**RW610 NCP User Manual**

**NXP Semiconductors.**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Authors | Revision | Description |
| 06/21/2022 | Qiankun Li | 1.0 | Initial draft. |
| 02/15/2023 | Hui Bai | 1.1 | Add wlan-roaming command. |

Table of Contents

[1. RW610 UART 4](#_Toc127366500)

[1.1. RW610 --- RW610(FC0 --- FC0) 4](#_Toc127366501)

[1.2. RW610 --- PC 4](#_Toc127366502)

[2. The format of command and response 5](#_Toc127366503)

[3. Command Reference – WLAN Commands 6](#_Toc127366507)

[3.1. Command Set 6](#_Toc127366510)

[3.2. WLAN Commands 6](#_Toc127366511)

[3.2.1. wlan-scan 6](#_Toc127366517)

[3.2.2. wlan-version 8](#_Toc127366518)

[3.2.3. wlan-roaming 9](#_Toc127366519)

[3.3. wlan-connect 10](#_Toc127366520)

[3.3.1. Usage 10](#_Toc127366522)

[3.3.2. Command Format 10](#_Toc127366523)

[3.4. get-connect-result 11](#_Toc127366524)

[3.4.1. Usage 11](#_Toc127366526)

[3.4.2. Command Format 11](#_Toc127366527)

[3.4.3. Response Format 11](#_Toc127366528)

[3.5. wlan-disconnect 13](#_Toc127366529)

[3.5.1. Usage 13](#_Toc127366531)

[3.5.2. Command Format 13](#_Toc127366532)

[3.5.3. Response Format 13](#_Toc127366533)

[3.6. ping 14](#_Toc127366534)

[3.6.1. Usage 14](#_Toc127366536)

[3.6.2. Command Format 14](#_Toc127366537)

[3.7. get-ping-result 15](#_Toc127366538)

[3.7.1. Usage 15](#_Toc127366540)

[3.7.2. Command Format 15](#_Toc127366541)

[3.7.3. Response Format 15](#_Toc127366542)

[3.8. iperf 16](#_Toc127366543)

[3.8.1. Usage 16](#_Toc127366545)

[3.8.2. Command Format 17](#_Toc127366546)

[3.9. get-iperf-result 18](#_Toc127366547)

[3.9.1. Usage 18](#_Toc127366549)

[3.9.2. Command Format 18](#_Toc127366550)

[3.9.3. Response Format 19](#_Toc127366551)

1. RW610 UART

|  |  |  |
| --- | --- | --- |
| **Board** | **FC0 RX/TX** | **FC0 CTS/RTS** |
| BGA | **Y**  Option 1: Remove R437  Option 2: Connect JP5 to GND, unload JP23, load JP9 | **Y**   1. Remove R437 |
| QFN | **Y**   1. Connect JP30 to GND\* | **Not support**   1. GPIO0 not supported in QFN |
| CSP | **Y**   1. Connect JP30 to GND\* | **Not work** |

*\*Note: Connecting JP30 to GND only needs to be GND once after power-on, and then it can be removed.*

* 1. RW610 --- RW610(FC0 --- FC0)

The connection is as follows:

