NEW RADIO COMMUNICATION FOR NEW ERA

NRC7394 Evaluation Kit User Guide

(AT-command)

Ultra-low power & Long-range Wi-Fi

Ver 1.1
Aug. 16, 2023

NEWRACOM, Inc.

NRC7394 Evaluation Kit User Guide (AT-command)
Ultra-low power & Long-range Wi-Fi

Office
Newracom, Inc.
505 Technology Drive, Irvine, CA 92618 USA
http://www.newracom.com

Contents

List of Tables

List of Figures

1 Overview

This document introduces the NRC7394 AT-command. The NRC7394 AT-command allows users to apply fine controls over the NRC7394 modules such as: checking the modem status, scanning, connecting to an AP, opening sockets, and exchanging data.

2 Basic Setup

2.1 Hardware

The AT-command communication is achieved via the UART or SPI interface between the NRC7394 and an external host.

Figure 2.1 shows the NRC7394 Evaluation Board (EVB). Figure 2.1 shows the NRC7394 Evaluation Kit (EVK) using a Raspberry Pi 4 model B as host.

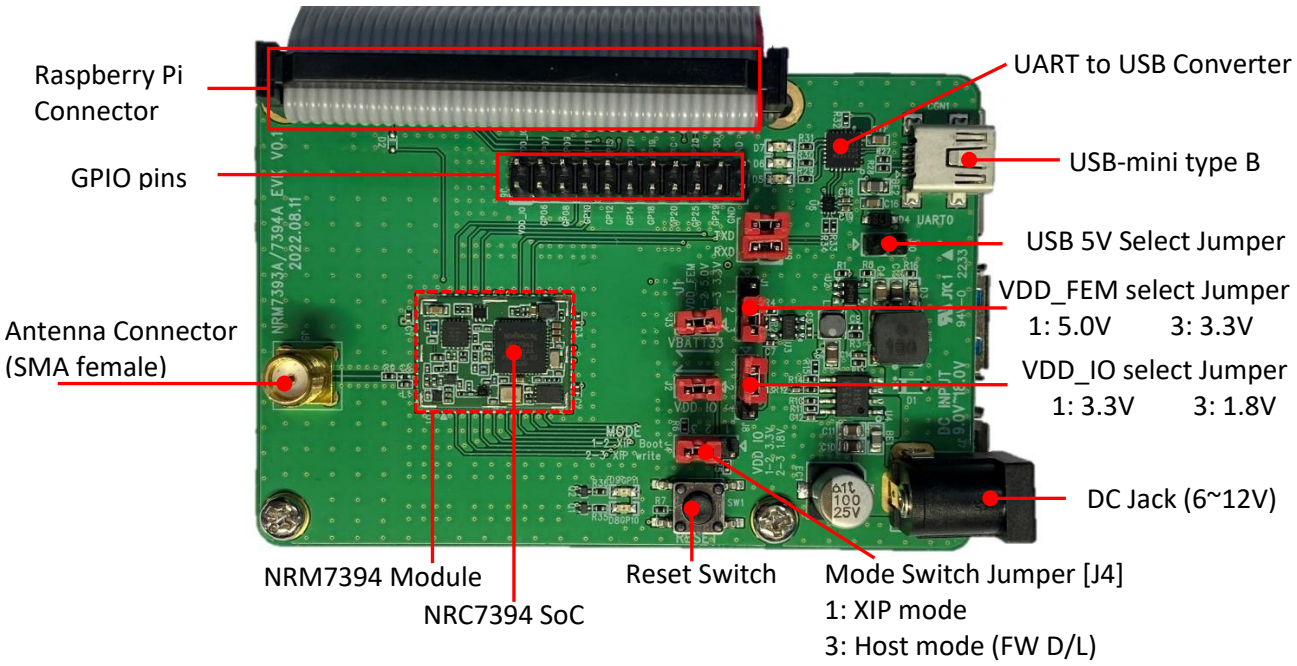


Figure 2.1 NRC7394 Evaluation Board

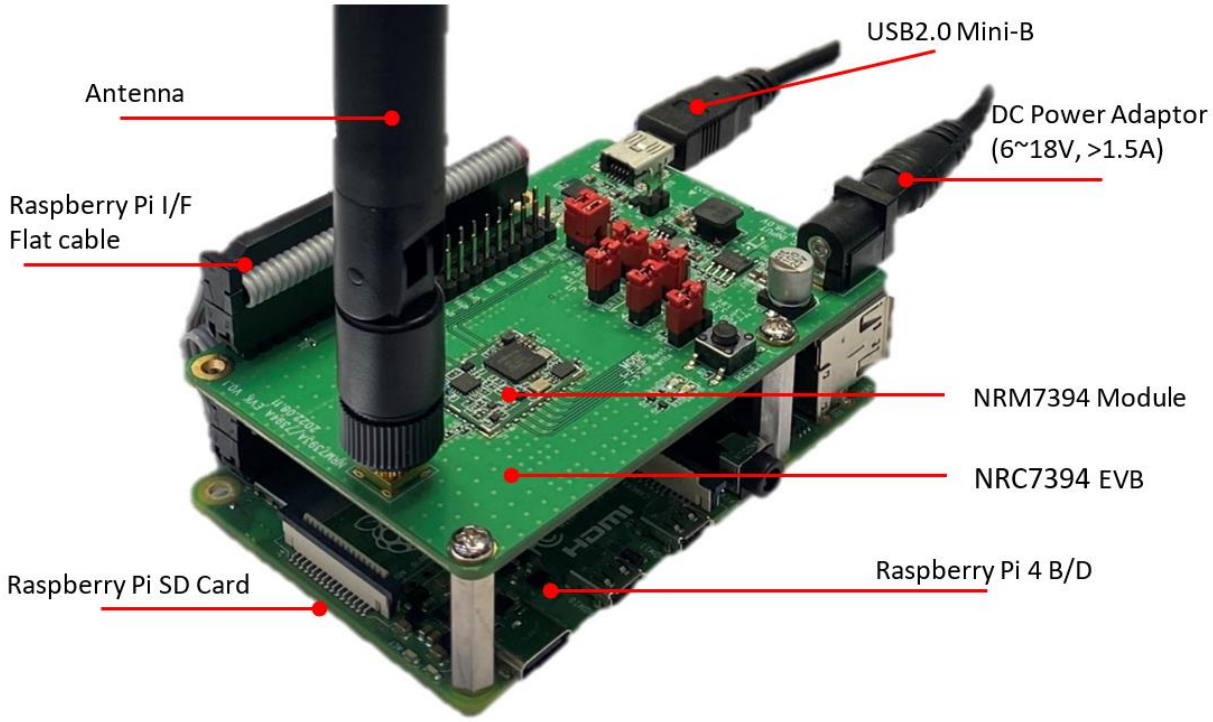


Figure 2.2 NRC7394 Evaluation Kit with Raspberry Pi 4 model B

Figure 2.3 shows the pin maps of the 20-pin and 40-pin headers on the NRC7394 EVB.

	PIN#		
VDD_IO	1	2	VDD_IO
HSPI_MOSI/GP06	3	4	HSPI_CLK/GP07
UART0_TXD/GP08	5	6	UART0_RXD/GP09
TMS/GP10/SWD_IO	7	8	TCK/GP11/SWD_CLK
TDO/GP12/UART1_TXD	9	10	TDI/GP13/UART1_RXD
GP14/UART1_CTS	11	12	GP17/ADC0
GP18/ADC1	13	14	MODE/GP19
GP20/UART1_RTS	15	16	GP24
GP25	17	18	HSPI_CS/GP28
HSPI_MISO/GP29	19	20	HSPI_EIRQ/GP30

	PIN#		
	1	2	5V
	3	4	5V
	5	6	GND
RESET	7	8	UART1_RXD
GND	9	10	UART1_TXD
UART1_CTS	11	12	
	13	14	GND
	15	16	
	17	18	
HSPI_MOSI	19	20	GND
HSPI_MISO	21	22	
HSPI_CLK	23	24	HSPI_CS
GND	25	26	
	27	28	
HSPI_EIRQ	29	30	GND
	31	32	
	33	34	GND
	35	36	UART1_RTS
	37	38	
GND	39	40	

Figure 2.3 Pin maps of the 20-pin and 40-pin headers on the NRC7394 EVB

Figure 2.4 shows the pin map of the 40-pin header on the Raspberry Pi board.

	PIN#		
3.3V	1	2	5V
GPIO 2 (SDA)	3	4	5V
GPIO 3 (SCL)	5	6	GND
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
GND	9	10	GPIO 15 (RXD)
GPIO 17 (RTS)	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	GND
GPIO 22	15	16	GPIO 23
3.3V	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	GND
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
GND	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	GND
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	GND
GPIO 19 (PCM_FS)	35	36	GPIO 16 (CTS)
GPIO 26	37	38	GPIO 20 (PCM_DIN)
GND	39	40	GPIO 21 (PCM_DOUT)

Figure 2.4 Pin map of the 40-pin header on the Raspberry Pi board

NOTE:

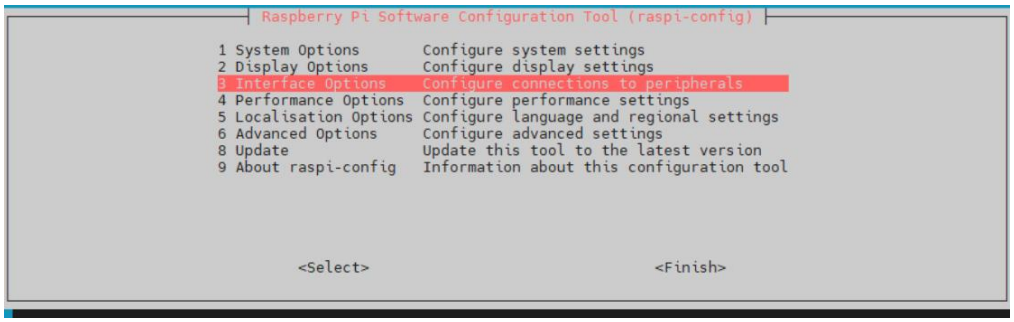
If the host is connected with a 20-pin header, detach the Raspberry Pi board from the EVB first before proceeding. The EVB must be used as a standalone for stable AT communication.

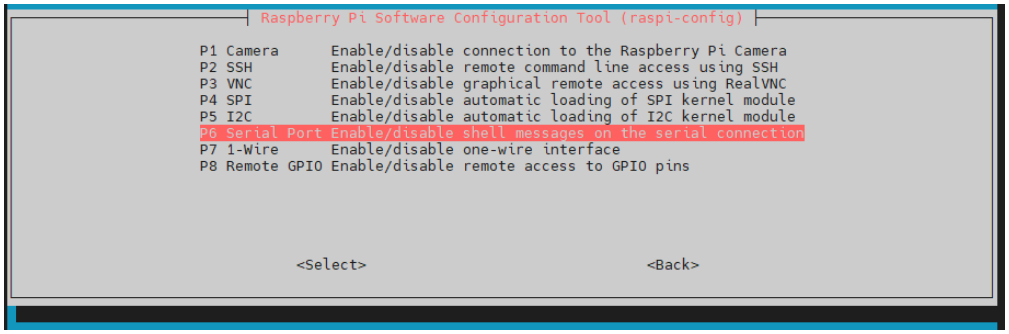
2.1.1 UART

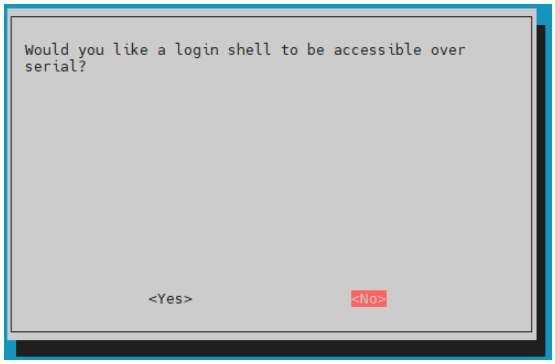
The NRC7394 AT command firmware uses UART channel 1. RTS/CTS is optional and is required to use baudrate greater than 115,200 bps.

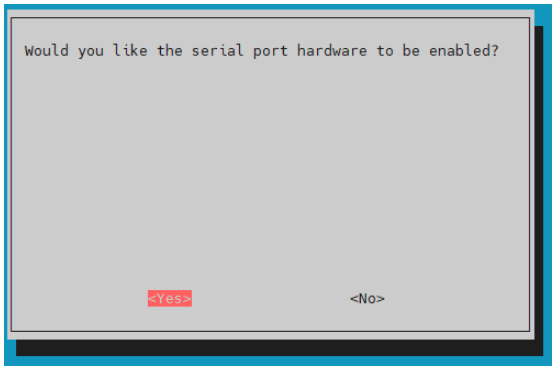
To perform AT command communication through UART on Raspberry Pi, Serial Port must be enabled in the Raspberry Pi configuration tool.

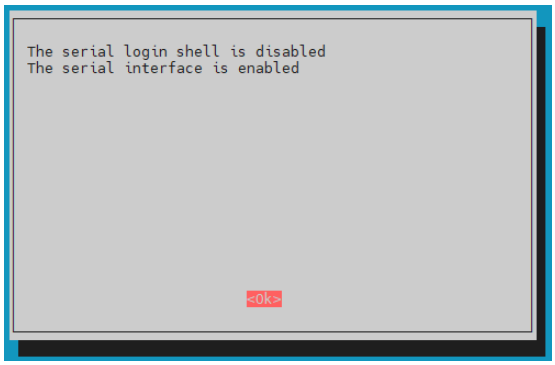
```
# sudo raspi-config
```











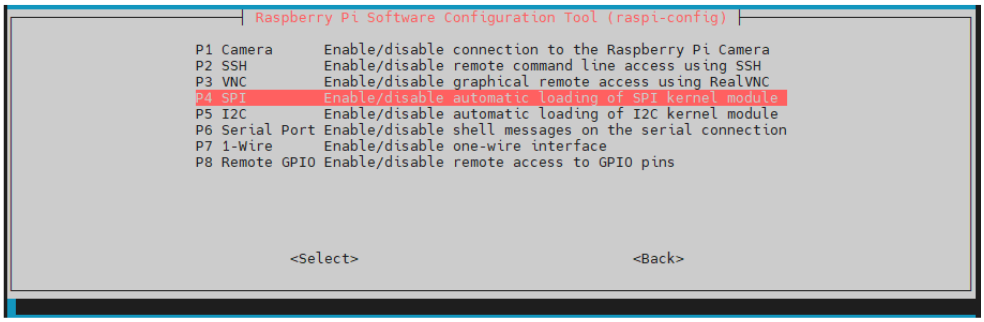
2.1.2 HSPI

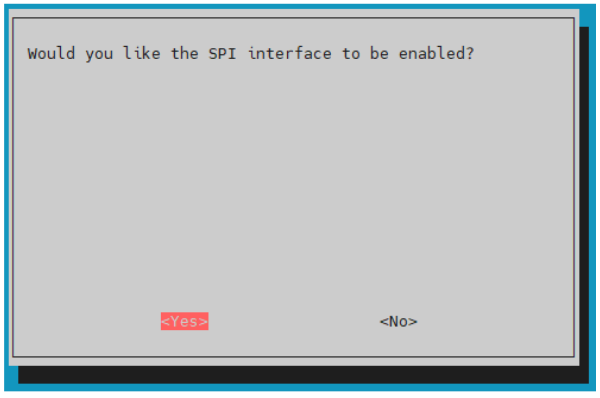
The NRC7394 has a dedicated SPI slave controller for high speed. HSPI_EIRQ is optional.

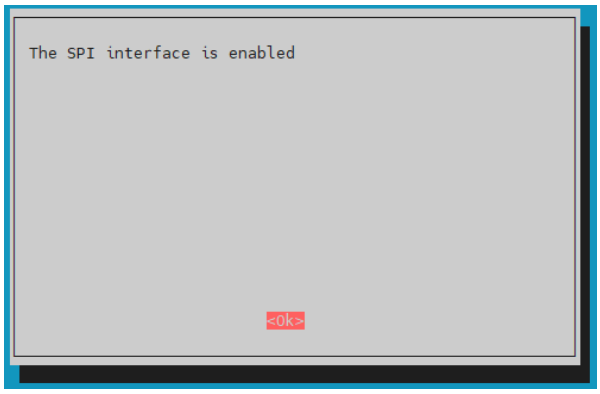
To perform AT command communication through SPI on Raspberry Pi, spidev (User mode SPI device driver) must be enabled.

