NEW RADIO COMMUNICATION FOR NEW ERA

NRC7394 Evaluation Kit User Guide

(AT Command)

Ultra-low power & Long-range Wi-Fi

Ver 1.3.1
Dec. 10, 2024

NEWRACOM, Inc.

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

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

Contents

List of Tables

List of Figures

1 Overview

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

2 Basic Setup

2.1 Hardware

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

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

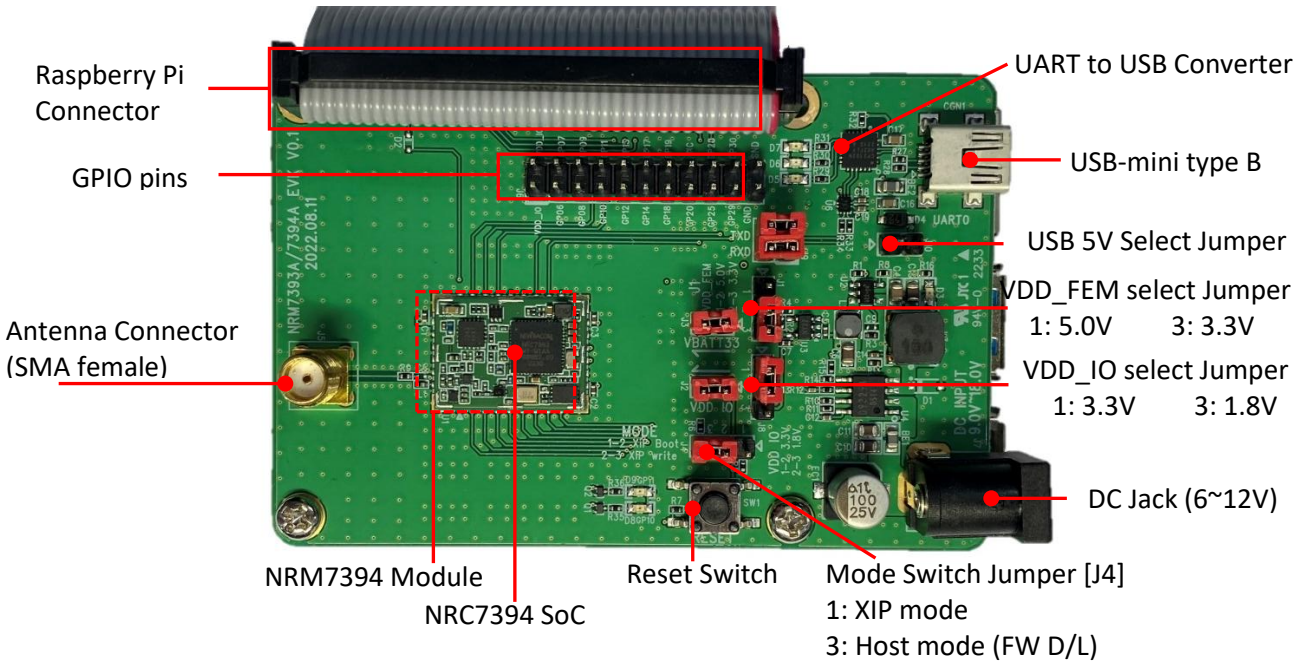


Figure 2.1 NRC7394 Evaluation Board

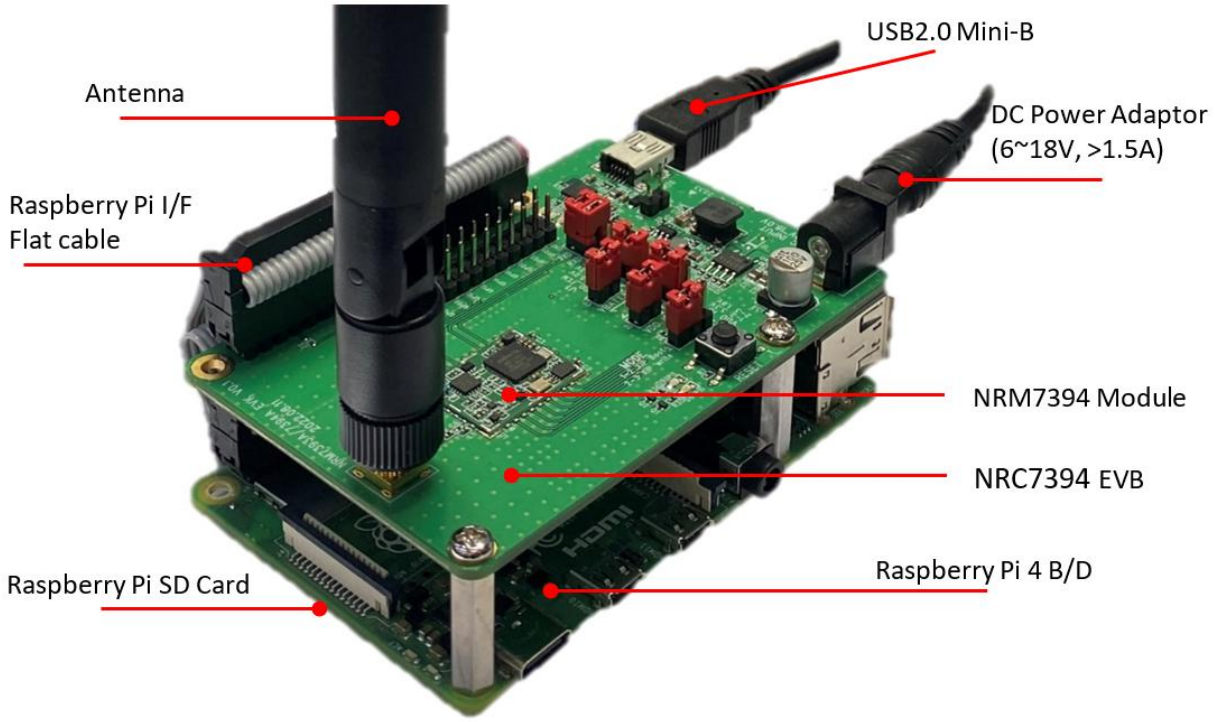


Figure 2.2 NRC7394 Evaluation Kit with Raspberry Pi 4 model B

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

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

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

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

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

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

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

NOTE:

