).

4.12.8 Set MAC Address

Description

This command is used to set the MAC Address of the module – overriding the MAC address stored in the module's non-volatile memory. This command should be issued to the module at every power-up if the user wants to override the MAC address stored in the memory. Host should ensure that this command is issued before the **mode select** command.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

The following is the structure for the Set MAC Address Request:

```
struct WLAN_CNFG_MAC_Addr {
UINT8 aMacAddr[6];
};
```

Structure Member Name	**Description**
aMacAddr	6-byte MAC Address that the module needs to use for all network communication.

Response

The structure of the response for this request frame is as follows:

```
struct WLAN_EVT_MAC_Addr {
UINT32 uErrorcode;
};
```

Structure Member Name	**Description**
uErrorcode	0: Success. 1: This error code is returned if the command is given after WLAN configuration (after "Join" command).

4.12.9 Query MAC Address

Description

This command is used to query the MAC address of the module. The Host may query the MAC address of the module at any time after the band and init commands.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

Indicates the type of Control Data Frame, should be set to 0x0F.

Response

The first Word (first two bytes) of the Frame body are used to indicate the type of Control Data Frame Response.

First Word = 0x0012. Refer

For response to Query MAC address , the MAC address is in Frame body after this first 2 bytes in a structure described below:

```
typedef struct WLAN_EVT_GET_MAC_Addr
{
        UINT8  aMacAddr[6];
        UINT32 reserved;
} WLAN_EVT_GET_MAC_Addr_t;
```

The “reserved field” always returns ‘0’.

4.13 Connecting to a Preconfigured Access Point

The RS9110-N-11-2X module can connect to a pre-configured access point after the module boots up. This feature facilitates fast connection to a known network. The corresponding commands are described below.

4.13.1Store Configuration in Flash Memory

Description

This command is used to save the parameters of an access point to internal flash memory. This command should be given after successfully executing the JOIN and IP config commands. This will enable the module to join immediately to this access point the next time the module powers up (within 200 ms of power up), provided the automatic join feature is enabled by the command store config enable.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

This command does not have the frame body

Parameters

None

Response

```
struct WiFi_CNFG_SaveConfigRsp
{
  UINT16 uErrorCode;
};
```

Result code (in Descriptor)	Description
UErrorCode	0- Success 1- Flash access failure 2- Not connected

4.13.2 Enable automatic joining to pre-configured Access Point

Description

This command is used to enable or disable the feature of automatic joining to a pre-configured Access Point on power up.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

```
struct WiFi_CNFG_Enable_Config
{
  UINT8 enable;
};
```

The valid values for the parameter for this command (enable) are as follows:

0 – Disable

1 – Enable

Response

```
struct WiFi_CNFG_EnableConfigRsp
{
  UINT16 uErrorCode;
};
```

Result code (in Descriptor)	Description
UErrorCode	0- Success 1- Flash access failure 2- Not connected

4.13.3 Get Information about Stored Configuration

Description

This command is used to get the configuration values that have been stored in the module's memory. The parameters returned in this command are the parameters used in automatically joining to a pre-configured network when the module boots up in the auto-join mode.

Relevance

This command is relevant both when the TCP/IP stack is enabled or bypassed.

Frame Body

This command does not have the frame body

Response

```
typedef struct flash_config_s
{
  UINT16 valid_flag;
  UINT8  cnum;
  UINT8  NetworkType;
  UINT8 secMode;
  UINT8 uRate;
  UINT8 uTxPower;
  UINT8 psk[64];
  UINT8 join_ssid[32];
  UINT8 uibss_mode;
  UINT8 uchannel;
  UINT8 uAction;
  UINT8 dhcp_enable;
  #define IP_ADDRESS_SZ 4
  UINT8  ip[IP_ADDRESS_SZ];
  UINT8  sn_mask[IP_ADDRESS_SZ];
  UINT8  dgw[IP_ADDRESS_SZ];
  UINT32 feature_select_bitmap;
  UINT8  wep_mode;
  UINT8 wep_index;
  UINT8 wep_keys[3][32];
```

}flash_config_t;

Structure of Response

Structure Member Name	**Description**
Valid_Flag	0 - Invalid configuration 1 - Valid configuration
CNum	It returns the channel number of the stored access point or network, according to Channel Number Parameter (2.4 Ghz) and Channel Number Parameter (5 Ghz).
NetworkType	1 - Infrastructure 0 - Ad-hoc
SecMode	0 – Open (No Security) 1 – WPA 2 – WPA2 3 - WEP
uRate	Data Rate of the module
uTxPower	0 – Low power (7dBm) 1 – Medium power (10dBm) 2 – High power (16 to 17dBm)
PSK	64 bytes, ASCII. The last byte of the parameter should always be ignored. Personal Shared Key. If the actual length is not 64 bytes, then filler bytes (0x00) are appended to the PSK to make it 64 bytes. For example, if the key is '12345678', the return value is 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x00 0x00 …..0x00 (to make 64 bytes)
join_ssid	SSID of the network to which the module should get connected. Returned as ASCII string. If the actual length is not 32 bytes, then filler bytes (0x00) is appended to the SSID to make it 32 bytes. For example, if the SSID is 'abcd', the return value is 0x61 0x62 0x63 0x64 0x00 0x00 …..0x00 (to make 32 bytes)
Dhcp_enable	0 - DHCP Disable 1 - DHCP Enable
Ip	IP Address of the module
sn_mask	Subnet mask
Dgw	Default gateway
feature_select_bitmap	Value of feature select bitmap.

Structure Member Name	Description
wep_mode[1]	'1'- WEP Shared suthentication '0'- WEP Open authentication
wep_index[2]	Index of the WEP key
Wep_psk[3][32][3]	WEP 3 keys

[1] Valid only when 7th bit of feature select bitmap is set.
[2] Valid only when 7th bit of feature select bitmap is set.
[3] Valid only when 7th bit of feature select bitmap is set.

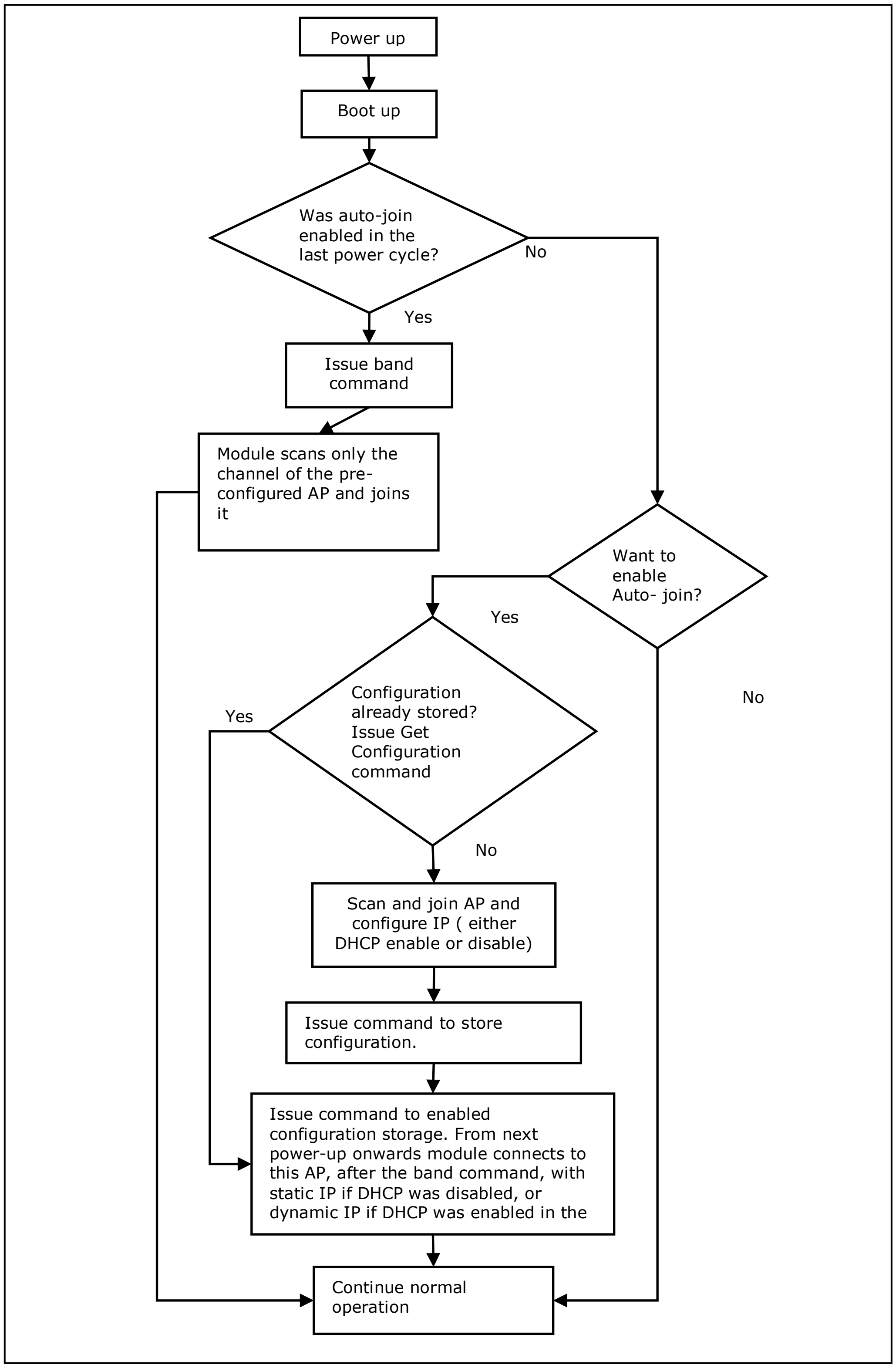

Power up
Boot up
Was auto-join enabled in the last power cycle?
No
Yes
Issue band command
Module scans only the channel of the pre-configured AP and joins it
Want to enable Auto- join?
Yes
No
Configuration already stored? Issue Get Configuration command
Yes
No
Scan and join AP and configure IP (either DHCP enable or disable)
Issue command to store configuration.
Issue command to enabled configuration storage. From next power-up onwards module connects to this AP, after the band command, with static IP if DHCP was disabled, or dynamic IP if DHCP was enabled in the
Continue normal operation

Figure 38: Connecting to Pre-configured AP

5 Firmware Upgrade and Bootload

The firmware of the RS9110-N-11-2X module resides in a Flash memory on the module. The host has to load a compact software bootloader that in turn loads the functional firmware from the Flash to the on-chip memory.

NOTE: Writing code from a scratch to execute these sequences may be time consuming. The user is strongly advised to use the API set, the source code of which is provided with the software package. Reference files are at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/API_Lib/. Documentation for the API set is available at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/Documentation/

5.1 Firmware Upgrade Process

Firmware upgrade of the image (functional firmware) on the Flash requires an understanding of the functionalities of the three images that are to be loaded on to the RS9110-N-11-2X module.

The firmware images are classified as follows:

Functional firmware: This is the actual firmware, that has to be either loaded from the module's flash memory or has to be upgraded.
Software Bootloader: This image, when loaded onto the module from the Host, loads the existing functional firmware from the Flash on to the on-chip memory and executes it.
Image upgrader: This image, when loaded onto the module from the Host, is used to load the new Functional firmware image to the Flash.

NOTE: Functional firmware is represented by ffinst1, ffinst2, and ffdata.

Software bootloader is represented by sbinst1, sbinst2, sbdata1 and sbdata2

Image upgrader is represented by iuinst1, iuinst2, and iudata.

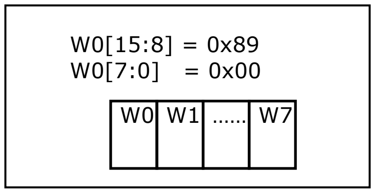
In the firmware upgrade process based on a new release, the user should take the complete set of above mentioned files from the same release package. The firmware and image files are written into the module while the bootloader files are updated into the host system and used during the regular booting up subsequently.

Please also note that these files are hex representation of files so do not take the size shown on PC.The file size can be calculated using sizeof() function or by counting the actual bytes in the file.

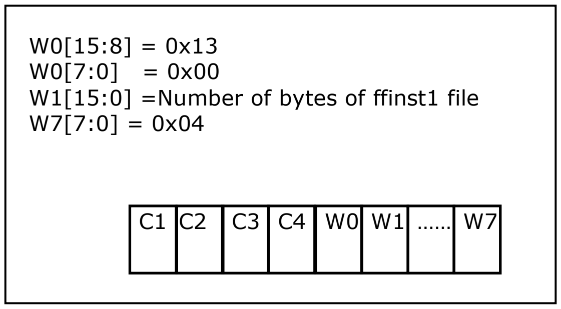
The general outline for upgrading firmware is shown below. These points will be referred to in the following sections where more details on firmware upgrade are presented.