First, SPI interface must be enabled in the Raspberry Pi configuration tool.

sudo raspi-config







If spidev0.0 and spidev0.1 are not created under /dev directory, open and check the /boot/config.txt.

```
# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on
enable_uart=1
dtoverlay=pi3-disable-bt
dtoverlay=pi3-disable-wifi
#dtoverlay=pi3-disable-spidev
```

After rebooting the Raspberry Pi, spidev0.0 and spidev0.1 could be accessible from the userspace.

```
pi@raspberrypi:~ $ ls /dev
autofs           gpiochip2  loop7              ram0   random     tty11  tty26  tty40  tty55      uhid      vcsa2
block            gpiomem    loop-control       ram1   raw        tty12  tty27  tty41  tty56      uinput    vcsa3
btrfs-control    hidraw0    mapper             ram10  rfkill     tty13  tty28  tty42  tty57      urandom   vcsa4
bus              hidraw1    mem                ram11  serial0    tty14  tty29  tty43  tty58      vchiq     vcsa5
cachefiles       hwrng      memory_bandwidth   ram12  serial1    tty15  tty3   tty44  tty59      vcio      vcsa6
char             initctl    mmcblk0            ram13  shm        tty16  tty30  tty45  tty6       vc-mem    vcsa7
console          input      mmcblk0p1          ram14  snd        tty17  tty31  tty46  tty60      vcs       vcsm
cpu_dma_latency  kmsg       mmcblk0p2          ram15  spidev0.0  tty18  tty32  tty47  tty61      vcs1      vhci
cuse             log        mqueue             ram2   spidev0.1  tty19  tty33  tty48  tty62      vcs2      watchdog
disk             loop0      net                ram3   stderr     tty2   tty34  tty49  tty63      vcs3      watchdog0
fb0              loop1      network_latency    ram4   stdin      tty20  tty35  tty5   tty7       vcs4      zero
fd               loop2      network_throughput ram5   stdout     tty21  tty36  tty50  tty8       vcs5
full             loop3      null               ram6   tty        tty22  tty37  tty51  tty9       vcs6
fuse             loop4      ppp                ram7   tty0       tty23  tty38  tty52  ttyAMA0    vcs7
gpiochip0        loop5      ptmx               ram8   tty1       tty24  tty39  tty53  ttyprintk  vcsa
gpiochip1        loop6      pts                ram9   tty10      tty25  tty4   tty54  ttyS0      vcsa1
```

2.2 Software

Users need to download the firmware binary onto the flash on the NRC7394 module to enable AT-command communication via UART or SPI.

Refer to the user guide **UG-7394-004-Standalone SDK.pdf** for instructions on how to download the firmware binary. (3 How to download compiled binaries)

3 AT Command Type

There are four types of AT-commands: HELP, GET, SET and RUN.

Type	Format	Description
HELP	**AT+<CMD>=?**	**List the input argument format and description.**
SET or **RUN**	**AT+<CMD>** OR **AT+<CMD>=<X1,X2,....>**	**Run with no argument.** OR **Set or run with the given arguments.**
GET	**AT+<CMD>?** OR **AT+<CMD>?=<X1,X2,…>**	**Query the current values with no argument.** OR **Query the current values with the given arguments.**

Table 3.1 AT-command type

- String input parameter values must be enclosed between double quotation marks (“).
- Parameters enclosed between a pair of square brackets ‘[]’ indicate optional parameters.
- Optional parameters may be nested.
- All AT commands must be in upper-case letters and terminated by CR-LF.
- Default optional values in the parameter descriptions are indicated by the asterisk ‘*’ characters.

4 Return for Commands

Return Message	Description
OK	The operation for command completes successfully.
ERROR	The command is not supported.
+<CMD>:1 ERROR	The parameter for command is not valid.
+<CMD>:2 ERROR	The previous operation for command is in progress.
+<CMD>:3 ERROR	The operation for command failed with some error.
+<CMD>:4 ERROR	The operation for command is still in progress after the specified time.

5 Basic AT Commands

Commands	Description
AT	Check the AT serial interface status.
ATE	Enable or disable echo.
ATZ	Reset the hardware and restart the firmware.
AT+VER	Fetch the AT firmware version and software package version.
AT+UART	Configure the serial UART parameters.
AT+GPIOCONF	Configure the GPIO pin mode, direction and pull-up option.
AT+GPIOVAL	Read or write the output GPIO pin level.
AT+ADC	Fetch the ADC value at the selected ADC channel index.

5.1 AT

Command	AT
Response	OK
Description	Check the AT serial interface status.
Example	AT OK

5.2 ATE

Command	ATE0 or ATE1
Response	OK
Description	Enable (ATE1) or disable (ATE0) echo. (default: disable) NOTE: Echo should typically be enabled for manual communication via a terminal.
Example	ATE1 OK ATE0 OK

5.3 ATZ

Command	ATZ
Response	
Description	Reset the hardware and restart the firmware.
Example	ATZ

5.4 AT+VER

Command	**GET** AT+VER?
Response	**GET** +VER: <sdk_version>,<command_version>

	OK
Description	Fetch the version information of current firmware.
Example	AT+VER? +VER:"1.0.0","1.23.5" OK

5.5 AT+UART

Command	**SET** AT+UART=<baud_rate>,<HFC> **GET** AT+UART?
Response	**SET** OK **GET** +UART:<baud_rate>,<data_bits>,<stop_bits>,<parity>,<HFC> OK
Parameters	**<baud rate>** 9600, 19200, 38400, 57600, 115200*, 230400, 460800, 500000, 576000, 921600, 1000000, 1152000, 1500000, 2000000 **<data bits>** Always 8 (8-bit)* **<stop bits>** Always 1 (1-bit)* **<parity>** Always 0 (None)* **<HFC>** 0 : disable RTS/CTS* 1 : enable RTS/CTS
Description	Configure the baud rate and HFC for the UART.

Example	AT+UART=115200,1 OK AT+UART? +UART:115200,8,1,0,1 OK

5.6 AT+GPIOCONF

Command	**SET** AT+GPIOCONF=<number>,<direction>,<pull-up> **GET** AT+GPIOCONF? AT+GPIOCONF?=<number>
Response	**SET** OK **GET** +GPIOCONF=<number>,<direction>,<pull-up> : OK
Parameters	**<number>** GPIO pin number **<direction>** 0 : input 1 : output **<pull-up> (input pin only)** 0 : pull-down 1 : pull-up
Description	Configure the GPIO pin direction and pull-up option.
Example	AT+GPIOCONF=10,1,1 OK

Host Interface Type	Available GPIO numbers
HSPI	10, 11, 12, 13, 14, 20, 25
UART	6, 7, 10, 11, 25, 28, 29, 30

	AT+GPIOCONF=11,0,0 OK AT+GPIOCONF? : +GPIOCONF:10,1,1 +GPIOCONF:11,0,0 : OK AT+GPIOCONF?=10 +GPIOCONF:10,1,1 OK

5.7 AT+GPIOVAL

Command	**SET** AT+GPIOVAL=<number>,<level> **GET** AT+GPIOVAL? AT+GPIOVAL?=<number>
Response	**SET** OK **GET** +GPIOVAL:<number>,<level> OK
Parameters	**<number>** GPIO pin number **<level>** 0 : low 1 : high

Host Interface Type	Available GPIO numbers
HSPI	10, 11, 12, 13, 14, 20, 25
UART	6, 7, 10, 11, 25, 28, 29, 30

Description	Read or write the output GPIO pin level.
Example	AT+GPIOVAL? : +GPIOVAL:10,1 +GPIOVAL:11,0 : OK AT+GPIOVAL?=10 +GPIOVAL:10,1 OK

5.8 AT+ADC

Command	**SET** AT+ADC=<controller> **GET** AT+ADC? AT+ADC?=<channel>
Response	**GET** +ADC:<channel>,<value> : OK
Parameters	**<controller>** 0 : disable 1 : enable **<channel>** 0, 1 **<value>** 0 ~ 1023 (10-bits)
Description	Fetch the ADC value at the selected ADC channel.
Example	AT+ADC=1 OK

	AT+ADC? +ADC:0,396 +ADC:1,448 OK AT+ADC?=0 +ADC:2,384 OK AT+ADC=0 OK AT+ADC? ERROR

6 Wi-Fi AT Commands

Commands	Description
AT+WMACADDR	Read the MAC address.
AT+WCOUNTRY	Configure the Wi-Fi country code
AT+WTXPOWER	Set the transmission power level.
AT+WRXSIG	Fetch or monitor the RSSI (dBm) and SNR (dB) values.
AT+WRATECTRL	Toggle the MCS rate control option.
AT+WMCS	Set the MCS index.
AT+WDUTYCYCLE	Configure duty cycle operation.
AT+WCCATHRESHOLD	Set CCA threshold.
AT+WTXTIME	Set carrier sense time and pause time.
AT+WTSF	Read the elapsed TSF timer duration.
AT+WSCAN	Perform Wi-Fi scanning.
AT+WCONN	Connect to a new AP.
AT+WDISCONN	Disconnect from the AP or abort an on-going connection process.
AT+WSOFTAP	Run as the AP mode.
AT+WBSSMAXIDLE	Configure the BSS Max idle service for SoftAP.
AT+WSTAINFO	Get information of associated STAs on AP mode.
AT+WIPADDR	Configure the IPv4 address.
AT+WDNS	Configure the IP address for the DNS server.
AT+WDHCP	Request dynamic IP allocation from the DHCP server.
AT+WDHCPS	Run the DHCP sever in SoftAP mode.
AT+WPING	Send ICMP ECHO_REQUEST to network hosts with IPv4 address.
AT+WDEEPSLEEP	Configure deep-sleep mode to save power.
AT+WFOTA	Enable or disable Firmware Over-the-Air (FOTA).
AT+WCTX	Send dummy data frames for continuous TX without connecting to AP.

AT+WTIMEOUT	Configure the response timeout for the specified command.
+WEVENT	Asynchronously raised Wi-Fi event logs.

6.1 AT+WMACADDR

Command	**GET** AT+WMACADDR?
Response	**GET** +WMACADDR:”<MAC address>” OK
Parameters	**<MAC address>** The MAC address ‘HH:HH:HH:HH:HH:HH’ where H is a hexadecimal character.
Description	Read the MAC address.
Example	AT+ WMACADDR? +WMACADDR:”2F:33:4F:65:11:20” OK

6.2 AT+WCOUNTRY

Command	**SET** AT+WCOUNTRY=”<country code>” **GET** AT+WCOUNTRY?
Response	**SET** OK **GET** +WCOUNTRY=”<country code>” OK
Parameters	**<country code>** - AU : Australia - CN : China - EU : Europe - JP : Japan - NZ : New Zealand - TW : Taiwan - US : United States - K0 : Korea USN (2M BW support) - K1 : Korea USN - K2 : Korea MIC

Description	Configure the Wi-Fi country code NOTE: The country code may need to be set after booting.
Example	AT+ WCOUNTRY =”US” OK AT+WCOUNTRY? +WCOUNTRY:”US” OK

6.3 AT+WTXPOWER

Command	**SET** AT+WTXPOWER=<txpower> **GET** AT+WTXPOWER?
Response	**SET** OK **GET** +WTXPOWER:<txpower>
Parameters	**<txpower>** Transmission Power Level (unit : dBm) (1 ~ 30)
Description	Set or get the transmission power level. Set to 0 to use AUTO mode, not FIXED mode. AUTO mode sets TX power automatically according to MCS. Default is AUTO mode. NOTE: Depending on the country and channel frequency, the maximum allowed TX power may be limited to less than 30 dBm.
Example	AT+WTXPOWER? +WTXPOWER:16 <--- TX power for the last transmission. OK **< FIXED mode >**

	AT+WTXPOWER=10 OK AT+WTXPOWER? +WTXPOWER:10 OK **< AUTO mode >** AT+WTXPOWER=0 OK AT+WTXPOWER? +WTXPOWER:10 <--- TX power for the last transmission. OK

6.4 AT+WRXSIG

Command	**SET** AT+WRXSIG =<time> **GET** AT+WRXSIG?
Response	**SET** +WRXSIG:<RSSI>,<SNR> … +WRXSIG:<RSSI>,<SNR> OK **GET** +WRXSIG:<RSSI>,<SNR> OK
Parameters	**<time>** Monitoring duration in seconds.
Description	Fetch or monitor the RSSI (dBm) and SNR (dB) values.
Example	AT+WRXSIG? +WRXSIG:-68,31 OK AT+WRXSIG=10 +WRXSIG:-68,31

	+WRXSIG:-68,30 +WRXSIG:-68,32 +WRXSIG:-68,32 +WRXSIG:-68,32 +WRXSIG:-68,32 +WRXSIG:-68,30 +WRXSIG:-68,31 +WRXSIG:-68,32 +WRXSIG:-68,32 OK

6.5 AT+WRATECTRL

Command	**SET** AT+WRATECTRL=<mode> **GET** AT+WRATECTRL?
Response	**SET** OK **GET** +WRATECTRL=<mode> OK
Parameters	**<mode>** 0 : disable 1 : enable*
Description	Toggle the MCS rate control option.
Example	AT+WRATECTRL? +WRATECTRL:1 OK AT+WRATECTRL=0 OK AT+WRATECTRL? +WRATECTRL:0 OK

6.6 AT+WMCS

Command	**SET** AT+WMCS=<index> **GET** AT+WMCS?
Response	**SET** OK **GET** +WMCS=<index> OK
Parameters	**<index>** Modulation Coding Scheme index (0, 1, 2, 3, 4, 5, 6, 7 and 10)
Description	Set or get the MCS index. NOTE: The MCS index can only be set when rate control is disabled.
Example	AT+WRATECTRL? +WRATECTRL:1 OK AT+WMCS? +WMCS:7 <--- MCS index for the last transmission. OK AT+WMCS=0 ERROR AT+WRATECTRL=0 OK AT+WRATECTRL? +WRATECTRL:0 OK AT+WMCS? +WMCS:7 OK

	AT+WMCS=0 OK AT+WMCS? +WMCS:0 OK

6.7 AT+WDUTYCYCLE

Command	**SET** AT+WDUTYCYCLE=<window>[,<duration>[,<margin>]] **GET** AT+WDUTYCYCLE?
Response	**SET** OK **GET** +WDUTYCYCLE=<window>,<duration>,<margin> OK
Parameters	**<window>** Duty cycle window in microseconds **<duration>** TX duration in microseconds allowed within duty cycle window **<margin>** Duty margin in microseconds
Description	Configure duty cycle operation.
Example	AT+WDUTYCYCLE? +WDUTYCYCLE:0,0,0 OK AT+WDUTYCYCLE=1000000,100000 AT+WDUTYCYCLE? +WDUTYCYCLE:1000000,100000,0 OK

	AT+WDUTYCYCLE=0 OK AT+WDUTYCYCLE? +WDUTYCYCLE:0,0,0 OK

6.8 AT+WCCATHRESHOLD

Command	**SET** AT+WCCATHRESHOLD=<threshold> **GET** AT+WCCATHRESHOLD?
Response	**SET** OK **GET** +WCCATHRESHOLD=<threshold> OK
Parameters	**<threshold>** CCA threshold.(unit: dBm) (-100 ~ -35)
Description	Set CCA threshold.
Example	AT+WCCATHRESHOLD? +WCCATHRESHOLD:-75 OK AT+WCCATHRESHOLD=-80 OK AT+WCCATHRESHOLD? +WCCATHRESHOLD:-80 OK

6.9 AT+WTXTIME

Command	**SET** AT+WTXTIME=<cs_time>[,<pause_time>]

	GET AT+WTXTIME?
Response	**SET** OK **GET** +WTXTIME:<cs_time>,<pause_time> OK
Parameters	**<cs_time>** Carrier sensing time in microseconds (0 ~ 13260) **<pause_time>** Tx pause time in microseconds
Description	Set carrier sense time and pause time for Listen Before Talk
Example	AT+WTXTIME? +WTXTIME:0,0 OK AT+WTXTIME=128,2000 OK AT+WTXTIME? +WTXTIME:128,2000 OK

6.10 AT+WTSF

Command	**GET** AT+WTSF?
Response	**GET** +WTSF:<time> OK
Parameters	**<time>** Elapsed TSF timer duration in microseconds.
Description	Read the elapsed TSF timer duration.
Example	AT+WTSF?