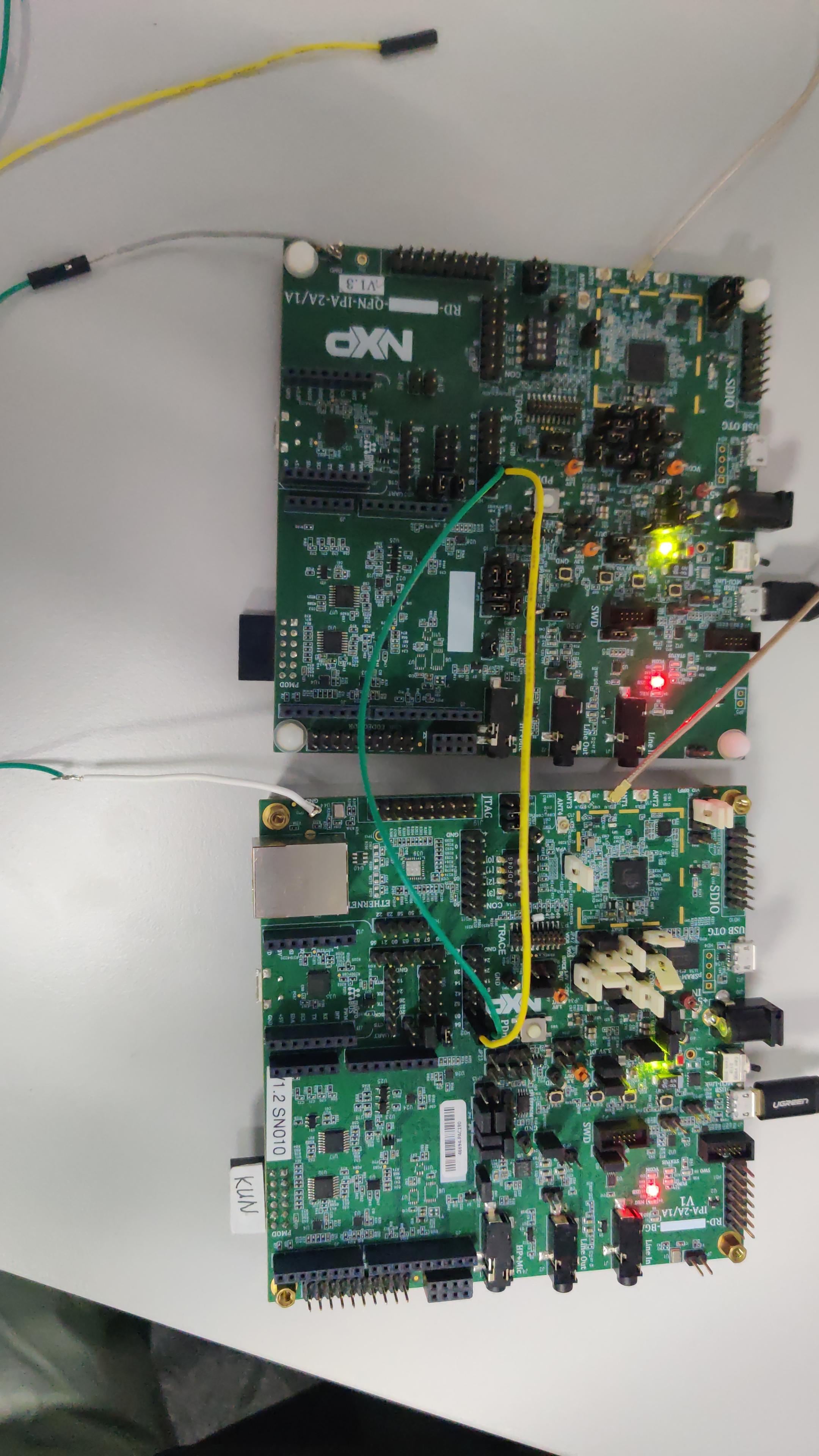
**FC0 RX/TX:**

**RW610 board1 RW610 board2**

HD2 2(FC0\_UART\_RXD) --- HD2 3(FC0\_UART\_TXD)

HD2 3(FC0\_UART\_TXD) --- HD2 2(FC0\_UART\_RXD)

HD2 GND --- HD2 GND



* 1. RW610 --- PC

The connection is as follows:

**FC0 RX/TX:**

HD2 2(FC0\_UART\_RXD) --- HD3(RX)

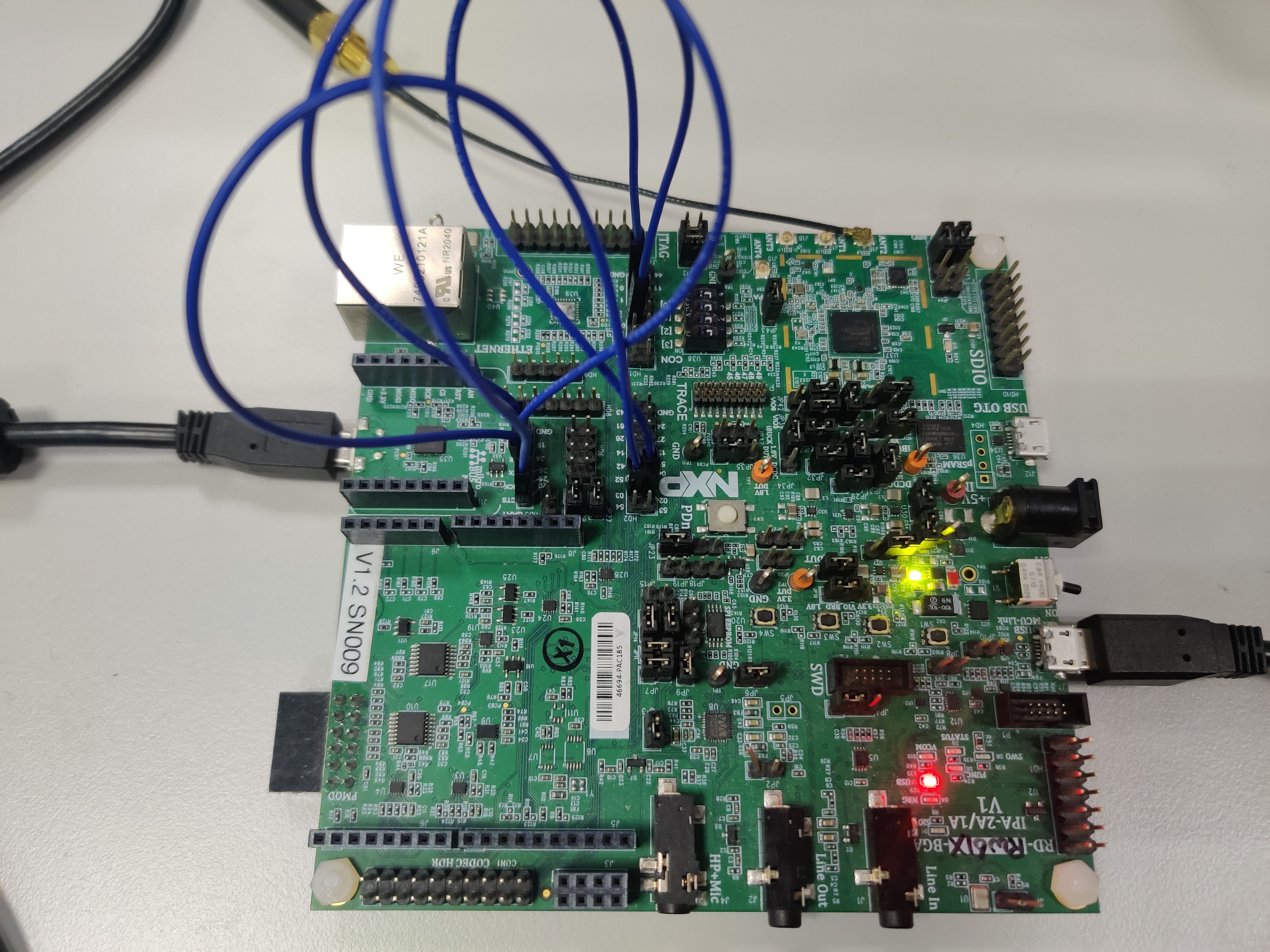
HD2 3(FC0\_UART\_TXD) --- HD3(TX)