1. Write 0x00000001 to the address 0x22000004 using Memory Write.

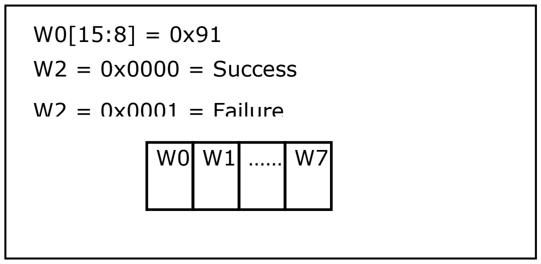
2. Load image upgrader file iuinst1 using Memory Write with address 0x00000000.
3. Load iuinst2 using Memory Write with address 0x02000000.
4. Load iudata using Memory Write with address 0x20003100.
5. Bring the module out of soft reset by writing 0 to the address 0x22000004 using Memory Write.
6. Read card ready status (by doing a 16 bytes slave read) after getting interrupt. The response is as shown below.


W0[15:8] = 0x89
W0[7:0]   = 0x00

7. Load ffinst1 using Memory Write with address 0x02008000.
8. Transmit ‘ffinst1 Upgrade’ management frame.


W0[15:8] = 0x13
W0[7:0]   = 0x00
W1[15:0] =Number of bytes of ffinst1 file
W7[7:0] = 0x04

Read status (by doing 16 bytes slave read) of ffinst1 upgrade success after getting an interrupt. Expected response is shown below.


W0[15:8] = 0x91
W2 = 0x0000 = Success
W2 = 0x0001 = Failure

9. Load ffinst2 using Memory Write with address 0x02008000.
10. Transmit ‘ffinst2 Upgrade’ management frame.

Read status of ffinst2 (by doing 16 bytes slave read) upgrade success after getting an interrupt. Expected response is shown below.

11. Load ffdata using Memory Write with address 0x02008000.
12. Transmit 'ffdata Upgrade' management frame.

W0[15:8] = 0x15
W0[7:0] = 0x00
W1[15:0] =Number of bytes of ffdata file
W7[7:0] = 0x04

C1	C2	C3	C4	W0	W1	……	W7

Read status of ffdata upgrade success (by doing 16 bytes slave read) after getting an interrupt. Expected response is shown below.

W0[15:8] = 0x93
W2 = 0x0000 = Success
W2 = 0x0001 = Failure

W0	W1	……	W7

NOTE: Firmware version 4.7.1 and above have the feature of wireless firmware upgrade and wireless configuration. The next section describes all the flows for firmware upgrade and bootloading taking this feature into account.

5.2 Boot-loading Process

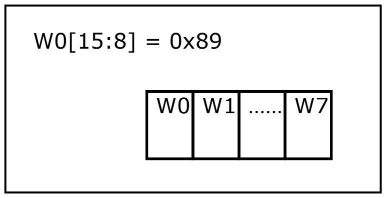
Once the firmware upgrade process is done, the module needs to be powered off and powered on to continue with the operation mentioned below.

The following procedure should be followed by the host driver/application in order to load the functional firmware from Flash (non-upgrade, normal mode of operation).

13. Load RS.CN.xx.GENR.SP.x.x.x/Firmware/sbinst1 using Memory Write with address 0x00000000.
14. Load RS.CN.xx.GENR.SP.x.x.x/Firmware/sbinst2 using Memory Write with address 0x02014010.
15. Load RS.CN.xx.GENR.SP.x.x.x/Firmware/sbdata1 using Memory Write with address 0x20003100.

16. Bring the module out of soft reset by writing 0x00000000 to the address 0x22000004 using Memory Write.
17. Read card ready status (by doing 16 bytes slave read) after getting interrupt.

    The response is as shown below.



18. Load RS.CN.xx.GENR.SP.x.x.x/Firmware/sbdata2 using Memory Write with address 0x02010000.

NOTE:

Irrespective of the firmware version (4.5.0 and 4.7.1 as on Nov, 2012) already present in the module, the user can use files RS.CN.xx.GENR.SP.4.7.1/Firmware/sb* and the above process and addresses for boot-loading the module. It is advised to not use RS.CN.xx.GENR.SP.4.5.x/Firmware/sb* files for bootloading if the user is not sure about the firmware version already present in the module, as this can cause issues if the firmware present is 4.7.x

.

The combination of firmware upgrade and boot-loading processes are shown below.

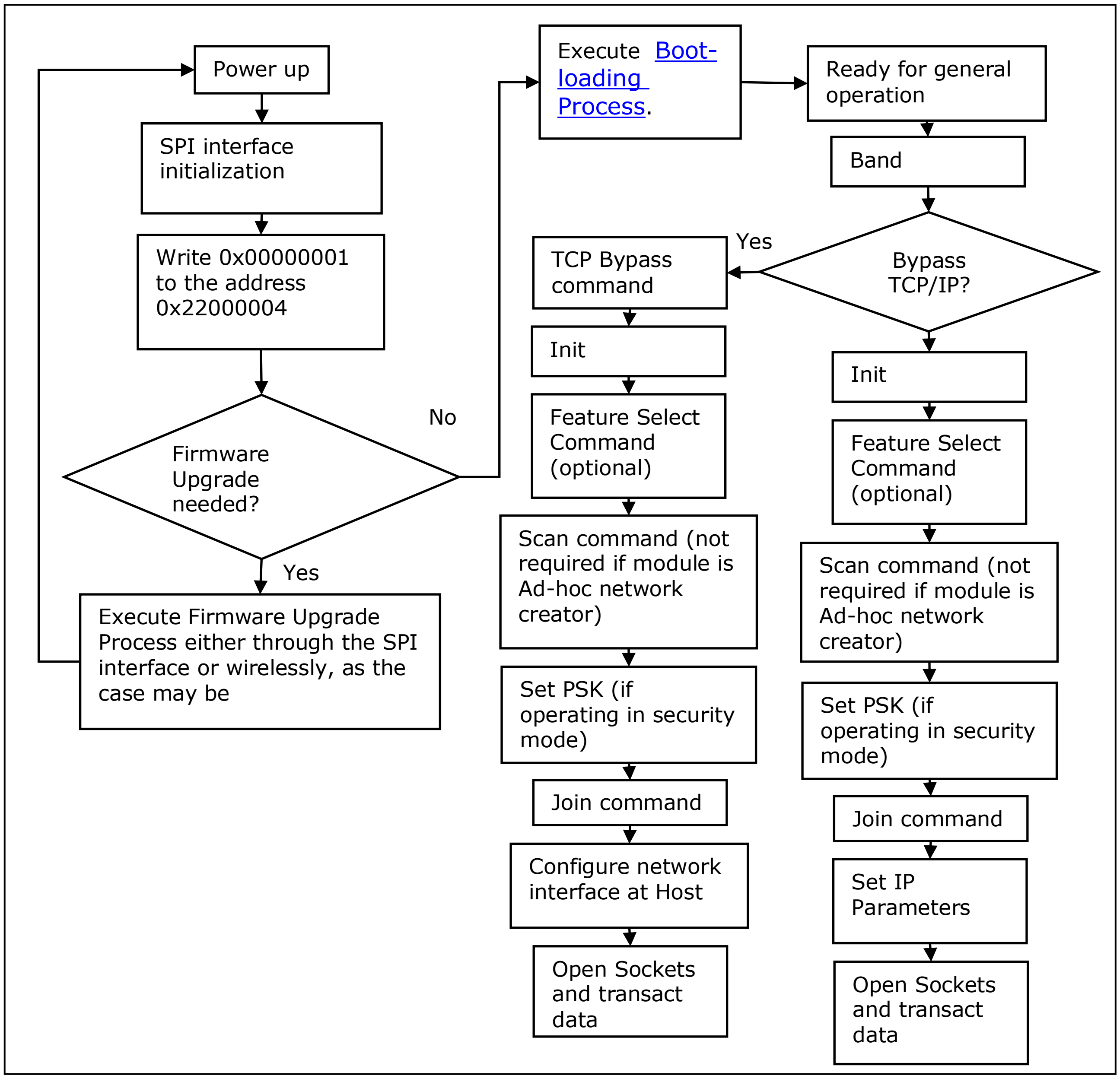

Power up
SPI interface initialization
Write 0x00000001 to the address 0x22000004
Firmware Upgrade needed?
Yes
Execute Firmware Upgrade Process either through the SPI interface or wirelessly, as the case may be
No
Execute Boot-loading Process.
Ready for general operation
Band
Bypass TCP/IP?
Yes
TCP Bypass command
Init
Feature Select Command (optional)
Scan command (not required if module is Ad-hoc network creator)
Set PSK (if operating in security mode)
Join command
Configure network interface at Host
Open Sockets and transact data
Init
Feature Select Command (optional)
Scan command (not required if module is Ad-hoc network creator)
Set PSK (if operating in security mode)
Join command
Set IP Parameters
Open Sockets and transact data

Figure 39: General Operation

NOTE: Writing code from a scratch to execute these sequences may be time consuming. The user is strongly advised to use the API set, the source code of which is provided with the software package. Reference files are RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/API_Lib/. Documentation/ for the API set is available at RS.CN.xx.GENR.SP.x.x.x/Driver/Driver_TCP<or nonTCP>/Documentation/

5.3 Firmware Upgrade Using SPI Interface without Wireless Firmware Upgrade feature

The following process should be followed to upgrade the firmware through SPI interface without wireless firmware upgrade feature.

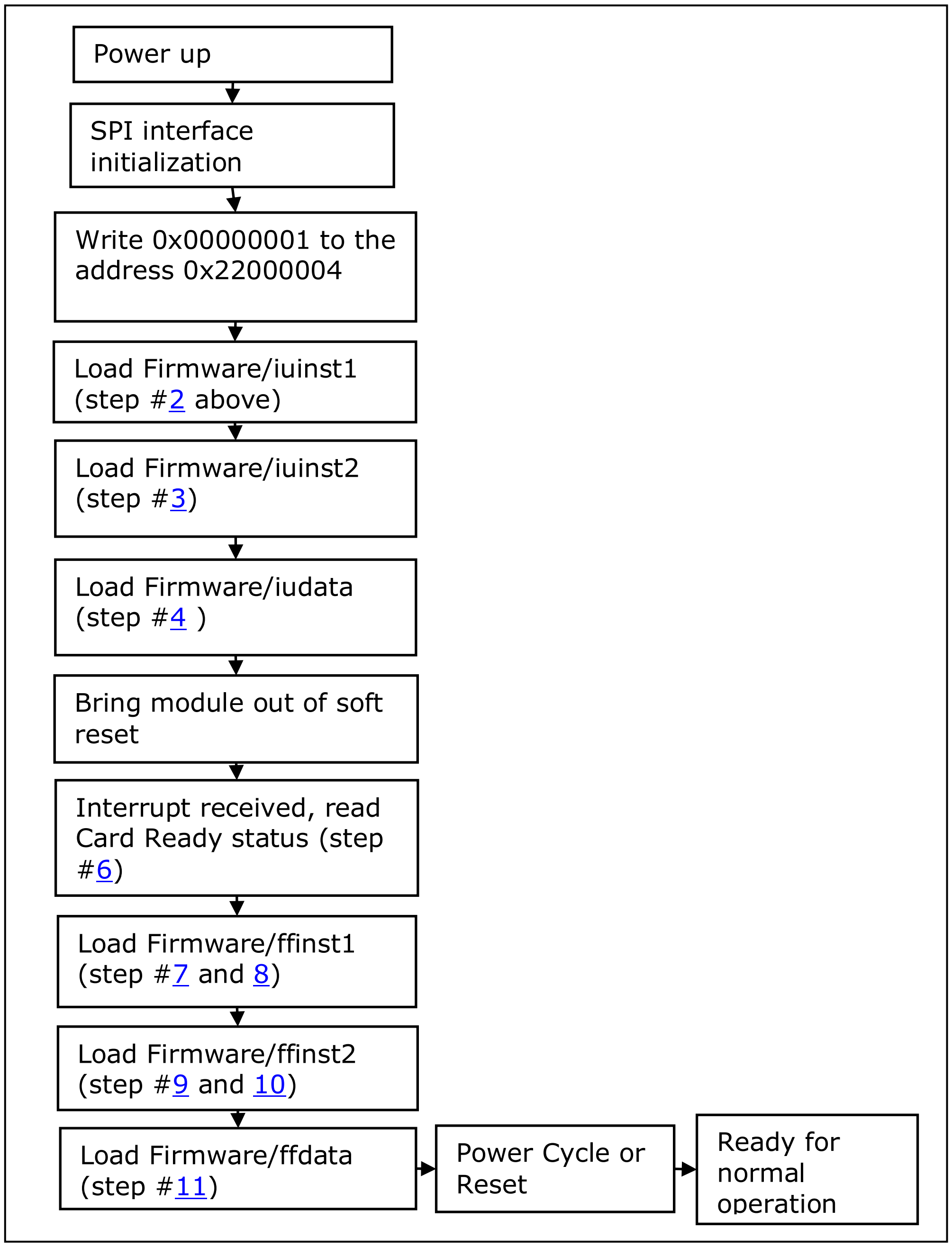

Power up
SPI interface initialization
Write 0x00000001 to the address 0x22000004
Load Firmware/iuinst1 (step #2 above)
Load Firmware/iuinst2 (step #3)
Load Firmware/iudata (step #4)
Bring module out of soft reset
Interrupt received, read Card Ready status (step #6)
Load Firmware/ffinst1 (step #7 and 8)
Load Firmware/ffinst2 (step #9 and 10)
Load Firmware/ffdata (step #11)
Power Cycle or Reset
Ready for normal operation

