NRC7292 Evaluation Kit User Guide

(AT-command)

Ultra-low power & Long-range Wi-Fi

Ver 1.13
Aug 04,2020

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

1 Overview 7

2 Basic Setup 7

2.1 Hardware connection 7

2.2 Building the firmware 10

2.3 Downloading the firmware and initializing the EVB 11

3 AT Command Type 14

4 Return for Commands 15

5 Basic AT Commands 15

5.1 AT 16

5.2 ATE 16

5.3 ATZ 16

5.4 AT+VER 16

5.5 AT+UART 17

5.6 AT+GPIOCONF 17

5.7 AT+GPIOVAL 18

5.8 AT+ADC 19

5.9 AT+SLEEP 20

6 Wi-Fi AT Commands 21

6.1 AT+WMACADDR 22

6.2 AT+WCOUNTRY 22

6.3 AT+WTXPOWER 23

6.4 AT+WRXSIG 23

6.5 AT+WRATECTRL 24

6.6 AT+WMCS 24

6.7 AT+WTSF 25

6.8 AT+WDHCP 25

6.9 AT+WDHCPS 26

6.10 AT+WIPADDR 26

6.11 AT+WSCAN 27

6.12 AT+WCONN 27

6.13 AT+WDISCONN 28

6.14 AT+WPING 28

6.15 AT+WSOFTAP 29

6.16 AT+WTIMEOUT 30

6.17 +WEVENT 31

7 Socket AT Commands 32

7.1 AT+SOPEN 33

7.2 AT+SCLOSE 34
7.3 AT+SLIST 34
7.4 AT+SRXLOGLEVEL 35
7.5 AT+SSEND 35
7.6 AT+STIMEOUT 37
7.7 +SEVENT 38
7.8 +RXD 39
8 Test Application 40
8.1 Command Line Interface (raspi-atcmd-cli) 40
8.2 Remote Server/Client (raspi-atcmd-remote) 50
9 Examples 53
9.1 Connect to 11ah AP and Send UDP Data to UDP Server 53
9.2 Connect to 11ah AP and Send TCP Data to TCP Server 55
10 Revision History 57

List of Tables

Table 3.1 AT-command type 14
Table 8.1 raspi-atcmd-cli source files........ 41
Table 8.2 raspi-atcmd-remote source files 50

List of Figures

Figure 2.1 NRC7292 evaluation board 7
Figure 2.2 UART connection between EVK and external host 8
Figure 2.3 SPI connection between EVK and external host 9
Figure 2.4 Download mode configuration 11
Figure 2.5 Detected serial port in the device manager 11
Figure 2.6 NRC7292 binary download tool 12
Figure 2.7 Error pop-up 12
Figure 2.8 Standalone mode configuration 13
Figure 2.9 Power switch 13
Figure 8.1 Pin map of 40-pin header for Raspberry Pi 40
Figure 8.2 Connection between NRM7292 EVB and Raspberry Pi 41
Figure 9.1 Configuration of Example1 53
Figure 9.2 Configuration of Example2 55

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.

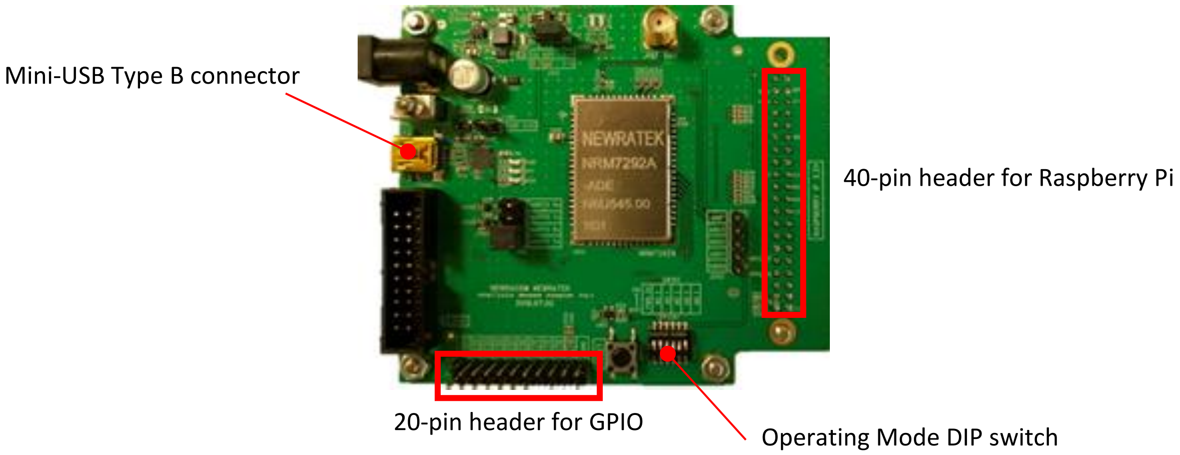

Mini-USB Type B connector
40-pin header for Raspberry Pi
20-pin header for GPIO
Operating Mode DIP switch
NEWRATEK
NRM7292A

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.

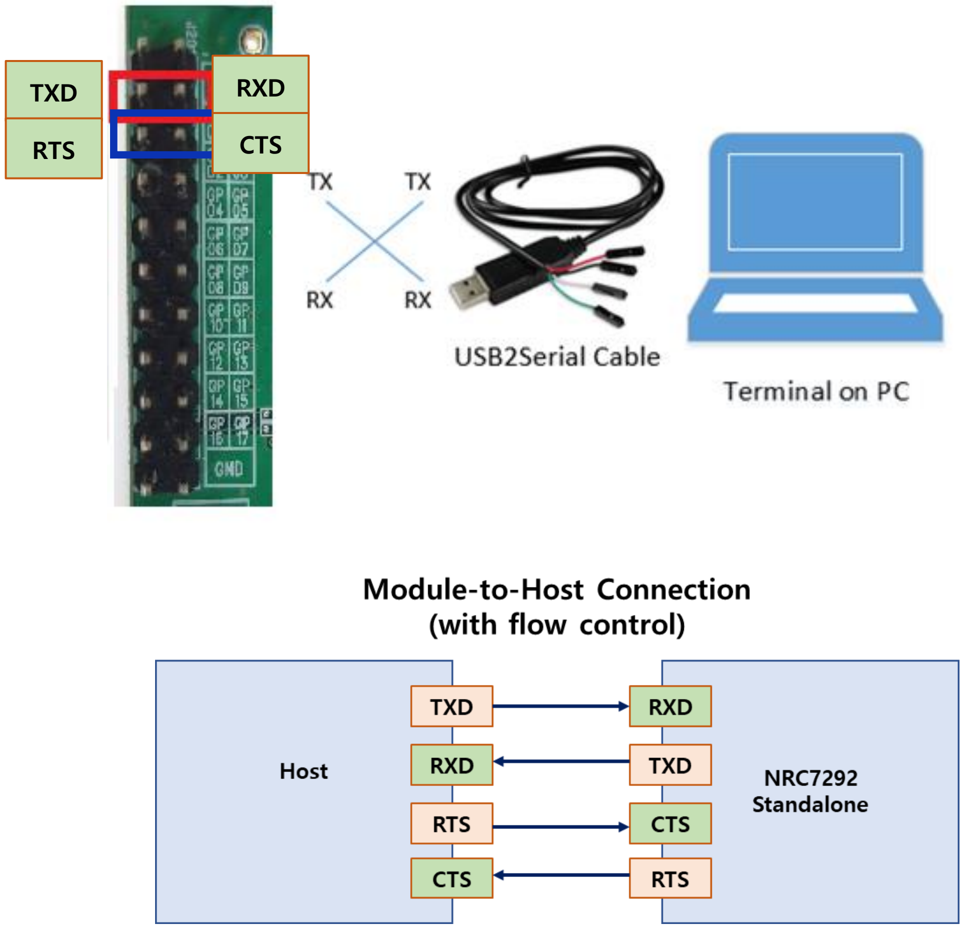


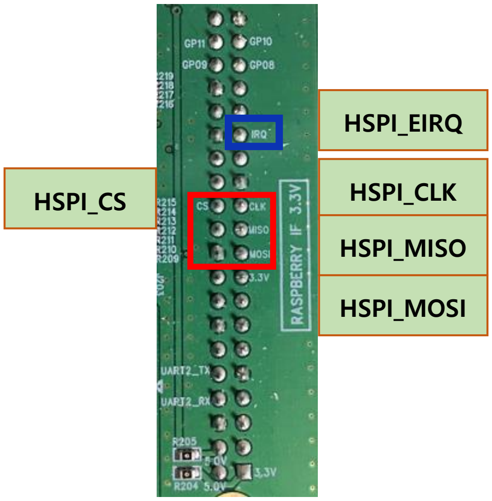
Figure 2.2 UART connection between EVK and external host

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.