**FC0 RTS/CTS: (Only for BGA board)**

HD11 0(FC0\_UART\_CTS) --- HD3(CTS)

HD11 5(FC0\_UART\_RTS) --- HD3(RTS)

And use a usb cable to connect the left uart(FC0) and PC in the figure below.



1. The format of command and response

The command format looks as follows:

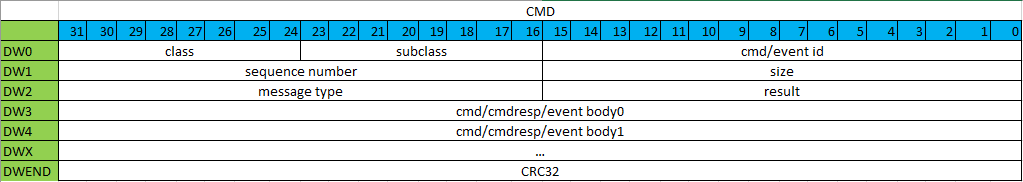


Figure 1. Command format

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| class | U8 | command class: Wi-Fi/BLE/15.4/Matter |
| subclass | U8 | command subclass |
| cmd/event id | U16 | command/event id |
| size | U16 | total bytes of command header and tlv |
| sequence number | U16 | command sequence number |
| result | U16 | Success/Fail/… |
| message type | U16 | 0, cmd; 1, cmd response; 2, event |
| cmd/cmdresp/event body | variable | / |
| CRC32 | U32 | command checksum |

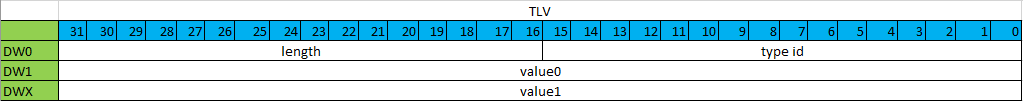


Figure 2. TLV format

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| type id | U16 | TLV type id |
| length | U16 | total bytes of the TLV header and body |
| value | variable |  |

1. Command Reference – WLAN Commands
3. 1. Command Set

|  |  |
| --- | --- |
| Instruction | Description |
| WLAN Commands |  |
| wlan-scan | WLAN scan |
| wlan-disconnect | Disconnect from network |
| wlan-version | Read firmware version |
| wlan-set-mac | Set STA and uAP mac address |
| wlan-get-mac | Get STA and uAP mac address |
| wlan-stat | Get WLAN connection status |
| wlan-info | Get the configured WLAN network information |

* 1. WLAN Commands

3. 2. 1. wlan-scan
         1. Usage

wlan-scan

* + - 1. Description

This command schedules a new scan and returns scan results of the available wlan networks.

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0001 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_SCAN |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 | NCP\_BRIDGE\_CMD\_RESULT\_OK |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_CMD |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0001 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_SCAN |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 if success  0x1 if fail | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_MPU\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| command response  body | NCP\_CMD\_SCAN  \_NETWORK\_INFO | sizeof(NCP\_CMD\_SCAN  \_NETWORK\_INFO) | scan results |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_RESP |
| CRC32 | U32 | variable | Command checksum |

**NCP\_CMD\_SCAN\_NETWORK\_INFO**

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| res\_cnt | uint8\_t | total number of scan results |
| res[30] | wlan\_bridge\_scan\_result | scan result |

wlan\_bridge\_scan\_result

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| ssid[33] | uint8\_t | the network ssid |
| ssid\_len | unsigned int | ssid length |
| bssid[6] | char | the network bssid |
| channel | unsigned int | the network channel |
| wmm:1 | unsigned | the network supports WMM |
| wps:1 | unsigned | the network supports WPS  (if defined CONFIG\_WPS2) |
| wps\_session | unsigned int | WPS Type PBS/PIN  (if defined CONFIG\_WPS2) |
| wpa2\_entp:1 | unsigned | WPA2 enterprise security |
| wep:1 | unsigned | WEP security |
| wpa:1 | unsigned | WPA security |
| wpa2:1 | unsigned | WPA2 security |
| wpa3\_sae:1 | unsigned | WPA3 SAE security |
| rssi | unsigned char | the signal strength of the beacon |
| trans\_ssid[33] | char | the network trans\_ssid |
| trans\_ssid\_len | unsigned int | trans\_ssid length |
| trans\_bssid[6] | char | the network trans\_bssid |
| beacon\_period | uint16\_t | beacon period |
| dtim\_period | uint8\_t | DTIM period |

* + - 1. Example

#wlan-scan

4 networks found

E0:40:07:4A:45:58 [hw-2g-9177]

channel: 1

rssi: -39 dBm

security: OPEN

WMM: YES

8C:A6:DF:B2:76:99 [huibai-2g]

channel: 1

rssi: -33 dBm

security: OPEN

WMM: YES

0C:9D:92:02:2A:98 [000\_ASUS\_BS\_wpa3]

channel: 1

rssi: -52 dBm

security: OPEN

WMM: YES

68:77:24:03:A4:86 [TPAX2G]

channel: 1

rssi: -56 dBm

security: OPEN

WMM: YES

* + 1. wlan-version
       1. Usage

wlan-version

* + - 1. Description

This command returns the Wi-Fi driver and firmware version. The command can be issued anytime once NCP bridge boots up successfully.