Figure 40: Firmware Upgrade - Flow 1

5.4 Firmware Upgrade Using SPI Interface with Wireless Firmware Upgrade feature[1]

The following process should be followed to upgrade the firmware through SPI interface with wireless firmware upgrade feature.[2]

[1] Wireless Firmware Upgrade Feature is available from 4.7.X onwards only.
[2] The folder "Wireless Upgrade" is referred to as WFU in the figure. For example WFU/cfdata means Wireless_Upgrade/cfdata

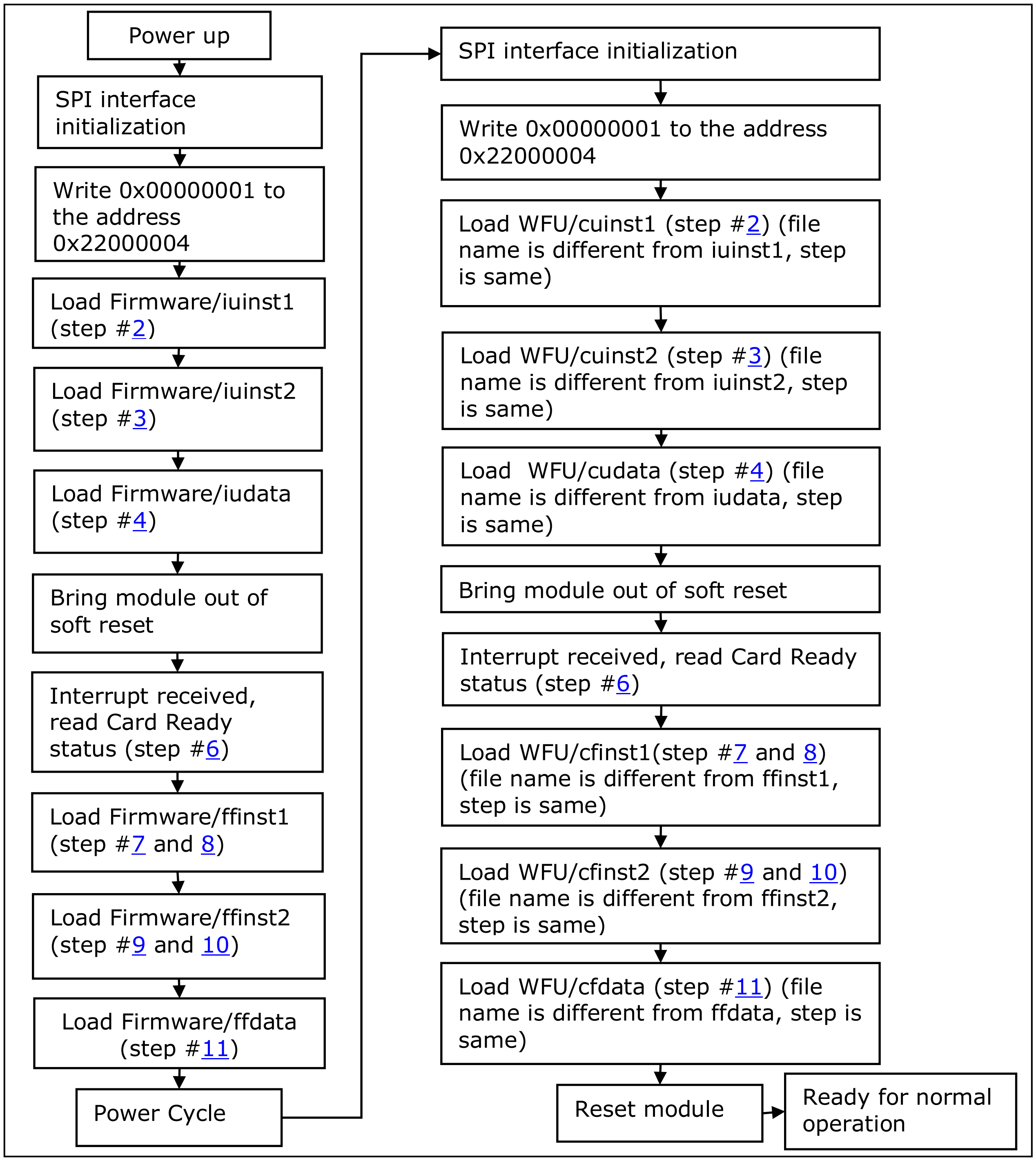

Power up
SPI interface initialization
Write 0x00000001 to the address 0x22000004
Load Firmware/iuinst1 (step #2)
Load Firmware/iuinst2 (step #3)
Load Firmware/iudata (step #4)
Bring module out of soft reset
Interrupt received, read Card Ready status (step #6)
Load Firmware/ffinst1 (step #7 and 8)
Load Firmware/ffinst2 (step #9 and 10)
Load Firmware/ffdata (step #11)
Power Cycle
SPI interface initialization
Write 0x00000001 to the address 0x22000004
Load WFU/cuinst1 (step #2) (file name is different from iuinst1, step is same)
Load WFU/cuinst2 (step #3) (file name is different from iuinst2, step is same)
Load WFU/cudata (step #4) (file name is different from iudata, step is same)
Bring module out of soft reset
Interrupt received, read Card Ready status (step #6)
Load WFU/cfinst1(step #7 and 8) (file name is different from ffinst1, step is same)
Load WFU/cfinst2 (step #9 and 10) (file name is different from ffinst2, step is same)
Load WFU/cfdata (step #11) (file name is different from ffdata, step is same)
Reset module
Ready for normal operation

Figure 41: Firmware Upgrade - Flow 2

5.5 Firmware Upgrade from 4.7.x to a higher version Wirelessly

The firmware of the modules RS9110-N-11-22(-04)(-05) and RS9110-N-11-28(-04) can be upgraded wirelessly. The wireless firmware upgrade feature is available in firmware version 4.7.1 and above. The following sections describe how to use this features.

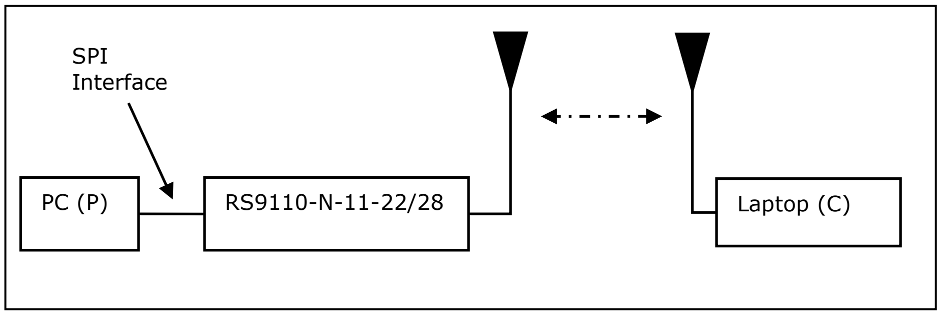


Figure 42: Wireless Firmware Upgrade Set-up

1. Power up the module and follow the processes in figure 43(Firmware Upgrade – Flow 3). The module now becomes an Access Point with open mode and SSID REDPINE_<MAC ID of the module>

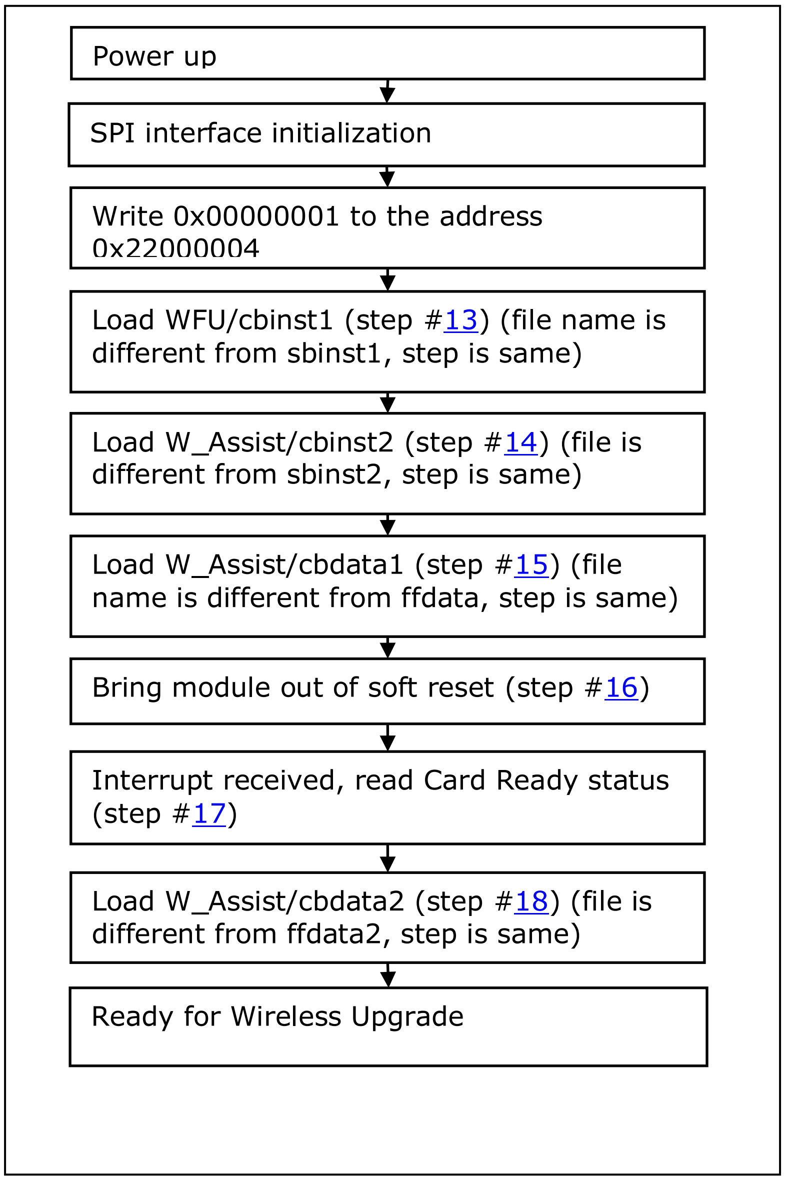
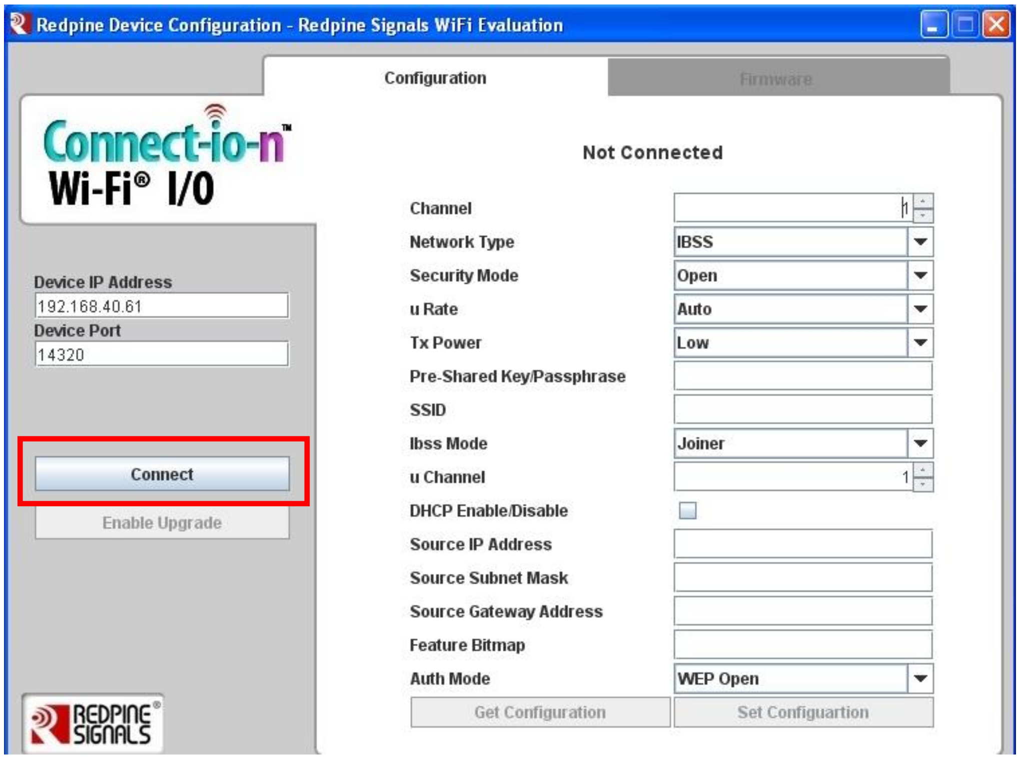