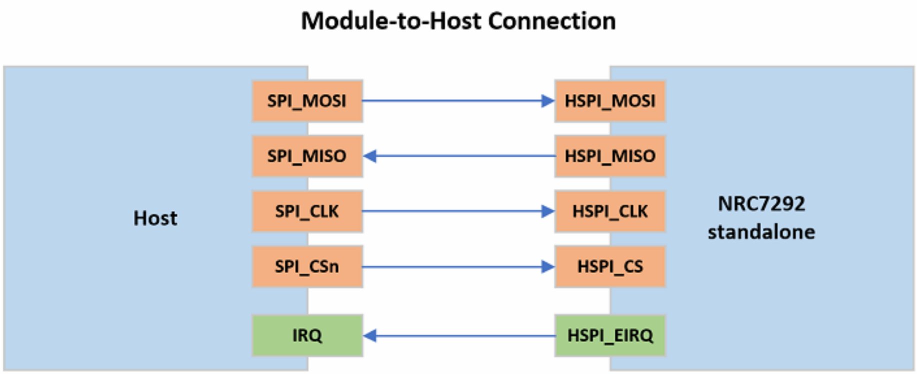
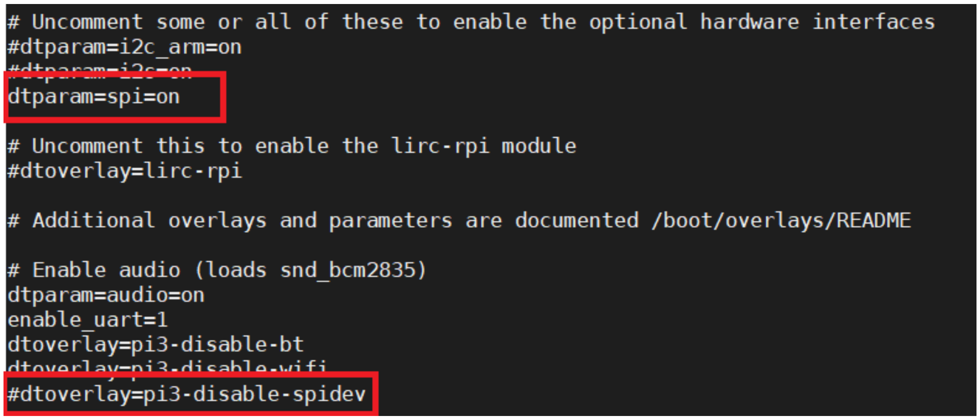


Figure 2.3 SPI connection between EVK and external host

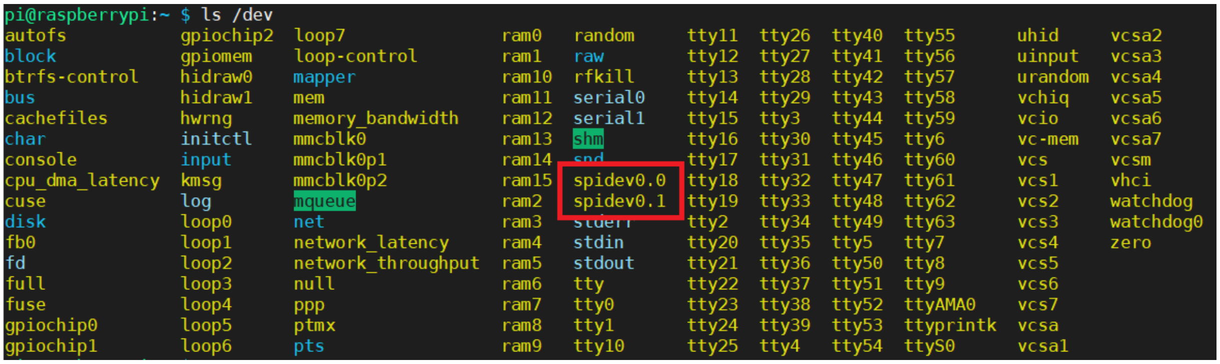
To perform AT command communication through SPI on Raspberry Pi, spidev of Raspberry Pi3 must be enabled.

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 a “doc/UG-7292-004-Standalone SDK.pdf” file provided with Standalone SDK.

(Chapter 2 Setup S/W build environment, NRC7292 Evaluation Kit User Guide (Standalone SDK))

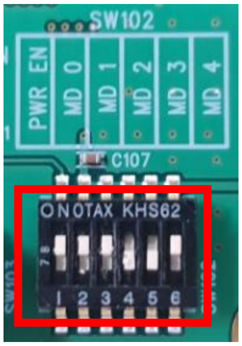
1. **HSPI mode:**
    - make select target=nrc7292.sdk.release APP_NAME=ATCMD_HSPI
2. **UART mode (without hardware flow control):**
    - make select target=nrc7292.sdk.release APP_NAME=ATCMD_UART
3. **UART mode (with hardware flow control):**
    - make select target=nrc7292.sdk.release APP_NAME=ATCMD_UART_HFC

2.3 Downloading the firmware and initializing the EVB

Follow the procedures outlined below to download the firmware binary onto the EVB:

1) DIP switch configuration for download mode

Configure the DIP switch to download mode as shown in Figure 2.4 below.



U U U D D U U(Up), D(Down)

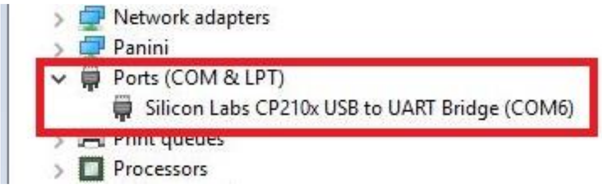
Figure 2.4 Download mode configuration

2) UART connection

A Mini-USB Type B connector cable is required to download the firmware onto the EVB from the PC. The user must install the Silicon Labs UART driver before the PC can recognize the module. The latest version for the driver is available on the website:

https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

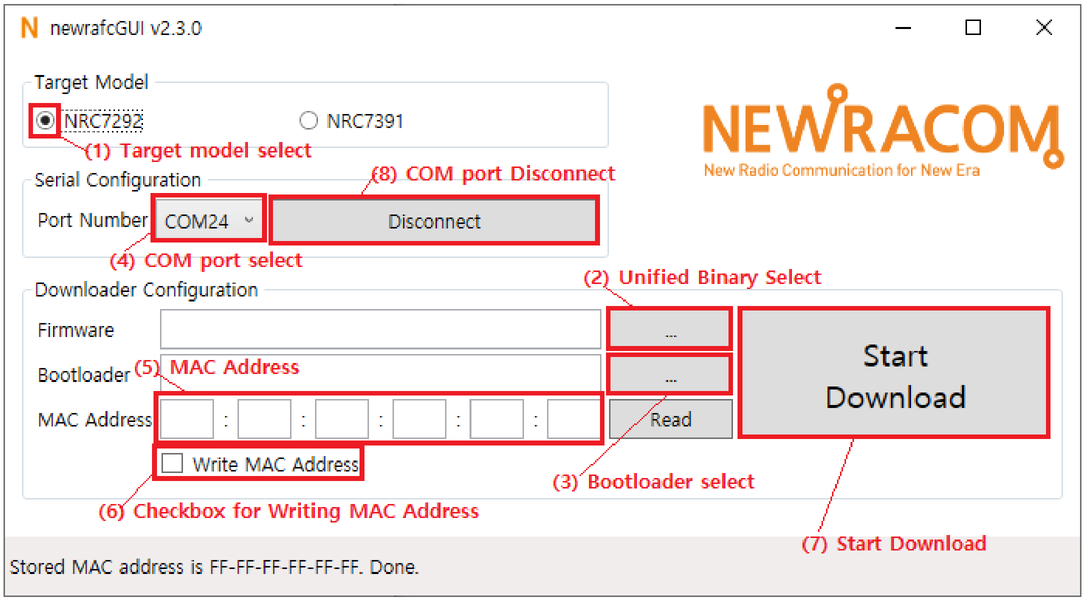
After installing the driver and connecting the EVB to the PC, the Silicon Labs USB to UART device should appear under Ports in Device Manager on the PC with the associated serial port name displayed in the form "COMXXX".



- Network adapters
- Panini
- Ports (COM & LPT)
  - Silicon Labs CP210x USB to UART Bridge (COM6)
- Print queues
- Processors

Figure 2.5 Detected serial port in the device manager

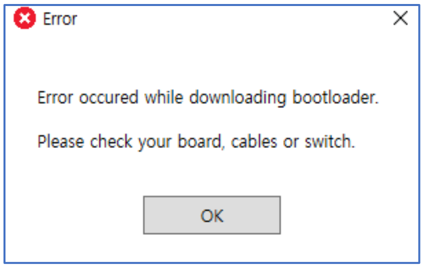
3) Downloading the firmware using the download tool



Stored MAC address is FF-FF-FF-FF-FF-FF. Done.

Figure 2.6 NRC7292 binary download tool

Start by launching the firmware download tool "newrafcGUI_v2.4.4.exe". Provide the paths to the firmware binary and the bootloader. The default bootloader path is "./bootloader/boot.bin". Select the serial port associated with the EVB and optionally provide the MAC address to be written onto the flash, if necessary. Be sure that MAC address will be written when 'Write MAC Address' checkbox was checked. Press the start button to start downloading the binary onto the flash.