	+WTSF:44142384 OK

6.11 AT+WSCAN

Command	**RUN** AT+WSCAN **SET** AT+WSCAN=[{+|-}]<freq>[@<bandwidth>][,<freq>[@<bandwidth>] ...] **GET** AT+WSCAN?
Response	**RUN** +WSCAN:<bssid>,<freq>,<sig_level>,<flags>,<ssid> : OK **SET** OK **GET** +WSCAN:<bandwidth>,<freq>[,<freq> …] : OK
Parameters	**<bssid>** The BSSID of the AP. **<freq>** The center frequency of the channel. (MHz) **<sig_level>** The RSSI (Received Signal Strength Indicator) in dBm. **<bandwidth>** The bandwidth of the channel. (1/2/4 MHz) **<flags>** Service set flags. **<ssid>** The SSID of the AP.

Description	**RUN** Perform Wi-Fi scanning. **SET/GET** Set the frequencies of the channel to scan or get a list of them. In the SET command, if the first frequency value has a '+' or '-' prefix, a new frequency is added or a specific frequency is excluded. "AT+WSCAN=0" command resets the scan frequency list to scan all supported channels. NOTE: The SET command cannot be used while connected to the AP and responds with ERROR. After "AT+WCOUNTRY" and "AT+WDISCONN" commands, the scan frequency list is reset to scan all supported channels.
Example	AT+WCOUNTRY="US" OK AT+WSCAN? +WSCAN:1,902.5,903.5,904.5,905.5,906.5,907.5,908.5,909.5,910.5,911.5 +WSCAN:1,912.5,913.5,914.5,915.5,916.5,917.5,918.5,919.5,920.5,921.5 +WSCAN:1,922.5,923.5,924.5,925.5,926.5,927.5 +WSCAN:2,903.0,905.0,907.0,909.0,911.0,913.0,915.0,917.0,919.0,921.0 +WSCAN:2,923.0,925.0,927.0 +WSCAN:4,906.0,910.0,914.0,918.0,922.0,926.0 OK AT+WSCAN +WSCAN:"02:00:eb:13:d3:4a",922.5,-39,"[ESS]","halow_open" +WSCAN:"68:27:eb:0e:07:27",922.5,-30,"[WPA2-PSK-CCMP][ESS]","halow_wpa2" +WSCAN:"8c:0f:fa:00:28:1f",906.0,-54,"[WPA3-SAE-CCMP][ESS]","halow_sae" +WSCAN:"8c:0f:fa:00:29:46",921.0,-75,"[WPA3-SAE-CCMP][ESS]","halow_sae2" OK AT+WSCAN=922.5 OK

	AT+WSCAN? +WSCAN:1,922.5 OK AT+WSCAN +WSCAN:"02:00:eb:13:d3:4a",922.5,-39,"[ESS]","halow_open" +WSCAN:"68:27:eb:0e:07:27",922.5,-30,"[WPA2-PSK-CCMP][ESS]","halow_wpa2" OK AT+WSCAN=+906,921 OK AT+WSCAN? +WSCAN:1922.5 +WSCAN:2,921.0 +WSCAN:4,906.0 OK AT+WSCAN +WSCAN:"02:00:eb:13:d3:4a",922.5,-39,"[ESS]","halow_open" +WSCAN:"68:27:eb:0e:07:27",922.5,-30,"[WPA2-PSK-CCMP][ESS]","halow_wpa2" +WSCAN:"8c:0f:fa:00:28:1f",906.0,-54,"[WPA3-SAE-CCMP][ESS]","halow_sae" +WSCAN:"8c:0f:fa:00:29:46",921.0,-75,"[WPA3-SAE-CCMP][ESS]","halow_sae2" OK AT+WSCAN=-921,922.5 OK AT+WSCAN? +WSCAN:4,906.0 OK AT+WSCAN +WSCAN:"8c:0f:fa:00:28:1f",906.0,-54,"[WPA3-SAE-CCMP][ESS]","halow_sae" OK AT+WSCAN=0 OK AT+WSCAN? +WSCAN:1,902.5,903.5,904.5,905.5,906.5,907.5,908.5,909.5,910.5,911.5 +WSCAN:1,912.5,913.5,914.5,915.5,916.5,917.5,918.5,919.5,920.5,921.5 +WSCAN:1,922.5,923.5,924.5,925.5,926.5,927.5

	+WSCAN:2,903.0,905.0,907.0,909.0,911.0,913.0,915.0,917.0,919.0,921.0 +WSCAN:2,923.0,925.0,927.0 +WSCAN:4,906.0,910.0,914.0,918.0,922.0,926.0 OK AT+WSCAN=922.5 OK AT+WSCAN +WSCAN:"02:00:eb:13:d3:4a",922.5,-39,"[ESS]","halow_open" +WSCAN:"68:27:eb:0e:07:27",922.5,-30,"[WPA2-PSK-CCMP][ESS]","halow_wpa2" OK AT+WCONN=”halow_open” OK AT+WSCAN? +WSCAN=1,922.5 OK AT+WSCAN=+906,921 ERROR AT+WDISCONN OK AT+WSCAN? +WSCAN:1,902.5,903.5,904.5,905.5,906.5,907.5,908.5,909.5,910.5,911.5 +WSCAN:1,912.5,913.5,914.5,915.5,916.5,917.5,918.5,919.5,920.5,921.5 +WSCAN:1,922.5,923.5,924.5,925.5,926.5,927.5 +WSCAN:2,903.0,905.0,907.0,909.0,911.0,913.0,915.0,917.0,919.0,921.0 +WSCAN:2,923.0,925.0,927.0 +WSCAN:4,906.0,910.0,914.0,918.0,922.0,926.0 OK ---------- AT+WCOUNTRY="JP" OK AT+WSCAN? +WSCAN:1,921.0,923.0,924.0,925.0,926.0,927.0 +WSCAN:2,923.5,924.5,925.5,926.5

	+WSCAN:4,924.5,925.5 OK AT+WSCAN=926,923,923.5,925.5 OK AT+WSCAN? +WSCAN:1,923.0,926.0 +WSCAN:2,923.5,925.5 OK AT+WSCAN=926,923,926.5,925.5@2,925.5@4,924.5@2 OK AT+WSCAN? +WSCAN:1,923.0,926.0 +WSCAN:2,924.5,925.5,926.5 +WSCAN:4,925.5 OK AT+WSCAN=-926.5,925.5@2 OK AT+WSCAN? +WSCAN:1,923.0,926.0 +WSCAN:2,924.5 +WSCAN:4,925.5 OK AT+WSCAN=+924.5@4,925 OK AT+WSCAN? +WSCAN:1,923.0,925.0,926.0 +WSCAN:2,924.5 +WSCAN:4,924.5,925.5 OK

6.12 AT+WCONN

Command	**SET**

	AT+WCONN=”<ssid|bssid>”[,”<security>”[,”<password>”]] **GET** AT+WCONN?
Response	**SET** OK **GET** +WCONN=”<ssid>”,”<bssid>”,”<security>”,”<password>”,”<state>” OK
Parameters	**<ssid>** The SSID of the AP. **<bssid>** The BSSID of the AP. **<security>** open*, wpa2-psk (or psk), wpa3-owe (or owe), wpa3-sae (or sae) **<password> (wpa2/wpa3-sae security option only)** The password when wpa2/wpa3-sae security option is used. (length : 8 ~ 63) **<state>** State indicator: “connecting”, “connected”, “disconnecting” or ”disconnected”
Description	Connect to a new AP or retrieves information about the current AP. NOTE: If an “ERROR” is returned with the error number INPROGRESS(2) or TIMEOUT(4), the AT-STA needs to be disconnected from the AP with the “AT+WDISCONN” command before a connection is attempted again with “AT+WCONN”.
Example	**OPEN :** AT+WSCAN +WSCAN:"8c:0f:fa:00:2b:a1",922.0,-13,"[ESS]","halow_ap” OK AT+WCONN="halow_ap” OK AT+WCONN? +WCONN:”halow_ap”,”8C:0F:FA:00:2B:A1”,”open”,””,”connected” OK

	WPA2-PSK : AT+WSCAN +WSCAN:"8c:0f:fa:00:2b:a1",922.0,-14,"[WPA2-PSK-CCMP][ESS]","halow_ap" OK AT+WCONN="halow_ap","wpa2-psk","12345678" OK AT+WCONN? +WCONN:"halow_ap","8C:0F:FA:00:2B:A1","wpa2-psk","12345678","connected" OK **WPA3-OWE :** AT+WSCAN +WSCAN:"8c:0f:fa:00:2b:a1",922.0,-13,"[WPA2-OWE-CCMP][ESS]","halow_ap" OK AT+WCONN="halow_ap","wpa3-owe" OK AT+WCONN? +WCONN:"halow_ap","8C:0F:FA:00:2B:A1","wpa3-owe","","connected" OK **WPA3-SAE :** AT+WSCAN +WSCAN:"8c:0f:fa:00:2b:a1",922.0,-14,"[WPA2-SAE-CCMP][ESS]","halow_ap" OK AT+WCONN="halow_ap","wpa3-sae","12345678" OK AT+WCONN? +WCONN:"halow_ap","8C:0F:FA:00:2B:A1","wpa3-sae","12345678","connected" OK

6.13 AT+WDISCONN

Command	**RUN** AT+WDISCONN
Response	**RUN** OK

Description	Disconnect from the AP or abort an on-going connection process.
Example	AT+WDISCONN OK

6.14 AT+WSOFTAP

Command	**SET** AT+WSOFTAP=<frequency>[@<bandwidth>],"<ssid>"[,"<security>"[,"<password>"]] **GET** AT+WSOFTAP?
Response	**SET** OK **GET** +WSOFTAP=<frequency>,"<ssid>","<security>","<password>"[,"dhcp"] OK
Parameters	**<frequency>** S1G channel frequency (MHz) **<bandwidth>** S1G channel bandwidth (1/2/4 MHz) **<ssid>** The SSID of the AP. **<security>** open*, wpa2-psk (or psk) **<password> (wpa2 security option only)** The password when wpa2 security option is used. (length : 8 ~ 63) **<dhcp>** Only included when the DHCP server is running.
Description	Run as the AP mode or retrieves information about the current settings. NOTE: The system should be reset to exit the AP mode. Software Reset is possible with the ATZ command.

Example	AT+WCOUNTRY=”JP” OK AT+WSCAN? +WSCAN:923.5,924.5,925.5,926.5,921.0,923.0,924.0,925.0,926.0,927.0 +WSCAN:924.5,925.5 OK AT+WSOFTAP=925.5@4,"halow_softap_psk","psk","12345678" OK AT+WSOFTAP? +WSOFTAP:4,925.5,"halow_softap_psk","wpa2-psk","12345678" OK AT+WDHCPS +WDHCPS:192.168.200.27,255.255.255.0,192.168.200.1 OK AT+WSOFTAP? +WSOFTAP:4,925.5,"halow_softap_psk","wpa2-psk","12345678","dhcp" OK

6.15 AT+WBSSMAXIDLE

Command	**SET** AT+WBSSMAXIDLE=<period>[,<retry>] **GET** AT+WBSSMAXIDLE?
Response	**SET** OK **GET** +WBSSMAXIDLE:<period>,<retry> OK
Parameters	**<period>** BSS MAX IDLE period in seconds (default: 0) **<retry>** retry count for receiving keep alive packet from STA (3 ~ 100, default: 3)

Description	Configure the BSS MAX IDLE service for SoftAP. SoftAP disconnects STA that is inactive for BSS MAX IDLE time. If the AP does not receive a keep alive packet from the STA for BSS MAX IDLE time, it is determined that the STA is in an inactive state. Example: - BSS max idle period = 60 secs - retry count = 5 - BSS max idle time = 60 x 5 = 300 secs If the period is set 0, the service is stopped.
Example	AT+WBSSMAXIDLE? +WBSSMAXIDLE:0,3 OK AT+WBSSMAXIDLE=60,60 OK AT+WBSSMAXIDLE? +WBSSMAXIDLE:60,60 OK AT+WSOFTAP=918.5,"halow_softap_wpa2","wpa2-psk","12345678" OK AT+WDHCPS +WDHCPS:"192.168.50.1","255.255.255.0","192.168.50.1" OK AT+WBSSMAXIDLE=60,5 OK AT+WBSSMAXIDLE? +WBSSMAXIDLE:60,5 OK AT+WBSSMAXIDLE=0 OK AT+WBSSMAXIDLE? +WBSSMAXIDLE:0,3 OK

6.16 AT+WSTAINFO

Command	**SET** AT+WSTAINFO=<aid> **GET** AT+WSTAINFO?
Response	+WSTAINFO=<aid>,"<mac_address>",<rssi>,<snr>,<mcs_index> OK
Parameters	**<aid>** Association ID **<mac_address>** Hardware address of associated station **<rssi>** Received Signal Strength indication **<snr>** Signal to Noise Ratio **<mcs_index>** Modulation Coding Scheme index
Description	Get information of associated STAs when the device is in AP mode.
Example	AT+WSOFTAP=918.5,"halow_softap","wpa2-psk","12345678" OK AT+WIPADDR="192.168.1.1","255.255.255.0","192.168.1.1" OK AT+WDHCPS +WDHCPS:"192.168.1.1","255.255.255.0","192.168.1.1" OK Wait for one or more stations to be associated … AT+WSTAINFO? +WSTAINFO:1,"8c:0f:fa:00:2b:a1",-34,31,7 +WSTAINFO:2,"8c:0f:fa:00:2b:a2",-45,34,7 +WSTAINFO:3,"8c:0f:fa:00:2b:a3",-16,21,7 OK AT+WSTAINFO=1

	+WSTAINFO:1,"8c:0f:fa:00:2b:a1",-33,34,7 OK

6.17 AT+WIPADDR

Command	**SET** AT+WIPADDR=”<address>”,”<netmask>”,”<gateway>” **GET** AT+WIPADDR?
Response	**SET** OK **GET** +WIPADDR=”<address>”,”<netmask>”,”<gateway>” OK
Parameters	**<address>,<netmask>,<gateway>** IPv4 address
Description	Configure the IPv4 address.
Example	AT+WIPADDR=”192.168.200.20”,”255.255.255.0”,”192.168.200.1” OK AT+WIPADDR? +WIPADDR=”192.168.200.20”,”255.255.255.0”,”192.168.200.1” OK

6.18 AT+WDNS

Command	**SET** AT+WDNS=”<DNS1>”[,”<DNS2>”] **GET** AT+WDNS?
Response	**SET** OK **GET** +WDNS=”<DNS1>”,”<DNS2>” OK
Parameters	**<DNS1>,<DNS2>** IPv4 address

Description	Configure the IP address of the DNS server.
Example	AT+WDNS? +WDNS=”192.168.200.1”,”0.0.0.0” OK AT+WDNS=”8.8.8.8” OK AT+WDNS? +WDNS=”8.8.8.8”,”0.0.0.0” OK AT+WDNS=”8.8.8.8”,”8.8.4.4” OK AT+WDNS? +WDNS=”8.8.8.8”,”8.8.4.4” OK

6.19 AT+WDHCP

Command	**RUN** AT+WDHCP **SET** AT+WDHCP=<mode> **GET** AT+WDHCP?
Response	**RUN** +WDHCP:”<address>”,”<netmask>”,”<gateway>” OK **SET** OK **GET** +WDHCP:{0|1} OK
Parameters	**<address>, <netmask>** and **<gateway>** IPv4 Address

	<mode> 0 : run manually after connection 1 : run automatically connection or reconnection
Description	Request dynamic IP allocation from the DHCP server. NOTE: Wi-Fi connection must be established before using this command.
Example	AT+WCONN="halow_ap","wpa3-sae","12345678" OK AT+WDHCP +WDHCP:″192.168.200.20″,″255.255.255.0″,″192.168.200.1″ OK AT+WDISCONN OK AT+WDHCP? +WDHCP:0 OK AT+WDHCP=1 OK AT+WCONN="halow_ap","wpa3-sae","12345678" OK +WEVENT:"DHCP_RUN" +WEVENT:"DHCP_SUCCESS","192.168.200.18","255.255.255.0","192.168.200.1" +WEVENT:"DISCONNECT","","halow_ap","wpa3-sae" +WEVENT:"CONNECT_SUCCESS","","halow_ap","wpa3-sae" +WEVENT:"DHCP_RUN" +WEVENT:"DHCP_SUCCESS","192.168.200.18","255.255.255.0","192.168.200.1"

6.20 AT+WDHCPS

Command	**RUN** AT+WDHCPS
Response	**RUN** +WDHCPS:”<IP>,”netmask>”,”<gateway>” OK
Parameters	**<IP>, <netmask>** and **<gateway>**

	'A.B.C.D' where A, B, C and D are between 0 and 255, inclusive.
Description	Run the DHCP sever in SoftAP mode. NOTE: SoftAP must be established before using this command. Refer to chapter 6.15. (AT+WSOFTAP)
Example	AT+WDHCPS +WDHCPS:"192.168.50.1","255.255.255.0","192.168.50.1" OK

6.21 AT+WPING

Command	**SET** AT+WPING="<remote address>"[,<time>] **GET** AT+WPING?
Response	**SET** +WPING:<size>,"<remote address>",<sequence number>,<TTL>,<elapsed time> : +WPING:<size>,"<remote address>",<sequence number>,<TTL>,<elapsed time> OK **GET** +WPING:"<remote address>",<time>
Parameters	**<remote address>** The remote IPv4 address of the recipient. **<time>** Monitoring duration in seconds. (Default: 5) **<sequence number>** ICMP sequence number. **<TTL>** Time to leave (TTL). **<elapsed time>** Time since the start of the session in seconds.