Power up
SPI interface initialization
Write 0x00000001 to the address 0x22000004
Load WFU/cbinst1 (step #13) (file name is different from sbinst1, step is same)
Load W_Assist/cbinst2 (step #14) (file is different from sbinst2, step is same)
Load W_Assist/cbdata1 (step #15) (file name is different from ffdata, step is same)
Bring module out of soft reset (step #16)
Interrupt received, read Card Ready status (step #17)
Load W_Assist/cbdata2 (step #18) (file is different from ffdata2, step is same)
Ready for Wireless Upgrade

Figure 43:Firmware Upgrade - Flow 3

2. Connect Laptop (C) to the Access Point created by the module. The Laptop will acquire an IP address from the AP.
3. Open the application RS.CN.xx.GENR.SP.x.x.x\Software\Wireless_Upgrade\ DeviceConfigGUI.jar in Laptop (C).By default, the "Device IP Address" is 192.168.40.61 and "Device Port" is 14320. Keep them unchanged and click "Connect". This makes the laptop establish a TCP socket

connection with the AP. The AP has an IP hardcoded into itself – 192.168.40.61. Ignore the values on the right side of the GUI.



Not Connected

Device IP Address: 192.168.40.61
Device Port: 14320

Channel	1
Network Type	IBSS
Security Mode	Open
u Rate	Auto
Tx Power	Low
Pre-Shared Key/Passphrase	
SSID	
Ibss Mode	Joiner
u Channel	1
DHCP Enable/Disable	☐
Source IP Address	
Source Subnet Mask	
Source Gateway Address	
Feature Bitmap	
Auth Mode	WEP Open

Figure 44: Configuration GUI

4. Click "Enable Upgrade" button and then click on the "Firmware" tab.

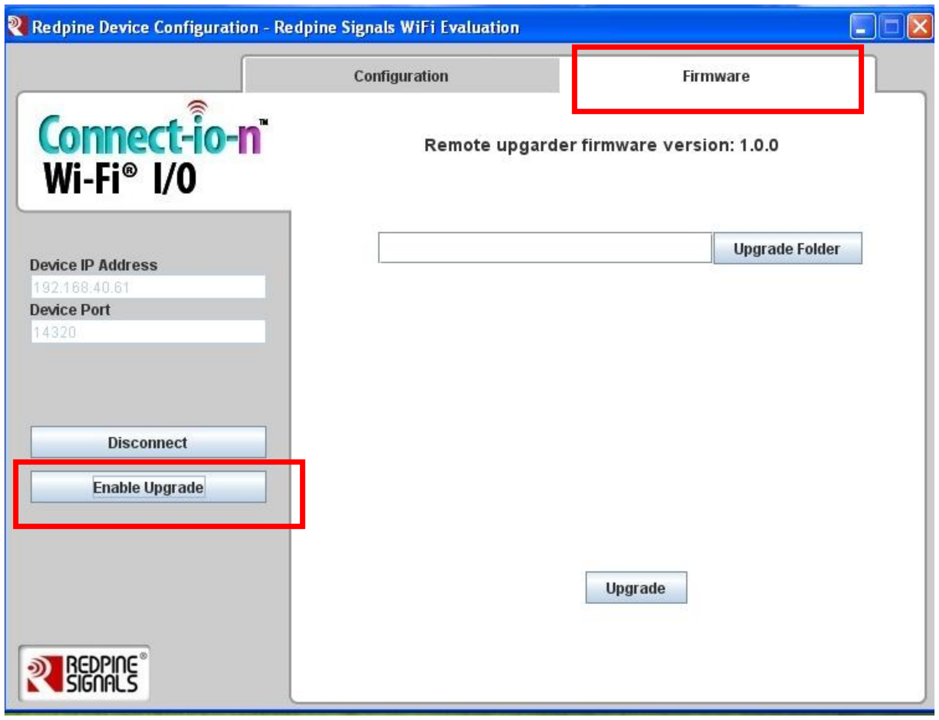
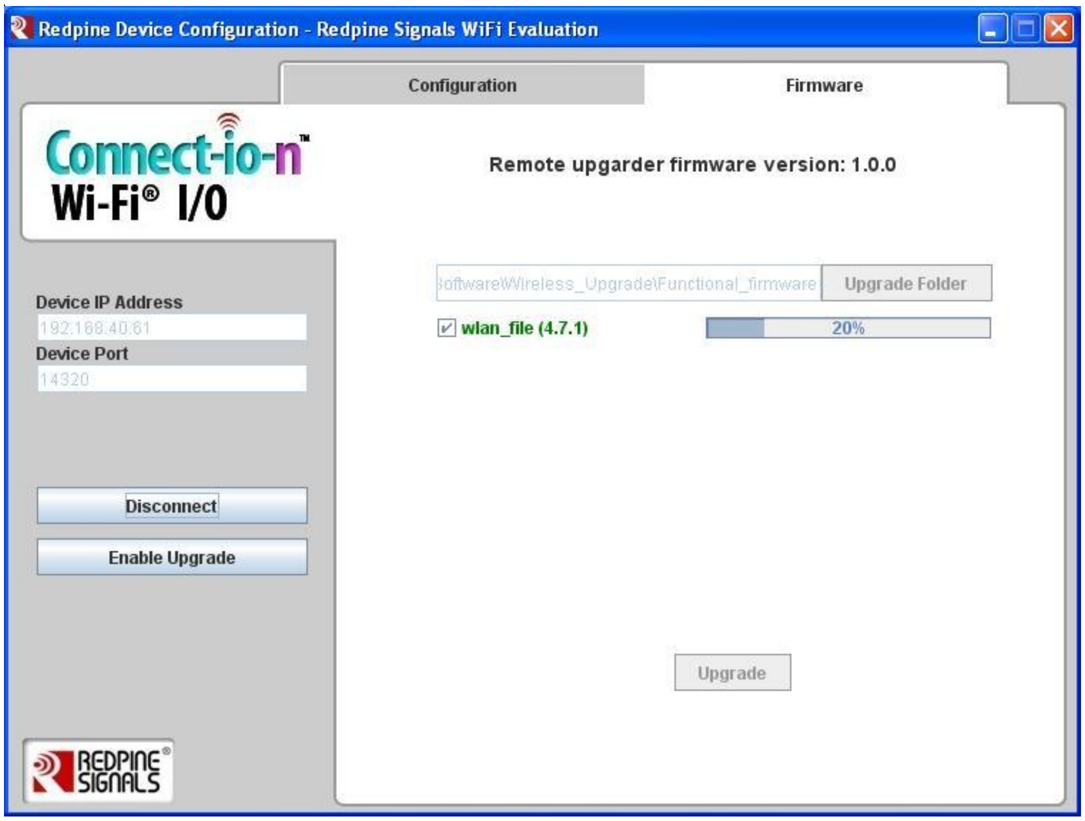


Figure 45: Configuration GUI(contd..)

5. Click on "Upgrade Folder" and select the FOLDER RS.CN.xx.GENR.SP.x.x.x\Software\Wireless_Upgrade\. Then click on the check box for the file wlan_file.rps that appears on the GUI. Click on "Upgrade" button.

NOTE: The file RS.CN.xx.GENR.SP.x.x.x\Software\Wireless_Upgrade\wlan_file.rps has the same contents as the files RS.CN.xx.GENR.SP.x.x.x\Software\Firmware\ff*, but is formatted appropriately for wireless upgrade



Remote upgarder firmware version: 1.0.0

Device IP Address: 192.168.40.61

Device Port: 14320

oftware\Wireless_Upgrade\Functional_firmware

☑ wlan_file (4.7.1) 20%

9. The upgrade process starts. After successful upgrade, an interrupt is sent from the module to the Host. The Host should read the status in the same way as 17 and should get the read value as 0x38.
10. The module is now upgraded with the latest firmware. It can be now power cycled or hard reset and used in functional mode as shown in figure General Operation.

5.6 Using the Wireless Configuration Feature

Using this feature, the module can be configured to

1. Join a specific Access Point
2. Join a specific ad-hoc network

This will be referred to as "Auto-connect".

5.6.1 Using the Wireless Configuration in AP mode

This section describes wireless configuration by putting the module initially into Access Point mode. The next section describes wireless configuration when the module is already connected to an AP.

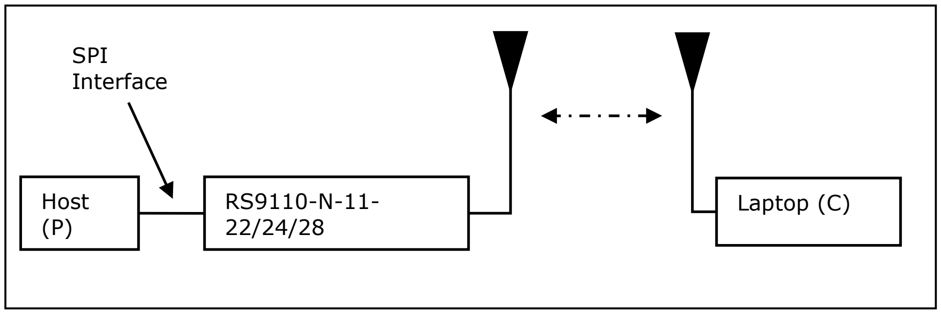
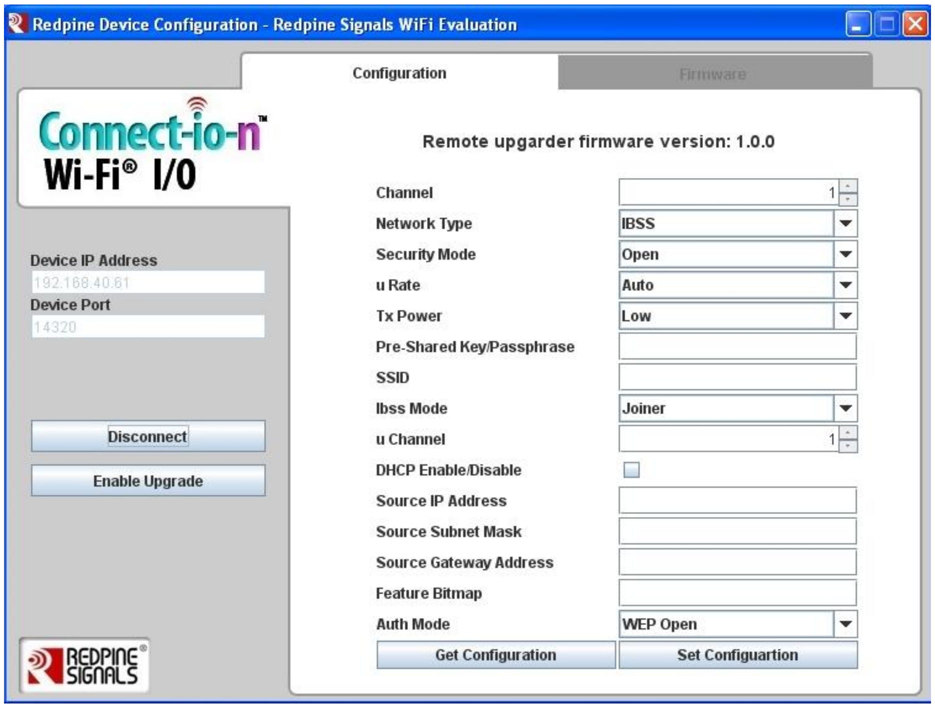


Figure 46: Wireless Configuration Set-up

1. Power up the module. Follow the same process as in figure 43(Firmware Upgrade – Flow 3).The module comes up as an Access Point with open mode and SSID REDPINE_<last 3 bytes of MAC ID of the module>
2. Connect a Laptop (C) to the AP created by the module. The Laptop should be configured as a DHCP client. The Laptop will acquire the IP address from the AP. Leave the "Device IP Address" and "Device Port" unchanged to 192.168.40.61 and 14320 respectively. After clicking on "Connect" Button, click on the "Configuration" tab on top.



