NEWRACOM
A New semiconductor Company for New Era

NRC7292 Evaluation Kit User Guide

(AT-command)

Ultra-low power & Long-range Wi-Fi

Ver 1.19.1
Jul 29, 2021

NEWRACOM, Inc.

NRC7292 Evaluation Kit User Guide (AT-command)
Ultra-low power & Long-range Wi-Fi

Office
Newracom, Inc.
25361 Commercentre Drive, Lake Forest, CA 92630 USA
http://www.newracom.com

Contents

List of Tables

List of Figures

1 Overview

This document introduces the NRC7292 AT-command. The NRC7292 AT-command allows users to apply fine controls over the NRC7292 modules such as: checking the modem status, scanning, connecting to an AP, opening sockets, and exchanging data.

2 Basic Setup

The AT-command package with a custom firmware binary to enable AT-command feature is required along with the firmware download tool. Users need to download the firmware binary onto the flash on the NRC7292 module to enable AT-command communication via UART or SPI.

2.1 Hardware connection

Figure 2.1 shows an NRC7292 evaluation board (EVB). The AT-command communication is achieved via the UART or SPI interface between an external host and the EVB.

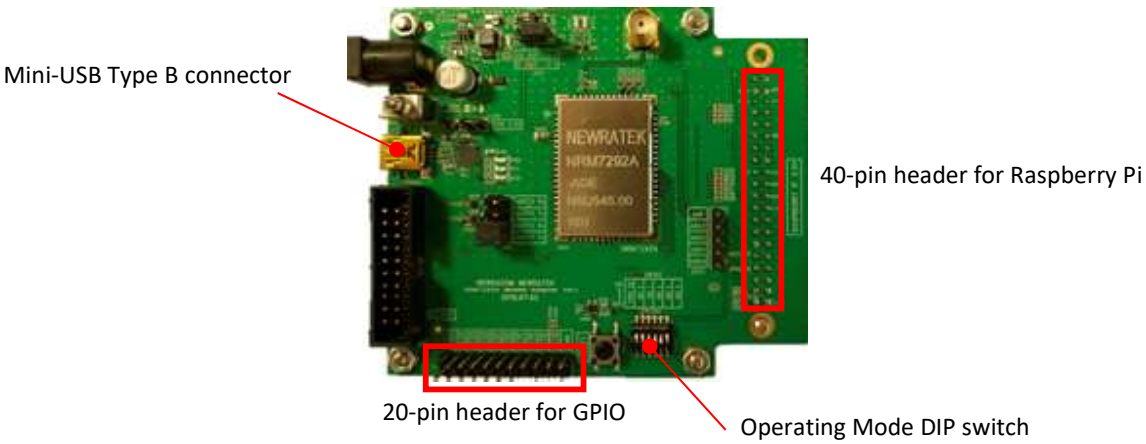
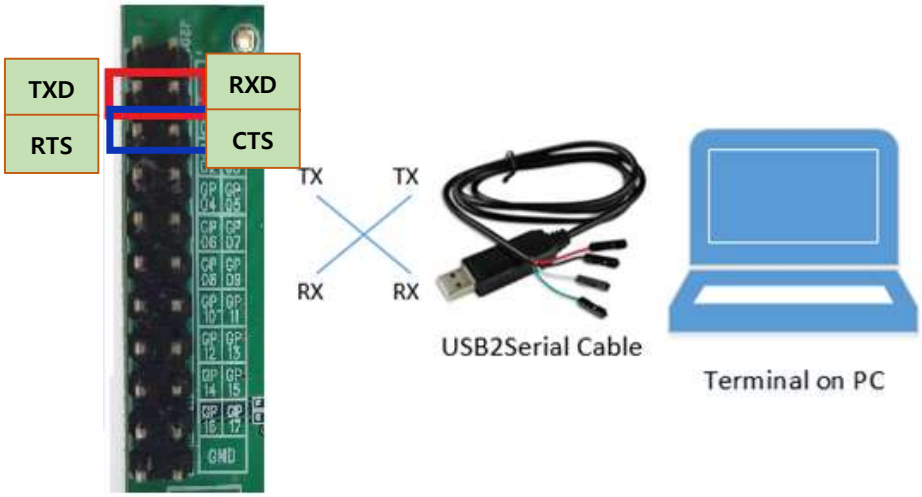


Figure 2.1 NRC7292 evaluation board

IMPORTANT: If the EVB is mounted on a Raspberry Pi host, detach the board from the Raspberry Pi host first before proceeding. The EVB must be used as a standalone for stable AT communication.

1) UART

The AT-command uses UART channel 2. The TX and RX of UART channel 2 are placed in a 20-pin header for GPIO.



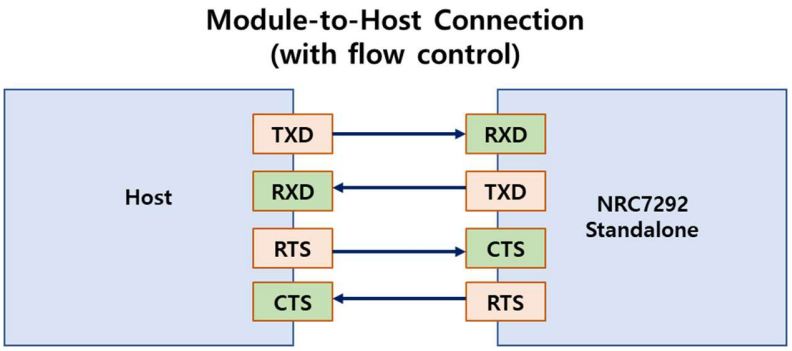


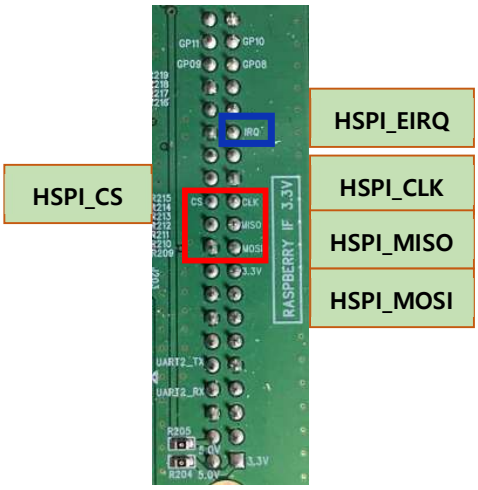
Figure 2.2 UART connection between EVK and external host

The GP00 and GP01 pins on the 20-pin header correspond to TX and RX of UART channel 2, respectively.

※ (Optional) GP02 and GP03 pins correspond to RTS and CTS of UART channel 2 for HW Flow Control

2) HSPI

The NRC7292 has a dedicated SPI slave controller for high speed. The SPI signals are placed in a 40-pin header for Raspberry Pi. The CLI application described in chapter 8 is available to perform AT-command communication via the SPI on Raspberry Pi.



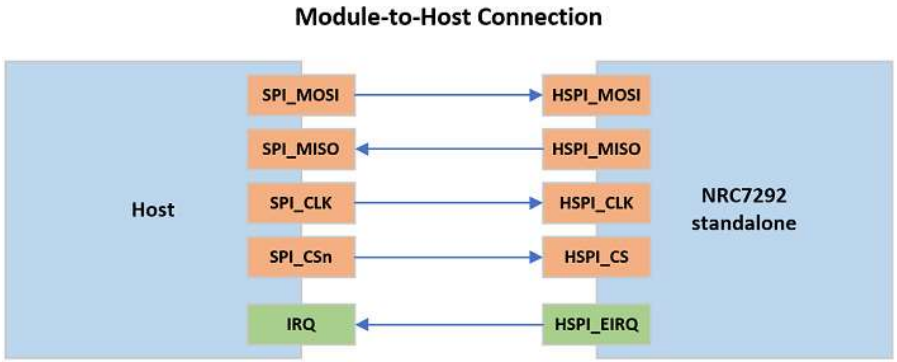


Figure 2.3 SPI connection between EVK and external host

To perform AT command communication through SPI on Raspberry Pi, spidev of Raspberry Pi3 must be enabled.