Description	Send ICMP ECHO_REQUEST to network hosts with IPv4 address. - Interval Time : 1 sec - Packet Size : 64-bytes
Example	AT+WPING =”192.168.200.1”,10 +WPING:64,”192.168.200.1”,1,64,4 +WPING:64,”192.168.200.1”,2,64,4 : +WPING:64,”192.168.200.1”,9,64,4 +WPING:64,”192.168.200.1”,10,64,4 OK

6.22 AT+WDEEPSLEEP

Command	**SET** AT+WDEEPSLEEP=<timeout>[,<gpio>]
Response	**SET** OK
Parameters	**<timeout>** Time in milliseconds. 0 for TIM mode. **<gpio>** GPIO number to use as external signal input. Available GPIO numbers are between 8 and 17.
Description	Configure deep-sleep mode to save power. Deep sleep mode powers off most peripherals to use minimal power. The RTC and retention RAM are always powered. The CPU is powered only in TIM mode to run the uCode stored in the retention RAM. And the GPIO may be powered for external signal input. In TIM mode, the NRC7292 wakes up when there are frames to receive. However, in Non-TIM mode, it cannot be woken up until a timeout. If there are frames to send, the NRC7292 can only be woken up via the GPIO input. The GPIO input level should be low in active mode. If it is high in deep sleep mode, the NRC7292 wakes up.

	After waking up, the CPU resets and the firmware reboots. When the firmware reboot is finished, the host application or terminal program will receive a "DEEPSLEEP_WAKEUP" event message.
Example	**< Deep Sleep, TIM mode >** AT+WCONN="halow_ap","wpa2-psk","12345678" OK AT+WDHCP +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1" OK AT+WDEEPSLEEP=0,11 OK +WEVENT:”DEEPSLEEP_WAKEUP” AT+WCONN="halow_ap","wpa2-psk","12345678" OK AT+WDHCP +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1" OK AT+WPING=”192.168.200.1”,2 +WEVENT:"PING",64,"192.168.200.1",1,64,5 +WEVENT:"PING",64,"192.168.200.1",2,64,4 OK **< Deep Sleep, Non-TIM mode >** AT+WCONN="halow_ap","wpa2-psk","12345678" OK AT+WDHCP +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1" OK AT+WDEEPSLEEP=5000,11 OK +WEVENT:”DEEPSLEEP_WAKEUP” AT+WCONN="halow_ap","wpa2-psk","12345678"

	OK AT+WDHCP +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1" OK AT+WPING=”192.168.200.1”,2 +WEVENT:"PING",64,"192.168.200.1",1,64,6 +WEVENT:"PING",64,"192.168.200.1",2,64,4 OK

6.23 AT+WFOTA

Command	**SET** AT+WFOTA=<check_time>[,\"<server_url>\"] AT+WFOTA=<check_time>[,\"<server_url>\",\"<bin_name>\",<bin_crc32>] **GET** AT+WFOTA? **RUN** AT+WFOTA
Response	**SET** OK **GET** +WFOTA:<check_time>,”<server_url>”,”<bin_name>”,<bin_crc32> OK **RUN** OK
Parameters	**<check_time>** Interval time in seconds to get new firmware information from the server. Set to 0 to stop the getting or get manually. Set to -1 to disable FOTA operation. **<server_url>** HTTP or HTTPS Server URL **<bin_name>** Firmware binary name with extension .bin. **<bin_crc32>** 32bit CRC value to detect data corruption of downloaded firmware.

	A hexadecimal number prefixed with 0x.
Description	FOTA(Firmware Over-the-Air) is enabled with the SET command and disabled by AT+WFOTA=-1 command. When FOTA is enabled, the current firmware starts checking for new firmware on the server. The server check interval can be controlled through the <check_time> parameter. To check for new firmware, the current firmware downloads the fota.json file from the server. The server should have a fota.json file as well as firmware binary. The contents of the fota.json file are as follows.  After getting information about new firmware from the server, the current firmware sends a FOTA_VERSION event to the terminal or host. +WEVENT:"FOTA_VERSION","<sdk_version>","<atcmd_version>" After receiving the FOTA_VERSION event, the terminal or host can use the RUN command to download new firmware from the server. If there is no fota.json file on the server, the firmware information to be downloaded can be set with the bin_name and bin_crc32 parameters. And the terminal or host can use the RUN command without receiving the FOTA_VERSION event. The terminal or host can check the download process through FOTA_BINARY and FOTA_DOWNLOAD events from the current firmware. +WEVENT: "FOTA_BINARY","<binary_name>" +WEVENT: "FOTA_DOWNLOAD",<total_size>,<download_size> When the download is complete and ready to update, the terminal or host will receive a FOTA_UPDATE event from the current firmware. +WEVENT: "FOTA_UPDATE" If an error occurs during the above process, the terminal or host will receive a FOTA_FAIL event from the current firmware. +WEVENT: "FOTA_FAIL" And FOTA will be automatically disabled.

```
{
    "AT_SDK_VER" : "10.10.10",
    "AT_CMD_VER" : "10.10.10",

    "AT_HSPI_BIN" : "nrc7292_standalone_xip_ATCMD_HSPI.bin",
    "AT_HSPI_CRC" : "750243d8",

    "AT_UART_BIN" : "nrc7292_standalone_xip_ATCMD_UART.bin",
    "AT_UART_CRC" : "793066ec",

    "AT_UART_HFC_BIN" : "nrc7292_standalone_xip_ATCMD_UART_HFC.bin",
    "AT_UART_HFC_CRC" : "8f564369"
}
```

If there are no errors, the current firmware will be replaced with the new firmware after a software reset. A software reset is possible with the ATZ command.

Firmware replacement will take about 10 seconds or more.

If an error occurs while accessing the flash memory for firmware replacement, the current firmware cannot be restored. If the error still occurs after a hardware reset, the firmware can only be restored through the download tool.

NOTE:

Whether or not the firmware in the server is the latest version can be determined by comparing the version confirmed by the AT+VER command and the FOTA_VERSION event.

EVENT:

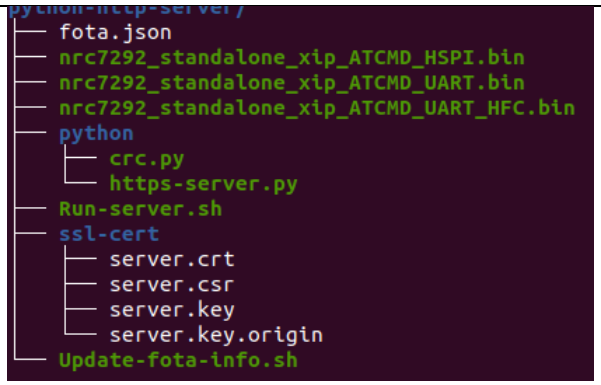
Name	Description
FOTA_VERSION	The version of new firmware on the server. - User SDK version - AT Command Set version
FOTA_BINARY	The binary name of new firmware to download from the server.
FOTA_DOWNLOAD	The binary size of new firmware being downloaded from the server. - Total size - Downloaded size
FOTA_UPDATE	The current firmware is ready to be replaced with the new firmware.
FOTA_FAIL	An error occurred during the FOTA process.

TEST:

The AT+WFOTA command can be tested using the python-http-server package in the SDK.

Path : atcmd/host/python-http-server

This package has the shell and python scripts to run HTTP/HTTPS server.



Shell Script	Description
Run-sever.sh	Run HTTP or HTTPS server. Usage: $ ./Run-server.sh http $ ./Run-server.sh https

Update-fota-info.sh	Calculate the CRC value of firmware binaries and update the fota.json file.
	Usage:
	$ ./Update-fota-info.sh [options]
	Firmware version and binary name can be set by editing this file.
	`SDK_VER="10.10.10"`
	`CMD_VER="10.10.10"`
	`HSPI_BIN="nrc7292_standalone_xip_ATCMD_HSPI.bin"`
	`UART_BIN="nrc7292_standalone_xip_ATCMD_UART.bin"`
	`UART_HFC_BIN="nrc7292_standalone_xip_ATCMD_UART_HFC.bin"`
	Alternatively, it can be set as options when executing the script. Available options can be checked with the -h or --help option. Values set as options overwrite values set in the file.
	If a binary is replaced with a new one, the fota.json should be updated by Update-fota-info.sh.

Example

AT+VER?
+VER:”1.0.0”,”1.23.5”
OK

AT+WFOTA?
+WFOTA:0,"","",0x0
OK

< Get new firmware information from fota.json file >

```
AT+WFOTA=10,"https://192.168.200.1:4443"
AT+WFOTA=10,"https://192.168.200.1:4443"
OK
AT+WFOTA?
+WFOTA:10,"https://192.168.200.1:4443","",0x0
OK
+WEVENT:"FOTA_VERSION","10.10.10","10.10.10"
+WEVENT:"FOTA_VERSION","10.10.10","10.10.10"
+WEVENT:"FOTA_VERSION","10.10.10","10.10.10"
```

*Stop the getting to switch manually.

```
AT+WFOTA=0
OK
AT+WFOTA=0
OK
+WEVENT:"FOTA_VERSION","10.10.10","10.10.10"
```

< Set new firmware information without fota.json file >

```
AT+WFOTA=0,"https://192.168.200.1:4443","nrc7394_atcmd_hspi.bin",0x3e47cf92
OK
AT+WFOTA?
+WEVENT:0,"https://192.168.200.1:4443","nrc7394_atcmd_hspi.bin",0x3E47CF92
OK
```

< Download the firmware binary >

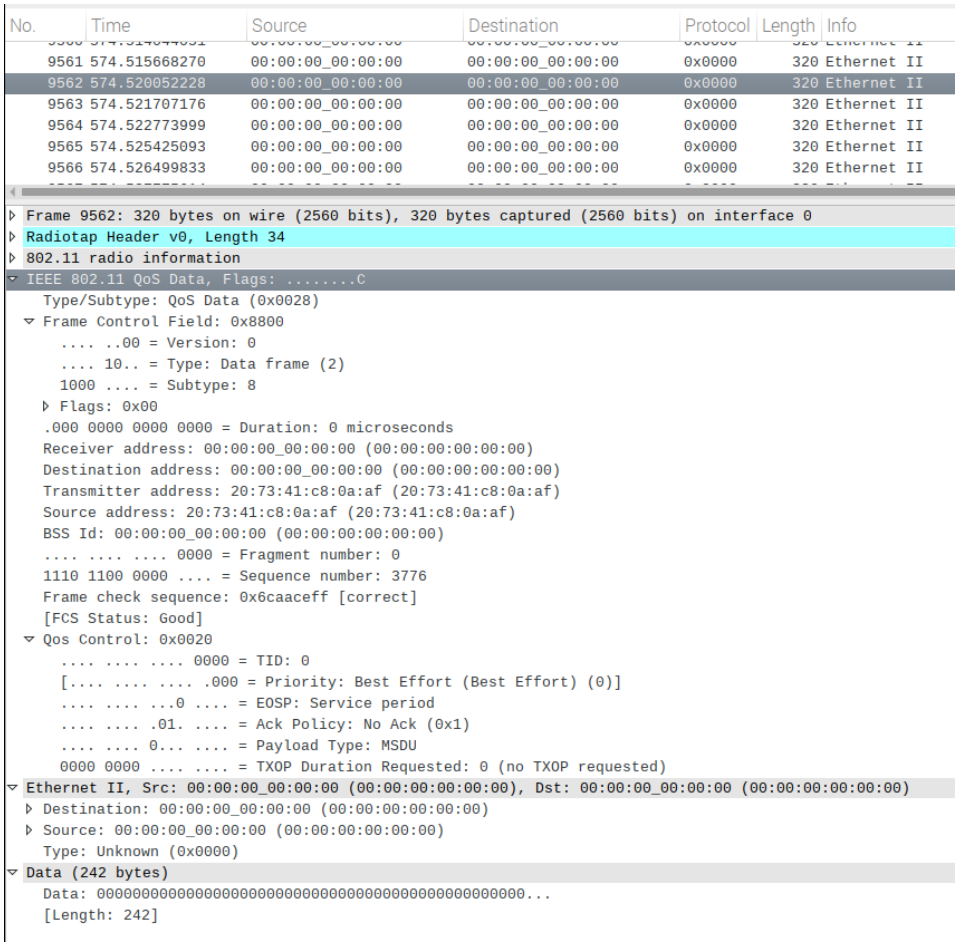
```
AT+WFOTA
OK
+WEVENT:"FOTA_BINARY","nrc7394_atcmd_hspi.bin"
+WEVENT:"FOTA_DOWNLOAD",897632,90112
+WEVENT:"FOTA_DOWNLOAD",897632,180224
+WEVENT:"FOTA_DOWNLOAD",897632,270336
                    :
+WEVENT:"FOTA_DOWNLOAD",897632,720896
+WEVENT:"FOTA_DOWNLOAD",897632,811008
+WEVENT:"FOTA_DOWNLOAD",897632,897632
```

	+WEVENT:"FOTA_UPDATE" **< Reset and update >** ATZ

6.24 AT+WCTX

Command	**RUN** AT+WCTX **SET** AT+WCTX=<frequency>,<bandwidth>,<mcs>,<txpower> **GET** AT+WCTX?
Response	**RUN/SET** OK **GET** +WCTX: <frequency>,<bandwidth>,<mcs>,<txpower> OK
Parameters	**<frequency>** Channel frequency in units of 100 KHz **<bandwidth>** S1G channel bandwidth (1, 2 and 4 MHz) **<mcs>** Modulation Coding Scheme index (0, 1, 2, 3, 4, 5, 6, 7 and 10) **<txpower>** Transmission Power Level (1 ~ 30 dBm)
Description	Send dummy data frames for continuous TX without connecting to AP. NOTE: This command is for testing purposes only.

Format of dummy data frame captured with Wireshark :



Example

AT+WCOUNTRY=”US”

OK

< Set parameters for continuous TX >

AT+WCTX=9180,4,7,17

OK

AT+WCTX?