Error occured while downloading bootloader.

Please check your board, cables or switch.

Figure 2.7 Error pop-up

If the download fails, make sure that:

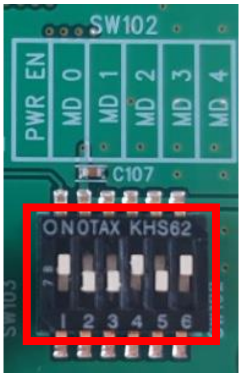
1. no other process such as a terminal emulator is occupying the serial port,
2. the EVB is powered on,
3. the DIP switch is configured to download mode,

and press the reset button before trying again.

Note that the download procedure only needs to be done once.

4) DIP switch configuration for standalone operation mode

After downloading the firmware binary onto the EVB, configure the DIP switch to standalone operation mode as shown in Figure 2.8.



U(Up), D(Down)

Figure 2.8 Standalone mode configuration

5) Switching on the EVB and enabling the UART AT-command communication

Turn on the power and press the reset button to start the module.

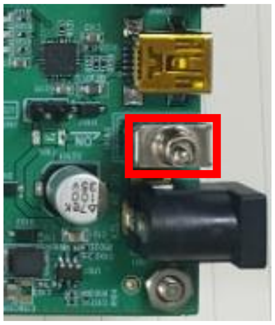


Figure 2.9 Power switch

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

<table>
<tr><td><b>Command</b></td><td colspan="2">ATE0 or ATE1</td></tr>
<tr><td><b>Response</b></td><td colspan="2">OK</td></tr>
<tr><td><b>Description</b></td><td colspan="2">Enable (ATE1) or disable (ATE0) echo. (default: disable)
Note: Echo should typically be enabled for manual communication via a terminal.</td></tr>
<tr><td><b>Example</b></td><td>ATE1
OK</td><td>ATE0
OK</td></tr>
</table>

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:<AT firmware version>,<S/W package version> OK
Description	Fetch the AT firmware version and software package version.
Example	AT+VER? +VER:”1.0.0”,”2.0.0”

	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 or retrieves information about the current AP.
AT+WDISCONN	Disconnect from the AP or abort an on-going connection process.
AT+WPING	Initiate a ping session.
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. *) 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>”[,”<security>”[,”<password>”]] **GET** AT+WCONN?
Response	**SET** OK **GET**

	+WCONN=”<ssid>”,”<security>”,”<password>”,”<state>” OK
Parameters	**<ssid>** The SSID of the AP. **<security>** open*, wpa2 **<password> (wpa2 security option only)** The password when wpa2 security option is used. **<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 disconnect AT-command “AT+WDISCONN” before a connection is attempted again with “AT+WCONN”.
Example	AT+WCONN=”demo_ap”,”wpa2”,”kds3f3” OK AT+WCONN? +WCONN:”demo_ap”,”wpa2”,”kds3f3”,”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+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. **<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","kds3f3" OK AT+WIPADDR="192.168.1.1","255.255.255.0","192.168.1.1" OK AT+DHCPS +WDHCP:"192.168.1.1","255.255.255.0","192.168.1.1" OK AT+WSOFTAP? +WCONN:918.5,"halow_softap","wpa2","kds3f3","dhcp" OK

6.16 AT+WTIMEOUT

Command	**SET** AT+WTIMEOUT="<command>",<timeout> **GET** AT+WTIMEOUT?
Response	**SET** OK **GET**

	+WTIMEOUT:”<command>”,<timeout> OK	
Parameters	**<command>** “WSCAN”, “WCONN”, “WDISCONN” **<timeout>** Timeout in seconds. (0: no timeout, default: 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

6.17 +WEVENT

Response	+WEVENT:<event>
Parameters	**<event>** “SCAN_DONE” “CONNECT_SUCCESS” “CONNECT_FAIL” “DISCONNECT”
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.

7.1AT+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	**<u>SET</u>** 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 number of raw bytes to send. (Max: 2048) In normal mode, the payload length must be explicitly provided. In passthrough mode, the payload length must NOT be provided.
Description	Send data through a socket. In normal mode, the <length> must be explicitly provided. The byte sequence of <length> bytes must be directly followed by "AT+SSEND=<socket ID>,<length>\r\n". The byte sequence does not have to be followed by "\r" or "\n". In passthrough mode, the module enters the continuous transmission state when the "AT+SSEND=<socket ID>\r\n" command is used without the length argument. Once the module enters the continuous transmission state, any byte sequence fed into the UART input stream will be copied in real-time to the corresponding socket stream. There should be no additional "AT+SSEND=<socket ID>\r\n" prefix fed into the UART input stream except for the very first one for state transition, as the prefix characters themselves will be treated as actual data bytes. To leave the continuous transmission state and return to the regular state in which AT-commands can be handled again, wait for SSEND timeout duration (See AT+STIMEOUT) without feeding any bytes to the UART input stream. And feed the magic bytes "AT\r\n" to exit the continuous transmission state after +SEVENT:"SEND_IDLE" notification is received. Exiting the state will trigger the +SEVENT:"SEND_EXIT" notification, at which point the module is ready to receive and handle AT-commands again. *Note: The data payload should be transmitted to the module after receiving OK in response to the AT+SSEND command. The passthrough mode is not supported for UART without H/W flow control.

Example	**[Normal Mode]** AT+SSEND=0,6 OK Hello! **[Passthrough mode]** AT+SSEND=0 OK Hello! Nice to meet you. [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 **[Normal Mode]** AT+SSEND=0,6 OK Hello!

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

<table>
<tr><td><b>Response</b></td><td>+SEVENT:<event>,<socket ID>[,<parameter 1>,…,<parameter N&gt;]</td></tr>
<tr><td><b>Parameters</b></td><td><b>&lt;event&gt;</b>
“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>

<b>&lt;socket ID&gt;</b>
Socket ID

<b>&lt;length&gt;</b>
The total length of the payload sent over the socket.
The SEND_DROP event indicates the length of the dropped payload.

<b>&lt;error&gt;</b>
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><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><b>Description</b></td><td>Asynchronously raised socket event logs.</td></tr>
</table>

Example	+SEVENT:”CONNECT”,1 +SEVENT:”SEND_INIT”,1,1500 +SEVENT:”SEND_ERROR”,1,1000,-103

7.8+RXD

Response	**RX mode (Terse)** +RXD:<socket ID>,<actual read length>,<raw bytes> **RX mode (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 mode (Terse)** +RXD=0,15,ABCDE12345,.?=+ **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.

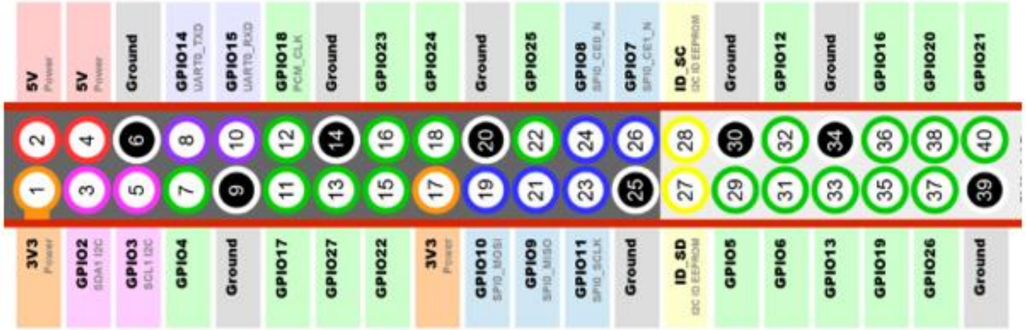
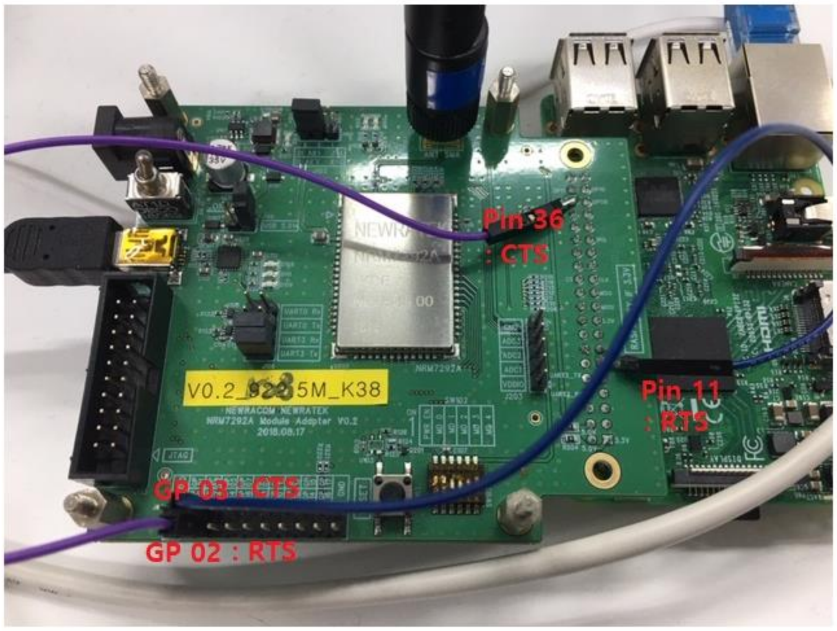


Figure 8.1 Pin map of 40-pin header for Raspberry Pi

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.