1. Modify /boot/config.txt and enable spi hardware interface configuration

```
# 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
```

2. After rebooting the Raspberry Pi3, 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 Building the firmware

Refer to the user guide **UG-7292-004-Standalone SDK.pdf** for instructions on how to build the firmware binary. (2. Setup S/W build environment)

1. **HSPI mode:**
   - make select target=nrc7292.sdk.release APP_NAME=ATCMD_HSPI
2. **UART mode (without hardware flow control):**
   - make select target=nrc7292.sdk.release APP_NAME=ATCMD_UART
3. **UART mode (with hardware flow control):**
   - make select target=nrc7292.sdk.release APP_NAME=ATCMD_UART_HFC

2.3 Downloading the firmware and starting the module

Refer to the user guide **UG-7292-004-Standalone SDK.pdf** for instructions on how to download the firmware binary. (3.2 Download the unified binary)

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.
AT+SLEEP	Configure deep sleep mode

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. (restarting time : 3 secs)
Example	ATZ

5.4 AT+VER

Command	**GET** AT+VER?
Response	**GET** +VER: <S/W package version>,<AT firmware version> OK
Description	Fetch the AT firmware version and software package version.

Example	AT+VER? +VER:”1.3.1”,”1.15.3” 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>** 19200, 38400, 57600, 115200*, 230400, 380400, 460800, 500000, 576000, 921600, 1000000, 1152000, 1500000 or 2000000 **<data bits>** Always 8 (8-bit)* **<stop bits>** Always 1 (1-bit)* **<parity>** Always 0 (None)* **<HFC>** 0 (RTS/CTS disabled)* or 1 (RTS/CTS enabled)	
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=<index>,<direction>[,<pull-up>] **GET** AT+GPIOCONF? AT+GPIOCONF?=<index>	
Response	**SET** OK **GET** +GPIOCONF=<index>,<direction>,<pull-up> OK	
Parameters	**<index>** The GPIO pin index. (8, 9, 10, 11, 12, 13, 14, 15, 16, 17) **<direction>** 0 (input)*, 1 (output) **<pull-up> (input pin only)** 0 (floating)*, 1 (pull-up)	
Description	Configure the GPIO pin direction and pull-up option.	
Example	AT+GPIOCONF=8,1 OK AT+GPIOCONF=11,0,0 OK AT+GPIOCONF=17,0,1 OK	AT+GPIOCONF? +GPIOCONF:8,1,0 : +GPIOCONF:11,0,0 : +GPIOCONF:17,0,1 OK AT+GPIOCONF?=13 +GPIOCONF:13,0,0 OK

5.7 AT+GPIOVAL

Command	**SET** AT+GPIOVAL=<index>,<level> **GET** AT+GPIOVAL? AT+GPIOVAL?=<index>	
Response	**SET** OK **GET** +GPIOVAL:<index>,<level> OK	
Parameters	**<GPIO pin index>** The GPIO pin index. (8, 9, 10, 11, 12, 13, 14, 15, 16, 17) **<GPIO pin level>** 0 (low)*, 1 (high)	
Description	Read or write the output GPIO pin level.	
Example	AT+GPIOVAL=8,1 OK AT+GPIOVAL=13,1 OK AT+GPIOVAL=17,0 OK	AT+GPIOVAL? +GPIOVAL:8,1 : +GPIOVAL:17,0 OK AT+GPIOVAL?=13 +GPIOVAL:13,1 OK

5.8 AT+ADC

Command	**SET** AT+ADC=<ADC channel index>
Response	**SET** +ADC:<ADC channel index>,<ADC value> OK

Parameters	**<ADC channel index>** 1, 2, 3 **<ADC value>** 0 ~ 511
Description	Fetch the ADC value at the selected ADC channel index.
Example	AT+ADC=1 +ADC:1,23 OK

5.9 AT+SLEEP

Command	**SET** AT+SLEEP=<rtc>[,<gpio>]
Response	**SET** OK
Parameters	**<rtc>** Use RTC interrupt to wake up from deep sleep by TIM in beacon frame. 0 : disable 1 : enable **<gpio>** Use GPIO interrupt to wake up from deep sleep. Available GPIO numbers : 8 ~ 17
Description	Configure deep sleep mode. In deep sleep mode, retention RAM and 32.768KHz OSC are powered on. And the others are powered off.
Example	AT+SLEEP=1 OK AT+SLEEP=1,15 OK

6 Wi-Fi AT Commands

Commands	Description
AT+WMACADDR	Read the MAC address.
AT+WCOUNTRY	Configure the Wi-Fi country code
AT+WTXPOWER	Configure 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	Configure the MCS index.
AT+WTSF	Read the elapsed TSF timer duration.
AT+WIPADDR	Configure the IP address, netmask and gateway.
AT+WDHCP	Request dynamic IP allocation from the DHCP server.
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+WPING	Initiate a ping session.
AT+WROAM	Enable or disable Wi-Fi roaming.
AT+WSOFTAP	Run as the AP mode.
AT+WSTAINFO	Get information of associated STAs on AP mode.
AT+WFOTA	Enable or disable Firmware Over-the-Air (FOTA).
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>** US, KR*, JP, CN, TW, EU	
Description	Configure the Wi-Fi country code	
Example	AT+ WCOUNTRY =”US” OK	AT+WCOUNTRY? +WCOUNTRY:”US” OK

6.3 AT+WTXPOWER

Command	**SET** AT+WTXPOWER=<power in dBm> **GET** AT+WTXPOWER?	
Response	**SET** OK **GET** +WTXPOWER:<power in dBm >	
Parameters	**<power in dBm>** 8, 9, … ,17*, 18	
Description	Configure the transmission power level.	
Example	AT+WTXPOWER=11 OK	AT+WTXPOWER? +WTXPOWER:11 OK

6.4 AT+WRXSIG

Command	**GET** AT+WRXSIG? **SET** AT+WRXSIG =<time>	
Response	**GET** +WRXSIG:<RSSI>,<SNR> OK **SET** +WRXSIG:<RSSI>,<SNR> … +WRXSIG:<RSSI>,<SNR> OK	
Parameters	**<time>** Monitoring duration in seconds.	
Description	Fetch or monitor the RSSI (dBm) and SNR (dB) values.	
Example	AT+WRXSIG?	AT+WRXSIG=2

	+WRXSIG:-50,25 OK	+WRXSIG:-62,20 +WRXSIG:-82,9 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 =1 OK	AT+WRATECTRL? +WRATECTRL:1 OK

6.6 AT+WMCS

Command	**SET (Only when the MCS control is enabled)** AT+WMCS=<MCS index> **GET** AT+WMCS?
Response	**SET (Only when the MCS control is enabled)** OK **GET** +WMCS:<value> OK
Parameters	**<MCS index>** 0~7, 10

Description	Configure the MCS index.	
Example	AT+WMCS=7 OK	AT+WMCS? +WMCS:2 OK

6.7 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

6.8 AT+WDHCP

Command	**RUN** AT+WDHCP
Response	**RUN** +WDHCP:”<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	Request dynamic IP allocation from the DHCP server. NOTE: Wi-Fi connection must be established before using this command.
Example	AT+WDHCP +WDHCP:”192.168.200.20”,”255.255.255.0”,”192.168.200.1” OK

6.9 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

6.10 AT+WIPADDR

Command	**SET** AT+WIPADDR="<IP>","<netmask>","<gateway>" **GET** AT+WIPADDR?
Response	**SET** OK **GET** +WIPADDR="<IP>","<netmask>","<gateway>" OK
Parameters	**<IP>,<netmask>,<gateway>** 'A.B.C.D' where A, B, C and D are between 0 and 255, inclusive.
Description	Configure the IP address, netmask and gateway.
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