Remote upgarder firmware version: 1.0.0

Device IP Address: 192.168.40.61
Device Port: 14320

Channel	1
Network Type	IBSS
Security Mode	Open
u Rate	Auto
Tx Power	Low
Pre-Shared Key/Passphrase	
SSID	
Ibss Mode	Joiner
u Channel	1
DHCP Enable/Disable	☐
Source IP Address	
Source Subnet Mask	
Source Gateway Address	
Feature Bitmap	
Auth Mode	WEP Open

Figure 47: Configuration GUI

3. Click on the "Get Configuration" button. It displays the currently stored configuration parameters. These parameters can be ignored for now.
4. Fill the values of the parameters
    a. Channel: Channel number of the AP or Adhoc network that the module should connect to. Refer tables Channel Number Parameter (2.4 Ghz) and Channel Number Parameter (5 Ghz).
    b. Network Type
    c. Security Mode- Security mode of the Access Point or Adhoc network
    d. uRate (refer table Data Rate Parameter)
    e. Tx Power
    f. Pre shared key- This can be left unfilled if the security mode is "Open".
    g. SSID- This is the SSID of the Access Point or Adhoc network to which the module should join
    h. IBSS mode (Adhoc)
    i. uChannel
    j. DHCP Enable/Disable- Used in infrastructure mode
    k. Source IP Address, Source Subnet Mask, Source Gateway Address: If DHCP is disabled, an IP address, subnet mask and gateway address should be entered for "Source IP Address", "Source Subnet Mask" and "Source Gateway" respectively. The module will get configured with these parameters.
    l. Feature Bitmap- The decimal value of the parameter *feature_select_bitmap* of the command Feature Select can be entered here. It can be set to '0' for most use cases.
    m. Auth mode- If "WEP" is selected in "Security Mode", then the options of "WEP Open" or "WEP Shared" should be used here.
5. Click on the button "Set Configuration". After the configuration is successfully stored in the module, it sends an interrupt to the Host. The Host should read the status in the same way as 17 and should get the read value as 0x18. The configuration information is stored in non-volatile memory of the module and can be accessed by the Host at any time using the command "Get Information about Stored Configuration".
6. The module should now be power cycled or hard reset. The "Band" command should be issued, after which the module will automatically connect to the configured AP.

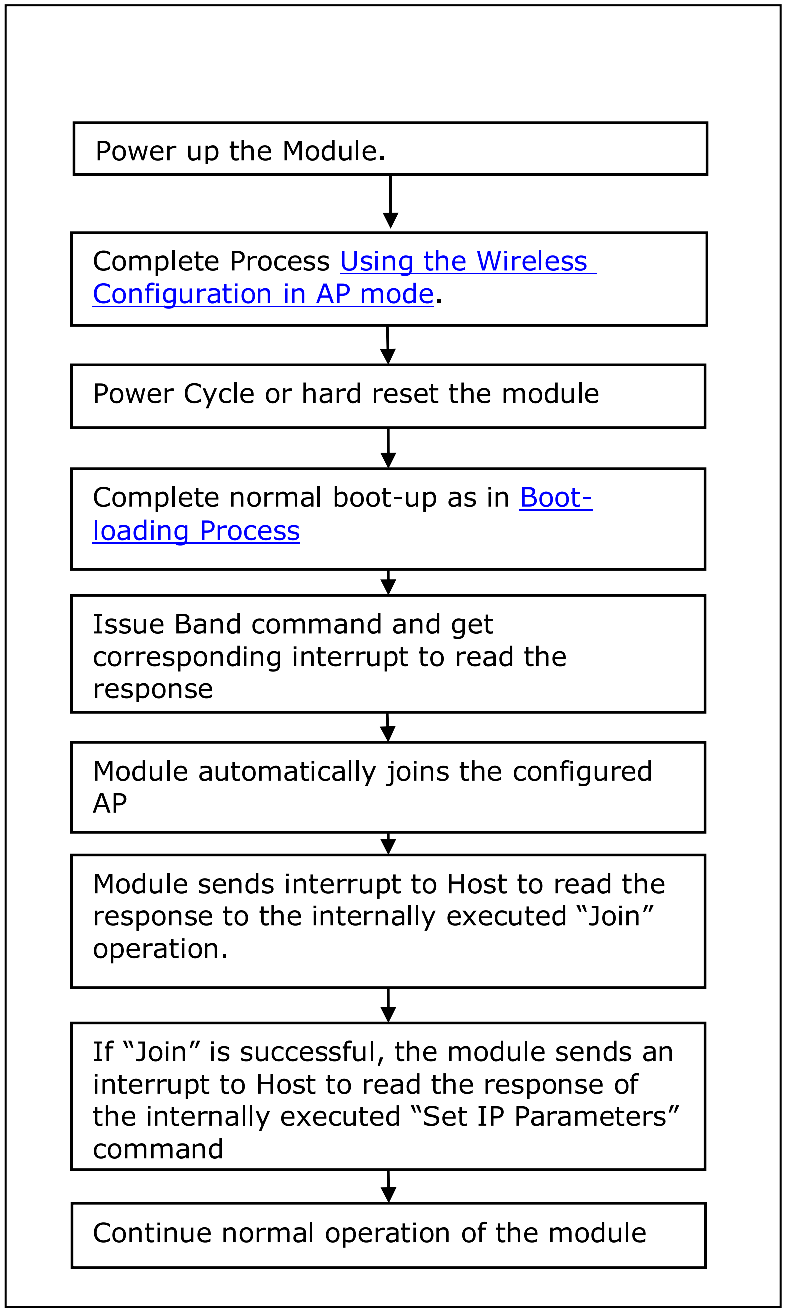

Power up the Module.
Complete Process Using the Wireless Configuration in AP mode.
Power Cycle or hard reset the module
Complete normal boot-up as in Boot-loading Process
Issue Band command and get corresponding interrupt to read the response
Module automatically joins the configured AP
Module sends interrupt to Host to read the response to the internally executed "Join" operation.
If "Join" is successful, the module sends an interrupt to Host to read the response of the internally executed "Set IP Parameters" command
Continue normal operation of the module

Figure 48: Wireless Configuration Flow-SPI

5.6.2 Using the Wireless Configuration when Module is connected to an AP

This section describes wireless configuration when the module is already connected to an AP.

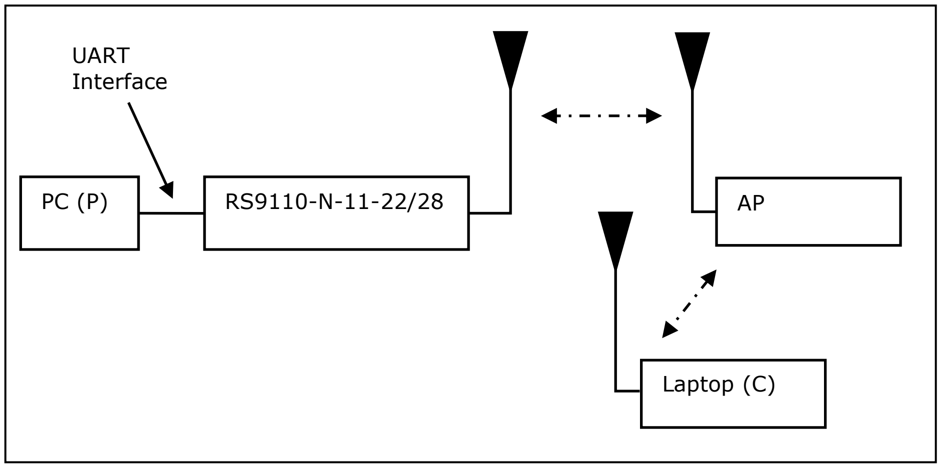
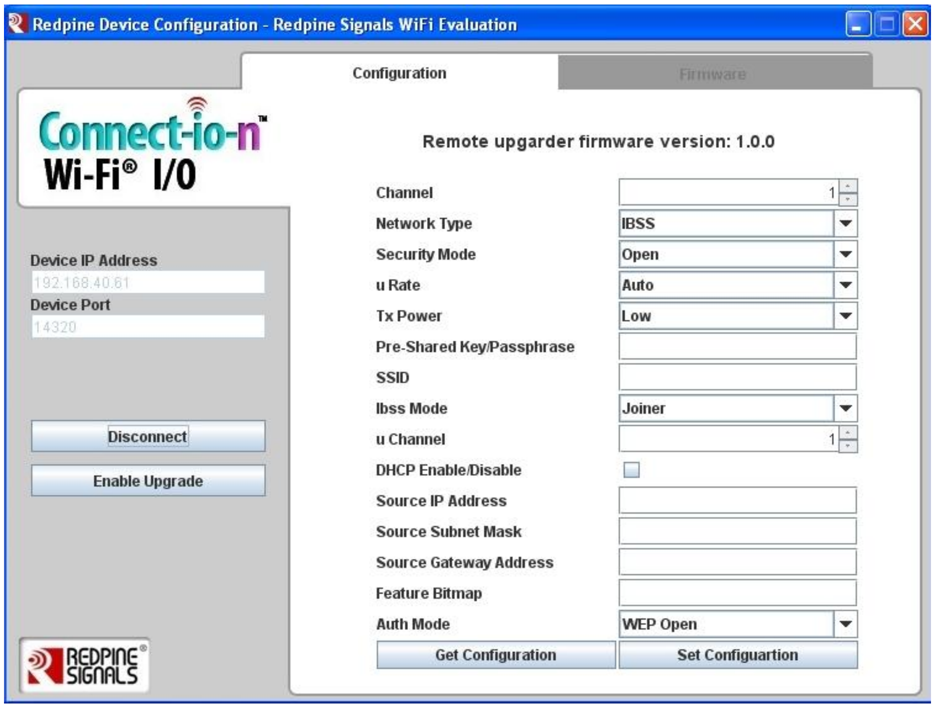


Figure 49: Wireless Configuration Set-up

1. Assuming the module is already connected to an AP after a normal boot-up as described in section Boot-loading Process, connect a Laptop to the same AP.
2. Open the application RS.CN.xx.GENR.SP.x.x.x\Software\Wireless_Upgrade\DeviceConfigGUI.jar in Laptop (C) and click on the connect button. Enter the IP address of the module in the field "Device IP Address". Keep the "Device Port" unchanged to 14320. After clicking on "Connect" Button, click on the "Configuration" tab on top.