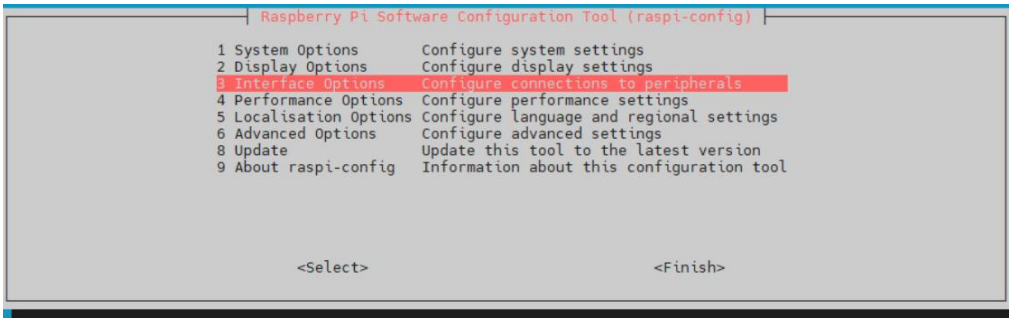
If the host is connected with a 20-pin header, detach the Raspberry Pi board from the EVB first before proceeding. The EVB must be used as a standalone for stable AT communication.

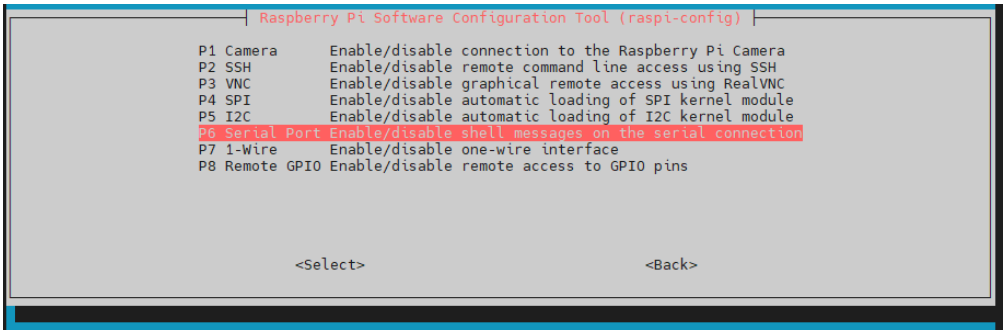
2.1.1 UART

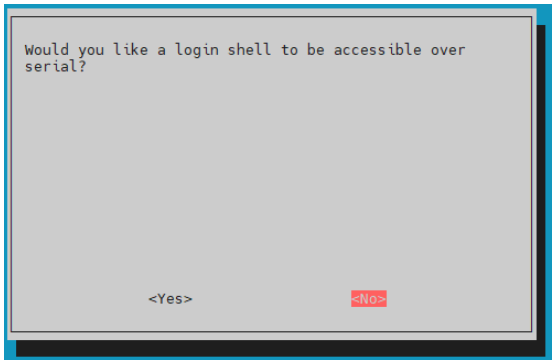
The NRC7394 AT command firmware uses UART channel 1. RTS/CTS is optional and is required to use baudrate greater than 115,200 bps.

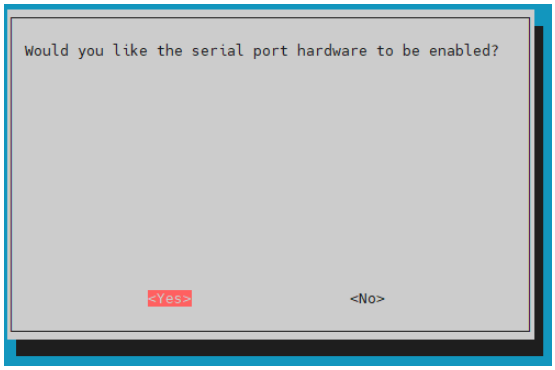
To perform AT command communication through UART on Raspberry Pi, Serial Port must be enabled in the Raspberry Pi configuration tool.

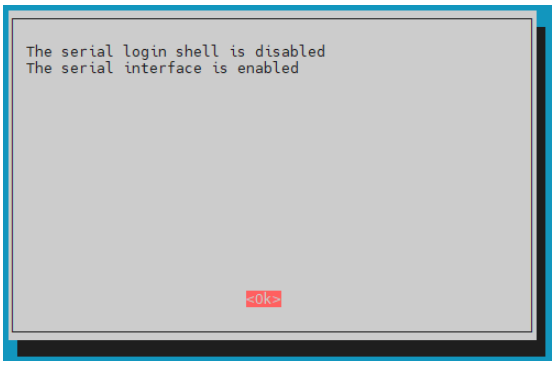
```
# sudo raspi-config
```











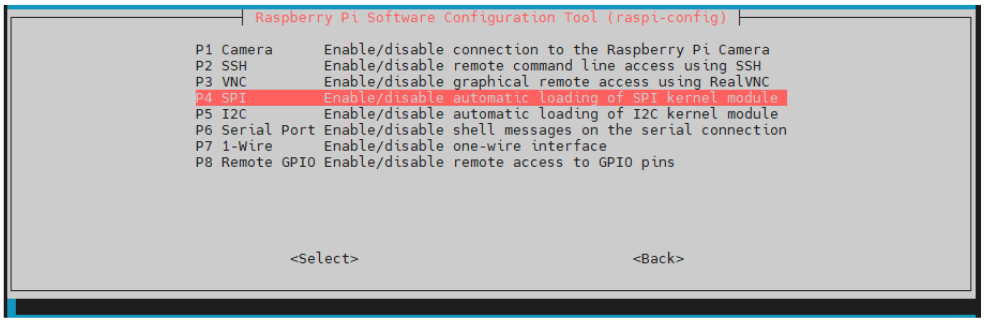
2.1.2 HSPI

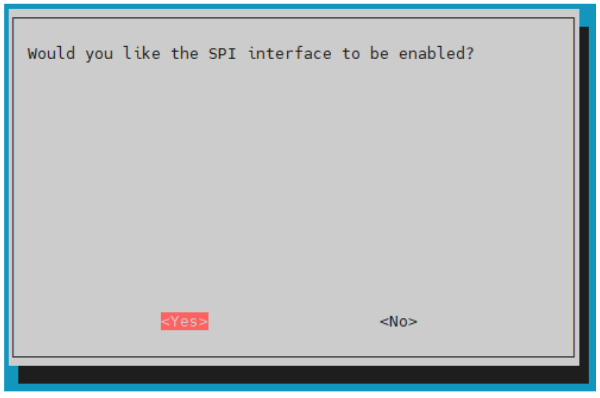
The NRC7394 has a dedicated SPI slave controller for high speed. HSPI_EIRQ is optional.