6.11 AT+WSCAN

Command	**RUN** AT+WSCAN
Response	**RUN** +WSCAN:<bssid>,<freq>,<sig_level>,<flags>,<ssid> : OK
Parameters	**<bssid>** The BSSID of the AP. **<freq>** The center frequency of the channel. **<sig_level>** The RSSI (Received Signal Strength Indicator) in dBm. **<flags>** Service set flags. **<ssid>** The SSID of the AP.
Description	Perform Wi-Fi scanning.
Example	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

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 **<password> (wpa2 security option only)** The password when wpa2 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: Before attempting to connect to the specified AP, the “AT+WSCAN” command should be executed to scan the APs. If the specified AP is not in the scan result list, this command will return an error. 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	AT+WCONN=”demo_ap”,”wpa2”,”12345678” OK AT+WDISCONN OK AT+WCONN=”02:00:A1:45:82:B4”,”wpa2”,”12345678” OK AT+WCONN? +WCONN:”demo_ap”,”02:00:A1:45:82:B4”,”wpa2”,”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+WPING

Command	**SET** AT+WPING=”<remote IP>”[,<time>]
Response	**SET** +WPING:<size>,”<remote IP>”,<sequence number>,<TTL>,<elapsed time> ……. +WPING:<size>,”<remote IP>”,<sequence number>,<TTL>,<elapsed time> OK
Parameters	**<remote IP>** The remote IP 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	Initiate a ping session.
Example	AT+ PING =”192.168.200.1”,10

	+PING:64,”192.168.200.1”,1,64,11 …… +PING:64,”192.168.200.1”,10,64,12 OK

6.15 AT+WROAM

Command	**SET** AT+WROAM=<scan_delay>,<rssi_threshold>,<rssi_level> AT+WROAM=0 **GET** AT+WROAM?
Response	**SET** OK **GET** +WROAM=<scan_delay>,<rssi_threshold>,<rssi_level> OK
Parameters	**<scan_delay>** Delay time between scan cycles that is executed to search for nearby APs. (msec) **<rssi_threshold>** Current AP’s minimum RSSI to move to a new AP. (dBm, -100 ~ 0) **<rssi_level>** The minimum signal strength of a new AP to leave the current AP. (dBm, 1 ~ 100)
Description	Enable or disable Wi-Fi roaming. Wi-Fi roaming can be enabled after connecting to the AP with AT+WCONN command and disabled with AT+WROAM=0 command. All APs that is scanned and connected for Wi-Fi roaming have the same SSID and security settings. Conditions for leaving current AP and moving to new AP. 1. Current AP’s RSSI <= rssi_threshold 2. (New AP’s RSSI - Current AP’s RSSI) >= rssi_level If it is disconnected from the current AP or the current AP is not scanned, it is connected to a new AP with the highest RSSI value among the scanned APs. When moving to a new AP during the roaming process, the following events occur sequentially. 1. +WEVENT:”DISOCNNECT”

	2. +WEVENT:”CONNNECT_SUCCESS” 3. +WEVENT:”ROAMING”,”<ssid>”,”<bssid>” At this time, the TCP connection may be disconnected and the TCP socket may be closed.
Example	AT+WSCAN +WSCAN:"02:00:eb:3a:a3:b4",922.5,-43,"[WPA2-PSK-CCMP][ESS]","wifi_roaming" +WSCAN:"02:00:eb:ee:9a:87",918.5,-35,"[WPA2-PSK-CCMP][ESS]","wifi_roaming" +WSCAN:"02:00:48:32:58:42",920.5,-59,"[WPA2-PSK-CCMP][ESS]","wifi_roaming" OK AT+WCONN="wifi_roaming","wpa2","12345678" OK AT+WCONN? +WCONN:"wifi_roaming","02:00:eb:ee:9a:87","wpa2","12345678","connected” OK AT+WROAM=10000,-30,10 OK AT+WROAM? +WROAM:10000,-30,10 OK +WEVENT:"DISCONNECT" +WEVENT:"CONNECT_SUCCESS" +WEVENT:"ROAMING","wifi_roaming","02:00:48:32:58:42" +WEVENT:"DISCONNECT" +WEVENT:"CONNECT_SUCCESS" +WEVENT:"ROAMING","wifi_roaming","02:00:eb:3a:a3:b4" AT+WROAM=0 OK

6.16 AT+WSOFTAP

Command	**SET** AT+WSOFTAP=<frequency>,”<ssid>”[,”<security>”[,”<password>”]] **GET**

	AT+WSOFTAP?
Response	**SET** OK **GET** +WSOFTAP=<frequency>,"<ssid>","<security>","<password>"[,"dhcp"] OK
Parameters	**<frequency>** S1G channel frequency (MHz) **<ssid>** The SSID of the AP. **<security>** open*, wpa2 **<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+WSOFTAP=918.5,"halow_softap","wpa2","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 AT+WSOFTAP? +WCONN:918.5,"halow_softap","wpa2","12345678","dhcp" OK

6.17 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","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

6.18 AT+WFOTA

Command	**SET** AT+WFOTA=<check_time>[,"<server_url>"] **GET** AT+WFOTA? **RUN** AT+WFOTA
Response	**SET** OK **GET** +WFOTA=<check_time>,"<server_url>" OK **RUN** OK
Parameters	**<check_time>** Interval time to periodically check new firmware on the server. Set to 0 to stop checking. **<server_url>** HTTP or HTTPS Server URL

Description	Enable or disable Firmware Over-the-Air (FOTA). FOTA 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.info file from the server. The server should have a fota.info file as well as firmware binary. The contents of the fota.info file are as follows.
	`AT_SDK_VER: 10.10.10`
	`AT_CMD_VER: 10.10.10`
	
	`AT_HSPI_BIN: nrc7292_standalone_xip_ATCMD_HSPI.bin`
	`AT_HSPI_CRC: 4c0ddd8e`
	
	`AT_UART_BIN: nrc7292_standalone_xip_ATCMD_UART.bin`
	`AT_UART_CRC: 155c2053`
	
	`AT_UART_HFC_BIN: nrc7292_standalone_xip_ATCMD_UART_HFC.bin`
	`AT_UART_HFC_CRC: e2ce889e`

If new firmware has a higher version, 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 and replace the current firmware with it.

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>

After the download is complete, the current firmware is replaced with new firmware. If the replacement is successful, a software reset is performed to run new firmware.

Before this process, the terminal or host receives 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 then FOTA will be automatically disabled.

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 replaced with new firmware, and then the system is reset to run new firmware. This process 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. Restoring the current firmware is only possible with the download tool.
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 : sdk/apps/atcmd/host/python-http-server

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

```
python-http-server/
├── fota.info
├── nrc7292_standalone_xip_ATCMD_HSPI.bin
├── nrc7292_standalone_xip_ATCMD_UART.bin
├── nrc7292_standalone_xip_ATCMD_UART_HFC.bin
├── python
│   ├── crc.py
│   └── https-server.py
├── Run-server.sh
├── ssl-cert
│   ├── server.crt
│   ├── server.csr
│   ├── server.key
│   └── server.key.origin
└── Update-fota-info.sh
```

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.info file.
	Usage:
	$ ./Update-fota-info.sh [options]
	Firmware version and binary name can be set by editing this file.
	`AT_SDK_VER: 10.10.10`
	`AT_CMD_VER: 10.10.10`
	`AT_HSPI_BIN: nrc7292_standalone_xip_ATCMD_HSPI.bin`
	`AT_HSPI_CRC: 4c0ddd8e`
	`AT_UART_BIN: nrc7292_standalone_xip_ATCMD_UART.bin`
	`AT_UART_CRC: 155c2053`
	`AT_UART_HFC_BIN: nrc7292_standalone_xip_ATCMD_UART_HFC.bin`
	`AT_UART_HFC_CRC: e2ce889e`
	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.info should be updated by Update-fota-info.sh.