Remote upgarder firmware version: 1.0.0

Device IP Address	192.168.40.61
Device Port	14320
Channel	1
Network Type	IBSS
Security Mode	Open
u Rate	Auto
Tx Power	Low
Pre-Shared Key/Passphrase	
SSID	
Ibss Mode	Joiner
u Channel	1
DHCP Enable/Disable	☐
Source IP Address	
Source Subnet Mask	
Source Gateway Address	
Feature Bitmap	
Auth Mode	WEP Open

Figure 50: Configuration GUI

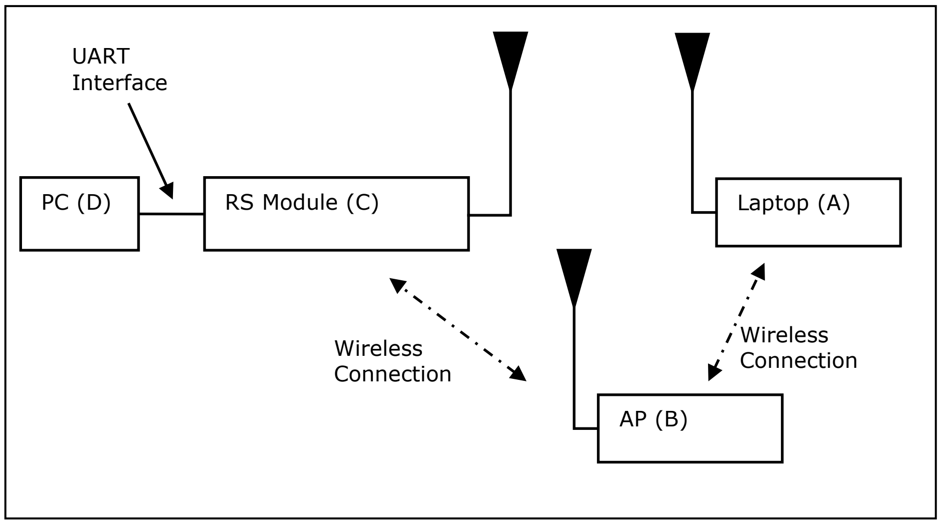
3. Click on the "Get Configuration" button. It displays the currently stored configuration parameters. These parameters can be ignored for now.
4. Fill the values of the parameters
    a. Channel: Channel number of the AP or Adhoc network that the module should connect to. Refer tables Channel Number Parameter (2.4 Ghz) and Channel Number Parameter (5 Ghz).
    b. Network Type
    c. Security Mode- Security mode of the Access Point or Adhoc network
    d. uRate (refer table Data Rate Parameter)
    e. Tx Power
    f. Pre shared key- This can be left unfilled if the security mode is "Open".
    g. SSID- This is the SSID of the Access Point or Adhoc network to which the module should join
    h. IBSS mode (Adhoc)
    i. uChannel
    j. DHCP Enable/Disable- Used in infrastructure mode
    k. Source IP Address, Source Subnet Mask, Source Gateway Address: If DHCP is disabled, an IP address, subnet mask and gateway address should be entered for "Source IP Address", "Source Subnet Mask" and "Source Gateway" respectively. The module will get configured with these parameters.
    l. Feature Bitmap- The decimal value of the parameter *feature_select_bitmap* of the command Feature Select can be entered here. It can be set to '0' for most use cases.
    m. Auth mode- If "WEP" is selected in "Security Mode", then the options of "WEP Open" or "WEP Shared" should be used here.
5. Click on the button "Set Configuration". After the configuration is successfully stored in the module, it sends an interrupt to the Host. The Host should read the status in the same way as 17 and should get the read value as 0x18. The configuration information is stored in non-volatile memory of the module and can be accessed by the Host at any time using the command "Get Information about Stored Configuration".
6. The module should now be power cycled or hard reset. The "Band" command should be issued, after which the module will automatically connect to the configured AP. Refer figure Wireless Configuration Flow-SPI.

NOTE: When the module is configured using wireless interface, the "Auto-join" feature is enabled. This makes the module join to the configured AP on the next power cycle or hard reset. If at any point of time, the user wants to disable the "Auto-join" feature, the command Enable automatic joining to pre-configured Access Point should be used.

6 Appendix A: Example Flow for AT Commands

This section gives an example flow of AT Commands for setting up Wi-Fi connection in Infrastructure and IBSS modes and then opening TCP/UDP/LTCP sockets and exchanging data. The commands are given according to the following network settings mentioned in each section.

Infrastructure Mode setup in WPA2(PSK) security mode



Frequency Band: 2.4GHz

SSID of AP: RSI_Infrastructure

Network Type: Infrastructure

Channel: 1

TX Rate: 0 (Autorate)

TX Power: 2 (High)

Security: WPA2-PSK

Pre-Shared Key: RSI_EVAL [To be configured in (B)]

IP Address of RS9110-N-11-2X Module: Assumed to be 192.168.1.175 in this set-up

IP Address of Access Point: Assumed to be 192.168.1.99 in this set-up

IP Address of Peer Computer: Assumed to be 192.168.1.172 in this set-up

The AT Command sequence is as follows:

1. Band selection: **at+rsi_band=0**\r\nInitialization: at+rsi_init\r\n
2. Scanning for Access Points : **at+rsi_scan=0**\r\n
3. Network Type: **at+rsi_network=INFRASTRUCTURE**\r\n

4. Autmode: **at+rsi_authmode=3**\r\n
5. Pre-Shared Key: **at+rsi_psk=RSI_EVAL**\r\n
6. Joining (Association): **at+rsi_join=RSI_Infrastructure,0,2**\r\n
7. IP Address Configuration (Manual): **at+rsi_ipconf=0, 192.168.1.175,255.255.255.0,192.168.1.99**\r\n
8. TCP server socket creation at (C): **at+rsi_tcp=5001**\r\n.
   The return value for this command would be the socket number, e.g., 1.
9. TCP client socket creation on (A): Run on command line of Laptop A- "**RS.CN.xx.GENR.UT.x.x.x\Software\Applications\Windows\TCP.exe c 2001 192.168.1.175 5001**.
10. Sending 10 bytes of data over TCP to (A): **at+rsi_snd=1,0,0,0,abcdefghij**\r\n

Usage of PC applications:
TCP.exe <s for server> <lport> <dipaddr> <dport>
TCP.exe <c for client> <lport> <dipaddr> <dport>
s – to open a server TCP socket
c – to open a client TCP socket
lport – Local port number
dipaddr – IP address of the destination
dport – Port number of the destination
UDP.exe is also used in the same way.

<u>Infrastructure Mode setup in WEP security mode</u>
The AT Command sequence is as follows:

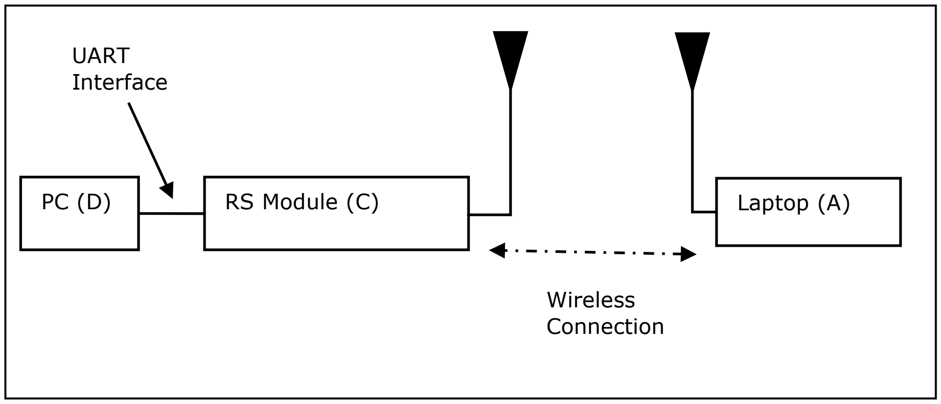
1. Band selection: **at+rsi_band=0**\r\n
2. Initialization: **at+rsi_init**\r\n
3. Scanning for Access Points : **at+rsi_scan=0**\r\n
4. Network Type: **at+rsi_network=INFRASTRUCTURE**\r\n
5. Autmode: **at+rsi_authmode=1**\r\n
6. Pre-Shared Key: **at+rsi_psk=RSI_EVAL**\r\n
7. Set alternative WEP keys:
   at+rsi_wep_keys=2,ABCDE12345,ABCDE12346,ABCDE12347\r\n
8. Joining (Association): **at+rsi_join=RSI_Infrastructure,0,2**\r\n
9. IP Address Configuration (Manual): **at+rsi_ipconf=0, 192.168.1.175,255.255.255.0,192.168.1.99**\r\n

10. TCP server socket creation at (C): **at+rsi_tcp=5001**\r\n. The return value for this command would be the socket number, e.g., 1.

11. TCP client socket creation on (A): Run on command line of Laptop A- "**RS.CN.xx.GENR.UT.x.x.x\Software\Applications\Windows\TCP.exe c 2001 192.168.1.175 5001**.

12.Sending 10 bytes of data over TCP to (A):
at+rsi_snd=1,0,0,0,abcdefghij\r\n

<u>IBSS Creator- Joiner Set-up Description (WEP Mode)</u>



Frequency Band: 2.4GHz

SSID (Network Name): RSI_IBSS

Network Type: IBSS

Channel: 6

Security Type: WEP

TX Rate: 0 (Autorate)

TX Power: 2 (High)

IP Address of RS9110-N-11-2X Module (C) Adhoc creator: Assumed to be 192.168.1.175 in this set-up

IP Address of Peer Computer (A) (Joiner): Assumed to be 192.168.1.172 in this set-up

The AT Commands' sequence is as follows (after Auto Baud Rate Detection is complete):

Commands on node (C)

1.Band selection: **at+rsi_band=0**\r\n

2.Initialization: **at+rsi_init**\r\n

3.Network Type: **at+rsi_network=IBSS_SEC,1,6**\r\n
4.Set PSK: **at+rsi_psk=12345ABCDE**\r\n

5.Create IBSS network: **at+rsi_join=RSI_IBSS,0,2**\r\n

6.IP Address Configuration (DHCP Manual): **at+rsi_ipconf=0, 192.168.1.175,255.255.255.0,192.168.1.1**\r\n
7.Connect to this network from node (A)

Note: Wireless adapter of node (A) should be in the same network. IP address used in this example is 192.168.1.172. Default gateway can be configured same as the RS module (192.168.1.1 in this case). The RS module cannot work as a DHCP server in ad-hoc mode.
Ping from A to C "ping 192.168.1.175" Note : Ping from D to A will not work as application for sending ping is not supported in RS module. It is recommended all other wireless connections, Ethernet adapters etc. in laptop A should be disabled, so that during ping, there is no clash between Ethernet and WLAN adapters.

8. TCP server socket creation at (C): **at+rsi_tcp=5001**\r\n. The return value for this command would be the socket number.

9. TCP client socket creation on (A): Run on command line of Laptop A- "**RS.CN.xx.GENR.UT.x.x.x\Software\Applications\Windows\TCP.exe c 2001 192.168.1.175 5001**.

10.Sending 10 bytes of data over TCP to (A):
at+rsi_snd=1,0,0,0,abcdefghij\r\n
For more detailed information, please follow the EVB user guide.