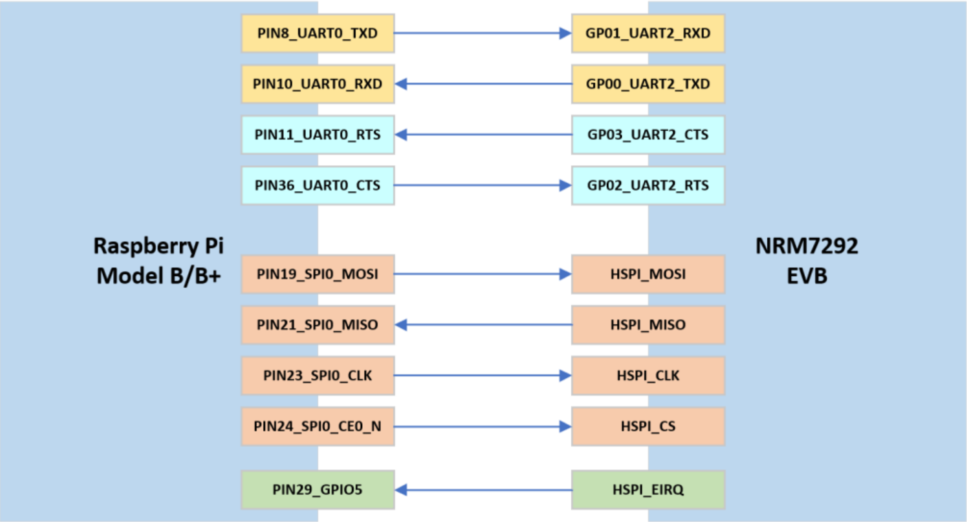
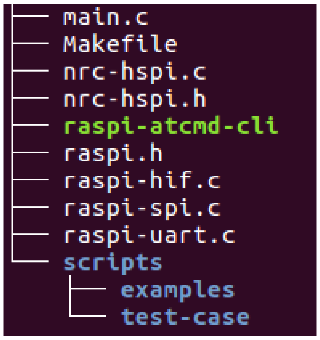


Figure 8.2 Connection between NRM7292 EVB and Raspberry Pi

1) Source files



```
├── main.c
├── Makefile
├── nrc-hspi.c
├── nrc-hspi.h
├── raspi-atcmd-cli
├── raspi.h
├── raspi-hif.c
├── raspi-spi.c
├── raspi-uart.c
└── scripts
    ├── examples
    └── test-case
```

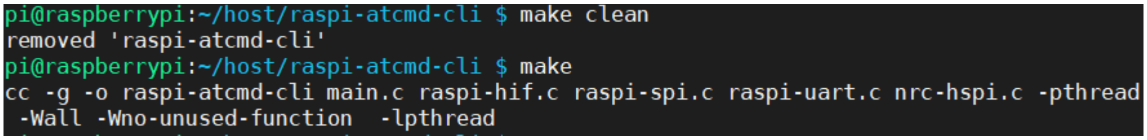
File	Description
main.c	CLI related functions.
Makefile	Make file for building.
nrc-hspi.c/h	Protocol driver for HSPI. ***Refer to this file to communicate with the ATCMD firmware via SPI from the host.**
raspi.h	Common header file for Raspberry Pi.
raspi-hif.c	Wrapper for user mode driver.
raspi-spi.c	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:~/host/raspi-atcmd-cli $ make clean
removed 'raspi-atcmd-cli'
pi@raspberrypi:~/host/raspi-atcmd-cli $ make
cc -g -o raspi-atcmd-cli main.c raspi-hif.c raspi-spi.c raspi-uart.c nrc-hspi.c -pthread
 -Wall -Wno-unused-function  -lpthread
```

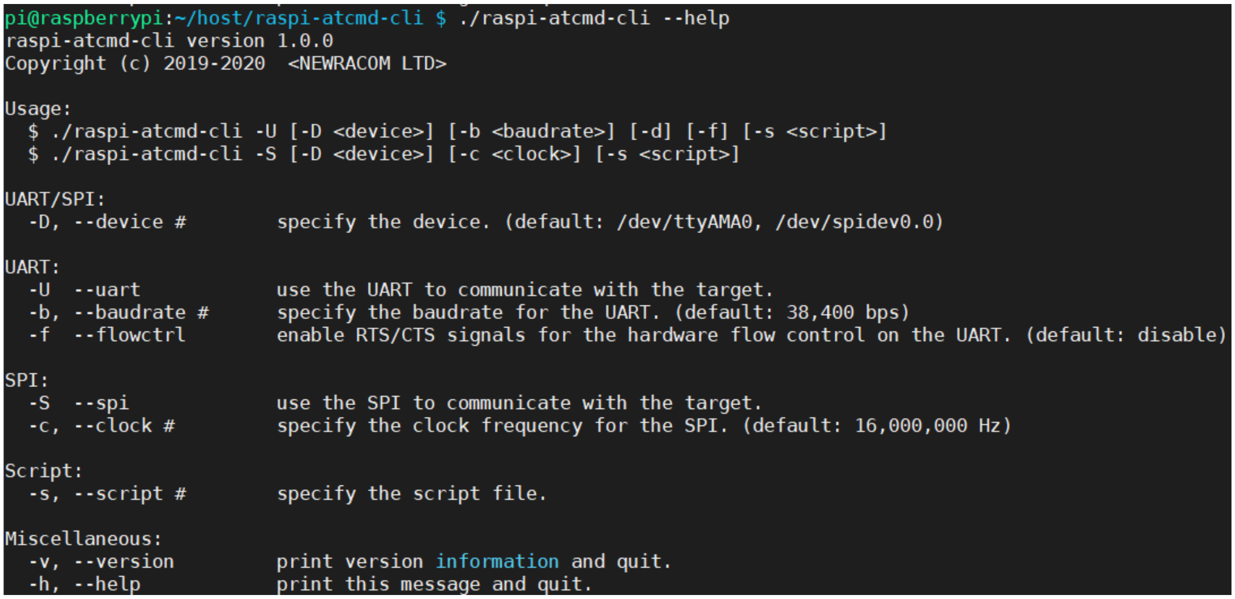
3) Run

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]
```

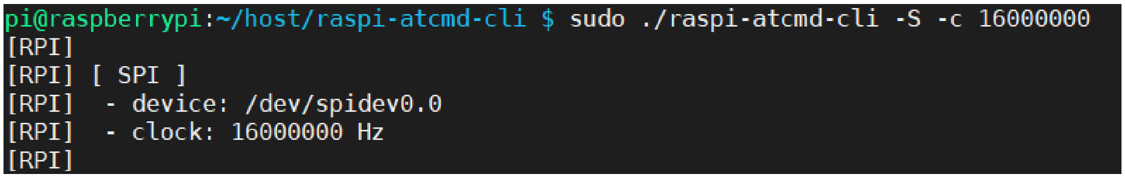


```
pi@raspberrypi:~/host/raspi-atcmd-cli $ ./raspi-atcmd-cli --help
raspi-atcmd-cli version 1.0.0
Copyright (c) 2019-2020  <NEWRACOM LTD>

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

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

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)

Script:
  -s, --script #        specify the script file.

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>]**

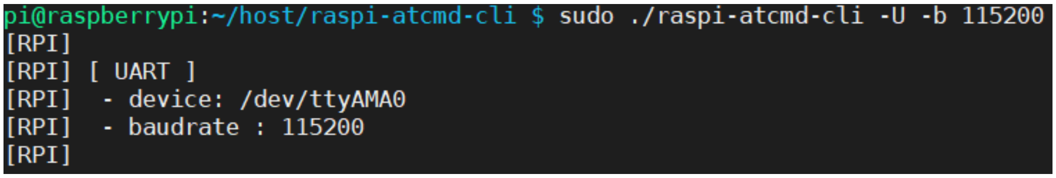