Example	AT+WSCAN +WSCAN:"02:00:eb:fa:49:99",921.5,-10,"[WPA2-PSK-CCMP][ESS]","halow_demo" OK AT+WCONN="halow_demo","wpa2","12345678" OK AT+WDHCP +WDHCP:"192.168.200.10","255.255.255.0","192.168.200.1" OK AT+WFOTA=5,"https://192.168.200.1:4443" OK AT+WFOTA? +WFOTA:5,"https://192.168.200.1:4443" 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" AT+WFOTA=0 OK AT+WFOTA OK +WEVENT:"FOTA_VERSION","10.10.10","10.10.10" +WEVENT:"FOTA_BINARY","nrc7292_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"

6.19 AT+WTIMEOUT

Command	**SET** AT+WTIMEOUT=”<command>”,<timeout> **GET** AT+WTIMEOUT?

Response	**SET** OK **GET** +WTIMEOUT:”<command>”,<timeout> OK	
Parameters	**<command>** “WSCAN”, “WCONN”, “WDISCONN”, “WDHCP” **<timeout>** Timeout in seconds. (0: no timeout, default: WDHCP=60, others=0)	
Description	Configure the response timeout for the specified command. A timeout event will trigger a Wi-Fi event notification “+WEVENT”.	
Example	AT+WTIMEOUT=”WCONN”,30 OK	AT+WTIMEOUT? +WTIMEOUT:”WSCAN”,0 +WTIMEOUT:”WCONN”,30 +WTIMEOUT:”WDISCONN”,0 OK

6.20 +WEVENT

Response	+WEVENT:<event>
Parameters	**<event>** “SCAN_DONE” “CONNECT_SUCCESS” “CONNECT_FAIL” “DISCONNECT” “ROAMING”,”<ssid>”,”<bssid>” “FOTA_VERSION”,”<sdk_version>”,”<atcmd_version>” “FOTA_BINARY”,”<binary_name>” “FOTA_DOWNLOAD”,”total_size”,”download_size” “FOTA_UPDATE” “FOTA_FAIL”
Description	Asynchronously raised Wi-Fi event logs.
Example	+WEVENT: ”CONNECT_SUCCESS”

7 Socket AT Commands

Commands	Description
AT+SOPEN	Create a TCP/UDP socket.
AT+SCLOSE	Close an existing socket.
AT+SLIST	List all currently open sockets.
AT+SRXLOGLEVEL	Configure the received packet event log level for +RXD.
AT+SSEND	Send data through a socket.
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> AT+SOPEN=”tcp”,<local_port> AT+SOPEN=”tcp”,”<remote IP>”,<remote port>
Response	**SET** +SOPEN=<socket ID> OK
Parameters	**<local_port> (UDP)** Optional argument to specify the outgoing local port. **<local_port> (TCP Server)** Local port to listen on. **<remote IP>,<remote port> (TCP Client)** The remote IP and remote port of the server.
Description	Create a TCP/UDP socket. For TCP, the server socket will listen on the given port in the background and asynchronously raise the event TCP_CONNECT to notify incoming connections.
Example	AT+ SOPEN =”TCP”,”192.168.100.109”,8088 +SOPEN=0 OK AT+ SOPEN =”TCP”,8088 +SOPEN=1 OK +SEVENT: “TCP_CONNECT”,2 AT+ SOPEN =”UDP”,8088 +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:3 OK

7.3 AT+SLIST

Command	**GET** AT+SLIST?
Response	**GET** +SLIST:<socket ID>,"<tcp-udp>","<remote IP>",<remote port>,<local port> : +SLIST:<socket ID>,"<tcp-udp>","<remote IP>",<remote port>,<local port> OK

Parameters	**<socket ID>** The ID allocated to the socket. **<tcp-udp>** TCP, UDP **<remote IP>,<remote port>,<local port>** The remote IP, remote port and local port associated with the socket.
Description	List all currently open sockets.
Example	AT+SLIST? +SLIST:1,"UDP","0.0.0.0",0,8088 +SLIST:3,"TCP", "192.168.100.109",8089,6000 OK

7.4 AT+SRXLOGLEVEL

Command	**SET** AT+SRXLOGLEVEL=<mode> **GET** AT+SRXLOGLEVEL?	
Response	**SET** +SRXLOGLEVEL:<mode> OK **GET** OK	
Parameters	**<mode>** 0 (terse)*, 1 (verbose)	
Description	Configure the received packet event log level for +RXD.	
Example	AT+SRXLOGLEVEL =1 OK	AT+SRXLOGLEVEL? + SRXLOGLEVEL:1 OK

7.5 AT+SSEND

Command	**SET** AT+SSEND =<socket ID>[,<length>]

	AT+SSEND =<socket ID>,"<remote IP>", <remote port>[,<length>]
Response	**SET** OK
Parameters	**<socket ID>** The ID allocated to the socket. **<remote IP>,<remote port> (UDP only)** The IP and port of the remote UDP server. **<length>** The (signed) number of raw bytes to send. (See the description)
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.
Example	**[Normal Mode]** AT+SSEND=0,6 OK Hello! **[Normal passthrough mode]** AT+SSEND=0,0 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",0,23 AT OK +SEVENT:"SEND_EXIT",0,23 **[Buffered passthrough Mode]** AT+SSEND=0,-8 TEST0001 (<- Without \r\n) TEST0002 (<- Without \r\n) TEST0003 (<- Without \r\n)

7.6 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. (default: [SOPEN=30, SSEND=1], disable: 0)	
Description	Configure the response timeout for the specified socket command. A timeout event will trigger a socket event notification "+SEVENT".	
Example	AT+STIMEOUT="SOPEN",60 OK	AT+STIMEOUT? +STIMEOUT:"SOPEN",60 +STIMEOUT:"SSEND",1 OK

7.7 +SEVENT

Response	+SEVENT:<event>,<socket ID>[,<parameter 1>,…,<parameter N>]
Parameters	**<event>** "CONNECT",<socket ID> "CLOSE",<socket ID> "SEND_IDLE",<socket ID>,<length> "SEND_DROP",<socket ID>,<length> "SEND_EXIT",<socket ID>,<length> "SEND_ERROR",<socket ID>,<length>,<error> "RECV_ERROR",<socket ID>,<error> **<socket ID>** Socket ID **<length>** The total length of the payload sent over the socket. The SEND_DROP event indicates the length of the dropped payload. **<error>** POSIX error code. If the SEND_ERROR and RECV_ERROR events occur due to the following errors, the corresponding socket is closed by the firmware.

<table>
<tr><td></td><td>
<table>
<tr><th>Event Name</th><th>Error Value</th><th>Description</th></tr>
<tr><td rowspan="2">SEND_ERROR</td><td>-107</td><td>Transport endpoint is not connected. (ENOTCONN)</td></tr>
<tr><td>-104</td><td>Connection reset by peer. (ECONNRESET)</td></tr>
<tr><td rowspan="2">RECV_ERROR</td><td>-107</td><td>Transport endpoint is not connected. (ENOTCONN)</td></tr>
<tr><td>-111</td><td>Connection refused. (ECONNREFUSED)</td></tr>
</table>
</td></tr>
<tr><td>Description</td><td>Asynchronously raised socket event logs.</td></tr>
<tr><td>Example</td><td>+SEVENT:”CONNECT”,1
+SEVENT:”SEND_INIT”,1,1500
+SEVENT:”SEND_ERROR”,1,1000,-103</td></tr>
</table>

7.8 +RXD

<table>
<tr><td>Response</td><td><u>RX mode (Terse)</u>
+RXD:<socket ID>,<actual read length>,<raw bytes>

<u>RX mode (Verbose)</u>
+RXD:<socket ID>,<actual read length>,”<remote IP>”,<remote port>,<raw bytes></td></tr>
<tr><td>Parameters</td><td><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.</td></tr>
<tr><td>Description</td><td>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.</td></tr>
<tr><td>Example</td><td><u>RX mode (Terse)</u>
+RXD=0,15,ABCDE12345,.?=+</td></tr>
</table>

	RX mode (Verbose) +RXD=0,12,”192.168.200.1”,5025,HELLO,WORLD!