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0004 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_VERSION |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 | NCP\_BRIDGE\_CMD\_RESULT\_OK |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_CMD |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0004 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_VERSION |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 if success  0x1 if fail | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_MPU\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| command response body | NCP\_CMD\_FW\_VERSION | sizeof(NCP\_CMD\_FW\_VERSION) | driver and firmware version |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_RESP |
| CRC32 | U32 | variable | Command checksum |

**NCP\_CMD\_FW\_VERSION**

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| driver\_ver\_str[16] | char | Driver version |
| fw\_ver\_str[128] | char | Firmware version |

* + - 1. Example

#wlan-version

WLAN Driver Version :v1.3.r34.p40

WLAN Firmware Version :rw610w-V0, RF878X, FP91, 18.91.1.p175, PVE\_FIX 1, RF878X, FP91, 18.91.1.p175

* + 1. wlan-roaming
       1. Usage

wlan-roaming <enable> <rssi\_threshold>

* + - 1. Description

This command is used to enable or disable roaming and configure RSSI threshold value to firmware. It will take effect after connecting to Ext-AP.

* + - 1. Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Description | Values | Mandatory |
| 1 | enable | 0 – disable roaming  1 – enable roaming | YES |
| 2 | rssi\_threshold | Default value is 70 if not specified(absolute value). This setting is valid only when enable = 1. | NO |

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| Subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| Command ID | U16 | 0x0008 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_ROAMING |
| Size | U16 | variable | Total length of command buffer except for CRC |
| Sequence Number | U16 | 0x0 | Command sequence number |
| Result | U16 | 0x0 | NCP\_BRIDGE\_CMD\_RESULT\_OK |
| Message Type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYEP\_CMD |
| Command Body | MPU\_NCP\_CMD\_ROAMING | sizeof(MPU\_NCP\_CMD\_ROAMING) | Roaming configuration. |
| CRC32 | U32 | variable | Command checksum |

**MPU\_NCP\_CMD\_ROAMING**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| enable | U32 | 0 – disable roaming  1 – enable roaming |
| rssi\_threshold | U32 | Rssi threshold. If RSSI is lower than this value, firmware will report RSSI\_LOW event to host. |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| Subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| Command ID | U16 | 0x0008 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_ROAMING |
| Size | U16 | variable | Total length of command response buffer except for CRC |
| Sequence Number | U16 | 0x0 | Command sequence number |
| Result | U16 | 0x0 if success  0x1 if fail | NCP\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| Message Type | U16 | 0x1 | NCP\_BRIDGE\_MSG\_TYPE\_RESP |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Example

# wlan-roaming 1 50 -- Enable roaming and set rssi threshold to 50dBm

# wlan-roaming 0 -- Disable roaming

* + 1. wlan-reset
       1. wlan-reset
       2. Usage

wlan-reset action

* + - 1. Description

This command is used to enable, disable and reset wlan driver.

* + - 1. Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Description | Values | Mandatory |
| 1 | action | 0 – disable wlan  1 – enable wlan  2 – reset wlan | YES |

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Value | Description |
| class | U8 | 0x0 | NCP\_MPU\_BRIDGE\_CMD\_WLAN |
| Subclass | U8 | 0x1 | NCP\_MPU\_BRIDGE\_CMD\_WLAN\_BASIC |
| Command ID | U16 | 0x00000001 | NCP\_MPU\_BRIDGE\_WLAN\_BASIC\_WLAN\_RESET |
| Size | U16 | variable | Total length of command buffer except for CRC |
| Sequence Number | U16 | 0x0 | Command sequence number |
| Result | U16 | 0x0 | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK |
| Message Type | U16 | 0x0 | NCP\_MPU\_BRIDGE\_MSG\_TYEP\_CMD |
| Command Body | MPU\_NCP\_CMD\_WLAN\_RESET\_CFG | sizeof(MPU\_NCP\_CMD\_WLAN\_RESET\_CFG) | Wlan reset. |
| CRC32 | U32 | variable | Command checksum |

**MPU\_NCP\_CMD\_WLAN\_RESET\_CFG**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| action | U32 | 0 – disable wlan  1 – enable wlan  2 – reset wlan |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Value | Description |
| class | U8 | 0x0 | NCP\_MPU\_BRIDGE\_CMD\_WLAN |
| Subclass | U8 | 0x1 | NCP\_MPU\_BRIDGE\_CMD\_WLAN\_BASIC |
| Command ID | U16 | 0x00000001 | NCP\_MPU\_BRIDGE\_WLAN\_BASIC\_WLAN\_RESET |
| Size | U16 | variable | Total length of command response buffer except for CRC |
| Sequence Number | U16 | 0x0 | Command sequence number |
| Result | U16 | 0x0 if success  0x1 if fail | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_MPU\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| Message Type | U16 | 0x1 | NCP\_MPU\_BRIDGE\_MSG\_TYPE\_RESP |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Example

# wlan-reset 0 – Disable the wlan

External host console result:

wlan-reset 0

Sent command successfully!

wlan\_reset is ok!