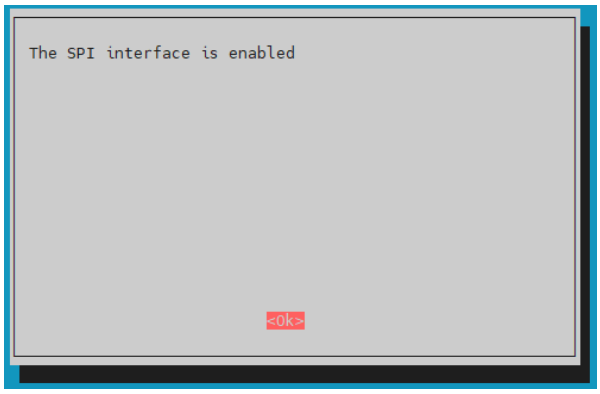
To perform AT command communication through SPI on Raspberry Pi, spidev (User mode SPI device driver) must be enabled.

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

sudo raspi-config







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

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

# Uncomment this to enable infrared communication.
#dtoverlay=gpio-ir,gpio_pin=17
#dtoverlay=gpio-ir-tx,gpio_pin=18

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

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

[pi4]
# Enable DRM VC4 V3D driver on top of the dispmanx display stack
dtoverlay=vc4-fkms-v3d
max_framebuffers=2