8 Test Application

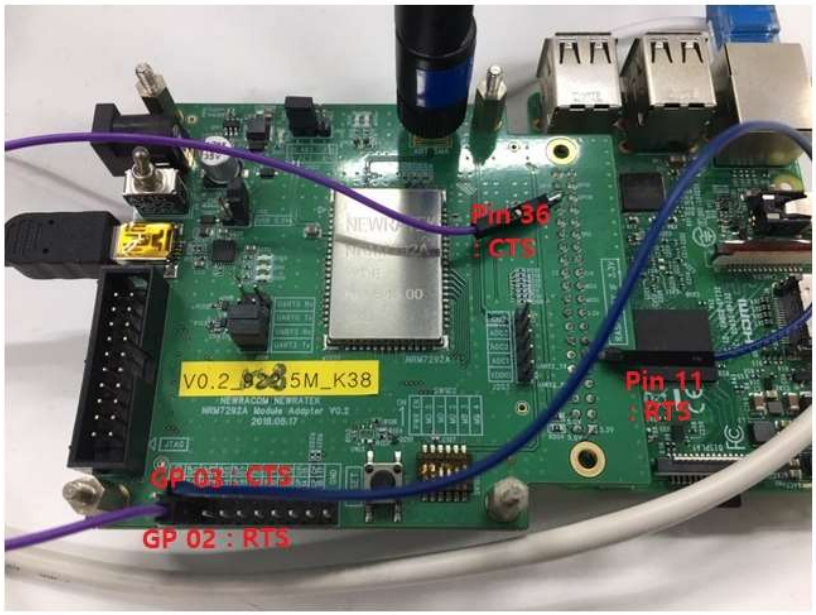
8.1 Command Line Interface (raspi-atcmd-cli)

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.

The NRM7292 EVB can use the Raspberry Pi as a host. The Raspberry Pi board is connected to the NRM7292 EVB through a 40-pin header. The 40-pin header has signals for UART and SPI.

Figure 8.1 Pin map of 40-pin header for Raspberry Pi

The NRM7292 EVB and Raspberry Pi board is connected as shown in the Figure 8.2.
Both PIN11_UART0_RTS and PIN36_UART0_CTS used for hardware flow control on the UART needs to be directly connected to a 20-pin header in the NRM7292 EVB by a jumper wire.



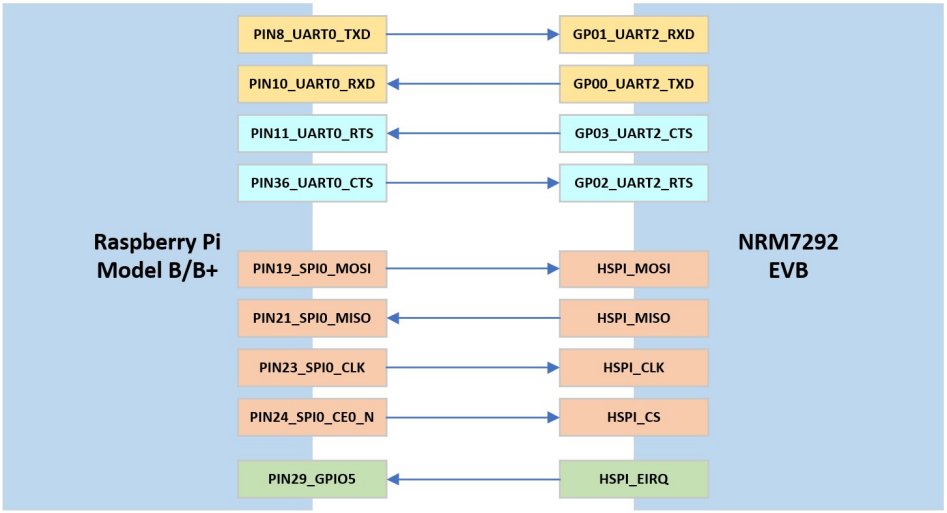
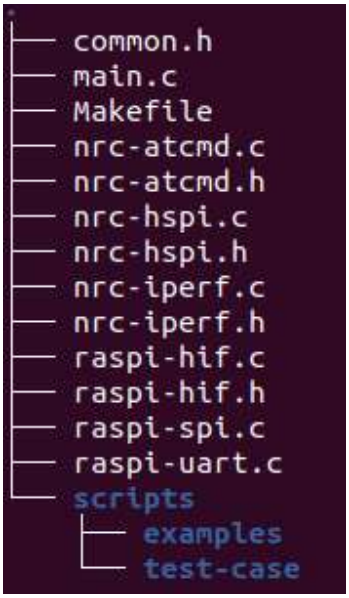


Figure 8.2 Connection between NRM7292 EVB and Raspberry Pi

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	Protocol driver for HSPI.

nrc-hspi.h	***Refer to this file to communicate with the ATCMD firmware via SPI from the host.**
nrc-iperf.c nrc-iperf.h	Iperf server/client
raspi-hif.c raspi-hif.h	Wrapper for user mode driver.
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

2) Build

Copy all source files from standalone/sdk/apps/atcmd/host to the Raspberry Pi's home directory. And build the CLI application with the make command.

```
$ cd $HOME
$ cd host/raspi-atcmd-cli
$ make [clean]
```

```
pi@raspberrypi:~/raspi-atcmd-cli $ make clean
removed 'raspi-atcmd-cli'
pi@raspberrypi:~/raspi-atcmd-cli $ make
cc -g -o raspi-atcmd-cli raspi-spi.c raspi-uart.c raspi-hif.c nrc-hspi.c nrc-atcmd.c
 nrc-iperf.c main.c -pthread -Wall -Wno-unused-function -lpthread
```

3) Run

A. It needs be executed on Raspberry Pi3 or Host which can use AT command through UART or SPI

Run the CLI application using a raspi-atcmd-cli file. And enter the AT command as in Terminal program.

- **Help**

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

```
pi@raspberrypi:~/raspi-atcmd-cli $ ./raspi-atcmd-cli -h
raspi-atcmd-cli version 1.2.0
Copyright (c) 2019-2020  <NEWRACOM LTD>

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

UART/SPI:
  -D, --device #        specify the device. (default: /dev/ttyAMA0, /dev/spidev0.0)
  -s, --script #        specify the script file.

UART:
  -U  --uart            use the UART to communicate with the target.
  -b, --baudrate #      specify the baudrate for the UART. (default: 38,400 bps)
  -f  --flowctrl        enable RTS/CTS signals for the hardware flow control on the UART. (default: disable)

SPI:
  -S  --spi             use the SPI to communicate with the target.
  -c, --clock #         specify the clock frequency for the SPI. (default: 16,000,000 Hz)

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

- **SPI**

  A clock of the SPI master is up to 20MHz. The clock default setting is 16 MHz.

  $ sudo ./raspi-atcmd-cli **-S [-c <clock>]**

```
pi@raspberrypi:~/raspi-atcmd-cli $ sudo ./raspi-atcmd-cli -S -c 16000000

 [ SPI ]
  - device: /dev/spidev0.0
  - clock: 16000000 Hz

# 
```

- **UART**

  A default setting for baud rate is 115200bps without the hardware flow control.

  $ sudo ./raspi-atcmd-cli **-U [-b <baudrate>]**

```
pi@raspberrypi:~/raspi-atcmd-cli $ sudo ./raspi-atcmd-cli -U -b 115200

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

# 
```

  If the baud rate setting is more than 115200bps, the hardware flow control needs to be enabled on the UART.

  $ sudo ./raspi-atcmd-cli **-U -f [-b <baudrate>]**