# wlan-reset 1 – Enable the wlan

wlan-reset 1

Sent command successfully!

wlan\_reset is ok!

# wlan-reset 2 – Reset the wlan

wlan-reset 2

Sent command successfully!

wlan\_reset is ok!

* + 1. wlan-disconnect
       1. Usage

wlan-disconnect

* + - 1. Description

This command disconnects from a WLAN network it is connected to. The connection status can be checked using “wlan-stat” command.

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0003 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_DISCONNECT |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 | NCP\_BRIDGE\_CMD\_RESULT\_OK |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_CMD |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0003 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_DISCONNECT |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 if success  0x1 if fail | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_MPU\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_RESP |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Example

#wlan-disconnect -- Disconnect from the current network it is connected to.

* + 1. wlan-stat
       1. Usage

wlan-stat

* + - 1. Description

This command provides the WLAN connection status.

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0007 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_ CONNECT\_STAT |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 | NCP\_BRIDGE\_CMD\_RESULT\_OK |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_CMD |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0007 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_ CONNECT\_STAT |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 if success  0x1 if fail | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_MPU\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_RESP |
| command response  body | NCP\_CMD\_  CONNECT\_STAT | sizeof(NCP\_CMD\_  CONNECT\_STAT) | wlan connnection state |
| CRC32 | U32 | variable | Command checksum |

**NCP\_CMD\_CONNECT\_STAT**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| ps\_mode | U8 | station power save mode |
| uap\_conn\_stat | U8 | WLAN uAP connection status |
| sta\_conn\_stat | U8 | WLAN station connection status |

* + - 1. Example

#wlan-stat

Station disconnected (Active)

uAP stopped

* + 1. wlan-info
       1. Usage

wlan-info

* + - 1. Description

This command returns the configured WLAN network information.

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_NETWORK |
| command id | U16 | 0x0001 | NCP\_BRIDGE\_CMD\_WLAN\_NETWORK\_INFO |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 | NCP\_BRIDGE\_CMD\_RESULT\_OK |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_CMD |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_NETWORK |
| command id | U16 | 0x0001 | NCP\_BRIDGE\_CMD\_WLAN\_NETWORK\_INFO |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 if success  0x1 if fail | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_MPU\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_RESP |
| command response  body | NCP\_CMD\_  NETWORK\_INFO | sizeof(NCP\_CMD\_  NETWORK\_INFO) | wlan network info |
| CRC32 | U32 | variable | Command checksum |

**NCP\_CMD\_NETWORK\_INFO**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| uap\_conn\_stat | U8 | WLAN uAP connection status |
| sta\_conn\_stat | U8 | WLAN station connection status |
| uap\_network | wlan\_bridge\_network | WLAN uAP Network Profile |
| sta\_network | wlan\_bridge\_network | WLAN station Network Profile |

**wlan\_bridge\_network**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| name[32] | char | The name of this network profile. |
| ssid[33] | char | ssid of the AP to which NCP Bridge is connected |
| bssid[6] | char | bssid of the AP to which NCP Bridge is connected |
| channel | unsigned int | channel no. of the AP to which NCP Bridge is connected |
| acs\_band | uint16\_t | The ACS band if set channel to 0. |
| rssi\_threshold | short | Rssi threshold(if defined  CONFIG\_SCAN\_WITH\_RSSIFILTER) |
| type | uint8\_t | BSS type |
| role | uint8\_t | Set this to specify what type of wlan network mode to use. (uAP or STA) |
| security\_type | uint8\_t | Type of network security to use specified |
| enable\_11ax:1 | uint8\_t | enable 802.11ax |
| enable\_11ac:1 | uint8\_t | enable 802.11ac |
| enable\_11n:1 | uint8\_t | enable 802.11n |
| ipv6[3] | wlan\_bridge\_ipv6\_config | The network IPv6 address configuration (if defined CONFIG\_IPV6) |
| ipv4 | wlan\_bridge\_ipv4\_config | The network IPv4 address configuration |
| is\_sta\_ipv4\_connected | uint8\_t | whether ipv4 is connected |
| identity[256] | char | (if defined CONFIG\_WPA2\_ENTP) |
| ssid\_specific:1 | unsigned | If set to 1, the ssid field contains the specific SSID for this network. |
| bssid\_specific:1 | unsigned | If set to 1, the bssid field contains the specific BSSID for this network. |
| channel\_specific | unsigned | If set to 1, the channel field contains the specific channel for this network. |
| security\_specific | unsigned | If set to 0, any security that matches is used. |
| wps\_specific | unsigned | This indicates this network is used as an internal network for WPS. (if defined CONFIG\_WPS2) |
| beacon\_period | uint16\_t | Beacon period of associated BSS |
| dtim\_period | uint8\_t | DTIM period of associated BSS |
| wlan\_capa | uint8\_t | (if defined CONFIG\_WIFI\_CAPA) |