+WCTX:9180,4,7,17

OK

< Start continuous TX >

AT+WCTX

	OK **< Stop continuous TX >** AT+WCTX=0 OK

6.25 AT+WTIMEOUT

Command	**SET** AT+WTIMEOUT=”<command>”,<timeout> **GET** AT+WTIMEOUT?
Response	**SET** OK **GET** +WTIMEOUT:”<command>”,<timeout> … OK
Parameters	**<command>** “WCONN”, “WDISCONN”, “WDHCP” **<timeout>** Timeout in seconds. (0: no timeout)
Description	Configure the response timeout for the specified command. Default timeout : - WCONN : 60 secs - WDISCONN : 60 secs - WDHCP : 60 secs
Example	AT+WTIMEOUT? +WTIMEOUT:″WCONN″,60 +WTIMEOUT:″WDISCONN″,60 +WTIMEOUT:″WDHCP″,60 OK AT+WTIMEOUT=″WCONN″,120

	OK AT+WTIMEOUT? +WTIMEOUT:"WCONN",120 +WTIMEOUT:"WDISCONN",60 +WTIMEOUT:"WDHCP",60 OK

6.26 +WEVENT

Response	+WEVENT:<event>
Parameters	**<event>** "CONNECT_SUCCESS", "<bssid>","<ssid>","<security>" "DISCONNECT", "<bssid>","<ssid>" ,"<security>" "DHCP_START" "DHCP_STOP" "DHCP_BUSY" "DHCP_FAIL" "DHCP_SUCCESS","<address>","<netmask>","<gateway>" "DHCP_TIMEOUT",<time> "STA_CONNECT","<mac_addr>" "STA_DISCONNECT","<mac_addr>" "FOTA_VERSION","<sdk_version>","<atcmd_version>" "FOTA_BINARY","<binary_name>" "FOTA_DOWNLOAD","total_size","download_size" "FOTA_UPDATE" "FOTA_FAIL" "DEEPSLEEP_WAKEUP"
Description	Asynchronously raised Wi-Fi event logs.
Example	+WEVENT:"CONNECT_SUCCESS","8c:0f:fa:00:2b:a1","halow_sae","wpa3-sae" +WEVENT:"DISCONNECT","8c:0f:fa:00:2b:a1","halow_sae","wpa3-sae"

	+WEVENT:"DHCP_START" +WEVENT:"DHCP_STOP" +WEVENT:"DHCP_BUSY" +WEVENT:"DHCP_FAIL" +WEVENT:"DHCP_SUCCESS","192.168.200.18","255.255.255.0","192.168.200.1" +WEVENT:"DHCP_TIMEOUT",60 +WEVENT:"STA_CONNECT","8C:0F:FA:00:39:0D" +WEVENT:"STA_DISCONNECT","8C:0F:FA:00:39:0D" +WEVENT:"FOTA_VERSION","10.10.10","10.10.10" +WEVENT:"FOTA_BINARY","nrc7292_atcmd_hspi.bin" +WEVENT:"FOTA_DOWNLOAD",897632,90112 +WEVENT:"FOTA_UPDATE" +WEVENT:"FOTA_FAIL" +WEVENT:”DEEPSLEEP_WAKEUP”

7 Socket AT Commands

Commands	Description
AT+SOPEN	Create a TCP/UDP socket for IPv4 domain.
AT+SCLOSE	Close an existing socket.
AT+SLIST	List all currently open sockets.
AT+SSEND	Send data through a socket.
AT+SRECV	Read buffered data from the network stack (lwip).
AT+SRECVMODE	Configures how data is read from the network stack (lwip).
AT+SRECVINFO	Configure the information level of “+RXD” message.
AT+SADDRINFO	Check the IP address from the domain name.
AT+STCPKEEPALIVE	Enable or disable TCP keepalive.
AT+STCPNODELAY	Enable or disable TCP Nagle’s algorithm.
AT+STIMEOUT	Configure the response timeout for the specified socket command.
+SEVENT	Asynchronously raised socket event logs.
+RXD	An event log for a received packet with payload.

7.1 AT+SOPEN

Command	**SET** AT+SOPEN=”udp”,<local_port>[,<reuse_addr>] AT+SOPEN=”tcp”,<local_port>[,<reuse_addr>] AT+SOPEN=”tcp”,”<server address>”,<server port>[,<reuse_addr>]
Response	**SET** +SOPEN=<socket ID> OK
Parameters	**<local_port> (UDP)** The outgoing local port. **<local_port> (TCP Server)** Local port to listen on. **<server address>,<server port> (TCP Client)** The IPv4 address and port number of the TCP server. **<reuse_addr>** SO_REUSEADDR option (0:disable, 1:enable) **<socket ID>** The ID allocated to the socket.
Description	Create a TCP/UDP socket for IPv4 domain. A socket for TCP server will listen on the given port in the background and asynchronously raise the event CONNECT to notify incoming connections.
Example	AT+SOPEN=”UDP”,60000 +SOPEN=0 OK AT+SOPEN=”TCP”,50000 +SOPEN=1 OK +SEVENT: “CONNECT”,2 AT+SOPEN=”TCP”,”192.168.200.100”,5001 +SOPEN=3

	OK

7.2 AT+SCLOSE

Command	**SET** AT+SCLOSE=<socket ID> **RUN** AT+SCLOSE
Response	**SET** +SCLOSE:<socket ID> OK **RUN** +SCLOSE:<socket ID> : +SCLOSE:<socket ID> OK
Parameters	**<socket ID>** The ID allocated to the socket.
Description	Close an existing socket. To close all existing sockets, run a command without the parameter <socket ID>. If a server socket is closed, all client sockets connected to the server socket will close automatically.
Example	AT+SCLOSE=1 +SCLOSE:1 OK AT+SCLOSE +SCLOSE:0 +SCLOSE:2 +SCLOSE:3 OK

7.3 AT+SLIST

Command	**GET** AT+SLIST?
Response	**GET** +SLIST:<socket ID>,”<protocol>”,”<remote address>”,<remote port>,<local port> :

	+SLIST:<socket ID>,"<protocol>","<remote address>",<remote port>,<local port> OK
Parameters	**<socket ID>** The ID allocated to the socket. **<protocol>** TCP or UDP **<remote address>,<remote port>,<local port>** The remote address, remote port and local port associated with the socket.
Description	List all currently open sockets.
Example	AT+SLIST? +SLIST:0,"UDP","0.0.0.0",0,60000 +SLIST:1,"TCP","0.0.0.0",0,50000 +SLIST:2,"TCP","192.168.200.100",55354,0 +SLIST:3,"TCP","192.168.200.100",5001,52433 OK

7.4 AT+SSEND

Command	**SET** AT+SSEND =<ID>[,<length>[,<done_event>]] AT+SSEND =<ID>,"<remote host>", <remote port>[,<length>[,<done_event>]]
Response	**SET** OK
Parameters	**<ID>** The ID allocated to the socket. **<remote host> (UDP only)** IPv4 address or domain name of the UDP server/client. **<remote port> (UDP only)** Port number of the UDP server/client. **<length>** The (signed) number of raw bytes to send. (See the description) **<done_event>**

	SEND_DONE event. (0:disable, 1:enable)
Description	Send data through a socket. In synchronous send mode, the value of the <length> parameter must be positive, and its maximum value is 2048. The payload byte sequence of <length> bytes must be directly followed by “AT+SSEND=<socket ID>,<length>\r\n”. The payload byte sequence does not have to be followed by “\r” or “\n” and the next payload byte sequence can be sent again after receiving the “OK\r\n\ response code from the firmware. In normal passthrough send mode, the value of the <length> parameter must be 0, so that the command takes the form “AT+SSEND=<socket ID>,0\r\n”. As soon as the firmware receives the command, the firmware enters the active passthrough state; all bytes fed into the AT stream is redirected to the associated socket stream. To exit the passthrough state, no byte should be fed into the AT stream for the duration of SSEND timeout duration in seconds (default: 1 second) to transition the active passthrough state to the idle passthrough state. The transition is notified by the +SEVENT:”SEND_IDLE” event. Upon receiving the idle event notification, the four magic bytes “AT\r\n” should be fed into the AT stream to exit the passthrough state. The magic bytes themselves will not be regarded as part of the payload as long as they are fed into the AT stream following the idle event notification, but if the characters following the idle event notification are different from the magic bytes, the fed bytes will indeed be regarded as part of the payload. The +SEVENT:”SEND_EXIT” event is raised upon exiting the passthrough mode. In buffered passthrough send mode, the value of the <length> parameter must be positive, and its maximum value is 2048. The command takes the form “AT+SSEND=<socket ID>,-<length>\r\n”, with the “-” sign preceding the <length> parameter. The buffered passthrough mode operates similarly to the normal passthrough mode. However, unlike the normal passthrough mode, the firmware maintains an internal byte buffer of size <length> and transfers the buffered byte onto the send queue only when the byte buffer is full. However, using this mode still does not guarantee that the receiver will always receive the payload in <length> bytes without fragmentation, as other factors such as the MTU size limit and other implementation-dependent features may affect the payload transfer process differently. NOTE: UART without HFC supports only normal send mode.
Example	**[Synchronous Mode]**

	AT+SSEND=0,6 OK Hello! AT+SSEND=0,6,1 OK Hello! +SEVENT:”SEND_DONE”,6 **[Passthrough Mode]** AT+SSEND=0 Hello, World! Goodbye, World! [Wait for SSEND timeout duration to change the internal state to receive magic bytes and exit the continuous transmission state] +SEVENT:”SEND_IDLE”,0,28,0,0 AT OK +SEVENT:”SEND_EXIT”,0,28,0 **[Buffered Passthrough Mode]** AT+SSEND=0,-8,1 TEST0001 +SEVENT:”SEND_DONE”,8 TEST0002 +SEVENT:”SEND_DONE”,8 +SEVENT:”SEND_IDLE”,0,16,0,0 TEST0003 +SEVENT:”SEND_DONE”,8 +SEVENT:”SEND_IDLE”,0,24,0,0 AT OK +SEVENT:”SEND_EXIT”,0,24,0

7.5 AT+SRECV

Command	**SET** AT+SRECV=<socket ID>[,<length>] **GET** AT+SRECV?
Response	**SET** OK **GET** +SRECV:<socket_ID>,<bufferd_length> … OK
Parameters	**<socket ID>** The ID allocated to the socket. **<length>** The maximum number of raw bytes to read **<bufferd_length>** The number of raw bytes currently buffered If omitted or set to 0, it is set to the maximum value supported by the firmware.
Description	Read buffered data from the network stack (lwip). NOTE: 1) AT+SRECV command can be used only when passive mode is set with AT+SRECVMODE command. 2) If it is UDP data, it will be lost when the buffer is full.
Example	AT+SLIST? +SLIST:0,"TCP","192.168.200.1",50000,0 +SLIST:1,"UDP","0.0.0.0",0,60001 OK +SEVENT:"RECV_READY",0,1024 +SEVENT:"RECV_READY",1,1024 AT+SRECV? +SRECV:0,7168 +SRECV:1,7168

	OK AT+SRECV=0 +RXD:0,4096,"192.168.200.1",50000 OK AT+SRECV=1 +RXD:1,1024,"192.168.200.1",60000 OK +SEVENT:"RECV_READY",0,3072 +SEVENT:"RECV_READY",1,6144

7.6 AT+SRECVMODE

Command	**SET** AT+SRECVMODE=<mode>[,<event>] **GET** AT+SRECVMODE?
Response	**SET** OK **GET** +SRECVMODE:<mode>,<event> OK
Parameters	**<mode>** 0 : active* 1 : passive **<event>** 0 : ready event disable 1 : ready event enable*
Description	Configures how data is read from the network stack (lwip). If the event parameter is set to 1 in passive mode, a RECV_READY event occurs when there is buffered data. The event does not occur again until the buffered data is read with the AT+SRECV command.
Example	AT+SRECVMODE=1 OK AT+SRECVMODE? +SRECVMODE:1,0

	OK AT+SRECVMODE=1,1 OK AT+SRECVMODE? +SRECVMODE:1,1 OK AT+SRECVMODE=0 OK AT+SRECVMODE? +SRECVMODE:0,0 OK

7.7 AT+SRECVINFO

Command	**SET** AT+SRECVINFO=<mode> **GET** AT+SRECVINFO?
Response	**SET** OK **GET** +SRECVINFO:<mode> OK
Parameters	**<mode>** 0 : terse* 1 : verbose
Description	Configure the information level of “+RXD” message. NOTE: The AT+SRECVINFO command is the same as the previous AT+SRXLOGLEVEL command. Only the command name is different.
Example	AT+SRECVINFO =1 OK AT+SRECVINFO? + SRECVINFO:1

	OK

7.8 AT+SADDRINFO

Command	**SET** AT+SADDRINFO=”<domain_name>”
Response	**SET** +SADDRINFO:”<address>” OK
Parameters	**<domain_name>** Domain name **<address>** IPv4 address
Description	Check the IP address from the domain name.
Example	AT+SADDRINFO =”www.google.com” +SADDRINFO:”142.250.199.100” OK

7.9 AT+STCPKEEPALIVE

Command	**SET** AT+STCPKEEPALIVE=<socket ID>,<keepalive>[,<keepidle>,<keepcnt>,<keepintvl>] **GET** AT+STCPKEEPALIVE? AT+STCPKEEPALIVE?=<socket ID>
Response	**SET** OK **GET** +STCPKEEPALIVE:<socket_ID>,<keepalive>,<keepidle>,<keepcnt>,<keepintvl> : OK
Parameters	**<socket ID>** The ID allocated to the socket for TCP client. **<keepalive>** 0 : disable 1 : enable

	<keepidle> The time to wait before sending out the first probe in seconds. (default : 7200) **<keepcnt>** The number of probes that are sent and unacknowledged. (default : 9) **<keepintvl>** The interval between subsequent keepalive probes in seconds. (default : 75)
Description	Enable or disable TCP keepalive.
Example	**< TCP Server >** AT+SOPEN=”TCP”,50000 +SOPEN=0 OK +SEVENT:”CONNECT”,1 AT+SLIST? +SLIST:0,”TCP”,”0.0.0.0”,0,50000 +SLIST:1,”TCP”,”192.168.200.2”,52432,0 OK AT+STCPKEEPALIVE? +STCPKEEPALIVE:1,0,7200,9,75 OK AT+STCPKEEPALIVE=1,0,60,5,30 OK AT+STCPKEEPALIVE? +STCPKEEPALIVE:1,0,60,5,30 OK AT+STCPKEEPALIVE=1,1 OK AT+STCPKEEPALIVE? +STCPKEEPALIVE:1,1,60,5,30 OK **< TCP Client >** AT+SOPEN=”TCP”,”192.168.200.1”,50000 +SOPEN:0 OK AT+SLIST? +SLIST:0,”TCP”,”192.168.200.1”,50000,0 OK AT+STCPKEEPALIVE?

	+STCPKEEPALIVE:0,0,7200,9,75 OK AT+STCPKEEPALIVE=0,1,60,5,30 OK AT+STCPKEEPALIVE?=0 +STCPKEEPALIVE:0,1,60,5,30 OK

7.10 AT+STCPNODELAY

Command	**SET** AT+STCPNODELAY=<socket ID>,{0|1} **GET** AT+STCPNODELAY?
Response	**SET** OK **GET** +STCPNODELAY:<socket_ID>,<status> OK
Parameters	**<socket ID>** The ID allocated to the socket. **<status>** 0 : disable 1 : enable
Description	Enable or disable TCP Nagle's algorithm.
Example	**< TCP Server >** AT+SOPEN="TCP",50000 +SOPEN=0 OK +SEVENT:"CONNECT",1 AT+SLIST? +SLIST:0,"TCP","0.0.0.0",0,50000 +SLIST:1,"TCP","192.168.200.2",52432,0 OK AT+STCPNODELAY? +STCPNODELAY:1,0

	OK AT+STCPNODELAY=1,1 OK AT+STCPNODELAY? +STCPNODELAY:1,1 OK **< TCP Client >** AT+SOPEN=”TCP”,”192.168.200.1”,50000 +SOPEN:0 OK AT+SLIST? +SLIST:0,”TCP”,”192.168.200.1”,50000,0 OK AT+STCPNODELAY? +STCPNODELAY:0,0 OK AT+STCPNODELAY=0,1 OK AT+STCPNODELAY? +STCPNODELAY:0,1 OK

7.11 AT+STIMEOUT

Command	**SET** AT+STIMEOUT=”<command>”,<timeout> **GET** AT+STIMEOUT?
Response	**SET** OK **GET** +STIMEOUT:”<command>”,<timeout> … OK
Parameters	**<command>** “SOPEN”, “SSEND” **<timeout>** Timeout in seconds. (0 : no timeout)

Description	Configure the response timeout for the specified socket command. Default timeout : - SOPEN : 30 secs - SSEND : 1 sec
Example	AT+STIMEOUT? +STIMEOUT:"SOPEN",30 +STIMEOUT:"SSEND",1 OK AT+STIMEOUT="SOPEN",60 OK AT+STIMEOUT="SSEND",3 OK AT+STIMEOUT? +STIMEOUT:"SOPEN",60 +STIMEOUT:"SSEND",3 OK

7.12 +SEVENT

Response	+SEVENT:<event>,<socket ID>[,<parameter 1>,…,<parameter N>]
Parameters	**<event>** "CONNECT",<socket ID> "CLOSE",<socket ID>,<error>,"<description>" "SEND_DONE",<socket ID>,<done> "SEND_DROP",<socket ID>,<drop> "SEND_IDLE",<socket ID>,<done>,<drop>,<wait> "SEND_EXIT",<socket ID>,<done>,<drop> "SEND_ERROR",<socket ID>,<error>,"<description>" "RECV_READY",<socket ID>,<length> "RECV_ERROR",<socket ID>,<error>,"<description>" **<socket ID>** Socket ID

	<done> The length of the sent payload. **<drop>** The length of the dropped payload. **<wait>** The length of the buffered payload. **<length>** The length of the receivable payload. **<error>** error code **<description>** string describing the error code NOTE: The error code may not match the POSIX error code. The error code defined in the errno.h file included in the ARM Toolchain is different from the POSIX error code.
Description	Asynchronously raised socket event logs.
Example	+SEVENT:”CONNECT”,1 +SEVENT:”CLOSE”,1,128,"Socket is not connected" +SEVENT:”SEND_DONE”,1,152 +SEVENT:”SEND_DROP”,1,152 +SEVENT:”SEND_IDLE”,1,1500,152,200 +SEVENT:”SEND_EXIT”,1,1700,152 +SEVENT:”SEND_ERROR”,1, 104,”Connection reset by peer” +SEVENT:”RECV_READY”,1,1488 +SEVENT:”RECV_ERROR”,1, 128,”Socket is not connected”

7.13 +RXD

Response	**RX Log Level (Terse)** +RXD:<socket ID>,<actual read length> <raw bytes> **RX Log Level (Verbose)** +RXD:<socket ID>,<actual read length>,"<remote IP>",<remote port> <raw bytes>
Parameters	**<socket ID>** The ID allocated to the socket. **<max read length>** The maximum number of bytes to read. (Max: 2048) **<actual read length>** Actual number of bytes read. **<remote IP>,<remote port>** The remote IP and port. **<raw bytes>** The received raw bytes (0x00~0xFF) payload.
Description	An event log for a received packet with payload. Upon receiving packets, +RXD event logs will automatically appear on the terminal output. Note that there will be no 'OK' message following the event log.
Example	**RX Log Level (Terse)** +RXD=0,15 ABCDE12345,.?=+ **RX Log Level (Verbose)** +RXD=0,12,"192.168.200.1",5025 HELLO,WORLD!