```
pi@raspberrypi:~/raspi-atcmd-cli $ sudo ./raspi-atcmd-cli -U -f -b 2000000

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

# 
```

- **Log**

  Getting the informations.

```
pi@raspberrypi:~/raspi-atcmd-cli $ sudo ./raspi-atcmd-cli -S

 [ SPI ]
  - device: /dev/spidev0.0
  - clock: 16000000 Hz


# AT
 SEND: AT
 RECV: OK

# AT+VER?
 SEND: AT+VER?
 RECV: +VER:"1.3.2","1.18.3"
 RECV: OK

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

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

# AT+WMACADDR?
 SEND: AT+WMACADDR?
 RECV: +WMACADDR:"8c:0f:fa:00:2b:73"
 RECV: OK

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

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

#
```

Connecting to an AP.

```
# AT+WSCAN
 SEND: AT+WSCAN
 RECV: +WSCAN:"8c:0f:fa:00:2b:a1",918.0,-16,"[ESS]","halow_atcmd_open"
 RECV: OK

# AT+WCONN="halow_atcmd_open"
 SEND: AT+WCONN="halow_atcmd_open"
 RECV: OK

# AT+WDHCP
 SEND: AT+WDHCP
 RECV: +WDHCP:"192.168.200.23","255.255.255.0","192.168.200.1"
 RECV: OK

# AT+WIPADDR?
 SEND: AT+WIPADDR?
 RECV: +WIPADDR:"192.168.200.23","255.255.255.0","192.168.200.1"
 RECV: OK

# AT+WPING
 SEND: AT+WPING
 RECV: +WPING:64,"192.168.200.1",1,64,6
 RECV: +WPING:64,"192.168.200.1",2,64,6
 RECV: +WPING:64,"192.168.200.1",3,64,18
 RECV: +WPING:64,"192.168.200.1",4,64,3
 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",50000,0
 RECV: OK

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

# ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SEND: DATA 10

#  RECV: +RXD:0,10


# AT
 SEND: AT
 RECV: OK

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

# ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SEND: DATA 26

#  RECV: +RXD:0,26
 RECV: +SEVENT:"SEND_IDLE",0,26


# DKAJFDKLAJFEKJAKEJA'A'D'AFLMEKFJAIDJFAKJEAFJDFJADKJFAEKJFADKJFADF
 SEND: DATA 65

#  RECV: +RXD:0,65
 RECV: +SEVENT:"SEND_IDLE",0,91


# AT
 SEND: AT
 RECV: OK

#  RECV: +SEVENT:"SEND_EXIT",0,91


#
```

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

```
# AT+SOPEN="udp",60000
 SEND: AT+SOPEN="udp",60000
 RECV: +SOPEN:1
 RECV: OK

# AT+SLIST?
 SEND: AT+SLIST?
 RECV: +SLIST:0,"TCP","192.168.200.1",50000,0
 RECV: +SLIST:1,"UDP","0.0.0.0",0,60000
 RECV: OK

# AT+SSEND=1,"192.168.200.1",55000,10
 SEND: AT+SSEND=1,"192.168.200.1",55000,10
 RECV: OK

# ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SEND: DATA 10

#  RECV: +RXD:1,10


# AT
 SEND: AT
 RECV: OK

# AT+SSEND=1,"192.168.200.1",55000
 SEND: AT+SSEND=1,"192.168.200.1",55000
 RECV: OK

# ABCDEFGHIJKLMNOPQRSTUVWXYZ
 SEND: DATA 26

#  RECV: +RXD:1,26
 RECV: +SEVENT:"SEND_IDLE",1,26


# AKDJFAOIEJFAI3EJFKALDJFAGJADFKGJA90E9RFA398RU4EYAGFAKJEJADKGJ3
 SEND: DATA 62

#  RECV: +RXD:1,62
 RECV: +SEVENT:"SEND_IDLE",1,88


# AT
 SEND: AT
 RECV: OK

#  RECV: +SEVENT:"SEND_EXIT",1,88


#
```

Closing all sockets.

```
# AT+SLIST?
 SEND: AT+SLIST?
 RECV: +SLIST:0,"TCP","192.168.200.1",50000,0
 RECV: +SLIST:1,"UDP","192.168.200.1",55000,60000
 RECV: OK

# AT+SCLOSE
 SEND: AT+SCLOSE
 RECV: +SCLOSE:0
 RECV: +SCLOSE:1
 RECV: OK

# EXIT
```

4) Run with script file

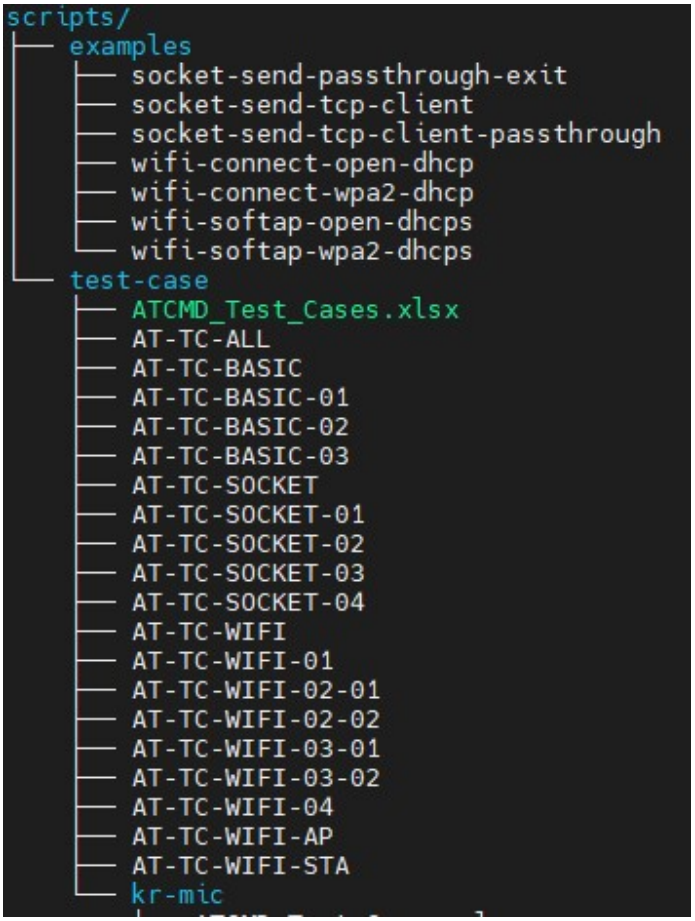
CLI application provides the option to run the script files. (-s/--script)

```
UART/SPI:
  -D, --device #        specify the device. (default: /dev/ttyAMA0, /dev/spidev0.0)
  -s, --script #        specify the script file.
```

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

Command	Description	Example
ECHO “<message>”	Print a message.	ECHO “AT Command”
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
CALL <script_file>	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
HOLD	Pause until there is keyboard input.	ECHO “Run an AP in open mode” HOLD

***) Users can refer to the script files under the scripts directory.**



- **SPI**

  $ sudo ./raspi-atcmd-cli -S [-c <clock>] **-s <script_file>**

  (Example) $sudo ./raspi-atcmd-cli -S -s scripts/test-case/AT-TC-ALL

- **UART**

  $ sudo ./raspi-atcmd-cli -U [-b <baudrate>] **-s <script_file>**

  (Example) $sudo ./raspi-atcmd-cli -U -s scripts/test-case/AT-TC-ALL

- **UART with H/W flow control**

  $ sudo ./raspi-atcmd-cli -U [-b <baudrate>] -f **-s <script_file>**

  (Example) $sudo ./raspi-atcmd-cli -U -s scripts/test-case/AT-TC-ALL

5) Iperf

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

iperf {-h|--help}

```
# iperf -h

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 in passthrough mode (always negative in UDP)

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 the following AT commands in the CLI application.

```
# AT
 SEND: AT
 RECV: OK

# AT+WSCAN
 SEND: AT+WSCAN
 RECV: +WSCAN:"8c:0f:fa:00:30:73",921.5,-10,"[WPA2-PSK-CCMP][ESS]","halow_atcmd_wpa2"
 RECV: OK

# AT+WCONN="halow_atcmd_wpa2","wpa2","12345678"
 SEND: AT+WCONN="halow_atcmd_wpa2","wpa2","12345678"
 RECV: OK

# AT+WDHCP
 SEND: AT+WDHCP
 RECV: +WDHCP:"192.168.200.31","255.255.255.0","192.168.200.1"
 RECV: OK
```