```
pi@raspberrypi:~/host/raspi-atcmd-cli $ sudo ./raspi-atcmd-cli -S -c 16000000
[RPI]
[RPI] [ SPI ]
[RPI]  - device: /dev/spidev0.0
[RPI]  - clock: 16000000 Hz
[RPI]
```

- **UART**

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

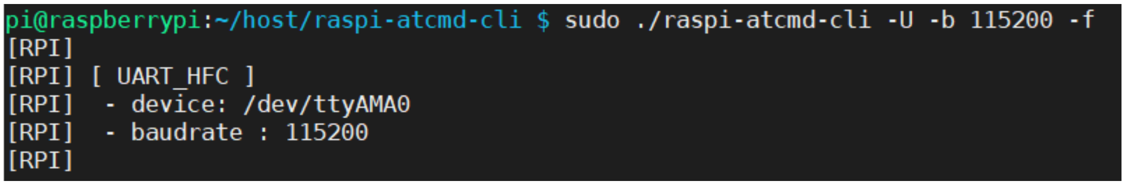
$ sudo ./raspi-atcmd-cli **-U [-b <baudrate>]**



```
pi@raspberrypi:~/host/raspi-atcmd-cli $ sudo ./raspi-atcmd-cli -U -b 115200
[RPI]
[RPI] [ UART ]
[RPI]  - device: /dev/ttyAMA0
[RPI]  - baudrate : 115200
[RPI]
```

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

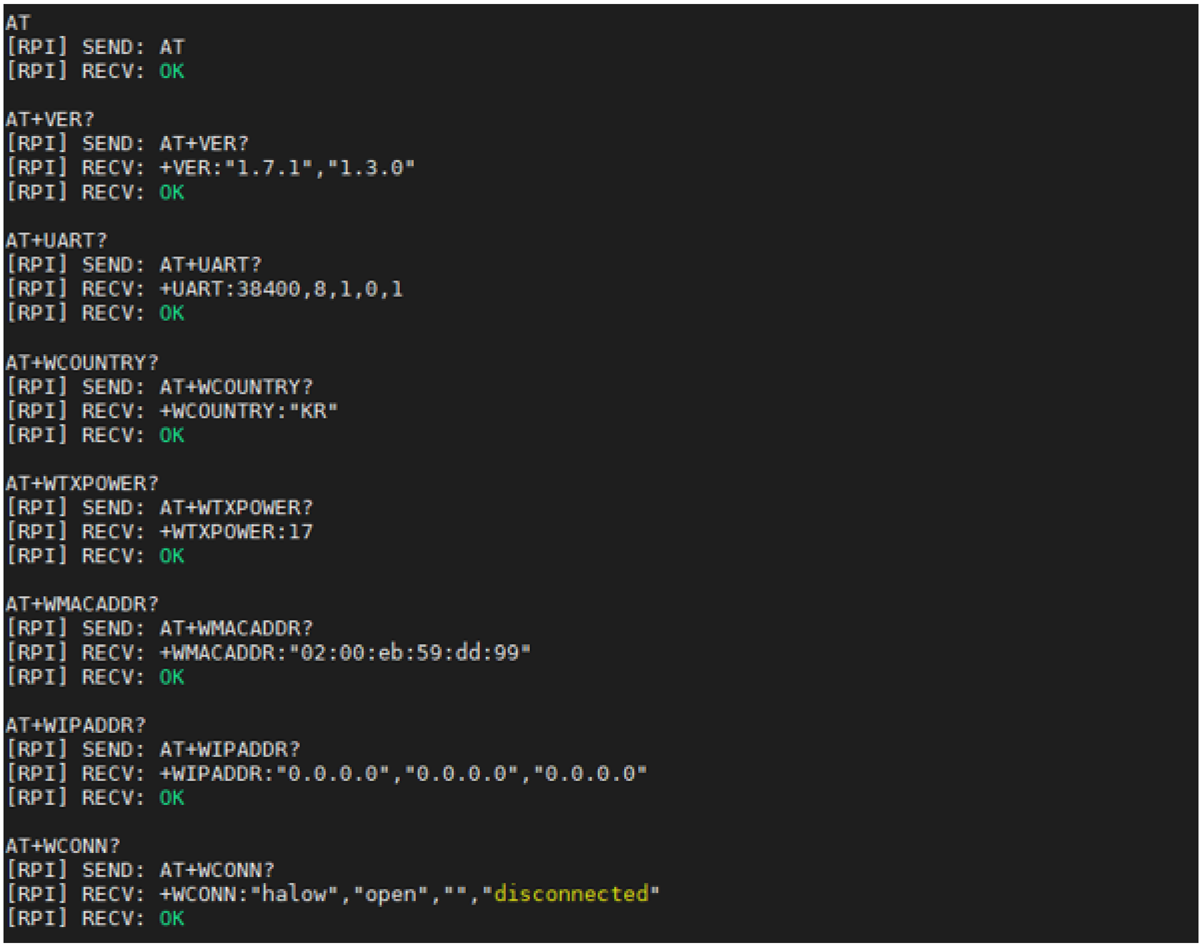
$ sudo ./raspi-atcmd-cli **-U [-b <baudrate>] -f**



```
pi@raspberrypi:~/host/raspi-atcmd-cli $ sudo ./raspi-atcmd-cli -U -b 115200 -f
[RPI]
[RPI] [ UART_HFC ]
[RPI]  - device: /dev/ttyAMA0
[RPI]  - baudrate : 115200
[RPI]
```

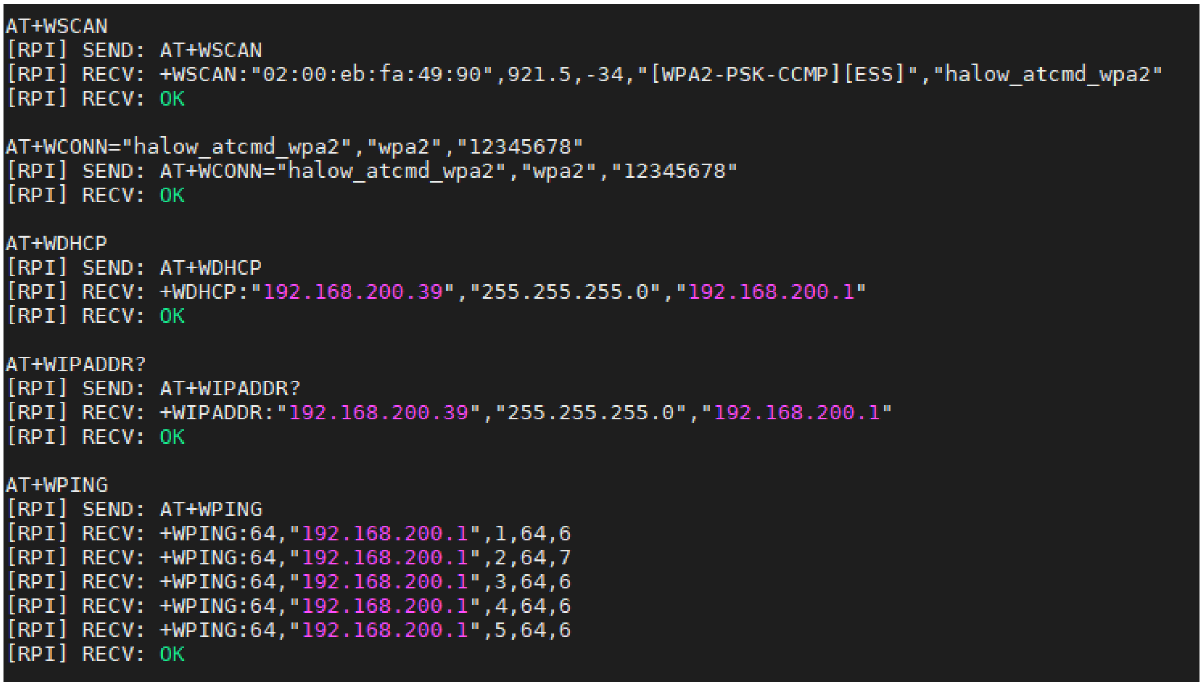
- **Log**

Getting the informations.



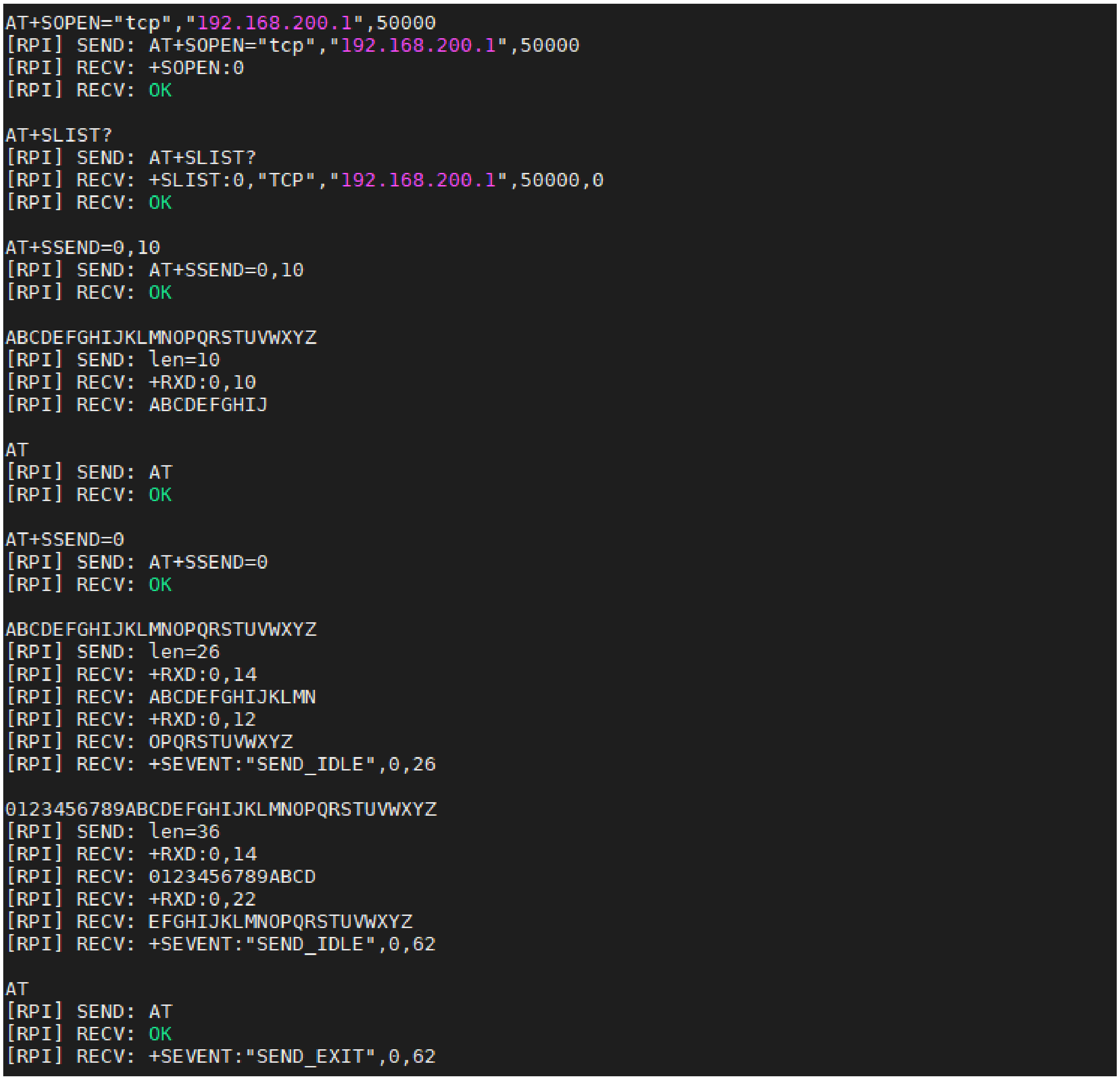
```
AT
[RPI] SEND: AT
[RPI] RECV: OK