8 Test Application

8.1 Command Line Interface (raspi-atcmd-cli)

CLI application is a Linux program running on Raspberry Pi for AT-command communication via UART or SPI. In the CLI application, as in terminal program via UART, the user can enter the AT command and check the response to the command.

8.1.1 Source files

File	Description
common.h	Common header file
main.c	CLI related functions.
Makefile	Make file for building.
nrc-atcmd.c nrc-atcmd.h	AT command handler
nrc-hspi.c nrc-hspi.h	Protocol driver for HSPI. ***Refer to this file to communicate with the ATCMD firmware via HSPI.**
nrc-iperf.c nrc-iperf.h	Iperf server/client
raspi-hif.c raspi-hif.h	Wrapper for user mode driver.
raspi-eirq.c	User mode driver for GPIO EIRQ.
raspi-spi.c	User mode driver for SPI.
raspi-uart.c	User mode driver for UART.
scripts/	Script files

Table 8.1 raspi-atcmd-cli source files

8.1.2 Build

Copy the “atcmd/host/raspi-atcmd-cli” directory to the Raspberry Pi's home directory. And build the CLI application with the make command.

```
$ cd $HOME

$ cd raspi-atcmd-cli
```

$ make clean

```
removed 'raspi-atcmd-cli'
```

$ make

```
cc -g -o raspi-atcmd-cli raspi-spi.c raspi-uart.c raspi-eirq.c raspi-hif.c nrc-hspi.c nrc-atcmd.c nrc-iperf.c main.c
-pthread -Wall    -lpthread
```

8.1.3 Run

- **Help**

  $ ./raspi-atcmd-cli [-h|--help]

```
raspi-atcmd-cli version 1.3.3
Copyright (c) 2019-2023    <NEWRACOM LTD>

Usage:
   $ ./raspi-atcmd-cli -S [-D <device>] [-E <trigger>] [-c <clock>] [-s <script> [-n]]
   $ ./raspi-atcmd-cli -U [-D <device>] [-b <baudrate>] [-s <script> [-n]]
   $ ./raspi-atcmd-cli -U -f [-D <device>] [-b <baudrate>] [-s <script> [-n]]

UART/SPI:
   -D, --device #          Specify the device. (default: /dev/spidev0.0, /dev/ttyAMA0)
   -s, --script #          Specify the script file.
   -n, --noexit #          Do not exit the script when the AT command responds with an error.

SPI:
   -S    --spi             Use the SPI to communicate with the target.
   -E, --eirq #            Use EIRQ mode for the SPI. (0:low, 1:high, 2:falling, 3:rising)
   -c, --clock #           Specify the clock frequency for the SPI. (default: 20000000 Hz)

UART:
   -U    --uart            Use the UART to communicate with the target.
   -f    --flowctrl        Enable RTS/CTS signals for the hardware flow control on the UART. (default: off)
   -b, --baudrate #        Specify the baudrate for the UART. (default: 115200 bps)

Miscellaneous:
   -v, --version           Print version information and quit.
   -h, --help              Print this message and quit.
```

- **SPI**

  The maximum clock frequency is 20MHz.

  $ sudo ./raspi-atcmd-cli **-S [-D <device>] [-E <trigger>] [-c <clock>] [-s <script> [-n]]**

```
$ sudo ./raspi-atcmd-cli -S -c 20000000 -E 2

  [ SPI ]
    - device: /dev/spidev0.0
    - clock: 20000000 Hz
    - eirq: falling


#
```

- **UART**

  The maximum baud rate is 115,200bps without the hardware flow control.

  $ sudo ./raspi-atcmd-cli **-U [-D <device>] [-b <baudrate>] [-s <script> [-n]]**

```
$ sudo ./raspi-atcmd-cli -U -b 115200

  [ UART ]
    - device: /dev/ttyAMA0
    - baudrate : 115200


#
```

- **UART_HFC**

  If the baud rate setting is more than 115,200bps, the hardware flow control needs to be enabled with -f option on the UART.

  $ sudo ./raspi-atcmd-cli **-U -f [-D <device>] [-b <baudrate>] [-s <script> [-n]]**

```
$ sudo ./raspi-atcmd-cli -U -f -b 2000000

  [ UART_HFC ]
    - device: /dev/ttyAMA0
    - baudrate : 2000000


#
```

- **Examples**

  Getting the informations.

```
# AT
  SEND: AT
```

```
  RECV: OK

# AT+VER?
  SEND: AT+VER?
  RECV: +VER:"1.0.0","1.23.5"
  RECV: OK

# AT+WMACADDR?
  SEND: AT+WMACADDR?
  RECV: +WMACADDR:"8c:0f:fa:00:29:43"
  RECV: OK

# AT+WCOUNTRY?
  SEND: AT+WCOUNTRY?
  RECV: +WCOUNTRY:"US"
  RECV: OK

# AT+WTXPOWER?
  SEND: AT+WTXPOWER?
  RECV: +WTXPOWER:17
  RECV: OK

# AT+WRATECTRL?
  SEND: AT+WRATECTRL?
  RECV: +WRATECTRL:1
  RECV: OK

# AT+WIPADDR?
  SEND: AT+WIPADDR?
  RECV: +WIPADDR:"0.0.0.0","0.0.0.0","0.0.0.0"
  RECV: OK
```

Connecting to an AP.

```
# AT+WCONN?
  SEND: AT+WCONN?
  RECV: +WCONN:"halow","00:00:00:00:00:00","open","","disconnected"
  RECV: OK

# AT+WSCAN
  SEND: AT+WSCAN
  RECV: +WSCAN:"8c:0f:fa:00:28:1f",906.0,-39,"[WPA3-SAE-CCMP][ESS]","halow_atcmd_sae"
  RECV: +WSCAN:"8c:0f:fa:00:28:11",925.0,-68,"[WPA3-OWE-CCMP][ESS]","halow_fota"
  RECV: +WSCAN:"8c:0f:fa:00:28:1e",903.5,-93,"[ESS]","halow_s1g_demo_open"
  RECV: OK

# AT+WCONN="halow_atcmd_sae","sae","12345678"
  SEND: AT+WCONN="halow_atcmd_sae","sae","12345678"
  RECV: OK
```

```
# AT+WCONN?
  SEND: AT+WCONN?
  RECV: +WCONN:"halow_atcmd_sae","8c:0f:fa:00:28:1f","wpa3-sae","12345678","connected"
  RECV: OK

# AT+WDHCP
  SEND: AT+WDHCP
  RECV: +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1"
  RECV: OK

# AT+WIPADDR?
  SEND: AT+WIPADDR?
  RECV: +WIPADDR:"192.168.200.18","255.255.255.0","192.168.200.1"
  RECV: OK

# AT+WPING="192.168.200.1"
  SEND: AT+WPING="192.168.200.1"
  RECV: +WPING:64,"192.168.200.1",1,64,5
  RECV: +WPING:64,"192.168.200.1",2,64,5
  RECV: +WPING:64,"192.168.200.1",3,64,149
  RECV: +WPING:64,"192.168.200.1",4,64,4
  RECV: +WPING:64,"192.168.200.1",5,64,5
  RECV: OK
```

Sending and receiving the data with a socket for TCP client.

```
# AT+SOPEN="TCP","192.168.200.1",50000
  SEND: AT+SOPEN="TCP","192.168.200.1",50000
  RECV: +SOPEN:0
  RECV: OK

# AT+SLIST?
  SEND: AT+SLIST?
  RECV: +SLIST:0,"TCP","192.168.200.1",50000,52432
  RECV: OK

# AT+SSEND=0,10
  SEND: AT+SSEND=0,10
  RECV: OK

# ABCDEFGHIJKLMNOPQRSTUVWXYZ
  SEND: DATA 10

#    RECV: +RXD:0,10

# AT+SSEND=0
  SEND: AT+SSEND=0
  RECV: OK
```

```
# DAJFKDAJFKDAJFDKAJFAKFJDK
  SEND: DATA 25

#    RECV: +RXD:0,25
  RECV: +SEVENT:"SEND_IDLE",0,25,0,0


# DKAJFKDAJFEKJAFKDJFADKJFAKDJFAKEJFKADJFAKEJFKAJDFKDJAFDKJFADK
  SEND: DATA 61

#    RECV: +RXD:0,61
  RECV: +SEVENT:"SEND_IDLE",0,86,0,0


# AT
  SEND: AT
  RECV: OK

#    RECV: +SEVENT:"SEND_EXIT",0,86,0
```

Closing all sockets.

```
# AT+SLIST?
  SEND: AT+SLIST?
  RECV: +SLIST:0,"TCP","192.168.200.1",50000,52432
  RECV: OK

# AT+SCLOSE
  SEND: AT+SCLOSE
  RECV: +SCLOSE:0
  RECV: OK

# EXIT
```

8.1.4 Run with a script

CLI application provides the option to run the script file. (-s/--script)

```
UART/SPI:
   -s, --script #          Specify the script file.
   -n, --noexit #          Do not exit the script when the AT command responds with an error.
```

The script file can be created using the AT command and script command.

Command	Description	Example

CALL <script_file>	Read and run the specified script file.	CALL wifi_connect CALL wifi/connect
LOOP <line> <count>	Repeat next lines. <line>: number of lines to repeat <count>: number of repetitions.	LOOP 2 5 AT+SSEND=0,1024 DATA 1024
DATA <length>	Send payload with random value.	DATA 1024
WAIT <time>{s|m|u}	Wait for the specified time. s: sec m: msec u: usec	WAIT 1s WAIT 1000m WAIT 100u
ECHO "<message>"	Print a message.	ECHO "AT Command"
TIME	Print current time.	TIME
HOLD	Pause until there is keyboard input.	ECHO "Run an AP in open mode" HOLD
EXIT	Exit script.	EXIT

Users can refer to the script files under the "raspi-atcmd-cli/scripts" directory.

```
raspi-atcmd-cli/scripts/
 ├──── socket-tcp-client-send
 ├──── socket-tcp-client-send-passthrough
 ├──── socket-tcp-client-send-passthrough-buffered
 ├──── socket-tcp-server
 ├──── socket-tcp-server-send
 ├──── socket-tcp-server-send-passthrough
 ├──── socket-tcp-server-send-passthrough-buffered
 ├──── socket-udp-client-send
 ├──── socket-udp-client-send-passthrough
 ├──── socket-udp-client-send-passthrough-buffered
 ├──── socket-udp-server
 ├──── socket-udp-server-send
 ├──── socket-udp-server-send-passthrough
 ├──── socket-udp-server-send-passthrough-buffered
 ├──── softap-tcp-client-send-normal
 ├──── softap-tcp-client-send-passthrough
```

```
├─── softap-tcp-server
├─── softap-udp-client-send-normal
├─── softap-udp-client-send-passthrough
├─── softap-udp-server
├─── sta-tcp-client-send-normal
├─── sta-tcp-client-send-passthrough
├─── sta-tcp-server
├─── sta-udp-client-send-normal
├─── sta-udp-client-send-passthrough
├─── sta-udp-server
├─── wifi-connect-open-dhcp-auto-kr-mic
├─── wifi-connect-open-dhcp-auto-us
├─── wifi-connect-open-dhcp-kr-mic
├─── wifi-connect-open-dhcp-kr-usn
├─── wifi-connect-open-dhcp-us
├─── wifi-connect-wpa2-psk-dhcp-auto-kr-mic
├─── wifi-connect-wpa2-psk-dhcp-auto-us
├─── wifi-connect-wpa2-psk-dhcp-kr-mic
├─── wifi-connect-wpa2-psk-dhcp-us
├─── wifi-connect-wpa3-owe-dhcp-auto-kr-mic
├─── wifi-connect-wpa3-owe-dhcp-auto-us
├─── wifi-connect-wpa3-owe-dhcp-kr-mic
├─── wifi-connect-wpa3-owe-dhcp-us
├─── wifi-connect-wpa3-sae-dhcp-auto-kr-mic
├─── wifi-connect-wpa3-sae-dhcp-auto-us
├─── wifi-connect-wpa3-sae-dhcp-kr-mic
├─── wifi-connect-wpa3-sae-dhcp-us
├─── wifi-softap-open-dhcps-kr-mic
├─── wifi-softap-open-dhcps-kr-usn
├─── wifi-softap-open-dhcps-us
├─── wifi-softap-wpa2-psk-dhcps-kr-mic
└─── wifi-softap-wpa2-psk-dhcps-us
```

8.1.5 Iperf

The CLI application supports the iperf2 command used for network performance measurement. However, the available options are limited as shown below.

iperf {-h|--help}

```
Usage: iperf {-s}|{-c <host>} [options]

Client/Server:
    -i, --interval #          seconds between periodic bandwidth reports (default: 1 sec)
    -p, --port #                server port to listen on/connect to (default: 5001)
    -u, --udp                     use UDP rather than TCP

Server specific:
    -s, --server                run in server mode

Client specific:
    -c, --client <host>     run in client mode, connecting to <host>
    -t, --time #                time in seconds to transmit for (default: 10 sec)
    -P, --passthrough         transmit in passthrough mode
    -N, --negative              use negative length for buffered passthrough mode (always negative in UDP)
    -D, --done_vent            enable SEND_DONE event


Miscellaneous:
    -h, --help                    print this message and quit
```

The iperf command can be run after completing the Wi-Fi connection and IP setup.

Wi-Fi connection and IP setup can be done in one of two ways:

- Enter AT command in the CLI application.

```
# AT+WSCAN
  SEND: AT+WSCAN
  RECV: +WSCAN:"8c:0f:fa:00:28:1f",914.0,-38,"[WPA3-SAE-CCMP][ESS]","halow_atcmd_sae"
  RECV: OK

# AT+WCONN="halow_atcmd_sae","sae","12345678"
  SEND: AT+WCONN="halow_atcmd_sae","sae","12345678"
  RECV: OK

# AT+WDHCP
  SEND: AT+WDHCP
  RECV: +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1"
  RECV: OK
```

- Specify a script file containing AT command with the -s option when running the CLI application.