7 Appendix B: SPI Data Transactions between a MCU Host and the WiFi module

The following sample logs are for a RS22/28 module with the TCP/IP stack enabled. This log may not be exactly same with different firmware versions.

```
-------------------------------------------
SPI Interface Init
Send : 15          Recv : 00
Send : 00          Recv : 58

System Init Done
-------------------------------------------

Boot Loader Start

Set Soft Reset
Send : 74          Recv : 00
Send : 00          Recv : 58
Send : 04          Recv : 58
Send : 00          Recv : 58
Send : 04          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 22          Dummy Recv : 58
Send : 01          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Loading SBINST1
Send : 74          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : 00          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
SBINST1 Bytes start here
Send : 07          Recv : 58
Send : 10          Recv : 58
Send : 08          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : 4c          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 4c          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 07          Dummy Recv : 58
Send : 10          Dummy Recv : 58
Send : 63          Dummy Recv : 58
Send : 21          Dummy Recv : 58
Send : 4c          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 4c          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Loading SBINST1 Done

Loading SBINST2, is 1008 Bytes
```

```
Send : 74          Recv : 00
Send : 00          Recv : 58
Send : f0          Recv : 58
Send : 03          Recv : 58
Send : 10          Recv : 58
Send : 40          Recv : 58
Send : 01          Dummy Recv : 58
Send : 02          Dummy Recv : 58
SBINST2 Bytes start here

Send : 03          Recv : 58
Send : 20          Recv : 58
Send : 80          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : 7c          Dummy Recv : 58
Send : 12          Dummy Recv : 58
Send : 04          Dummy Recv : 58
Send : 20          Dummy Recv : 58
............. All bytes of SBINST2 are not put in this log. This is for illustrative purposes only
.............
Send : 00          Dummy Recv : 58
Send : 5c          Dummy Recv : 58
Send : 80          Dummy Recv : 58
Send : 08          Dummy Recv : 58
Send : 30          Dummy Recv : 58
Send : 98          Dummy Recv : 58
Send : 1c          Dummy Recv : 58
Send : 0c          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Loading SBINST2 Done


Loading SBDATA1, is 32 Bytes
Send : 74          Recv : 00
Send : 00          Recv : 58
Send : 20          Recv : 58
Send : 00          Recv : 58
Send : 00          Recv : 58
Send : 31          Recv : 58
Send : 00          Dummy Recv : 58
Send : 20          Dummy Recv : 58

SBDATA1 Bytes start here
Send : 00          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : b2          Dummy Recv : 58
Send : 0f          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : d4          Dummy Recv : 58
Send : 01          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 5e          Dummy Recv : 58
Send : 2b          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : d4         Dummy Recv : 58
Send : 01         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58
Loading SBDATA1 Done


Remove Soft Reset
Send : 74         Recv : 00
Send : 00         Recv : 58
Send : 04         Recv : 58
Send : 00         Recv : 58
Send : 04         Recv : 58
Send : 00         Recv : 58
Send : 00         Dummy Recv : 58
Send : 22         Dummy Recv : 58
Send : 00         Recv : 58
Send : 00         Recv : 58
Send : 00         Dummy Recv : 58
Send : 00         Dummy Recv : 58


Wait for Card Ready Interrupt

When Card Ready interrupt is received,
do a master read as shown below:
Send : 5c         Recv : 00
Send : 00         Recv : 58
Send : 10         Recv : 58
Send : 00         Recv : 58
wait for start token
Send : 00      Recv : 58
Send : 00      Recv : 58
Send : 00      Recv : 55 ........ Start token received
Read descriptor is
Send : 00      Recv : 00
Send : 00      Recv : 89 ........ Card ready code received
Send : 00      Recv : 00
Send : 00      Recv : 00
Send : 00      Recv : 00
Send : 00      Recv : 00
Send : 00      Recv : 00
Send : 00      Recv : 55
Send : 00      Recv : 88
Send : 00      Recv : 01
Send : 00      Recv : 00
Send : 00      Recv : 00
Send : 00      Recv : d1
Send : 00      Recv : 27
Send : 00      Recv : 04
Send : 00      Recv : 00

Loading SBDATA2
Send : 74         Recv : 80
Send : 00         Recv : 58
Send : 04         Recv : 58
Send : 00         Recv : 58           -- Address 0x02010000 starts here
Send : 00          Recv : 58
Send : 00          Recv : 58
Send : 01          Dummy Recv : 58
Send : 02          Dummy Recv : 58      -- Address 0x02010000 end here
SBDATA2 starts here
Send : 28          Recv : 58
Send : 14          Recv : 58
Send : 28          Dummy Recv : 58
Send : 44          Dummy Recv : 58
Loading SBDATA2 Done
BootLoading Operation Complete
--------------------------------------------

Band Command Start

Send Descriptor
Send : 7c          Recv : 00
Send : 04          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58

Send : 02          Recv : 58
Send : 18          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 04          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send Payload


Send : 7c      Recv : 00
Send : 04      Recv : 58
Send : 04      Recv : 58
Send : 00      Recv : 58
Send : 01      Recv : 58  .... Band_val = 1
Send : 00      Recv : 58
Send : 00      Dummy Recv : 58
Send : 00      Dummy Recv : 58


Wait for interrupt
Interrupt received, Do master read
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Read descriptor is
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 97  .........Band Success Response
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 88
Send : Dummy 00 Recv : 01
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : d1
Send : Dummy 00 Recv : 27
Send : Dummy 00 Recv : 04
Send : Dummy 00 Recv : 00

Band Response =97 Status=00
--------------------------------------------

Init Start
Send : 7c          Recv : 80
Send : 04          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 04          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Wait for interrupt
Interrupt received, Do master read
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Read descriptor is
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 94 ..... Init command response received
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 80
Send : Dummy 00 Recv : 51
Send : Dummy 00 Recv : 40
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : d0
Send : Dummy 00 Recv : 27
Send : Dummy 00 Recv : 04
Send : Dummy 00 Recv : 00

InitRsp 94
--------------------------------------------

Query FWVersion Start

Send Descriptor
Send : 7c          Recv : 80
Send : 02          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : 02          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 02          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send Payload
Send : 7c          Recv : 00
Send : 02          Recv : 58
Send : 04          Recv : 58
Send : 00          Recv : 58
Send : 0d          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58

wait for interrupt
Interrupt received, Do master read to read descriptor
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
.............................
.............................  continue till start token is received
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55 .... start token received
Descriptor data starts here
Send : Dummy 00 Recv : 16
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 02
Send : Dummy 00 Recv : 00

The first word, 0x0016 is payload length
Read the payload of length 0x0016
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 16          Recv : 58
Send : 00          Recv : 58
................................
................................  continue till start token is received
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55 .... start token received
read descriptor is
Send : Dummy 00 Recv : 0f ..... Firmware version query response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 34 ..... ASCII For '4'
Send : Dummy 00 Recv : 2e ..... ASCII For '.'
Send : Dummy 00 Recv : 34 ..... ASCII For '4'
Send : Dummy 00 Recv : 2e ..... ASCII For '.'
Send : Dummy 00 Recv : 36 ..... ASCII For '6'
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00

Query FWVersion Rsp=0f FWVersion=4.4.6


---------------------------------------------------
Query Mac Address

Send Descriptor
Send : 7c          Recv : 00
Send : 02          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : 02          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 02          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send Payload
Send : 7c          Recv : 00
Send : 02          Recv : 58
Send : 04          Recv : 58
Send : 00          Recv : 58
Send : 0f          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58

wait for interrupt
Interrupt received, Do master read
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
read descriptor is
Send : Dummy 00 Recv : 0c
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 02
Send : Dummy 00 Recv : 00

The first word, 0x000c is payload length
Read the payload of length 0x000c
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 0c          Recv : 58
Send : 00          Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 12 ..... MAC address query Response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00 ..... MAC address start
Send : Dummy 00 Recv : 23
Send : Dummy 00 Recv : a7
Send : Dummy 00 Recv : 1d
Send : Dummy 00 Recv : 0a
Send : Dummy 00 Recv : 6d ..... MAC address end
Send : Dummy 00 Recv : 00 .....
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00       Error code
Send : Dummy 00 Recv : 00 .....

Query MAC Rsp=0x0012

------------------------------------------------------------
Scan Start

Send descriptor
Send : 7c          Recv : 00
Send : 04          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : 24          Recv : 58
Send : 11          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 04          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send payload
Send : 7c          Recv : 00
Send : 04          Recv : 58
Send : 24          Recv : 58
Send : 00          Recv : 58

Send : 00          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58


wait for interrupt
Interrupt received, Do master read
Read descriptor is
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : ff
Send : Dummy 00 Recv : 95 .... Scan response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : d1
Send : Dummy 00 Recv : 27
Send : Dummy 00 Recv : da
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : d1
Send : Dummy 00 Recv : 27
Send : Dummy 00 Recv : 04
Send : Dummy 00 Recv : 00

ScanRsp=95 Status=00 SUCCESS

Read the payload
DataLen=00da
Send : 5c          Recv : 80
Send : 00          Recv : 58
Send : da          Recv : 58
Send : 00          Recv : 58
.....................................
.....................................  continue till start token is received
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 06
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
....................................
```

```
..................................  All bytes of received scan response is not put in this log. This is
for illustrative pruposes only
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00

---------------------------------------------

Join Start

Send descriptor
Send : 7c           Recv : 00
Send : 04           Recv : 58
Send : 10           Recv : 58
Send : 00           Recv : 58
Send : 48           Recv : 58
Send : 12           Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 04           Dummy Recv : 58
Send : 00           Dummy Recv : 58

send payload is
Send : 7c           Recv : 00
Send : 04           Recv : 58
Send : 48           Recv : 58
Send : 00           Recv : 58

Send : 01           Recv : 58

Send : 00           Recv : 58

Send : 00           Dummy Recv : 58

Send : 02           Dummy Recv : 58

Send : 77           Dummy Recv : 58
Send : 61           Dummy Recv : 58
Send : 69           Dummy Recv : 58
Send : 6d           Dummy Recv : 58
Send : 61           Dummy Recv : 58
Send : 6e           Dummy Recv : 58
Send : 61           Dummy Recv : 58
Send : 6c           Dummy Recv : 58
Send : 6f           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00           Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send : 6d          Dummy Recv : 58
Send : 75          Dummy Recv : 58
Send : 64          Dummy Recv : 58
Send : 70          Dummy Recv : 58
Send : 6f          Dummy Recv : 58
Send : 74          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send : 00          Dummy Recv : 58

Send : 06          Dummy Recv : 58

Send : 00          Dummy Recv : 58


wait for interrupt
Interrupt received, Do master read
Read descriptor
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
```

```
..........................................
..........................................  continue till start token is received
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 96 .... Join Response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00 .... Status code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : c8
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 54
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 04
Send : Dummy 00 Recv : 00

JoinRsp 96 Status=00

--------------------------------------------

IPPARAM Start

Send descriptor
Send : 7c        Recv : 00
Send : 02        Recv : 58
Send : 10        Recv : 58
Send : 00        Recv : 58
Send : 10        Recv : 58
Send : 00        Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 02        Dummy Recv : 58
Send : 00        Dummy Recv : 58

Send payload
Send : 7c        Recv : 00
Send : 02        Recv : 58
Send : 10        Recv : 58
Send : 00        Recv : 58
Send : 01        Recv : 58
Send : 00        Recv : 58
Send : 00        Dummy Recv : 58
Send : c0        Dummy Recv : 58
Send : a8        Dummy Recv : 58
Send : 01        Dummy Recv : 58
Send : 78        Dummy Recv : 58
Send : ffDummy Recv : 58
Send : ffDummy Recv : 58
Send : ffDummy Recv : 58
Send : 00          Dummy Recv : 58
Send : c0          Dummy Recv : 58
Send : a8          Dummy Recv : 58
Send : 01          Dummy Recv : 58
Send : 01          Dummy Recv : 58
Send : 00          Dummy Recv : 58

wait for interrupt
Interrupt received, Do master read
Read descriptor
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 18
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 02
Send : Dummy 00 Recv : 00

Read the payload
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 18          Recv : 58
Send : 00          Recv : 58
....................................
....................................  continue till start token is received

Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 01 .... IPconfig command response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00 ....
Send : Dummy 00 Recv : 23
Send : Dummy 00 Recv : a7
Send : Dummy 00 Recv : 1d      6 Bytes MAC address
Send : Dummy 00 Recv : 0a
Send : Dummy 00 Recv : 6d ....
Send : Dummy 00 Recv : c0 ....
Send : Dummy 00 Recv : a8
Send : Dummy 00 Recv : 01      IP address
Send : Dummy 00 Recv : 78 ....
Send : Dummy 00 Recv : ff ....
Send : Dummy 00 Recv : ff
Send : Dummy 00 Recv : ff      Subnet mask
Send : Dummy 00 Recv : 00 ....
Send : Dummy 00 Recv : c0 ....
Send : Dummy 00 Recv : a8
Send : Dummy 00 Recv : 01      Gate way
Send : Dummy 00 Recv : 01 ....
Send : Dummy 00 Recv : 00 ....
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00      Error code
Send : Dummy 00 Recv : 00 ....

IPPARAMR   =01
Len        =0018
MAC Addr   =00:23:a7:1d:0a:6d
IP Addr    =192.168.1.120
Netmask    =255.255.255.0
Gateway    =192.168.1.1
ErrorCode  =00000000


--------------------------------------------

Socket create

Send descriptor
Send : 7c          Recv : 00
Send : 02          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : 0c          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 02          Dummy Recv : 58
Send : 00          Dummy Recv : 58

Send payload
Send : 7c          Recv : 00
Send : 02          Recv : 58
Send : 0c          Recv : 58
Send : 00          Recv : 58
Send : 02          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 3e          Dummy Recv : 58
Send : 27          Dummy Recv : 58
Send : 3e          Dummy Recv : 58
Send : 27          Dummy Recv : 58
Send : c0          Dummy Recv : 58
Send : a8          Dummy Recv : 58
Send : 01          Dummy Recv : 58
Send : 64          Dummy Recv : 58

wait for interrupt
Interrupt received, Do master read
Read descriptor
```

```
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 12
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 02
Send : Dummy 00 Recv : 00

Read payload
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 12          Recv : 58
Send : 00          Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 02 .... Socket create response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 01
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 3e
Send : Dummy 00 Recv : 27
Send : Dummy 00 Recv : c0
Send : Dummy 00 Recv : a8
Send : Dummy 00 Recv : 01
Send : Dummy 00 Recv : 78
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : b4
Send : Dummy 00 Recv : 05

SocketRsp=02
ErrorCode=00000000
Module IP Addr=192.168.1.120
Socket descriptor is : 01

--------------------------------------------

Receive data from remote peer over socket 01

wait for interrupt
Interrupt received, Do master read
```

```
Read descriptor
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 59
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 04
Send : Dummy 00 Recv : 84
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 30
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 59
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : cf
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 02
Send : Dummy 00 Recv : 00
Read payload
Send : 5c          Recv : 80
Send : 00          Recv : 58
Send : 59          Recv : 58
Send : 00          Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 07 .... Receive response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 01
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 23
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 36
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 3e
Send : Dummy 00 Recv : 27
Send : Dummy 00 Recv : c0
Send : Dummy 00 Recv : a8
Send : Dummy 00 Recv : 01
Send : Dummy 00 Recv : 64
Send : Dummy 00 Recv : 00-------
Send : Dummy 00 Recv : 4b
Send : Dummy 00 Recv : 04
Send : Dummy 00 Recv : 67
Send : Dummy 00 Recv : 40
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 80
Send : Dummy 00 Recv : 06
Send : Dummy 00 Recv : 72
Send : Dummy 00 Recv : 19
Send : Dummy 00 Recv : c0
Send : Dummy 00 Recv : a8
Send : Dummy 00 Recv : 01
Send : Dummy 00 Recv : 64
Send : Dummy 00 Recv : c0
Send : Dummy 00 Recv : a8
Send : Dummy 00 Recv : 01    Receive TCP data offset buf
Send : Dummy 00 Recv : 78
Send : Dummy 00 Recv : 27
Send : Dummy 00 Recv : 3e
Send : Dummy 00 Recv : 27
Send : Dummy 00 Recv : 3e
Send : Dummy 00 Recv : ca
Send : Dummy 00 Recv : 54
Send : Dummy 00 Recv : b8
Send : Dummy 00 Recv : aa
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 01
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 04
Send : Dummy 00 Recv : 50
Send : Dummy 00 Recv : 18
Send : Dummy 00 Recv : 44
Send : Dummy 00 Recv : 70
Send : Dummy 00 Recv : 42
Send : Dummy 00 Recv : 5a
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00------
Send : Dummy 00 Recv : 74------
Send : Dummy 00 Recv : 68
Send : Dummy 00 Recv : 69
Send : Dummy 00 Recv : 73
Send : Dummy 00 Recv : 20
Send : Dummy 00 Recv : 69
Send : Dummy 00 Recv : 73
Send : Dummy 00 Recv : 20
Send : Dummy 00 Recv : 64
Send : Dummy 00 Recv : 61
Send : Dummy 00 Recv : 74
Send : Dummy 00 Recv : 61
Send : Dummy 00 Recv : 20
Send : Dummy 00 Recv : 66
Send : Dummy 00 Recv : 72
Send : Dummy 00 Recv : 6f     Received data of 35 bytes
Send : Dummy 00 Recv : 6d
Send : Dummy 00 Recv : 20
Send : Dummy 00 Recv : 72
Send : Dummy 00 Recv : 65
Send : Dummy 00 Recv : 6d
Send : Dummy 00 Recv : 6f
Send : Dummy 00 Recv : 74
Send : Dummy 00 Recv : 65
Send : Dummy 00 Recv : 20
Send : Dummy 00 Recv : 74
Send : Dummy 00 Recv : 6f
Send : Dummy 00 Recv : 20
Send : Dummy 00 Recv : 72
Send : Dummy 00 Recv : 65
Send : Dummy 00 Recv : 64
Send : Dummy 00 Recv : 70
Send : Dummy 00 Recv : 69
Send : Dummy 00 Recv : 6e
Send : Dummy 00 Recv : 65------

Received Response type=07

Descriptor 16 bytes -  (2bytes - response code,
                        2bytes - receive socket descriptor,
                        4bytes - receive data size,
                        2bytes - header length,
                        2bytes - port number,
                        4bytes - Ip address)
Received data of 35 bytes is - this is data from remote terminal to redpine

----------------------------------------------

Sending data to remote peer over socket 1

Send descriptor
Send : 7c          Recv : 00
Send : 02          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : 59          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 02          Dummy Recv : 58
Send : 00          Dummy Recv : 58
send payload
Send : 7c          Recv : 00
Send : 02          Recv : 58
Send : 5c          Recv : 58
Send : 00          Recv : 58
Send : 03          Recv : 58     -------------------------------------
Send : 00          Recv : 58
Send : 01          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 23          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 36          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58                    descriptor + offset Header + Data
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 74          Dummy Recv : 58----
Send : 68          Dummy Recv : 58
Send : 69          Dummy Recv : 58
Send : 73          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : 69          Dummy Recv : 58
Send : 73          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : 64          Dummy Recv : 58
Send : 61          Dummy Recv : 58
Send : 74          Dummy Recv : 58
Send : 61          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : 66          Dummy Recv : 58
Send : 72          Dummy Recv : 58
Send : 6f          Dummy Recv : 58   sending data
Send : 6d          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : 72          Dummy Recv : 58
Send : 65          Dummy Recv : 58
Send : 64          Dummy Recv : 58
Send : 70          Dummy Recv : 58
Send : 69          Dummy Recv : 58
Send : 6e          Dummy Recv : 58
Send : 65          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : 74          Dummy Recv : 58
Send : 6f          Dummy Recv : 58
Send : 20          Dummy Recv : 58
Send : 72          Dummy Recv : 58
Send : 65          Dummy Recv : 58
Send : 6d          Dummy Recv : 58
Send : 6f          Dummy Recv : 58
Send : 74          Dummy Recv : 58
Send : 65          Dummy Recv : 58----
Send : 00          Dummy Recv : 58----
Send : 00          Dummy Recv : 58  padding
Send : 00          Dummy Recv : 58------------------------------------------
```

```
Sent Packet Count = 01

Sending data - this is data from redpine to remote terminal
-----------------------------------------------------------------------------------------------------------
-------
```