[all]
#dtoverlay=vc4-fkms-v3d
enable_uart=1

dtoverlay=disable-bt
dtoverlay=disable-wifi
#dtoverlay=newracom
```

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

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

2.2 Software

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

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

3 AT Command Type

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

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

Table 3.1 AT-command type

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

4 Return for Commands

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

5 Basic AT Commands

Commands	Description
AT	Check the AT serial interface status.
ATE	Enable or disable echo.
ATZ	Reset the hardware and restart the firmware.
AT+VER	Fetch the AT firmware version and software package version.
AT+BOOT	Fetch the cause of the most recent system boot.
AT+XTAL	Get the status of the crystal.
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+FWUPDATE	Set the information required for firmware update.
AT+FWBINDL	Download the firmware binary data to RAM and write it to FLASH.
AT+SFUSER	Read, write and erase the user data area of Flash memory.
AT+SFSYSUSER	Read the user factory area of Flash memory.
+BEVENT	Asynchronously raised event messages.

5.1 AT

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

5.2 ATE

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

5.3 ATZ

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

5.4 AT+VER

Command	**GET** AT+VER?
Response	**GET** +VER: <SDK>,<ATCMD>

	OK
Parameters	**<SDK>** SDK version **<ATCMD>** AT Command Set version
Description	Fetch the version information of current firmware.
Example	AT+VER? +VER:"1.0.0","1.23.5" OK

5.5 AT+BOOT

Command	**GET** AT+BOOT?
Response	**GET** +BOOT: <reason> OK
Parameters	**<reason>** The cause of the system boot. 1. "POR" : Power On Reset 2. "WDT" : Watchdog Timer 3. "PMC" : Power Management Controller 4. "HSPI" : HSPI controller
Description	Fetch the cause of the most recent system boot. Boot Cause: POR — This indicates a Power-On Reset (POR), which can occur due to one of the following: ● Powering on the system. ● Triggering the reset pin manually. ● A software command that writes to the reset register. Boot Cause: WDT — The system was reset due to a Watchdog Timer (WDT) event. This typically happens when the system fails to respond or hang for a prolonged period, causing the watchdog timer to reset the system to prevent it from freezing.

	PMC	This indicates a reset caused by the Power Management Controller (PMC), which is responsible for managing power-related functions in the system.
	HSPI	This indicates a reset caused by the HSPI controller. A host application can request a firmware reset by writting a software reset register in the HSPI controller.
	NOTE: After booting, the AT Command firmware writes the message "+BOOT:<reason>" to the host interface (UART or HSPI).	
Example	+BOOT:"POR" : : AT+BOOT? +BOOT:"POR" OK	

5.6 AT+XTAL

Command	**GET** AT+XTAL?
Response	**GET** +XTAL: <status> OK
Parameters	**<status>** 0 : Crystal status not checked. 1 : Crystal is working. 2 : Crystal is not working.
Description	Get the status of the crystal.
Example	AT+XTAL? +XTAL:1 OK

5.7 AT+UART

Command	**SET** AT+UART=<baud_rate>,<HFC> **GET** AT+UART?
Response	**SET** OK **GET** +UART:<baud_rate>,<data_bits>,<stop_bits>,<parity>,<HFC> OK
Parameters	**<baud rate>** 9600, 19200, 38400, 57600, 115200*, 230400, 460800, 500000, 576000, 921600, 1000000, 1152000, 1500000, 2000000 **<data bits>** Always 8 (8-bit)* **<stop bits>** Always 1 (1-bit)* **<parity>** Always 0 (None)* **<HFC>** 0 : disable RTS/CTS* 1 : enable RTS/CTS
Description	Configure the baud rate and HFC for the UART. NOTE : For higher baud rates, it is recommended to enable hardware flow control. When hardware flow control is disabled, the AT+SSEND command can only set synchronous send mode.
Example	AT+UART=115200,1 OK

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

5.8 AT+GPIOCONF

Command	**SET** AT+GPIOCONF=<number>,<direction>,<pull-up> **GET** AT+GPIOCONF? AT+GPIOCONF?=<number>
Response	**SET** OK **GET** +GPIOCONF=<number>,<direction>,<pull-up> : OK
Parameters	**<number>** GPIO pin number Host Interface Type: HSPI — Available GPIO numbers: 10, 11, 12, 13, 14, 20, 25 Host Interface Type: UART — Available GPIO numbers: 6, 7, 10, 11, 25, 28, 29, 30 **<direction>** 0 : input 1 : output **<pull-up> (input pin only)** 0 : pull-down 1 : pull-up
Description	Configure the GPIO pin direction and pull-up option.
Example	AT+GPIOCONF=10,1,1 OK AT+GPIOCONF=11,0,0

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

5.9 AT+GPIOVAL

Command	**SET** AT+GPIOVAL=<number>,<level> **GET** AT+GPIOVAL? AT+GPIOVAL?=<number>
Response	**SET** OK **GET** +GPIOVAL:<number>,<level> OK
Parameters	**<number>** GPIO pin number **<level>** 0 : low 1 : high
Description	Read or write the output GPIO pin level.
Example	AT+GPIOVAL?

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

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

5.10 AT+ADC

Command	**SET** AT+ADC=<controller> **GET** AT+ADC? AT+ADC?=<channel>
Response	**GET** +ADC:<channel>,<value> : OK
Parameters	**<controller>** 0 : disable 1 : enable **<channel>** 0, 1 **<value>** 0 ~ 1023 (10-bits)
Description	Fetch the ADC value at the selected ADC channel.
Example	AT+ADC=1 OK AT+ADC? +ADC:0,396

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

5.11 AT+FWUPDATE

Command	**RUN** AT+FWUPDATE **SET** AT+FWUPDATE=<length>,<crc32>[,<verify>] **GET** AT+FWUPDATE?
Response	**RUN** OK **SET** OK **GET** +FWUPDATE:<length>,<crc32>,<verify> OK