**wlan\_bridge\_ipv6\_config**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| address[4] | unsigned | The system's IPv6 address in network order. |
| addr\_type\_str[16] | unsigned char | The address type string: linklocal, site-local or global. |
| addr\_state\_str[32] | unsigned char | The state string of IPv6 address (Tentative, Preferred, etc). |

**wlan\_bridge\_ipv4\_config**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| addr\_type:2 | uint32\_t | use DHCP to obtain the IP address or to use a static IP. |
| address | uint32\_t | The system's IP address in network order. |
| gw | uint32\_t | The system's default gateway in network order. |
| netmask | uint32\_t | The system's subnet mask in network order. |
| dns1 | uint32\_t | The system's primary dns server in network order. |
| dns2 | uint32\_t | The system's secondary dns server in network order. |

* + - 1. Example

#wlan-info

Station not connected

uAP started as:

"testuAP"

SSID: ncp

BSSID: 00:50:43:99:FF:11

channel: 1

role: uAP

security: WPA3 SAE

wifi capability: 11ax

user configure: 11ax

IPv4 Address

address: STATIC

IP: 192.168.1.1

gateway: 192.168.1.1

netmask: 255.255.255.0

dns1: 192.168.1.1

dns2: 0.0.0.0

IPv6 Addresses

Link-Local : fe80::250:43ff:fe99:ff11 (Tentative)

rssi threshold: 0

* + 1. wlan-set-mac
       1. Usage

wlan-set-mac <mac\_address>

* + - 1. Description

This command sets the MAC address.

* + - 1. Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Description | Values | Mandatory |
| 1 | mac\_address | xx:xx:xx:xx:xx:xx | YES |

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0005 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_ SET\_MAC |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 | NCP\_BRIDGE\_CMD\_RESULT\_OK |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_CMD |
| command body | NCP\_CMD\_MAC  \_ADDRESS | sizeof(NCP\_CMD\_  MAC\_ADDRESS) | MAC address |
| CRC32 | U32 | variable | Command checksum |

**NCP\_CMD\_MAC\_ADDRESS**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| mac\_addr[6] | uint8\_t | MAC address |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0005 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_ SET\_MAC |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 if success  0x1 if fail | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_MPU\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_RESP |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Example

#wlan-set-mac 00:50:43:99:FE:11 --- Set MAC address to “00:50:43:99:FE:11”.

* + 1. wlan-get-mac
       1. Usage

wlan-get-mac

* + - 1. Description

This command gets the MAC address of station and uAP.

* + - 1. Command Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0006 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_GET\_MAC |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 | NCP\_BRIDGE\_CMD\_RESULT\_OK |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_CMD |
| CRC32 | U32 | variable | Command checksum |

* + - 1. Response Format

|  |  |  |  |
| --- | --- | --- | --- |
| Filed Name | Type | Value | Description |
| class | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN |
| subclass | U8 | 0x0 | NCP\_BRIDGE\_CMD\_WLAN\_STA |
| command id | U16 | 0x0005 | NCP\_BRIDGE\_CMD\_WLAN\_STA\_ SET\_MAC |
| size | U16 | variable | total length of command buffer except for CRC |
| sequence number | U16 | 0x0 | command sequence number |
| result | U16 | 0x0 if success  0x1 if fail | NCP\_MPU\_BRIDGE\_CMD\_RESULT\_OK if success  NCP\_MPU\_BRIDGE\_CMD\_RESULT\_ERROR if failed |
| message type | U16 | 0x0 | NCP\_BRIDGE\_MSG\_TYPE\_RESP |
| command response  body | NCP\_CMD\_GET\_  MAC\_ADDRESS | sizeof(NCP\_CMD\_GET  \_MAC\_ADDRESS) | Get MAC address |
| CRC32 | U32 | variable | Command checksum |

**NCP\_CMD\_GET\_MAC\_ADDRESS**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| uap\_mac [6] | uint8\_t | uAP MAC address |
| sta\_mac [6] | uint8\_t | Station MAC address |

* + - 1. Example

#wlan-get-mac

MAC Address

STA MAC Address: 00:50:43:99:FE:11

UAP MAC Address: 00:50:43:99:FF:11

* 1. wlan-connect
  2. 1. Usage

wlan-connect <ssid>

(Only supports connecting to unencrypted APs.)

* + 1. Command Format

wlan\_connect\_command(char \*arg)(wifi\_bridge\_command.c) processes ‘wlan-connect <ssid>’ command and converts it the specified command format.

Para arg points to the ssid to be connected. The characters in ssid are converted to corresponding ASCII.

typedef struct SSID\_ParaSet

{

TypeHeader\_t header;

uint8\_t ssid[1];

}SSID\_tlv;

SSID\_tlv ssid\_tlv;

ssid\_tlv.header.type = 0x02;

ssid\_tlv.header.size = strlen(ssid);

ssid\_tlv.ssid = the ASCII of ssid(char\*);

0 8 16 24 31

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Command code | Size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| SeqNum | BSS | Result |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Action | type | size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x02 |
| Size | UNIT16 | BRIDGE\_COMMAND\_LEN + TLV\_HEADER\_LEN + ssid\_tlv.header.size |
| SeqNum | UINT8 | 0x00 |
| BSS | UINT8 | 0x00 |
| Result | UINT16 | 0x00 |
| Action | UINT8 | 0x01 |
| SSID\_Tlv | UINT8 | ssid tlv |

Function wlan\_bridge\_connect(t\_u8 \*cmd)(MCU bridge\_app) processes the connect command sent by LINUX APP. Para cmd points to the received command.