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

  $ sudo ./raspi-atcmd-cli -S -s scripts/example/wifi-connect-wpa2-dhcp

```
[ SPI ]
 - device: /dev/spidev0.0
 - clock: 16000000 Hz

CALL: scripts/examples/wifi-connect-wpa2-dhcp

SEND: AT+WDISCONN
RECV: OK

ECHO: Run an AP in wpa2 mode.
ECHO:  - SSID : halow_atcmd_wpa2
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:00:30:73",921.5,-10,"[WPA2-PSK-CCMP][ESS]","halow_atcmd_wpa2"
RECV: +WSCAN:"00:10:40:39:81:13",920.0,-73,"[WPA2-PSK-CCMP][ESS]","halow_tcp_test"
RECV: OK
SEND: AT+WDISCONN
RECV: OK
SEND: AT+WCONN="halow_atcmd_wpa2","wpa2","12345678"
RECV: OK
SEND: AT+WDHCP
RECV: +WDHCP:"192.168.200.31","255.255.255.0","192.168.200.1"
RECV: OK

DONE: scripts/examples/wifi-connect-wpa2-dhcp
```

- **Iperf TCP Client**

  # iperf -c <server_ip> [-t <time>] [-P]

```
# iperf -c 192.168.200.1 -t 5 -P

 [ IPERF OPTION ]
  - role: client
  - protocol: tcp
  - server_port: 5001
  - server_ip: 192.168.200.1
  - send_time: 5
  - send_passthrough: on
  - report_interval: 1


 SEND: AT+SRXLOGLEVEL=1
 RECV: OK
 SEND: AT+SOPEN="tcp","192.168.200.1",5001
 RECV: +SOPEN:0
 RECV: OK

 [ IPERF TCP Client ]
  Sending 1470 byte datagram ...
    Interval        Transfer      Bandwidth
    0.0 ~  1.0 sec  212.46 KBytes   1.74 Mbits/sec
    1.0 ~  2.0 sec  206.72 KBytes   1.69 Mbits/sec
    2.0 ~  3.0 sec  199.54 KBytes   1.63 Mbits/sec
    3.0 ~  4.0 sec  208.15 KBytes   1.70 Mbits/sec
    4.0 ~  5.0 sec  205.28 KBytes   1.67 Mbits/sec
    0.0 ~  5.0 sec   1.01 MBytes   1.69 Mbits/sec
  Sent 719 datagrams
  Done

 SEND: AT
 RECV: OK
 SEND: AT+SCLOSE
 RECV: +SEVENT:"SEND_EXIT",0,1056930
 RECV: +SCLOSE:0
 RECV: OK
 SEND: AT+SRXLOGLEVEL=0
 RECV: OK
```

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 iperf tcp server stops after 1 second.

```
pi@raspberrypi:~ $ 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 192.168.200.31 port 52436
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 1.0 sec   203 KBytes  1.66 Mbits/sec
[  4]  1.0- 2.0 sec   204 KBytes  1.67 Mbits/sec
[  4]  2.0- 3.0 sec   205 KBytes  1.68 Mbits/sec
[  4]  3.0- 4.0 sec   206 KBytes  1.69 Mbits/sec
[  4]  4.0- 5.0 sec   205 KBytes  1.68 Mbits/sec
[  4]  5.0- 6.0 sec  8.17 KBytes  66.9 Kbits/sec
[  4]  6.0- 7.0 sec  0.00 Bytes  0.00 bits/sec
[  4]  0.0- 7.0 sec  1.01 MBytes  1.20 Mbits/sec
```

- **Iperf UDP Client**

  # iperf -u -c <server_ip> [-t <time>] [-P]

```
# iperf -u -c 192.168.200.1 -t 5 -P

[ IPERF OPTION ]
 - role: client
 - protocol: udp
 - server_port: 5001
 - server_ip: 192.168.200.1
 - send_time: 5
 - send_passthrough: on
 - datagram_size: 1470
 - report_interval: 1


SEND: AT+SRXLOGLEVEL=1
RECV: OK
SEND: AT+SOPEN="udp",50000
RECV: +SOPEN:0
RECV: OK

[ IPERF UDP Client ]
 Sending 1470 byte datagrams ...
   Interval        Transfer      Bandwidth
   0.0 ~  1.0 sec  235.43 KBytes   1.91 Mbits/sec
   1.0 ~  2.0 sec  235.43 KBytes   1.93 Mbits/sec
   2.0 ~  3.0 sec  236.87 KBytes   1.94 Mbits/sec
   3.0 ~  4.0 sec  241.17 KBytes   1.95 Mbits/sec
   0.0 ~  5.0 sec   1.15 MBytes   1.92 Mbits/sec
 Sent 818 datagrams
 Server Report:
   0.0 ~  5.0 sec   1.15 MBytes   1.92 Mbits/sec  4.883 ms  0/818 (0%)
 Done

RECV: +SEVENT:"SEND_IDLE",0,0
SEND: AT
RECV: OK
SEND: AT+SCLOSE
RECV: +SEVENT:"SEND_EXIT",0,0
RECV: +SCLOSE:0
RECV: OK
SEND: AT+SRXLOGLEVEL=0
RECV: OK
```

- **Iperf TCP Server**

  # iperf -s

```
# iperf -s

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


SEND: AT+SRXLOGLEVEL=1
RECV: OK
SEND: AT+SOPEN="tcp",5001
RECV: +SOPEN:0
RECV: OK

[ IPERF TCP Server ]
 Connected with client: id=1
   Inerval         Transfer       Bandwidth
   0.0 ~  1.0 sec  196.00 KBytes   1.60 Mbits/sec
   1.0 ~  2.0 sec  184.74 KBytes   1.51 Mbits/sec
   2.0 ~  3.0 sec  183.38 KBytes   1.50 Mbits/sec
   3.0 ~  4.0 sec  188.00 KBytes   1.54 Mbits/sec
   4.0 ~  5.2 sec  225.48 KBytes   1.59 Mbits/sec
   0.0 ~  5.2 sec  977.60 KBytes   1.55 Mbits/sec
 Done

SEND: AT+SCLOSE
RECV: +SCLOSE:0
RECV: OK
SEND: AT+SRXLOGLEVEL=0
RECV: OK
```

- **Iperf UDP Server**

  # iperf -u -s

```
# iperf -u -s

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


SEND: AT+SRXLOGLEVEL=1
RECV: OK
SEND: AT+SOPEN="udp",5001
RECV: +SOPEN:0
RECV: OK

[ IPERF UDP Server ]
 Connected with client: 192.168.200.1 port 43056
   Inerval         Transfer       Bandwidth        Jitter     Lost/Total Datagrams
   0.0 ~  1.0 sec  235.43 KBytes   1.93 Mbits/sec   2.399 ms     0/  164 (0%)
   1.0 ~  2.0 sec  245.48 KBytes   2.01 Mbits/sec   1.527 ms     0/  171 (0%)
   2.0 ~  3.0 sec  241.17 KBytes   1.97 Mbits/sec   0.735 ms     1/  169 (0.59%)
   3.0 ~  4.0 sec  241.17 KBytes   1.97 Mbits/sec   1.365 ms     1/  169 (0.59%)
   4.0 ~  5.0 sec  252.66 KBytes   2.01 Mbits/sec   2.942 ms     2/  178 (1.1%)
   0.0 ~  5.0 sec   1.19 MBytes   1.98 Mbits/sec   2.942 ms     4/  851 (0.47%)
 Done

SEND: AT+SCLOSE
RECV: +SCLOSE:0
RECV: OK
RECV: +SEVENT:"RECV_ERROR",0,-9
SEND: AT+SRXLOGLEVEL=0
RECV: OK
```

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

A remote server/client application run one server or client at a time. This application is a Linux application that can be executed on Raspberry Pi. After running the AP, rasp-atcmd-remote must be run on the host running the AP. That is, the AP must run UDP or TCP server / client, and a user can send and receive data using socket-related AT commands through the CLI application(rasp-atcmd-cli).