Parameters	**<length>** Total length of firmware binary data. **<crc32>** A 32-bit hexadecimal value, prefixed with '0x' and calculated using the CRC-32 algorithm to detect data corruption. To determine the CRC value of the 'newFW.bin' file, you can use the 'crc.py' script located in the 'package\standalone\atcmd\host\python-http-server\python'

	directory. Simply run the command 'python crc.py newFW.bin' and add the '0x' prefix to the result. (ex) python crc.py newFW.bin 97cb8611 **<verify>** Enable or disable to check for data error. (*0 : disable, 1 : enable) If the "<verify>" option is set to 1, the written binary data is read to check for data error. And if a data error occurs, the binary data is written back to the erased flash memory.
Description	Set the information required for firmware update. The SET command sets the data length and CRC value before downloading the firmware binary data with the AT+FWBINDL command. The AT+FWUPDATE=0 command resets previous settings to 0. The RUN command is required after completing the download with the AT+FWBINDL command and before resetting the system. A system reset can be performed with the ATZ command. Replacing the old firmware with a new one is performed by the bootloader after a system reset.
Example	AT+FWUPDATE=0 OK AT+FWUPDATE=915320,0xDAE06D27 OK AT+FWUPDATE? +FWUPDATE: 915320,0xDAE06D27 OK !!! Download the firmware binary data with the AT+FWBINDL SET command !!! AT+FWUPDATE OK ATZ

5.12 AT+FWBINDL

Command	**SET** AT+FWBINDL=<offset>,<length> **GET** AT+FWBINDL?
Response	**SET** OK **GET** +FWBINDL:<total_length>,<done_length> OK
Parameters	**<offset>** Zero-based offset of the data to download. **<length>** Length of data to download. **<total_length>** Total length of firmware binary data. **<done_length>** The data length written to flash memory after downloading.
Description	Download the firmware binary data to RAM and write it to FLASH. Firmware binary data can be downloaded with multiple SET commands. After receiving the OK message for the SET command, data can be downloaded up to 4KB at a time. If no data is downloaded for 1 second, the FWBINDL_IDLE event is raised. At this time, the download can be canceled with the "AT\r\n" command without downloading the remaining data. +BEVENT:”FWBINDL_IDLE”,<offset>,<length>,<count> When a download is cancelled, the FWBINDL_DROP event is raised. However, the data downloaded with the previous SET command remains, so canceled data can be downloaded again. +BEVENT:”FWBINDL_DROP”, <offset>,<length>

	If the downloaded data cannot be written to FLASH or if the data written to FLASH does not match the downloaded data, the FWBINDL_FAIL event is raised. +BEVENT:”FWBINDL_FAIL”, <offset>,<length> If data is downloaded without cancellation, the FWBINDL_DONE event is raised. After the FWBINDL_DONE event, the next data can continue to be downloaded with the SET command. +BEVENT:”FWBINDL_DONE”, <offset>,<length>
Example	AT+FWUPDATE=915320,0xDAE06D27 OK AT+FWBINDL? +FWBINDL:915320,0 OK AT+FWBINDL=0,4096 OK < data > +BEVENT:"FWBINDL_DONE",0,4096 AT+FWBINDL=4096,4096 OK < data > +BEVENT:"FWBINDL_DONE",4096,4096 AT+FWBINDL=8192,4096 OK < data > +BEVENT:"FWBINDL_DONE",8192,4096 : : AT+FWBINDL=909312,4096 OK < data > +BEVENT:"FWBINDL_DONE",909312,4096 AT+FWBINDL=913408,1912 OK < data > +BEVENT:"FWBINDL_DONE",913408,1912

	AT+FWBINDL? +FWBINDL:915320,915320 OK

5.13 AT+SFUSER

Command	**SET** AT+SFUSER=<mode>[,<offset>,<length>] **GET** AT+SFUSER?
Response	**SET** OK **GET** +SFUSER:<address>,<size> OK
Parameters	**<mode>** 0 : Read 1 : Write 2 : Erase **<offset>** Offset from the start address of the user data area. **<length>** Amount of data in bytes to read, write, or erase. **<address>** Start address of the user data area. **<size>** Total size of the user data area in kilobytes.
Description	Read, write and erase the user data area of Flash memory. 1. Read ● Read data from the specified Flash memory offset. ● The read data is appended after the +RXD_SFUSER message.

<table>
<tr><td></td><td>+RXD_SFUSER:<offset>,<length>\r\n<data>

2. Write
● Write data to the specified Flash memory offset.
● During the write operation, several events may occur. These events indicate the status of the write process:
i. +BEVENT:"SFUSER_IDLE",<offset>,<length>,<count>
1. No data is received from the host interface (UART or SPI) for more than 1 second.
2. <count> indicates the amount of data received before the event.
ii. +BEVENT:"SFUSER_DROP", <offset>,<length>
1. An operation is canceled with the "AT\r\n" command after a "SFUSER_IDLE" event.
iii. +BEVENT:"SFUSER_FAIL", <offset>,<length>
1. Flash write operation failed.
iv. +BEVENT:f"SFUSER_DONE", <offset>,<length>
1. Write operation completed successfully.

3. Erase
● Erase the specified section of Flash memory.
● If no offset or length is specified, the entire user data area is erased.

NOTE:
The user data area may or may not be supported depending on the Flash memory map profile.
The total size of the available user data area can be checked with the GET command.
The total size of the user data area is 100KB or 8KB.</td></tr>
<tr><td>Example</td><td>AT+SFUSER?
+SFUSER:0x1E6000,100
OK

AT+SFUSER=0,0,128
OK
+RXD_SFUSER:0,128
<data>

AT+SFUSER=1,0,128
OK</td></tr>
</table>

	<data> +BEVENT:"SFUSER_DONE",0,128 AT+SFUSER=0,0,128 OK +RXD_SFUSER:0,128 <data> AT+SFUSER=2,0,128 OK AT+SFUSER=2 OK

5.14 AT+SFSYSUSER

Command	**SET** AT+SFSYSUSER=<offset>[,<length>] **GET** AT+SFSYSUSER?
Response	**SET** OK **GET** +SFSYSUSER:<address>,<size> OK