* 1. get-connect-result
  2. 1. Usage

get-connect-result

Use this command to get the connected network info. Because it takes some time to connect AP, it’s best to wait at least 10s to get network info after entering ‘wlan-connect <ssid>’.

* + 1. Command Format

wlan\_get\_connect\_res\_command()(wifi\_bridge\_command.c) processes this command and converts it the specified command format.

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x02 |
| Size | UINT16 | BRIDGE\_COMMAND\_LEN |
| SeqNum | UINT8 | 0x00 |
| BSS | UINT8 | 0x00 |
| Result | UINT16 | 0x00 |
| Action | UINT8 | 0x00 |

* + 1. Response Format

wlan\_bridge\_get\_connect()(MCU bridge\_app) processes the command and returns the connected network info to the LINUX APP in the specified command format.

typedef struct CONNECT\_RES

{

    TypeHeader\_t header;

    uint8\_t ip[16];

    uint8\_t ssid[32];

}CONNECT\_res\_tlv;

#define BRIDGE\_CONNECT\_RES\_TLV\_LEN 48

The format of connected result response:

0 8 16 24 31

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Command code | Size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| SeqNum | BSS | Result |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Action | type | size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ip address |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ip address |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ip address |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ip address |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid |

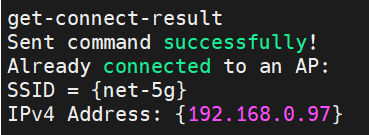
|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ssid |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x02 |
| Size | UINT16 | BRIDGE\_COMMAND\_LEN + TLV\_HEADER\_LEN + BRIDGE\_CONNECT\_RES\_TLV\_LEN |
| SeqNum | UINT8 | 0x00 |
| BSS | UINT8 | 0x00 |
| Result | UINT16 | 0:Success !0:Failed |
| Action | UINT8 | 0x00 |
| SSID\_res\_tlv | UINT8 | IPv4 address ssid |

Function wlan\_process\_con\_response()(wifi\_bridge\_command.c) processes the response sent by bridge\_app.



* 1. ping
  2. 1. Usage

ping [c <packet\_count>] <ip address>

* + 1. Command Format

wlan\_ping\_command()(wifi\_bridge\_command.c) processes this command and converts it the specified command format.

typedef struct PING\_ParaSet

{

TypeHeader\_t header;

uint16\_t packet\_count;

uint8\_t ping\_ip[1];

}PING\_tlv;

PING\_tlv ping\_tlv;

ping\_tlv.header.type = 0x04;

ping\_tlv.header.size = strlen(ping\_ip) + sizeof(packet\_count);

ping\_tlv.packet\_count = the count of ping packets;

ping\_tlv.ping\_ip = <IP Address>

0 8 16 24 31

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Command code | Size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| SeqNum | BSS | Result |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Action | type | size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| packet\_count | ping\_ip

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x04 |
| Size | UNIT16 | BRIDGE\_COMMAND\_LEN + TLV\_HEADER\_LEN + ping\_tlv.header.size |
| SeqNum | UINT8 | 0X00 |
| BSS | UINT8 | 0X00 |
| Result | UINT16 | 0X00 |
| Action | UINT8 | 0X01 |
| PING\_Tlv | UINT8 | The tlv of ping parameters |

Function wlan\_bridge\_ping(t\_u8 \*cmd)(MCU bridge\_app) processes ‘ping’ command sent by LINUX APP. Para cmd points to the received command.

* 1. get-ping-result
  2. 1. Usage

get-ping-result

Use this command to get the ping statistics info. Because it takes some time to ping, it’s best to wait at least 10s(If the packet count is 10.) to get ping info after entering ‘ping <ip address>’.

* + 1. Command Format

wlan\_get\_ping\_res\_command()(wifi\_bridge\_command.c) processes ‘get-ping-result’ command and converts it the specified command format.

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x04 |
| Size | UINT16 | BRIDGE\_COMMAND\_LEN |
| SeqNum | UINT8 | 0X00 |
| BSS | UINT8 | 0X00 |
| Result | UINT16 | 0X00 |
| Action | UINT16 | 0X00 |

* + 1. Response Format

wlan\_bridge\_get\_ping()(MCU bridge\_app) processes the command and returns the ping statistics info to the LINUX APP in the specified command format.

typedef struct PING\_RES

{

    TypeHeader\_t header;

    uint8\_t status;

    uint32\_t packet\_transmit;

    uint32\_t packet\_received;

}PING\_res\_tlv;

#define PING\_RES\_INFO 9

0 8 16 24 31

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Command code | Size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| SeqNum | BSS | Result |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Action | type | size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| status | packet\_trans |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|packet\_trans | packet\_recv |

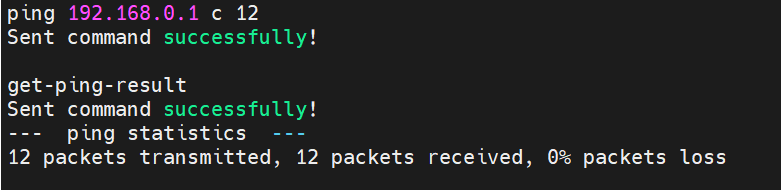
|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| packet\_recv |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x04 |
| Size | UNIT16 | BRIDGE\_COMMAND\_LEN + TLV\_HEADER\_LEN + PING\_RES\_INFO |
| SeqNum | UINT8 | 0X00 |
| BSS | UINT8 | 0X00 |
| Result | UINT16 | 0:Success !0:Failed |
| Action | UINT8 | 0X00 |
| PING\_Tlv | UINT8 | The tlv of ping parameters |

Function wlan\_process\_ping\_response()(wifi\_bridge\_command.c) processes the response sent by bridge\_app.