  $ sudo ./raspi-atcmd-cli -S -s scripts/example/wifi-connect-wpa3-sae-dhcp

```
CALL: scripts/examples/wifi-connect-wpa3-sae-dhcp

  SEND: AT
  RECV: OK
  SEND: AT+WDISCONN
  RECV: OK

  ECHO: Run an AP in WPA3-SAE.
  ECHO:    - SSID : halow_atcmd_sae
  ECHO:    - Password : 12345678
  ECHO:    - IP : 192.168.200.1
  ECHO:    - DHCP Server
  HOLD: Press ENTER to continue.

  SEND: AT+WSCAN
  RECV: +WSCAN:"8c:0f:fa:00:28:1f",906.0,-39,"[WPA3-SAE-CCMP][ESS]","halow_atcmd_sae"
  RECV: OK
  SEND: AT+WDISCONN
  RECV: OK
  SEND: AT+WCONN="halow_atcmd_sae","wpa3-sae","12345678"
  RECV: OK
  SEND: AT+WCONN?
  RECV: +WCONN:"halow_atcmd_sae","8c:0f:fa:00:28:1f","wpa3-sae","12345678","connected"
  RECV: OK
  SEND: AT+WDHCP
  RECV: +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1"
  RECV: OK


DONE: scripts/examples/wifi-connect-wpa3-sae-dhcp
```

- **Iperf TCP Client/Server**

```
# iperf -c 192.168.200.1

  [ IPERF OPTION ]
    - role: client
    - protocol: tcp
    - server_port: 5001
    - server_ip: 192.168.200.1
    - send_length: 1440
    - send_time: 10
    - send_passthrough: off
```

```
  - send_done_event: 0
  - report_interval: 1


 [ IPERF TCP Client ]
  Sending 1440 byte datagram ...
    Interval            Transfer          Bandwidth
    0.0 ~  1.0 sec   187.03 KBytes    1.53 Mbits/sec
    1.0 ~  2.0 sec   192.66 KBytes    1.57 Mbits/sec
    2.0 ~  3.0 sec   191.25 KBytes    1.56 Mbits/sec
    3.0 ~  4.0 sec   194.06 KBytes    1.59 Mbits/sec
    4.0 ~  5.0 sec   191.25 KBytes    1.56 Mbits/sec
    5.0 ~  6.0 sec   194.06 KBytes    1.58 Mbits/sec
    6.0 ~  7.0 sec   195.47 KBytes    1.59 Mbits/sec
    7.0 ~  8.0 sec   192.66 KBytes    1.57 Mbits/sec
    8.0 ~  9.0 sec   191.25 KBytes    1.56 Mbits/sec
    9.0 ~ 10.0 sec   187.03 KBytes    1.58 Mbits/sec
    0.0 ~ 10.0 sec    1.87 MBytes    1.57 Mbits/sec
  Sent 1363 datagrams
  Done


# iperf -c 192.168.200.1 -P

 [ IPERF OPTION ]
  - role: client
  - protocol: tcp
  - server_port: 5001
  - server_ip: 192.168.200.1
  - send_length: 1440
  - send_time: 10
  - send_passthrough: on
  - send_done_event: 0
  - report_interval: 1


 [ IPERF TCP Client ]
  Sending 1440 byte datagram ...
    Interval            Transfer          Bandwidth
    0.0 ~  1.0 sec   426.09 KBytes    3.47 Mbits/sec
    1.0 ~  2.0 sec   407.81 KBytes    3.34 Mbits/sec
    2.0 ~  3.0 sec   406.41 KBytes    3.32 Mbits/sec
    3.0 ~  4.0 sec   412.03 KBytes    3.37 Mbits/sec
    4.0 ~  5.0 sec   403.59 KBytes    3.30 Mbits/sec
    5.0 ~  6.0 sec   414.84 KBytes    3.40 Mbits/sec
    6.0 ~  7.0 sec   403.59 KBytes    3.29 Mbits/sec
    7.0 ~  8.0 sec   405.00 KBytes    3.31 Mbits/sec
    8.0 ~  9.0 sec   405.00 KBytes    3.31 Mbits/sec
    9.0 ~ 10.0 sec   409.22 KBytes    3.39 Mbits/sec
    0.0 ~ 10.0 sec    4.00 MBytes    3.35 Mbits/sec
  Sent 2911 datagrams
```

```
  Done


# iperf -c 192.168.200.1 -P -N

 [ IPERF OPTION ]
  - role: client
  - protocol: tcp
  - server_port: 5001
  - server_ip: 192.168.200.1
  - send_length: 1440
  - send_time: 10
  - send_passthrough: on (-)
  - send_done_event: 0
  - report_interval: 1


 [ IPERF TCP Client ]
  Sending 1440 byte datagram ...
    Interval            Transfer          Bandwidth
    0.0 ~  1.0 sec   348.75 KBytes    2.85 Mbits/sec
    1.0 ~  2.0 sec   343.12 KBytes    2.79 Mbits/sec
    2.0 ~  3.0 sec   340.31 KBytes    2.77 Mbits/sec
    3.0 ~  4.0 sec   334.69 KBytes    2.74 Mbits/sec
    4.0 ~  5.0 sec   337.50 KBytes    2.76 Mbits/sec
    5.0 ~  6.0 sec   336.09 KBytes    2.75 Mbits/sec
    6.0 ~  7.0 sec   330.47 KBytes    2.70 Mbits/sec
    7.0 ~  8.0 sec   337.50 KBytes    2.76 Mbits/sec
    8.0 ~  9.0 sec   341.72 KBytes    2.79 Mbits/sec
    9.0 ~ 10.0 sec   330.47 KBytes    2.77 Mbits/sec
    0.0 ~ 10.0 sec    3.30 MBytes     2.77 Mbits/sec
  Sent 2404 datagrams
  Done


# iperf -s

 [ IPERF OPTION ]
  - role: server
  - protocol: tcp
  - server_port: 5001
  - report_interval: 1


 [ IPERF TCP Server ]
  Connected with client: 192.168.200.1 port 52174
    Interval            Transfer          Bandwidth
    0.0 ~  1.0 sec   415.77 KBytes    3.41 Mbits/sec
    1.0 ~  2.0 sec   424.22 KBytes    3.47 Mbits/sec
    2.0 ~  3.0 sec   428.46 KBytes    3.51 Mbits/sec
    3.0 ~  4.0 sec   435.53 KBytes    3.57 Mbits/sec
```

```
      4.0 ~    5.0 sec    425.39 KBytes      3.48 Mbits/sec
      5.0 ~    6.0 sec    424.46 KBytes      3.48 Mbits/sec
      6.0 ~    7.0 sec    439.77 KBytes      3.60 Mbits/sec
      7.0 ~    8.0 sec    418.56 KBytes      3.43 Mbits/sec
      8.0 ~    9.0 sec    425.63 KBytes      3.49 Mbits/sec
      9.0 ~ 10.0 sec    416.91 KBytes      3.42 Mbits/sec
      0.0 ~ 10.0 sec     4.15 MBytes      3.49 Mbits/sec
   Done

Press ENTER to continue or type "quit" : quit


#
```

Remote Iperf TCP Server/Client

```
$ iperf -s -i 1
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[    4] local 192.168.200.1 port 5001 connected with 192.168.200.43 port 52432
[ ID] Interval            Transfer         Bandwidth
[    4]    0.0- 1.0 sec     187 KBytes    1.53 Mbits/sec
[    4]    1.0- 2.0 sec     193 KBytes    1.58 Mbits/sec
[    4]    2.0- 3.0 sec     190 KBytes    1.56 Mbits/sec
[    4]    3.0- 4.0 sec     194 KBytes    1.59 Mbits/sec
[    4]    4.0- 5.0 sec     191 KBytes    1.57 Mbits/sec
[    4]    5.0- 6.0 sec     193 KBytes    1.58 Mbits/sec
[    4]    6.0- 7.0 sec     194 KBytes    1.59 Mbits/sec
[    4]    7.0- 8.0 sec     191 KBytes    1.57 Mbits/sec
[    4]    8.0- 9.0 sec     191 KBytes    1.57 Mbits/sec
[    4]    9.0-10.0 sec     193 KBytes    1.58 Mbits/sec
[    4]    0.0-10.0 sec    1.87 MBytes    1.57 Mbits/sec
[    5] local 192.168.200.1 port 5001 connected with 192.168.200.43 port 52433
[    5]    0.0- 1.0 sec     408 KBytes    3.34 Mbits/sec
[    5]    1.0- 2.0 sec     405 KBytes    3.32 Mbits/sec
[    5]    2.0- 3.0 sec     408 KBytes    3.34 Mbits/sec
[    5]    3.0- 4.0 sec     412 KBytes    3.37 Mbits/sec
[    5]    4.0- 5.0 sec     400 KBytes    3.28 Mbits/sec
[    5]    5.0- 6.0 sec     418 KBytes    3.42 Mbits/sec
[    5]    6.0- 7.0 sec     402 KBytes    3.30 Mbits/sec
[    5]    7.0- 8.0 sec     403 KBytes    3.30 Mbits/sec
[    5]    8.0- 9.0 sec     406 KBytes    3.32 Mbits/sec
[    5]    9.0-10.0 sec     413 KBytes    3.39 Mbits/sec
[    5] 10.0-11.0 sec    18.2 KBytes     149 Kbits/sec
[    5]    0.0-11.3 sec    4.00 MBytes    2.98 Mbits/sec
[    4] local 192.168.200.1 port 5001 connected with 192.168.200.43 port 52434
[    4]    0.0- 1.0 sec     336 KBytes    2.75 Mbits/sec
```

```
[  4]  1.0- 2.0 sec   340 KBytes  2.78 Mbits/sec
[  4]  2.0- 3.0 sec   339 KBytes  2.78 Mbits/sec
[  4]  3.0- 4.0 sec   333 KBytes  2.73 Mbits/sec
[  4]  4.0- 5.0 sec   338 KBytes  2.77 Mbits/sec
[  4]  5.0- 6.0 sec   333 KBytes  2.72 Mbits/sec
[  4]  6.0- 7.0 sec   334 KBytes  2.73 Mbits/sec
[  4]  7.0- 8.0 sec   337 KBytes  2.76 Mbits/sec
[  4]  8.0- 9.0 sec   339 KBytes  2.78 Mbits/sec
[  4]  9.0-10.0 sec   338 KBytes  2.77 Mbits/sec
[  4] 10.0-11.0 sec  15.2 KBytes   124 Kbits/sec
[  4]  0.0-11.3 sec  3.30 MBytes  2.46 Mbits/sec
```

```
$ iperf -c 192.168.200.43 -i 1
------------------------------------------------------------
Client connecting to 192.168.200.43, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[  3] local 192.168.200.1 port 52174 connected with 192.168.200.43 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  1.0- 2.0 sec   384 KBytes  3.15 Mbits/sec
[  3]  2.0- 3.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  3.0- 4.0 sec   384 KBytes  3.15 Mbits/sec
[  3]  4.0- 5.0 sec   384 KBytes  3.15 Mbits/sec
[  3]  5.0- 6.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  6.0- 7.0 sec   384 KBytes  3.15 Mbits/sec
[  3]  7.0- 8.0 sec   384 KBytes  3.15 Mbits/sec
[  3]  8.0- 9.0 sec   512 KBytes  4.19 Mbits/sec
[  3]  9.0-10.0 sec   384 KBytes  3.15 Mbits/sec
[  3]  0.0-10.2 sec  4.25 MBytes  3.51 Mbits/sec
```

NOTE:

When sending data in passthrough mode with the -P option, the socket can only be closed after receiving the SEND_IDLE event. It takes more than 1 second after sending the last data. So, the remote iperf tcp server stops after 1 second.