Parameters	**<offset>** Offset from the start address of the user factory area **<length>** Amount of data in bytes to read, write, or erase. **<address>** Start address of the user factory area. **<size>** Total size of the user factory area in bytes.
Description	Read the user factory data in the 4KB SYSCONFIG area of Flash memory.

	The read data is appended after the +RXD_SFSYSUSER message. +RXD_SFSYSUSER:<offset>,<length>\r\n<data> NOTE: The total size of the available user factory area can be checked with the GET command.
Example	AT+SFSYSUSER? +SFSYSUSER:0x3FC100,512 OK AT+SFSYSUSER=0 OK +RXD_SFSYSUSER:0,512 <data> AT+SFSYSUSER=128 OK +RXD_SFSYSUSER:128,384 <data> AT+SFSYSUSER=256,128 OK +RXD_SFSYSUSER:256,128 <data>

5.15 +BEVENT

Response	+BEVENT:<event>[,<parameter 1>,…,<parameter N>]
Parameters	**<event>** ”FWBINDL_IDLE”,<offset>,<length>,<count> ”FWBINDL_DROP”, <offset>,<length> ”FWBINDL_FAIL”, <offset>,<length> ”FWBINDL_DONE”, <offset>,<length> “SFUSER_IDLE”,<offset>,<length>,<count> “SFUSER_DROP”, <offset>,<length> “SFUSER_FAIL”, <offset>,<length> “SFUSER_DONE”, <offset>,<length>
Description	Asynchronously raised event messages.

Example	+BEVENT:"FWBINDL_IDLE",102400,4096,1024 +BEVENT:"FWBINDL_DROP",102400,4096 +BEVENT:"FWBINDL_FAIL",102400,4096 +BEVENT:"FWBINDL_DONE",909312,4096 +BEVENT:"SFUSER_IDLE",128,1024,512 +BEVENT:"SFUSER_DROP",128,1024 +BEVENT:"SFUSER_FAIL",128,1024 +BEVENT:"SFUSER_DONE",128,1024

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 TX power level.
AT+WRXSIG	Fetch or monitor the RSSI (dBm) and SNR (dB) values.
AT+WRATECTRL	Toggle the MCS rate control option.
AT+WMCS	Set the MCS index.
AT+WDUTYCYCLE	Configure duty cycle operation.
AT+WCCATHRESHOLD	Set CCA threshold.
AT+WTXTIME	Set carrier sense time and pause time.
AT+WTSF	Read the elapsed TSF timer duration.
AT+WBI	Get the beacon interval of the connected AP in STA mode.
AT+WLI	Set the listen interval in STA mode.
AT+WSCAN	Perform Wi-Fi scanning.
AT+WSCANSSID	Perform Wi-Fi scanning with probe request frames that specify full SSID.
AT+WBGSCAN	Perform periodic background scans based on signal strength.
AT+WSAEPWE	Set the SAE PWE derivation method
AT+WCONN	Connect to a new AP.
AT+WDISCONN	Disconnect from the AP or abort an on-going connection process.
AT+WSOFTAP	Run as the AP mode.
AT+WSOFTAPSSID	Set how to specify the SSID in the beacon frame.
AT+WBSSMAXIDLE	Configure the BSS Max idle service for SoftAP.
AT+WSTAINFO	Get information of associated STAs on AP mode.
AT+WMAXSTA	Set the maximum number of STAs allowed in AP mode.
AT+WIPADDR	Configure the IPv4 address.

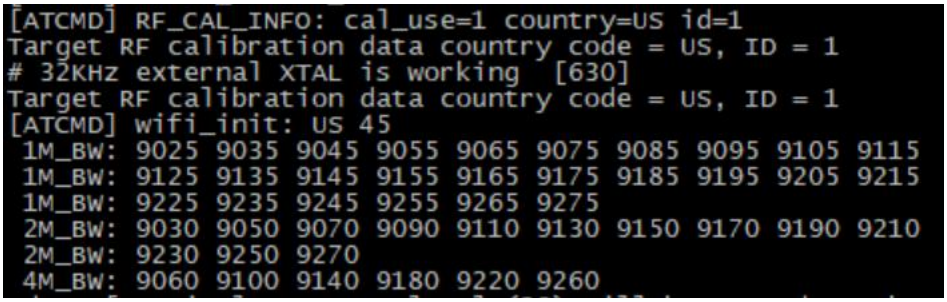
AT+WDNS	Configure the IP address for the DNS server.
AT+WDHCP	Request dynamic IP allocation from the DHCP server.
AT+WDHCPS	Run the DHCP sever in SoftAP mode.
AT+WPING	Send ICMP ECHO_REQUEST to network hosts with IPv4 address.
AT+WDEEPSLEEP	Configure deep-sleep mode to save power.
AT+WFOTA	Enable or disable Firmware Over-the-Air (FOTA).
AT+WCTX	Send dummy data frames for continuous TX without connecting to AP.
AT+WSTX	Start or stop sine wave TX.
AT+WRELAY	Run as the RELAY mode.
AT+WWPS	Enable WPS PBC mode and start WPS negotiation.
AT+WTIMEOUT	Configure the response timeout for the specified command.
+WEVENT	Asynchronously raised Wi-Fi event messages.

6.1 AT+WMACADDR

Command	**GET** AT+WMACADDR?
Response	**GET** +WMACADDR:”<mac_address_0>”, ”<mac_address_1>” OK
Parameters	**<mac_address_0> , <mac_address_1>** The MAC address ‘HH:HH:HH:HH:HH:HH’ where H is a hexadecimal character. mac_address_0 indicates the STA MAC address in STA mode, and the AP MAC address in AP and RELAY modes. mac_address_1 indicates the STA MAC address in RELAY mode.
Description	Read the MAC address
Example	AT+ WMACADDR? +WMACADDR:”2F:33:4F:65:11:20”, ”2F:33:4F:65:11:21” OK

6.2 AT+WCOUNTRY

Command	**SET** AT+WCOUNTRY=”<country_code>” **GET** AT+WCOUNTRY?
Response	**SET** OK **GET** +WCOUNTRY=”<country_code>” OK
Parameters	**<country_code>** - AU : Australia - CN : China - EU : Europe - JP : Japan - NZ : New Zealand - US : United States - K1 : Korea USN1

	- K2 : Korea USN5 - S8 : Singapore 860MHz band - S9 : Singapore 920MHz band - T8 : Taiwan 840MHz band - T9 : Taiwan 920MHz band
Description	Configure the Wi-Fi country code. Supported country codes can be retrieved with the “AT+WCOUNTRY=?” command. NOTE: If the nrc7394 module has RF calibration data, the country code is set during boot as shown in the firmware log below.  If not, the country code may need to be set with the SET command after boot.
Example	AT+WCOUNTRY=? +AT+WCOUNTRY="{US|JP|K1|T8|EU|CN|NZ|AU|K2|S8|S9|T9}" OK AT+ WCOUNTRY =”US” OK AT+WCOUNTRY? +WCOUNTRY:”US” OK

6.3 AT+WTXPOWER

Command	**SET** AT+WTXPOWER=<power>[,”<mode>”] **GET** AT+WTXPOWER?
Response	**SET** OK

<table>
<tr><td></td><td><u>GET</u>
+WTXPOWER:<power_0>,<power_1>,”<mode>”[,<limit_power>]
OK</td></tr>
<tr><td>Parameters</td><td><mode>
TX power mode
<table><tr><th>Mode</th><th>Description</th></tr><tr><td>“auto”</td><td>TX power is automatically adjusted based on signal quality. (default)</td></tr><tr><td>“fixed”</td><td>TX power remains fixed at the specified <power> level.</td></tr><tr><td>“limit”</td><td>TX power is adjusted within the specified <power>.</td></tr></table>
<power>
TX power level (1dBm to 30dBm)

<power_0>, <power_1>
TX power level at last transmission
<table><tr><th>Device Type</th><th>power_0 (wlan0)</th><th>power_1 (wlan1)</th></tr><tr><td>AP</td><td>AP TX Power</td><td>Always 0</td></tr><tr><td>STA</td><td>STA TX Power</td><td>Always 0</td></tr><tr><td>RELAY</td><td>Relay AP TX Power</td><td>Relay STA TX Power</td></tr></table>
<limit_power>
Maximum allowable TX power level when TX power mode is set to "limit."</td></tr>