AT+VER?
[RPI] SEND: AT+VER?
[RPI] RECV: +VER:"1.7.1","1.3.0"
[RPI] RECV: OK

AT+UART?
[RPI] SEND: AT+UART?
[RPI] RECV: +UART:38400,8,1,0,1
[RPI] RECV: OK

AT+WCOUNTRY?
[RPI] SEND: AT+WCOUNTRY?
[RPI] RECV: +WCOUNTRY:"KR"
[RPI] RECV: OK

AT+WTXPOWER?
[RPI] SEND: AT+WTXPOWER?
[RPI] RECV: +WTXPOWER:17
[RPI] RECV: OK

AT+WMACADDR?
[RPI] SEND: AT+WMACADDR?
[RPI] RECV: +WMACADDR:"02:00:eb:59:dd:99"
[RPI] RECV: OK

AT+WIPADDR?
[RPI] SEND: AT+WIPADDR?
[RPI] RECV: +WIPADDR:"0.0.0.0","0.0.0.0","0.0.0.0"
[RPI] RECV: OK

AT+WCONN?
[RPI] SEND: AT+WCONN?
[RPI] RECV: +WCONN:"halow","open","","disconnected"
[RPI] RECV: OK
```

Connecting to an AP.



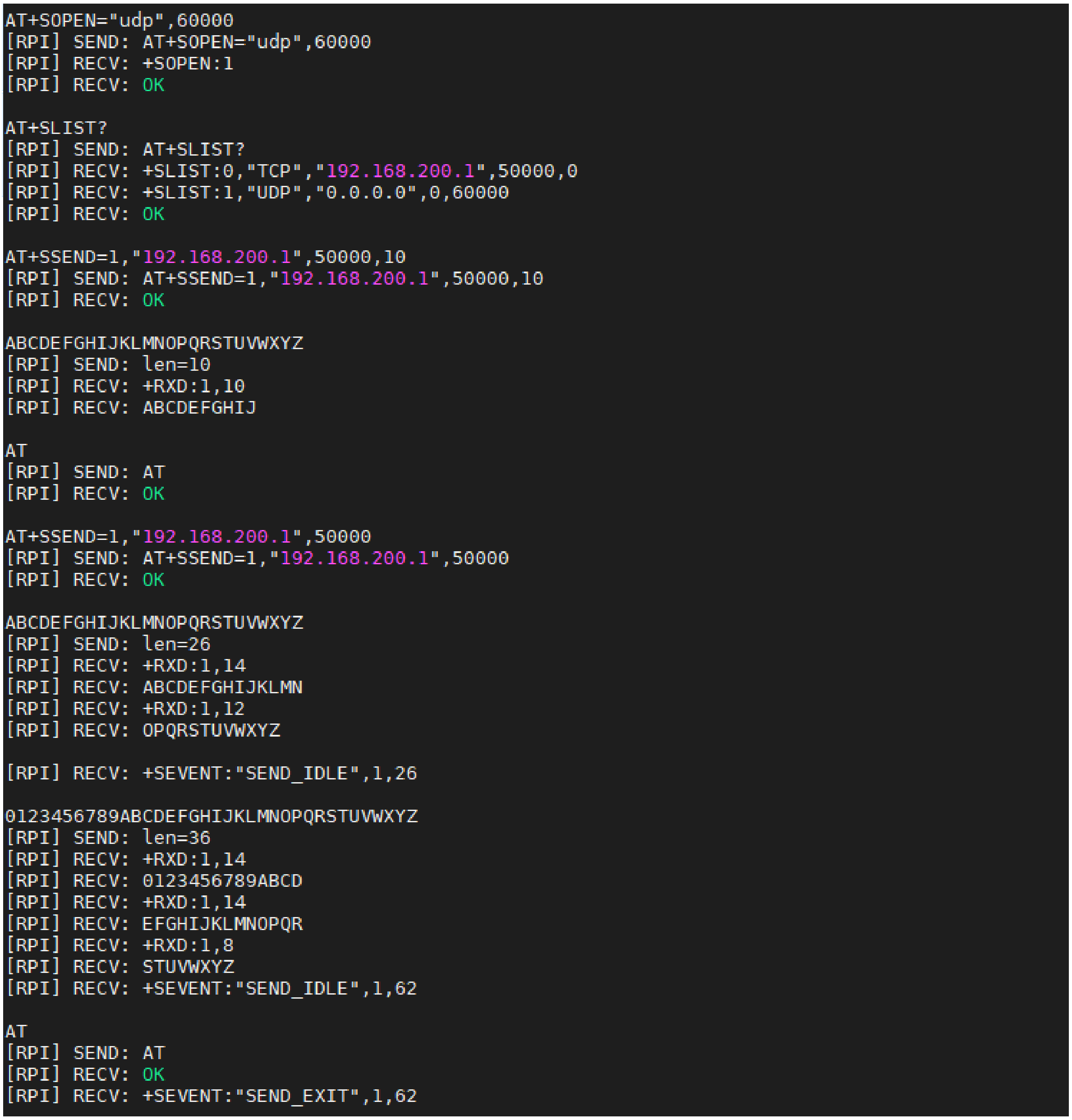
```
AT+WSCAN
[RPI] SEND: AT+WSCAN
[RPI] RECV: +WSCAN:"02:00:eb:fa:49:90",921.5,-34,"[WPA2-PSK-CCMP][ESS]","halow_atcmd_wpa2"
[RPI] RECV: OK

AT+WCONN="halow_atcmd_wpa2","wpa2","12345678"
[RPI] SEND: AT+WCONN="halow_atcmd_wpa2","wpa2","12345678"
[RPI] RECV: OK