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

2) Build

$ cd $HOME
$ cd host/raspi-atcmd-remote
$ make [clean]

```
pi@raspberrypi:~/host/raspi-atcmd-remote $ make clean
removed 'raspi-atcmd-remote'
pi@raspberrypi:~/host/raspi-atcmd-remote $ make
cc -g -o raspi-atcmd-remote main.c -Wall -Wno-unused-function
```

3) Run

A. It needs be executed on Raspberry Pi3 running as a Host mode AP.

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

```
pi@raspberrypi:~/host/raspi-atcmd-remote $ ./raspi-atcmd-remote
raspi-atcmd-remote version 1.0.0
Copyright (c) 2019-2020  <NEWRACOM LTD>

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

UDP:
  -u, --udp             use UDP.

TCP:
  -t, --tcp             use TCP
  -s, --server          run in server mode
  -c, --client #        run in client mode

UDP/TCP:
  -p, --port #          set port number (default: 50000)
  -e, --echo            enable echo for received packets (default: disable)

  -v, --version         print version information and quit.
  -h, --help            print this message and quit.
```

Examples:

Mode	Command
UDP Server or Client	$ ./raspi-atcmd-remote -u -p 50000 [-e]
TCP Server	$ ./raspi-atcmd-remote -t -s -p 50000 [-e]
TCP Client	$ ./raspi-atcmd-remote -t -c 192.168.200.39 -p 60000 [-e]

4) Log

UDP Server or Client ($ ./raspi-atcmd-remote -u -p 50000 -e)

```
[ UDP ]
  - bind_port : 50000
  - echo : on
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
```

TCP Server ($ ./raspi-atcmd-remote -t -s -p 50000 -e)

```
[ TCP_SERVER ]
  - listen_port : 50000
  - echo : on
LISTEN ...
CONNECT: addr=192.168.200.39 port=52433

RECV: addr=192.168.200.39 port=52433 len=16
SEND: addr=192.168.200.39 port=52433 len=16
RECV: addr=192.168.200.39 port=52433 len=16
SEND: addr=192.168.200.39 port=52433 len=16
RECV: addr=192.168.200.39 port=52433 len=16
SEND: addr=192.168.200.39 port=52433 len=16
RECV: addr=192.168.200.39 port=52433 len=16
SEND: addr=192.168.200.39 port=52433 len=16
RECV: addr=192.168.200.39 port=52433 len=16
SEND: addr=192.168.200.39 port=52433 len=16
```

TCP Client ($ ./raspi-atcmd-remote -t -c 192.168.200.39 -p 60000 -e)

```
[ TCP_CLIENT ]
  - server_ip : 192.168.200.39
  - server_port : 60000
  - echo : on
CONNECT: addr=192.168.200.39 port=60000

RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
```

9 Examples

9.1 Connect to 11ah AP and Send UDP Data to UDP Server

Configuration

- 11ah AP (IP: 192.168.200.1, SSID: halow_demo, Security: Open, DHCP Server: O)
- UDP Server (Port 8800, IP 192.168.200.10, DHCP Server)
- UDP Client (Port 1000, DHCP Client)

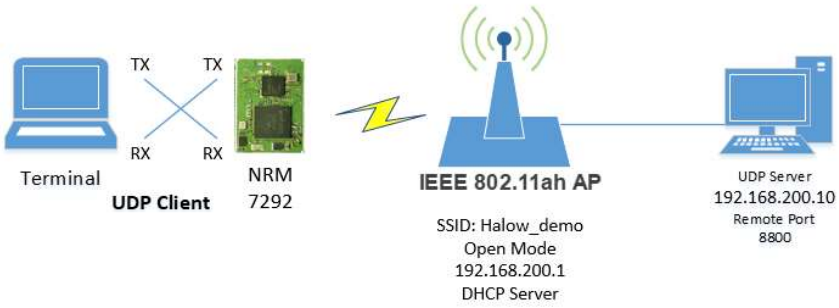


Figure 9.1 Configuration of Example1

[AT Command used for example1]

1) Find AP
   ➔ AT+WSCAN

2) Try to connection Wi-Fi AP (SSID: halow_demo, Open Mode)
   ➔ AT+WCONN=”halow_demo”

3) Try to DHCP
   ➔ AT+WDHCP

4) Check IP address after connection
   ➔ AT+WIPADDR?

5) Check Connection to AP using PING
   ➔ AT+WPING=”192.168.200.1”

6) Create UDP Client Socket to Server (Server Port 8800, Server IP 192.168.200.10)
   ➔ AT+SOPEN=”UDP”,1000

7) Check UDP Socket

➔ AT+SLIST?

8) Send Data to UDP Server
   ➔ AT+SSEND=0,"192.168.200.10",8800,10
   "0123456789"

9) Close UDP Socket
   ➔ AT+SCLOSE=0

10) Check UDP Socket
   ➔ AT+SLIST?

9.2 Connect to 11ah AP and Send TCP Data to TCP Server

Configuration

- 11ah AP (IP: 192.168.200.1, SSID: halow_demo, Security: WPA2, PW:12345678, NO DHCP)
- TCP Server (Port 8098, IP 192.168.200.10)
- TCP Client (IP 192.168.200.20)

Figure 9.2 Configuration of Example2

[AT Command used for example2]

1) Find AP
   ➔ AT+WSCAN

2) Set Static IP
   ➔ AT+WIPADDR=”192.168.200.20”,”255.255.255.0”,”192.168.200.1”

3) Try to connection Wi-Fi AP (SSID: halow_demo, Security Mode, Static IP)
   ➔ AT+WCONN=”halow_demo”,”wpa2”,”12345678”

4) Check IP address after connection
   ➔ AT+WIPADDR?

5) Check Connection to AP using PING
   ➔ AT+WPING=”192.168.200.1”

6) Create TCP Client Socket to Server (Server Port 8800, Server IP 192.168.200.10)
   ➔ AT+SOPEN=”TCP”,”192.168.200.10”,8098

7) Check UDP Socket
   ➔ AT+SLIST?

8) Send Data to TCP Server
   ➔ AT+SSEND=0,10
   "0123456789"

9) Close TCP Socket
   ➔ AT+SCLOSE=0

10) Check TCP Socket
   ➔ AT+SLIST?

10 Revision History

Revision No	Date	Comments
Ver 1.0	03/28/2019	Initial version for customer release created
Ver 1.1	07/02/2019	Sample Applications updated
Ver 1.2	08/01/2019	HW Flow Control added
Ver 1.3	09/17/2019	Additional AT-commands added
Ver 1.4	11/18/2019	Download binary update & remove description wpa security
Ver 1.5	02/14/2020	Improved command descriptions
Ver 1.6	03/25/2020	SPI connection and CLI application added
Ver 1.7	03/31/2020	AT+STXMODE, AT+SRXMODE, AT+SRXAVAIL and AT+SRECV commands removed
Ver 1.8	04/07/2020	Socket related events removed and added CLI application updated
Ver 1.9	05/15/2020	Ping size parameter removed Test Application added
Ver 1.10	05/22/2020	AT+WDHCPS, AT+WSOFTAP commands added
Ver 1.11	06/03/2020	AT+SLEEP command added
Ver 1.12	07/15/2020	“Chapter 2.2 Building the firmware” added
Ver 1.13	08/04/2020	UART default baudrate changed (38400 -> 115200) “4) Run with script file” in chapter 8.1 added
Ver1.14	08/13/2020	BSSID in AT+WCONN command added
Ver1.15	08/24/2020	AT+WROAM command added ROAMING event added
Ver1.16	09/02/2020	AT+WFOTA command added FOTA event added
Ver1.17	10/08/2020	In raspi-atcmd-cli application, Iperf command supported
Ver1.18	11/24/2020	FOTA updated - New events added - Get-bin-crc.sh removed - Update-fota-info.sh added
Ver1.19	06/15/2021	AT+WSTAINFO command added
Ver1.19.1	07/29/2021	Some examples fixed