- **Iperf UDP Client/Server**

```
# iperf -c 192.168.200.1 -u

 [ IPERF OPTION ]
  - role: client
  - protocol: udp
  - server_port: 5001
  - server_ip: 192.168.200.1
  - send_length: 1470
```

```
  - send_time: 10
  - send_passthrough: off
  - send_done_event: 0
  - report_interval: 1


 [ IPERF UDP Client ]
  Sending 1470 byte datagrams ...
    Interval            Transfer          Bandwidth
    0.0 ~   1.0 sec   215.33 KBytes     1.76 Mbits/sec
    1.0 ~   2.0 sec   216.77 KBytes     1.77 Mbits/sec
    2.0 ~   3.0 sec   222.51 KBytes     1.82 Mbits/sec
    3.0 ~   4.0 sec   219.64 KBytes     1.79 Mbits/sec
    4.0 ~   5.0 sec   222.51 KBytes     1.81 Mbits/sec
    5.0 ~   6.0 sec   222.51 KBytes     1.82 Mbits/sec
    6.0 ~   7.0 sec   216.77 KBytes     1.77 Mbits/sec
    7.0 ~   8.0 sec   213.90 KBytes     1.75 Mbits/sec
    8.0 ~   9.0 sec   215.33 KBytes     1.76 Mbits/sec
    9.0 ~ 10.0 sec   206.72 KBytes     1.74 Mbits/sec
    0.0 ~ 10.0 sec     2.12 MBytes     1.78 Mbits/sec
  Sent 1513 datagrams
  Done


# iperf -c 192.168.200.1 -u -P

 [ IPERF OPTION ]
  - role: client
  - protocol: udp
  - server_port: 5001
  - server_ip: 192.168.200.1
  - send_length: 1470
  - send_time: 10
  - send_passthrough: on (-)
  - send_done_event: 0
  - report_interval: 1


 [ IPERF UDP Client ]
  Sending 1470 byte datagrams ...
    Interval            Transfer          Bandwidth
    0.0 ~   1.0 sec   480.91 KBytes     3.94 Mbits/sec
    1.0 ~   2.0 sec   467.99 KBytes     3.83 Mbits/sec
    2.0 ~   3.0 sec   469.42 KBytes     3.84 Mbits/sec
    3.0 ~   4.0 sec   467.99 KBytes     3.83 Mbits/sec
    4.0 ~   5.0 sec   469.42 KBytes     3.83 Mbits/sec
    5.0 ~   6.0 sec   470.86 KBytes     3.83 Mbits/sec
    6.0 ~   7.0 sec   467.99 KBytes     3.83 Mbits/sec
    7.0 ~   8.0 sec   467.99 KBytes     3.83 Mbits/sec
    8.0 ~   9.0 sec   466.55 KBytes     3.82 Mbits/sec
    9.0 ~ 10.0 sec   462.25 KBytes     3.84 Mbits/sec
```

```
    0.0 ~ 10.0 sec      4.58 MBytes      3.84 Mbits/sec
  Sent 3268 datagrams
  Done


# iperf -c 192.168.200.1 -u -P -N

 [ IPERF OPTION ]
  - role: client
  - protocol: udp
  - server_port: 5001
  - server_ip: 192.168.200.1
  - send_length: 1470
  - send_time: 10
  - send_passthrough: on (-)
  - send_done_event: 0
  - report_interval: 1


 [ IPERF UDP Client ]
  Sending 1470 byte datagrams ...
    Interval            Transfer          Bandwidth
    0.0 ~  1.0 sec   483.78 KBytes     3.96 Mbits/sec
    1.0 ~  2.0 sec   467.99 KBytes     3.82 Mbits/sec
    2.0 ~  3.0 sec   470.86 KBytes     3.84 Mbits/sec
    3.0 ~  4.0 sec   467.99 KBytes     3.83 Mbits/sec
    4.0 ~  5.0 sec   469.42 KBytes     3.83 Mbits/sec
    5.0 ~  6.0 sec   470.86 KBytes     3.84 Mbits/sec
    6.0 ~  7.0 sec   470.86 KBytes     3.83 Mbits/sec
    7.0 ~  8.0 sec   467.99 KBytes     3.83 Mbits/sec
    8.0 ~  9.0 sec   470.86 KBytes     3.85 Mbits/sec
    9.0 ~ 10.0 sec   455.07 KBytes     3.84 Mbits/sec
    0.0 ~ 10.0 sec     4.59 MBytes     3.85 Mbits/sec
  Sent 3271 datagrams
  Done


# iperf -s -u

 [ IPERF OPTION ]
  - role: server
  - protocol: udp
  - server_port: 5001
  - report_interval: 1


 [ IPERF UDP Server ]
  Connected with client: 192.168.200.1 port 56129
    Interval            Transfer          Bandwidth          Jitter      Lost/Total Datagrams
    0.0 ~  1.0 sec   482.34 KBytes     3.95 Mbits/sec     0.964 ms      0/  336 (0%)
    1.0 ~  2.0 sec   490.96 KBytes     4.02 Mbits/sec     0.393 ms      0/  342 (0%)
```

```
    2.0 ~  3.0 sec  490.96 KBytes  4.02 Mbits/sec  0.276 ms  0/  342 (0%)
    3.0 ~  4.0 sec  489.52 KBytes  4.01 Mbits/sec  0.509 ms  0/  341 (0%)
    4.0 ~  5.0 sec  486.65 KBytes  3.98 Mbits/sec  0.280 ms  0/  339 (0%)
    5.0 ~  6.0 sec  486.65 KBytes  3.99 Mbits/sec  0.544 ms  0/  339 (0%)
    6.0 ~  7.0 sec  490.96 KBytes  4.02 Mbits/sec  0.454 ms  0/  342 (0%)
    7.0 ~  8.0 sec  489.52 KBytes  4.01 Mbits/sec  0.301 ms  0/  341 (0%)
    8.0 ~  9.0 sec  488.09 KBytes  3.99 Mbits/sec  0.607 ms  0/  340 (0%)
    9.0 ~ 10.0 sec  489.52 KBytes  4.01 Mbits/sec  0.807 ms  0/  341 (0%)
    0.0 ~ 10.0 sec  4.77 MBytes  4.00 Mbits/sec  0.807 ms  0/ 3403 (0%)
  Done: 3403/3403

Press ENTER to continue or type "quit" :

[ IPERF UDP Server ]
  Connected with client: 192.168.200.1 port 51030
    Interval        Transfer        Bandwidth       Jitter   Lost/Total Datagrams
    0.0 ~  1.0 sec  496.70 KBytes  4.07 Mbits/sec  0.477 ms  0/  346 (0%)
    1.0 ~  2.0 sec  501.01 KBytes  4.10 Mbits/sec  0.454 ms  0/  349 (0%)
    2.0 ~  3.0 sec  499.57 KBytes  4.09 Mbits/sec  0.550 ms  0/  348 (0%)
    3.0 ~  4.0 sec  499.57 KBytes  4.09 Mbits/sec  0.747 ms  0/  348 (0%)
    4.0 ~  5.0 sec  501.01 KBytes  4.10 Mbits/sec  0.507 ms  0/  349 (0%)
    5.0 ~  6.0 sec  501.01 KBytes  4.10 Mbits/sec  0.694 ms  0/  349 (0%)
    6.0 ~  7.0 sec  502.44 KBytes  4.12 Mbits/sec  0.448 ms  0/  350 (0%)
    7.0 ~  8.0 sec  499.57 KBytes  4.09 Mbits/sec  0.428 ms  0/  348 (0%)
    8.0 ~  9.0 sec  501.01 KBytes  4.10 Mbits/sec  0.588 ms  0/  349 (0%)
    9.0 ~ 10.0 sec  505.31 KBytes  4.12 Mbits/sec  1.007 ms  0/  352 (0%)
    0.0 ~ 10.0 sec  4.89 MBytes  4.10 Mbits/sec  1.007 ms  0/ 3488 (0%)
  Done: 3488/3488

Press ENTER to continue or type "quit" :

 [ IPERF UDP Server ]
  Connected with client: 192.168.200.1 port 39813
    Interval        Transfer        Bandwidth       Jitter   Lost/Total Datagrams
    0.0 ~  1.0 sec  492.39 KBytes  4.03 Mbits/sec  0.633 ms  3/  346 (0.87%)
    1.0 ~  2.0 sec  502.44 KBytes  4.11 Mbits/sec  0.402 ms  8/  358 (2.2%)
    2.0 ~  3.0 sec  503.88 KBytes  4.12 Mbits/sec  0.486 ms  7/  358 (2%)
    3.0 ~  4.0 sec  501.01 KBytes  4.10 Mbits/sec  0.627 ms  8/  357 (2.2%)
    4.0 ~  5.0 sec  501.01 KBytes  4.10 Mbits/sec  0.773 ms  7/  356 (2%)
    5.0 ~  6.0 sec  503.88 KBytes  4.13 Mbits/sec  0.404 ms  8/  359 (2.2%)
    6.0 ~  7.0 sec  502.44 KBytes  4.11 Mbits/sec  0.383 ms  7/  357 (2%)
    7.0 ~  8.0 sec  501.01 KBytes  4.10 Mbits/sec  0.487 ms  8/  357 (2.2%)
    8.0 ~  9.0 sec  499.57 KBytes  4.09 Mbits/sec  0.550 ms  8/  356 (2.2%)
    9.0 ~ 10.0 sec  515.36 KBytes  4.16 Mbits/sec  1.931 ms  7/  367 (1.9%)
    0.0 ~ 10.0 sec  4.91 MBytes  4.11 Mbits/sec  1.931 ms  72/ 3573 (2%)
  Done: 3500/3573

Press ENTER to continue or type "quit" : quit

#
```

Remote Iperf UDP Server/Client

```
$ iperf -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:    160 KByte (default)
------------------------------------------------------------
[    3] local 192.168.200.1 port 5001 connected with 192.168.200.43 port 50000
[ ID] Interval            Transfer         Bandwidth              Jitter     Lost/Total Datagrams
[    3]    0.0- 1.0 sec     218 KBytes    1.79 Mbits/sec     0.499 ms       0/    152 (0%)
[    3]    1.0- 2.0 sec     215 KBytes    1.76 Mbits/sec     0.465 ms       0/    150 (0%)
[    3]    2.0- 3.0 sec     223 KBytes    1.82 Mbits/sec     0.659 ms       0/    155 (0%)
[    3]    3.0- 4.0 sec     218 KBytes    1.79 Mbits/sec     0.726 ms       0/    152 (0%)
[    3]    4.0- 5.0 sec     221 KBytes    1.81 Mbits/sec     0.606 ms       0/    154 (0%)
[    3]    5.0- 6.0 sec     223 KBytes    1.82 Mbits/sec     0.658 ms       0/    155 (0%)
[    3]    6.0- 7.0 sec     217 KBytes    1.78 Mbits/sec     0.901 ms       0/    151 (0%)
[    3]    7.0- 8.0 sec     214 KBytes    1.75 Mbits/sec     0.799 ms       0/    149 (0%)
[    3]    8.0- 9.0 sec     214 KBytes    1.75 Mbits/sec     0.712 ms       0/    149 (0%)
[    3]    0.0-10.0 sec    2.12 MBytes    1.78 Mbits/sec     0.756 ms       0/ 1513 (0%)
[    4] local 192.168.200.1 port 5001 connected with 192.168.200.43 port 50000
[    4]    0.0- 1.0 sec     468 KBytes    3.83 Mbits/sec     2.071 ms       0/    326 (0%)
[    4]    1.0- 2.0 sec     467 KBytes    3.82 Mbits/sec     2.216 ms       0/    325 (0%)
[    4]    2.0- 3.0 sec     469 KBytes    3.85 Mbits/sec     2.175 ms       0/    327 (0%)
[    4]    3.0- 4.0 sec     468 KBytes    3.83 Mbits/sec     2.077 ms       0/    326 (0%)
[    4]    4.0- 5.0 sec     468 KBytes    3.83 Mbits/sec     2.053 ms       0/    326 (0%)
[    4]    5.0- 6.0 sec     468 KBytes    3.83 Mbits/sec     2.109 ms       0/    326 (0%)
[    4]    6.0- 7.0 sec     467 KBytes    3.82 Mbits/sec     2.329 ms       0/    325 (0%)
[    4]    7.0- 8.0 sec     467 KBytes    3.82 Mbits/sec     2.159 ms       0/    325 (0%)
[    4]    8.0- 9.0 sec     468 KBytes    3.83 Mbits/sec     2.121 ms       0/    326 (0%)
[    4]    9.0-10.0 sec     469 KBytes    3.85 Mbits/sec     2.180 ms       0/    327 (0%)
[    4]    0.0-10.0 sec    4.58 MBytes    3.83 Mbits/sec     2.072 ms       0/ 3268 (0%)
[    3] local 192.168.200.1 port 5001 connected with 192.168.200.43 port 50000
[    3]    0.0- 1.0 sec     469 KBytes    3.85 Mbits/sec     2.106 ms       0/    327 (0%)
[    3]    1.0- 2.0 sec     468 KBytes    3.83 Mbits/sec     2.252 ms       0/    326 (0%)
[    3]    2.0- 3.0 sec     467 KBytes    3.82 Mbits/sec     2.483 ms       0/    325 (0%)
[    3]    3.0- 4.0 sec     469 KBytes    3.85 Mbits/sec     2.064 ms       0/    327 (0%)
[    3]    4.0- 5.0 sec     467 KBytes    3.82 Mbits/sec     2.311 ms       0/    325 (0%)
[    3]    5.0- 6.0 sec     469 KBytes    3.85 Mbits/sec     2.323 ms       0/    327 (0%)
[    3]    6.0- 7.0 sec     468 KBytes    3.83 Mbits/sec     2.198 ms       0/    326 (0%)
[    3]    7.0- 8.0 sec     468 KBytes    3.83 Mbits/sec     2.018 ms       0/    326 (0%)
[    3]    8.0- 9.0 sec     468 KBytes    3.83 Mbits/sec     2.115 ms       0/    326 (0%)
[    3]    9.0-10.0 sec     468 KBytes    3.83 Mbits/sec     2.247 ms       0/    326 (0%)
[    3]    0.0-10.0 sec    4.59 MBytes    3.83 Mbits/sec     2.124 ms       0/ 3271 (0%)


$ iperf -c 192.168.200.43 -u -b 4M -i 1
------------------------------------------------------------
Client connecting to 192.168.200.43, UDP port 5001
Sending 1470 byte datagrams, IPG target: 2940.00 us (kalman adjust)
```

```
UDP buffer size:    160 KByte (default)
------------------------------------------------------------
[    3] local 192.168.200.1 port 56129 connected with 192.168.200.43 port 5001
[ ID] Interval            Transfer         Bandwidth
[    3]    0.0- 1.0 sec     491 KBytes    4.02 Mbits/sec
[    3]    1.0- 2.0 sec     488 KBytes    4.00 Mbits/sec
[    3]    2.0- 3.0 sec     488 KBytes    4.00 Mbits/sec
[    3]    3.0- 4.0 sec     488 KBytes    4.00 Mbits/sec
[    3]    4.0- 5.0 sec     488 KBytes    4.00 Mbits/sec
[    3]    5.0- 6.0 sec     488 KBytes    4.00 Mbits/sec
[    3]    6.0- 7.0 sec     488 KBytes    4.00 Mbits/sec
[    3]    7.0- 8.0 sec     490 KBytes    4.01 Mbits/sec
[    3]    8.0- 9.0 sec     488 KBytes    4.00 Mbits/sec
[    3]    9.0-10.0 sec      488 KBytes    4.00 Mbits/sec
[    3]    0.0-10.0 sec    4.77 MBytes    4.00 Mbits/sec
[    3] Sent 3403 datagrams
[    3] Server Report:
[    3]    0.0-10.0 sec    4.77 MBytes    4.00 Mbits/sec      0.807 ms        0/ 3403 (0%)

$ iperf -c 192.168.200.43 -u -b 4.1M -i 1
------------------------------------------------------------
Client connecting to 192.168.200.43, UDP port 5001
Sending 1470 byte datagrams, IPG target: 2868.29 us (kalman adjust)
UDP buffer size:    160 KByte (default)
------------------------------------------------------------
[    3] local 192.168.200.1 port 51030 connected with 192.168.200.43 port 5001
[ ID] Interval            Transfer         Bandwidth
[    3]    0.0- 1.0 sec     502 KBytes    4.12 Mbits/sec
[    3]    1.0- 2.0 sec     501 KBytes    4.10 Mbits/sec
[    3]    2.0- 3.0 sec     500 KBytes    4.09 Mbits/sec
[    3]    3.0- 4.0 sec     501 KBytes    4.10 Mbits/sec
[    3]    4.0- 5.0 sec     501 KBytes    4.10 Mbits/sec
[    3]    5.0- 6.0 sec     500 KBytes    4.09 Mbits/sec
[    3]    6.0- 7.0 sec     501 KBytes    4.10 Mbits/sec
[    3]    7.0- 8.0 sec     501 KBytes    4.10 Mbits/sec
[    3]    8.0- 9.0 sec     500 KBytes    4.09 Mbits/sec
[    3]    9.0-10.0 sec      501 KBytes    4.10 Mbits/sec
[    3]    0.0-10.0 sec    4.89 MBytes    4.10 Mbits/sec
[    3] Sent 3488 datagrams
[    3] Server Report:
[    3]    0.0-10.0 sec    4.89 MBytes    4.10 Mbits/sec      1.006 ms        0/ 3488 (0%)

$ iperf -c 192.168.200.43 -u -b 4.2M -i 1
------------------------------------------------------------
Client connecting to 192.168.200.43, UDP port 5001
Sending 1470 byte datagrams, IPG target: 2800.00 us (kalman adjust)
UDP buffer size:    160 KByte (default)
------------------------------------------------------------
[    3] local 192.168.200.1 port 39813 connected with 192.168.200.43 port 5001
[ ID] Interval            Transfer         Bandwidth
[    3]    0.0- 1.0 sec     515 KBytes    4.22 Mbits/sec
```

```
[  3]  1.0- 2.0 sec   512 KBytes   4.20 Mbits/sec
[  3]  2.0- 3.0 sec   512 KBytes   4.20 Mbits/sec
[  3]  3.0- 4.0 sec   512 KBytes   4.20 Mbits/sec
[  3]  4.0- 5.0 sec   512 KBytes   4.20 Mbits/sec
[  3]  5.0- 6.0 sec   512 KBytes   4.20 Mbits/sec
[  3]  6.0- 7.0 sec   512 KBytes   4.20 Mbits/sec
[  3]  7.0- 8.0 sec   514 KBytes   4.21 Mbits/sec
[  3]  8.0- 9.0 sec   512 KBytes   4.20 Mbits/sec
[  3]  9.0-10.0 sec   512 KBytes   4.20 Mbits/sec
[  3]  0.0-10.0 sec  5.01 MBytes   4.20 Mbits/sec
[  3] Sent 3573 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  4.91 MBytes   4.11 Mbits/sec   1.930 ms   72/ 3573 (2%)
```

8.2 Remote Server/Client (raspi-atcmd-remote)

A remote server/client application run one server or client. This application is a Linux application and can be executed on Raspberry Pi.

8.2.1 Source files

File	Description
main.c	UDP/TCP server/client related functions
Makefile	Make file for building

Table 8.2 raspi-atcmd-remote source files

8.2.2 Build

Copy the “atcmd/host/raspi-atcmd-remote” directory to the Raspberry Pi's home directory. And build the remote application with the make command.

```
$ cd $HOME

$ cd raspi-atcmd-remote

$ make clean
```

```
removed 'raspi-atcmd-remote'
```

$ make

```
cc -g -o raspi-atcmd-remote main.c -Wall -Wno-unused-function -DCONFIG_VERBOSE
```

8.2.3 Run

$ ./raspi-atcmd-remote [-h|--help]

```
raspi-atcmd-remote version 1.2.0
Copyright (c) 2019-2023    <NEWRACOM LTD>

Usage:
   $ ./raspi-atcmd-remote -s [-p <listen_port>] [-u] [-e]
   $ ./raspi-atcmd-remote -c <server_ip> [-p <server_port>] [-u] [-e]

Options:
   -s, --server              run in server mode
   -c, --client #            run in client mode
   -p, --port #              set server port to listen on or connect to (default: 50000)
   -u, --udp                 use UDP
   -e, --echo                enable echo for received packets (default: off)
   -v, --version             print version information and quit
   -h, --help                print this message and quit
```

Examples:

Mode	Protocol	Command
Server	TCP	$ ./raspi-atcmd-remote -s -p 50000 [-e]
	UDP	$ ./raspi-atcmd-remote -s -u -p 60000 [-e]
Client	TCP	$ ./raspi-atcmd-remote -c 192.168.200.1 -p 50000 [-e]
	UDP	$ ./raspi-atcmd-remote -c 192.168.200.1 -u -p 60000 [-e]

9 Revision History

Revision No	Date	Comments
Ver 1.0	8/4/2023	Initial version (AT Command Set v1.23.5)
Ver 1.1	8/16/2023	AT+WCTX command added