* 1. iperf
  2. 1. Usage

iperf s TCP Server

iperf su UDP Server

iperf c <IP address> t <time> TCP Client <time is optional, the default time is 10s>

iperf cu <IP address> t <time> UDP Client <time is optional, the default time is 10s>

iperf a Abort current iperf mode

* + 1. Command Format

wlan\_iperf\_command()(wifi\_bridge\_command.c) processes ‘iperf’ command and converts it the specified command format.

typedef struct IPERF\_ParaSet

{

TypeHeader\_t header;

uint16\_t time;

uint8\_t iperf\_ip [1];

}IPERF\_tlv;

IPERF\_tlv iperf\_tlv;

TCP Server:

iperf\_tlv.header.type = 0x51;

iperf\_tlv.header.size = 0;

UDP Server:

iperf\_tlv.header.type = 0x52;

iperf\_tlv.header.size = 0;

TCP Client:

iperf\_tlv.header.type = 0x53;

iperf\_tlv.header.size = strlen(<IP Address>) + sizeof(iperf\_tlv.time);

iperf\_tlv.time = the time of throughput test(default time is 10s);

iperf\_tlv.iperf\_ip = <IP Address>

UDP Client:

Iperf\_tlv.header.type = 0x54;

iperf\_tlv.header.size = strlen(<IP Address>) + sizeoof(iperf\_tlv.time);

iperf\_tlv.time = the time of throughput test(default time is 10s);

iperf\_tlv.iperf\_ip = <IP Address>

Iperf Abort

perf\_tlv.header.type = 0x55;

iperf\_tlv.header.size = 0;

0 8 16 24 31

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Command code | Size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-+

| SeqNum | BSS | Result |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Action | type | size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| time | iperf\_ip

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x05 |
| Size | UNIT16 | BRIDGE\_COMMAND\_LEN + TLV\_HEADER\_LEN + iperf\_tlv.header.size |
| SeqNum | UINT8 | 0x00 |
| BSS | UINT8 | 0x00 |
| Result | UINT16 | 0x00 |
| Action | UINT8 | 0x01 |
| IPERF\_Tlv | UINT8 | iperf parameter |

Function wlan\_bridge\_iperf(t\_u8 \*cmd)(MCU bridge\_app) processes ‘iperf’ command sent by LINUX APP. Para cmd points to the ‘iperf’ command.

* 1. get-iperf-result
  2. 1. Usage

get-iperf-result

Use this command to get the iperf statistics info. Because it takes some time to test throughput, it’s best to wait until the throughput test is over.

* + 1. Command Format

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x05 |
| Size | UINT16 | BRIDGE\_COMMAND\_LEN |
| SeqNum | UINT8 | 0x00 |
| BSS | UINT8 | 0x00 |
| Result | UINT16 | 0x00 |
| Action | UINT16 | 0x00 |

wlan\_bridge\_get\_iperf()(MCU bridge\_app) processes the command and returns ‘iperf’ statistics info to the LINUX APP in the specified command format.

* + 1. Response Format

typedef struct IPERF\_INFO

{

    uint8\_t   role;

    uint8\_t   local\_addr[4];

    uint16\_t local\_port;

    uint8\_t   remote\_addr[4];

    uint16\_t remote\_port;

    uint64\_t bytes\_transferred;

    uint32\_t ms\_duration;

    uint32\_t bandwidth\_Mbitpsec;

}IPERF\_info;

typedef struct IPERF\_RES

{

    TypeHeader\_t header;

    IPERF\_info iperf\_info;

}IPERF\_res\_tlv;

#define IPERF\_INFO\_LEN 29

0 8 16 24 31

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Command code | Size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| SeqNum | BSS | Result |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| Action | type | size |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| role | local\_addr |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| local\_addr | local \_port | remote\_addr |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| remote\_addr | remote\_port |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| remote\_port | bytes\_transfer |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| bytes\_transfer |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|bytes\_transfer| ms\_duration |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

| ms\_duration | bandwidth\_Mp |

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|bandwidth\_Mp|

|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|-+-+-+-+-+-+-+-|

|  |  |  |
| --- | --- | --- |
| Filed Name | Type | Description |
| Command Code | UINT16 | 0x05 |
| Size | UNIT16 | BRIDGE\_COMMAND\_LEN + TLV\_HEADER\_LEN + IPERF\_INFO\_LEN |
| SeqNum | UINT8 | 0x00 |
| BSS | UINT8 | 0x00 |
| Result | UINT16 | 0x00 |
| Action | UINT8 | 0x00 |
| IPERF\_res\_tlv | UINT8 | iperf throughput test info |

Function wlan\_process\_iperf\_response()(wifi\_bridge\_command.c) processes ‘iperf’ response sent by bridge\_app.