AT+WDHCP
[RPI] SEND: AT+WDHCP
[RPI] RECV: +WDHCP:"192.168.200.39","255.255.255.0","192.168.200.1"
[RPI] RECV: OK

AT+WIPADDR?
[RPI] SEND: AT+WIPADDR?
[RPI] RECV: +WIPADDR:"192.168.200.39","255.255.255.0","192.168.200.1"
[RPI] RECV: OK

AT+WPING
[RPI] SEND: AT+WPING
[RPI] RECV: +WPING:64,"192.168.200.1",1,64,6
[RPI] RECV: +WPING:64,"192.168.200.1",2,64,7
[RPI] RECV: +WPING:64,"192.168.200.1",3,64,6
[RPI] RECV: +WPING:64,"192.168.200.1",4,64,6
[RPI] RECV: +WPING:64,"192.168.200.1",5,64,6
[RPI] RECV: OK
```

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



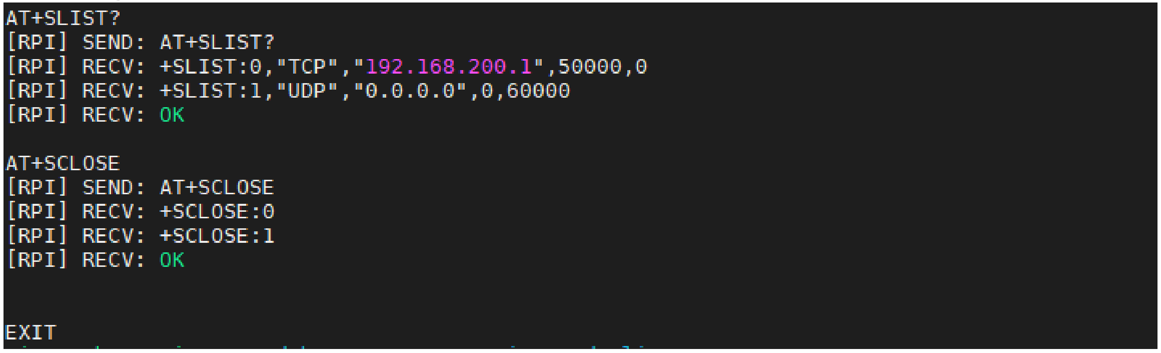
```
AT+SOPEN="tcp","192.168.200.1",50000
[RPI] SEND: AT+SOPEN="tcp","192.168.200.1",50000
[RPI] RECV: +SOPEN:0
[RPI] RECV: OK

AT+SLIST?
[RPI] SEND: AT+SLIST?
[RPI] RECV: +SLIST:0,"TCP","192.168.200.1",50000,0
[RPI] RECV: OK

AT+SSEND=0,10
[RPI] SEND: AT+SSEND=0,10
[RPI] RECV: OK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
[RPI] SEND: len=10
[RPI] RECV: +RXD:0,10
[RPI] RECV: ABCDEFGHIJ

AT
[RPI] SEND: AT
[RPI] RECV: OK

AT+SSEND=0
[RPI] SEND: AT+SSEND=0
[RPI] RECV: OK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
[RPI] SEND: len=26
[RPI] RECV: +RXD:0,14
[RPI] RECV: ABCDEFGHIJKLMN
[RPI] RECV: +RXD:0,12
[RPI] RECV: OPQRSTUVWXYZ
[RPI] RECV: +SEVENT:"SEND_IDLE",0,26

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
[RPI] SEND: len=36
[RPI] RECV: +RXD:0,14
[RPI] RECV: 0123456789ABCD
[RPI] RECV: +RXD:0,22
[RPI] RECV: EFGHIJKLMNOPQRSTUVWXYZ
[RPI] RECV: +SEVENT:"SEND_IDLE",0,62

AT
[RPI] SEND: AT
[RPI] RECV: OK
[RPI] RECV: +SEVENT:"SEND_EXIT",0,62
```

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



```
AT+SOPEN="udp",60000
[RPI] SEND: AT+SOPEN="udp",60000
[RPI] RECV: +SOPEN:1
[RPI] RECV: OK

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

AT+SSEND=1,"192.168.200.1",50000,10
[RPI] SEND: AT+SSEND=1,"192.168.200.1",50000,10
[RPI] RECV: OK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
[RPI] SEND: len=10
[RPI] RECV: +RXD:1,10
[RPI] RECV: ABCDEFGHIJ

AT
[RPI] SEND: AT
[RPI] RECV: OK

AT+SSEND=1,"192.168.200.1",50000
[RPI] SEND: AT+SSEND=1,"192.168.200.1",50000
[RPI] RECV: OK

ABCDEFGHIJKLMNOPQRSTUVWXYZ
[RPI] SEND: len=26
[RPI] RECV: +RXD:1,14
[RPI] RECV: ABCDEFGHIJKLMN
[RPI] RECV: +RXD:1,12
[RPI] RECV: OPQRSTUVWXYZ

[RPI] RECV: +SEVENT:"SEND_IDLE",1,26

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
[RPI] SEND: len=36
[RPI] RECV: +RXD:1,14
[RPI] RECV: 0123456789ABCD
[RPI] RECV: +RXD:1,14
[RPI] RECV: EFGHIJKLMNOPQR
[RPI] RECV: +RXD:1,8
[RPI] RECV: STUVWXYZ
[RPI] RECV: +SEVENT:"SEND_IDLE",1,62

AT
[RPI] SEND: AT
[RPI] RECV: OK
[RPI] RECV: +SEVENT:"SEND_EXIT",1,62
```

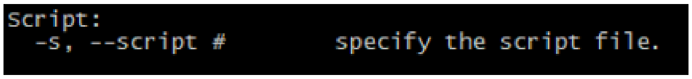
Closing all sockets.



```
AT+SLIST?
[RPI] SEND: AT+SLIST?
[RPI] RECV: +SLIST:0,"TCP","192.168.200.1",50000,0
[RPI] RECV: +SLIST:1,"UDP","0.0.0.0",0,60000
[RPI] RECV: OK

AT+SCLOSE
[RPI] SEND: AT+SCLOSE
[RPI] RECV: +SCLOSE:0
[RPI] RECV: +SCLOSE:1
[RPI] RECV: OK


EXIT
```

4) Run with script file

CLI application provides the option to run the script files.

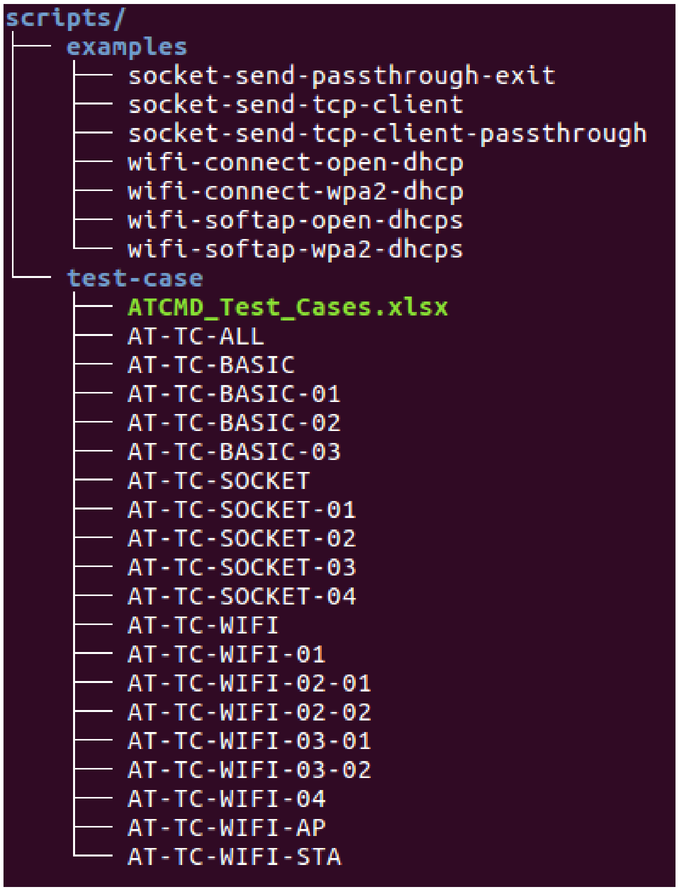


```
Script:
  -s, --script #       specify the script file.
```

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.**