The logs are same for TCP enabled and bypass modes, for commands that are relevant in both modes, except for the send data and receive data commands and the commands listed in Other Commands (Data Frames). Please refer to W2 in the table Frame Descriptor for Management/Data Frames in SPI mode. Some examples are given below.

```
-----------------------------------------------------------------------------
Query FWVersion Start

Send Descriptor
Send : 7c        Recv : 80
Send : 02        Recv : 58
Send : 10        Recv : 58
Send : 00        Recv : 58
Send : 02        Recv : 58
Send : 00        Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 44        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 02        Dummy Recv : 58
Send : 00        Dummy Recv : 58

Send Payload
Send : 7c        Recv : 00
Send : 02        Recv : 58
Send : 04        Recv : 58
Send : 00        Recv : 58
Send : 0d        Recv : 58
Send : 00        Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58

wait for interrupt
Interrupt received, Do master read to read descriptor
Send : 5c        Recv : 00
Send : 00        Recv : 58
Send : 10        Recv : 58
Send : 00        Recv : 58
.............................
.............................  continue till start token is received
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55 .... start token received
Descriptor data starts here
Send : Dummy 00 Recv : 16
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 44
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 02
Send : Dummy 00 Recv : 00

The first word, 0x0016 is payload length
Read the payload of length 0x0016
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 16          Recv : 58
Send : 00          Recv : 58
................................
................................  continue till start token is received
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55 .... start token received
read descriptor is
Send : Dummy 00 Recv : 0f ..... Firmware version query response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 34 ..... ASCII For '4'
Send : Dummy 00 Recv : 2e ..... ASCII For '.'
Send : Dummy 00 Recv : 34 ..... ASCII For '4'
Send : Dummy 00 Recv : 2e ..... ASCII For '.'
Send : Dummy 00 Recv : 36 ..... ASCII For '6'
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00

Query FWVersion Rsp=0f FWVersion=4.4.6

--------------------------------------------
Query MAC Address

Send Descriptor
Send : 7c          Recv : 00
Send : 02          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : 02          Recv : 58
Send : 00          Recv : 58
Send : 00          Dummy Recv : 58
Send : 00          Dummy Recv : 58
Send : 44          Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 02        Dummy Recv : 58
Send : 00        Dummy Recv : 58

Send Payload
Send : 7c        Recv : 00
Send : 02        Recv : 58
Send : 04        Recv : 58
Send : 00        Recv : 58
Send : 0f        Recv : 58
Send : 00        Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58

wait for interrupt
Interrupt received, Do master read
Send : 5c        Recv : 00
Send : 00        Recv : 58
Send : 10        Recv : 58
Send : 00        Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
read descriptor is
Send : Dummy 00 Recv : 0c
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 44
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 02
Send : Dummy 00 Recv : 00

The first word, 0x000c is payload length
Read the payload of length 0x000c
Send : 5c        Recv : 00
Send : 00        Recv : 58
Send : 0c        Recv : 58
Send : 00        Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 12 ..... MAC address query Response code
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00 ..... MAC address start
Send : Dummy 00 Recv : 23
Send : Dummy 00 Recv : a7
Send : Dummy 00 Recv : 1d
Send : Dummy 00 Recv : 0a
Send : Dummy 00 Recv : 6d ..... MAC address end
Send : Dummy 00 Recv : 00 .....
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00      Error code
Send : Dummy 00 Recv : 00 .....

Query MAC Rsp=0x0012




-----------------------------------------------------------------------------
Receive data from remote terminal

wait for interrupt
Interrupt received, Do master read

Read descriptor
Send : 5c          Recv : 00
Send : 00          Recv : 58
Send : 10          Recv : 58
Send : 00          Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 00 ...
Send : Dummy 00 Recv : 00 ... The first 12 bits 0th word of the descriptor gives the length
of received bytes
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 04
Send : Dummy 00 Recv : 84
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 30
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 59
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : cf
Send : Dummy 00 Recv : 00
Send : Dummy 00 Recv : 02
Send : Dummy 00 Recv : 00
Read payload
Send : 5c          Recv : 80
Send : 00          Recv : 58
Send : 59          Recv : 58
Send : 00          Recv : 58
wait for start token
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 58
Send : Dummy 00 Recv : 55
Send : Dummy 00 Recv : 65-----
Send : Dummy 00 Recv : 20
Send : Dummy 00 Recv : 74
Send : Dummy 00 Recv : 6f
Send : Dummy 00 Recv : 20
Send : Dummy 00 Recv : 72     Receive payload
Send : Dummy 00 Recv : 65
Send : Dummy 00 Recv : 64
Send : Dummy 00 Recv : 70
Send : Dummy 00 Recv : 69
Send : Dummy 00 Recv : 6e
Send : Dummy 00 Recv : 65------

All bytes of received data are not actual. These are for illustrative pruposes only

-------------------------------------------

Sending data to remote terminal

Send descriptor
Send : 7c        Recv : 00
Send : 02        Recv : 58
Send : 10        Recv : 58
Send : 00        Recv : 58
Send : 59        Recv : 58     ....
Send : 00        Recv : 58     .... First 12 bits of 0th word of the descriptor are the no. of
bytes that we are sending
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 02        Dummy Recv : 58
Send : 00        Dummy Recv : 58
send payload
Send : 7c        Recv : 00
Send : 02        Recv : 58
Send : 5c        Recv : 58
Send : 00        Recv : 58
Send : 00        Recv : 58     -----------------------------------
Send : 00        Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58  sending data
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58
Send : 00        Dummy Recv : 58------------------------------------

All bytes of sending data are not actual. These are for illustrative purposes only
---------------------------------------------------------------------------
```

8 Recent Revision History:

Rev.	Ver. No.	Date	Changes
	4.00	10 Dec, 2012	This version is built on top of 3.31
	4.10	31 Dec 2013	Modified description about at+rsi_wep_keys command Modified Bit[9] description for at+rsi_feat_sel command Changed description for at+rsi_cfgget? command response .Added wep_index wep_keys description.
	4.11	4 Jan 2013	Added Multicast details, Changes in the PSK to 63 Bytes, Firmware upgrade hex bytes,
	4.12	8 Jan 2013	Added new paragraph to firmware upgrade procedure
	4.13	12 Jan,2013	Number of sockets is decreased and the same is modified in the note. Description is modified for at+rsi_nwtype? And at+rsi_bssid? Commands. Added description for error code 0x9C in the error table.
	4.14	18 Jan 2013	Modified descrption for http response. Updated section Firmware upgrade and boot loading section in SPI. Added new error codes.
	4.15	23 Jan 2013	Removed 24-04 reference since not released yet.