<tr><td>Description</td><td>Configure the TX power level.</td></tr>
<tr><td>Example</td><td>AT+WTXPOWER?
+WTXPOWER:0,0,”auto”,30 <--- no transmission
OK
AT+WPING=”192.168.200.1”
:
OK
AT+WTXPOWER?
+WTXPOWER:20,0,”auto”,30
OK

< FIXED mode >
AT+WTXPOWER=10
OK</td></tr>
</table>

	AT+WPING=”192.168.200.1” : OK AT+WTXPOWER? +WTXPOWER:10,0,”fixed”,10 OK **< LIMIT mode >** AT+WTXPOWER=15,”limit” OK AT+WPING=”192.168.200.1” : OK AT+WTXPOWER? +WTXPOWER:15,0,”limit”,15 OK **< AUTO mode >** AT+WTXPOWER=0 OK AT+WPING=”192.168.200.1” : OK AT+WTXPOWER? +WTXPOWER:20,0,”auto”,30 OK

6.4 AT+WRXSIG

Command	**SET** AT+WRXSIG =<time> **GET** AT+WRXSIG?
Response	**SET** +WRXSIG:<rssi>,<snr> … +WRXSIG:<rssi>,<snr>

	OK **GET** +WRXSIG:<rssi>,<snr> OK
Parameters	**<time>** Monitoring time in seconds. **<rssi>** Received Signal Strength Indication **<snr>** Signal to Noise Ratio
Description	Fetch or monitor the RSSI (dBm) and SNR (dB) values in STA mode.
Example	AT+WRXSIG? +WRXSIG:-68,31 OK AT+WRXSIG=10 +WRXSIG:-68,31 +WRXSIG:-68,30 +WRXSIG:-68,32 +WRXSIG:-68,32 +WRXSIG:-68,32 +WRXSIG:-68,32 +WRXSIG:-68,30 +WRXSIG:-68,31 +WRXSIG:-68,32 +WRXSIG:-68,32 OK

6.5 AT+WRATECTRL

Command	**SET** AT+WRATECTRL=<mode> **GET** AT+WRATECTRL?
Response	**SET**

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

6.6 AT+WMCS

Command	**SET** AT+WMCS=<index> **GET** AT+WMCS?
Response	**SET** OK **GET** +WMCS=<tx_index_0>,<tx_index_1>,<rx_index_0>,<rx_index_1> OK
Parameters	**<index>** Modulation Coding Scheme index (0, 1, 2, 3, 4, 5, 6, 7 and 10) **<tx_index_0> , <tx_index_1>, <rx_index_0> , <rx_index_1>** MCS index at last transmission/reception

device type	tx_index_0, rx_index_0 (wlan0)	tx_index_1, rx_index_1 (wlan1)

	<table><tr><td>AP</td><td>AP MCS index</td><td>Always 0</td></tr><tr><td>STA</td><td>STA MCS index</td><td>Always 0</td></tr><tr><td>RELAY</td><td>Relay AP MCS index</td><td>Relay STA MCS index</td></tr></table>
Description	Set the MCS index. NOTE: The MCS index can only be set when rate control is disabled.
Example	AT+WRATECTRL? +WRATECTRL:1 OK AT+WMCS? +WMCS:7,0,7,0 OK AT+WMCS=0 ERROR AT+WRATECTRL=0 OK AT+WRATECTRL? +WRATECTRL:0 OK AT+WMCS? +WMCS:4,0,7,0 OK AT+WMCS=10 OK AT+WMCS? +WMCS:10,0,7,0 OK

6.7 AT+WDUTYCYCLE

Command	**SET** AT+WDUTYCYCLE=<window>[,<duration>[,<margin>]]

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

6.8 AT+WCCATHRESHOLD

Command	**SET**

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

6.9 AT+WTXTIME

Command	**SET** AT+WTXTIME=<cs_time>[,<pause_time>] **GET** AT+WTXTIME?
Response	**SET** OK **GET** +WTXTIME:<cs_time>,<pause_time> OK
Parameters	**<cs_time>** Carrier sensing time in microseconds (0 ~ 13260)

	<pause_time> Tx pause time in microseconds
Description	Set carrier sense time and pause time for Listen Before Talk
Example	AT+WTXTIME? +WTXTIME:0,0 OK AT+WTXTIME=128,2000 OK AT+WTXTIME? +WTXTIME:128,2000 OK

6.10 AT+WTSF

Command	**GET** AT+WTSF?
Response	**GET** +WTSF:<time_0>[,<time_1>] OK
Parameters	**<time_0> , <time_1>** Elapsed TSF timer duration in microseconds. time_0 indicates the STA time in STA mode, and the AP time in AP and RELAY modes. time_1 indicates the STA time index in RELAY mode and is excluded in AP and STA modes.
Description	Read the elapsed TSF timer duration.
Example	AT+WTSF? +WTSF:44142384 OK

6.11 AT+WBI

Command	**GET** AT+WBI?

Response	**GET** +WBI:<beacon_interval> OK
Parameters	**<beacon_interval>** Beacon interval expressed in Time Unit (TU) *1TU = 1024us
Description	Get the beacon interval of the connected AP in STA mode. The beacon Interval indicates the time between beacon frames transmitted by an AP. Since it is expressed in TU, the beacon interval time is calculated as follows. Beacon Interval Time (us) = <beacon_interval> x 1024 NOTE: If there is no connected AP, an ERROR message is returned.
Example	AT+WBI? ERROR AT+WCONN="halow_atcmd_open" OK AT+WBI? +WBI:100 OK

6.12 AT+WLI

Command	**SET** AT+WLI=<listen_interval> **GET** AT+WLI?
Response	**SET** OK **GET** +WLI:<listen_interval> OK

Parameters	**<listen_interval>** Listen interval expressed in Beacon Interval (BI)
Description	Set the listen interval in STA mode. The listen interval indicates how often the STA will wake to hear a beacon that includes a Traffic Indication Map (TIM) information element. Since it is expressed in BI, the listen interval time is calculated as follows. Listen Interval Time (us) = <listen_interval> x Beacon Interval Time = <listen_interval> x <beacon_interval> x 1024 If BSS MAX IDLE service is enabled in AP, the listen interval time should be less than BSS MAX IDLE time to avoid association-reject. NOTE: The listen interval can only be set before the AT+WCONN command. While connected to the AP, the SET command returns an ERROR message.
Example	AT+WLI? +WLI:0 OK AT+WLI=1000 OK AT+WLI? +WLI:1000 OK AT+WCONN="halow_atcmd_open" OK AT+WLI? +WLI:1000 OK AT+WLI=100

	ERROR

6.13 AT+WSCAN

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

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

	OK AT+WSCAN? +WSCAN:1,922.5 OK AT+WSCAN +WSCAN:"02:00:eb:13:d3:4a",922.5@1,-39,"[ESS]","halow_open" +WSCAN:"68:27:eb:0e:07:27",922.5@1,-30,"[WPA2-PSK-CCMP][ESS]","halow_wpa2" OK AT+WSCAN=+906,921 OK AT+WSCAN? +WSCAN:1,922.5 +WSCAN:2,921.0 +WSCAN:4,906.0 OK AT+WSCAN +WSCAN:"02:00:eb:13:d3:4a",922.5@1,-39,"[ESS]","halow_open" +WSCAN:"68:27:eb:0e:07:27",922.5@1,-30,"[WPA2-PSK-CCMP][ESS]","halow_wpa2" +WSCAN:"8c:0f:fa:00:28:1f",906.0@4,-54,"[WPA3-SAE-CCMP][ESS]","halow_sae" +WSCAN:"8c:0f:fa:00:29:46",921.0@2,-75,"[WPA3-SAE-CCMP][ESS]","halow_sae2" OK AT+WSCAN=-921,922.5 OK AT+WSCAN? +WSCAN:4,906.0 OK AT+WSCAN +WSCAN:"8c:0f:fa:00:28:1f",906.0@4,-54,"[WPA3-SAE-CCMP][ESS]","halow_sae" OK AT+WSCAN=0 OK AT+WSCAN?