```
scripts/
├── examples
│   ├── socket-send-passthrough-exit
│   ├── socket-send-tcp-client
│   ├── socket-send-tcp-client-passthrough
│   ├── wifi-connect-open-dhcp
│   ├── wifi-connect-wpa2-dhcp
│   ├── wifi-softap-open-dhcps
│   └── wifi-softap-wpa2-dhcps
└── test-case
    ├── ATCMD_Test_Cases.xlsx
    ├── AT-TC-ALL
    ├── AT-TC-BASIC
    ├── AT-TC-BASIC-01
    ├── AT-TC-BASIC-02
    ├── AT-TC-BASIC-03
    ├── AT-TC-SOCKET
    ├── AT-TC-SOCKET-01
    ├── AT-TC-SOCKET-02
    ├── AT-TC-SOCKET-03
    ├── AT-TC-SOCKET-04
    ├── AT-TC-WIFI
    ├── AT-TC-WIFI-01
    ├── AT-TC-WIFI-02-01
    ├── AT-TC-WIFI-02-02
    ├── AT-TC-WIFI-03-01
    ├── AT-TC-WIFI-03-02
    ├── AT-TC-WIFI-04
    ├── AT-TC-WIFI-AP
    └── AT-TC-WIFI-STA
```

- **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**

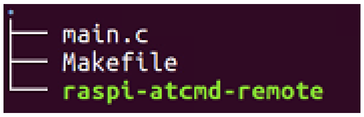
  $ sudo ./raspi-atcmd-cli -U [-b <baudrate>] -f **-s <script_file>**

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

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



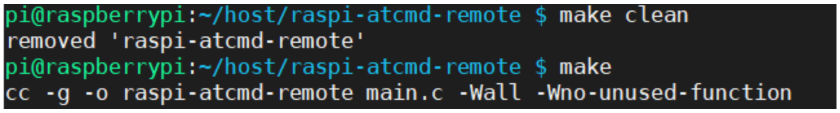
```
.
├── main.c
├── Makefile
└── raspi-atcmd-remote
```

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]

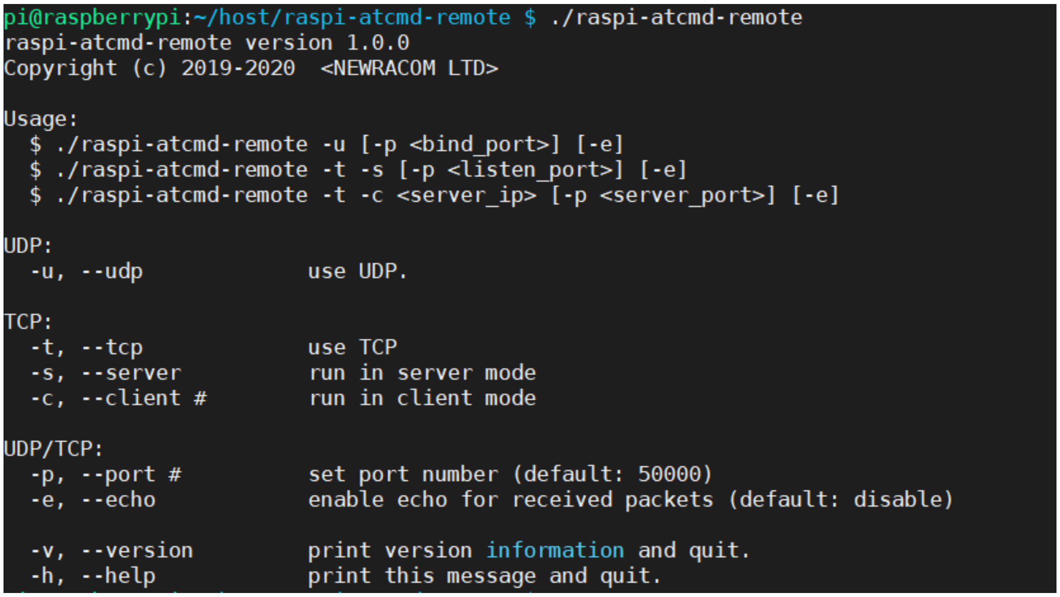


```
pi@raspberrypi:~/host/raspi-atcmd-remote $ make clean
removed 'raspi-atcmd-remote'
pi@raspberrypi:~/host/raspi-atcmd-remote $ make
cc -g -o raspi-atcmd-remote main.c -Wall -Wno-unused-function
```

3) Run

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

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



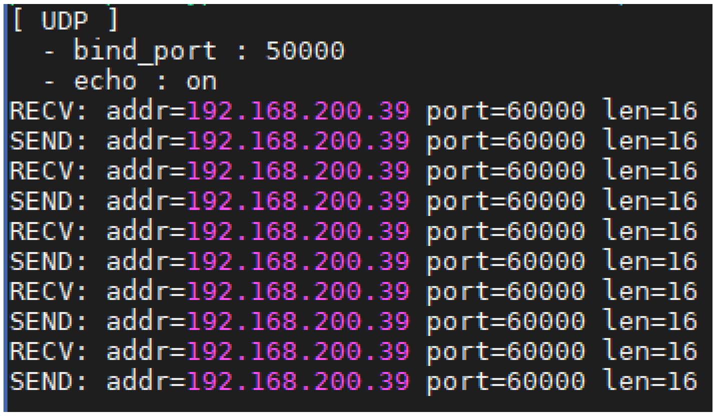
```
pi@raspberrypi:~/host/raspi-atcmd-remote $ ./raspi-atcmd-remote
raspi-atcmd-remote version 1.0.0
Copyright (c) 2019-2020  <NEWRACOM LTD>

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]

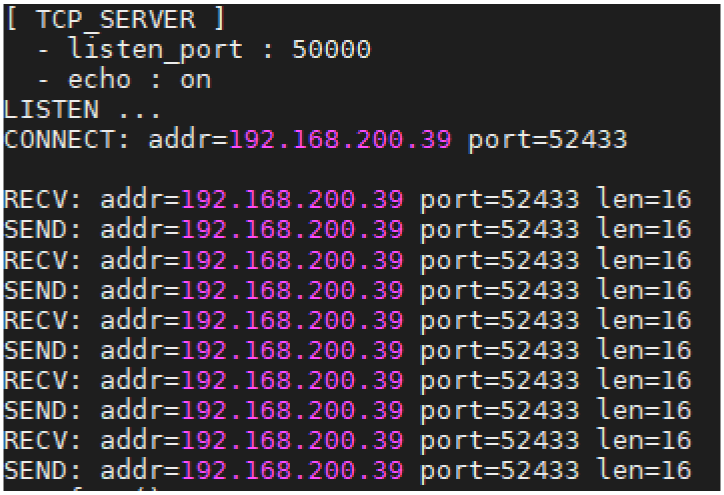
4) Log

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



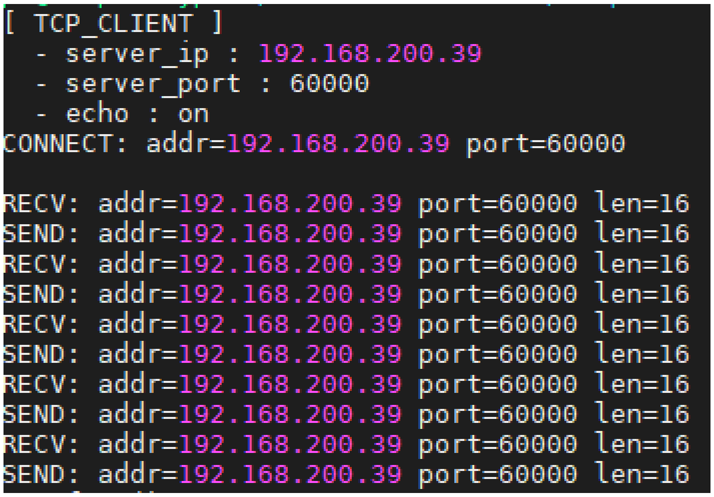
```
[ UDP ]
  - bind_port : 50000
  - echo : on
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000 len=16
RECV: addr=192.168.200.39 port=60000 len=16
SEND: addr=192.168.200.39 port=60000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

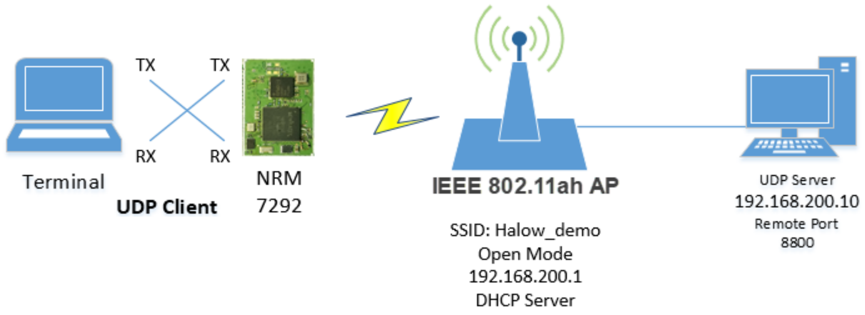


Figure 9.1 Configuration of Example1

[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)

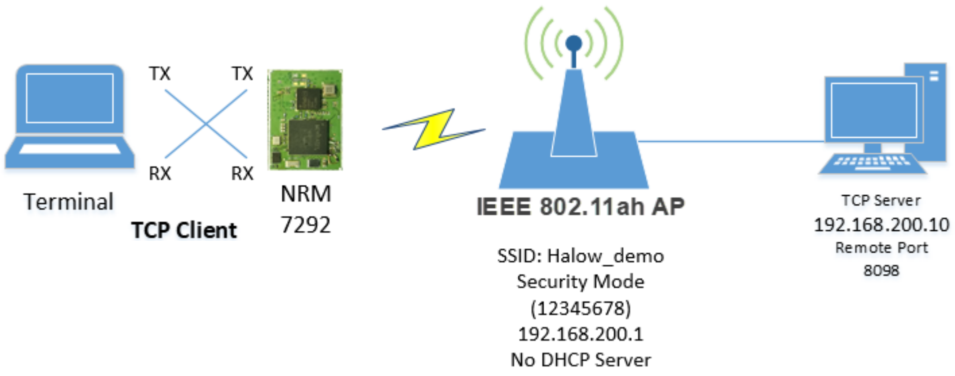


Figure 9.2 Configuration of Example2

[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