+WSCAN:1,902.5,903.5,904.5,905.5,906.5,907.5,908.5,909.5,910.5,911.5
+WSCAN:1,912.5,913.5,914.5,915.5,916.5,917.5,918.5,919.5,920.5,921.5
+WSCAN:1,922.5,923.5,924.5,925.5,926.5,927.5
+WSCAN:2,903.0,905.0,907.0,909.0,911.0,913.0,915.0,917.0,919.0,921.0
+WSCAN:2,923.0,925.0,927.0
+WSCAN:4,906.0,910.0,914.0,918.0,922.0,926.0
OK

AT+WSCAN=922.5
OK
AT+WSCAN
+WSCAN:"02:00:eb:13:d3:4a",922.5@1,-39,"[ESS]","halow_open"
+WSCAN:"68:27:eb:0e:07:27",922.5@1,-30,"[WPA2-PSK-CCMP][ESS]","halow_wpa2"
OK
AT+WCONN=”halow_open”
OK
AT+WSCAN?
+WSCAN=1,922.5
OK
AT+WSCAN=+906,921
ERROR

AT+WDISCONN
OK
AT+WSCAN?
+WSCAN:1,902.5,903.5,904.5,905.5,906.5,907.5,908.5,909.5,910.5,911.5
+WSCAN:1,912.5,913.5,914.5,915.5,916.5,917.5,918.5,919.5,920.5,921.5
+WSCAN:1,922.5,923.5,924.5,925.5,926.5,927.5
+WSCAN:2,903.0,905.0,907.0,909.0,911.0,913.0,915.0,917.0,919.0,921.0
+WSCAN:2,923.0,925.0,927.0
+WSCAN:4,906.0,910.0,914.0,918.0,922.0,926.0
OK

AT+WCOUNTRY="JP"

	OK AT+WSCAN? +WSCAN:1,921.0,923.0,924.0,925.0,926.0,927.0 +WSCAN:2,923.5,924.5,925.5,926.5 +WSCAN:4,924.5,925.5 OK AT+WSCAN=926,923,923.5,925.5 OK AT+WSCAN? +WSCAN:1,923.0,926.0 +WSCAN:2,923.5,925.5 OK AT+WSCAN=926,923,926.5,925.5@2,925.5@4,924.5@2 OK AT+WSCAN? +WSCAN:1,923.0,926.0 +WSCAN:2,924.5,925.5,926.5 +WSCAN:4,925.5 OK AT+WSCAN=-926.5,925.5@2 OK AT+WSCAN? +WSCAN:1,923.0,926.0 +WSCAN:2,924.5 +WSCAN:4,925.5 OK AT+WSCAN=+924.5@4,925 OK AT+WSCAN? +WSCAN:1,923.0,925.0,926.0 +WSCAN:2,924.5 +WSCAN:4,924.5,925.5 OK

6.14 AT+WSCANSSID

Command	**SET** AT+WSCANSSID=”<ssid>”
Response	**SET** +WSCANSSID:”<bssid>”,<freq>,<sig_level>,”<flags>”,”<ssid>” OK
Parameters	**<ssid>** The SSID of the AP
Description	Perform Wi-Fi scanning with probe request frame that specify full SSID.
Example	AT+WSCANSSID="halow_atcmd_open" +WSCANSSID:"8c:0f:fa:00:28:16",902.5,-74,"[ESS]","halow_atcmd_open" OK AT+WSCANSSID="halow_atcmd_sae" +WSCANSSID:"8c:0f:fa:00:28:16",906.0,-71,"[WPA3-SAE-CCMP][ESS]","halow_atcmd_sae" OK

6.15 AT+WBGSCAN

Command	**SET** AT+WBGSCAN=<short_interval>,<long_interval>,<signal_threshold> **GET** AT+WBGSCAN?
Response	**SET** OK **GET** +WBGSCAN=<scanning>,<short_interval>,<long_interval>,<signal_threshold> OK
Parameters	**<short_interval>** Short scan interval in seconds **<long_interval>** Long scan interval in seconds

	<signal_threshold> Minimum RSSI needed for connection
Description	Performs periodic background scans based on signal strength. Background scans can be performed for roaming purposes within an ESS, a single network where all APs use the same SSID. NOTE: Parameters for background scans must be set with the AT+WBGSCAN SET command before the AT+WCONN RUN/SET command.
Example	AT+WDHCP=1 OK AT+WBGSCAN=30,300,-50 OK AT+WBGSCAN? +WBGSCAN:1,30,300,-50 OK AT+WCONN=”halow_atcmd_open” OK

6.16 AT+WSAEPWE

Command	**SET** AT+WSAEPWE=<sae_pwe> **GET** AT+WSAEPWE?
Response	**SET** OK **GET** +WSAEPWE=<sae_pwe> OK
Parameters	**<sae_pwe>** SAE PWE derivation method (default : 2) ● 0 : Hunting-and-pecking loop only ● 1 : Hash-to-element only ● 2 : Both hunting-and-pecking loop and hash-to-element enabled

	This mode provides the broadest compatibility.
Description	Set the SAE (Simultaneous Authentication of Equals) PWE (Password Element) derivation method, which is how the password element is derived during the WPA3-SAE process. The SAE PWE derivation method can be set with the SET command before the AT+WCONN, AT+WSOFTAP, and AT+WRELAY commands.
Example	AT+WSAEPWE? +WSAEPWE:2 OK AT+WSAEPWE=1 OK AT+WSAEPWE? +WSAEPWE:1 OK **Run STA/SoftAP/Relay in WPA3-SAE mode.** ● STA : AT+WCONN command ● SoftAP : AT+WSOFTAP command ● Relay : AT+WRELAY command

6.17 AT+WCONN

Command	**SET** AT+WCONN=”<ssid|bssid>”[,”<security>”[,”<password>”]] **GET** AT+WCONN?
Response	**SET** OK **GET** +WCONN=”<ssid>”,”<bssid>”,”<security>”,”<password>”,”<state>” OK
Parameters	**<ssid>** The SSID of the AP. **<bssid>** The BSSID of the AP. **<security>** open*, wpa2-psk (or psk), wpa3-owe (or owe), wpa3-sae (or sae)

<table>
<tr><td></td><td>

<password> (wpa2/wpa3-sae security option only)
The password when wpa2/wpa3-sae security option is used. (length : 8 ~ 64)

<state>
State indicator: “connecting”, “connected”, “disconnecting” or ”disconnected”

NOTE:
For security reasons from AT Command Set v1.26.7, we decided to hide the password in the response message to the GET command. However, for compatibility with previous versions, the password field is displayed as "" or "*". "*" indicates that the AP information is recovered after waking up from deep sleep with the AT+WDEEPSLEEP command. And in this case, the AP information is initialized after disconnection.

</td></tr>
<tr><td>Description</td><td>

Connect to a new AP or retrieves information about the current AP.

If an ERROR is returned with the error number 2 (in progress) or 4 (timeout), the followings are required before a connection is attempted again with the AT+WCONN command.

<table>
<tr><th>Error number</th><th>Required operation</th></tr>
<tr><td>2 (in progress)</td><td>STA should be disconnected from the AP with the AT+WDISCONN command.</td></tr>
<tr><td>4 (timeout)</td><td>Amount of timeout should be increased with the AT+WTIMOEUT command.</td></tr>
</table>

</td></tr>
<tr><td>Example</td><td>

OPEN :
AT+WSCAN
+WSCAN:"8c:0f:fa:00:2b:a1",922.0@4,-13,"[ESS]","halow_ap”
OK
AT+WCONN="halow_ap”
OK
AT+WCONN?
+WCONN:”halow_ap”,”8C:0F:FA:00:2B:A1”,”open”,””,”connected”
OK

WPA2-PSK :
AT+WSCAN
+WSCAN:"8c:0f:fa:00:2b:a1",922.0@4,-14,"[WPA2-PSK-CCMP][ESS]","halow_ap"

</td></tr>
</table>

	OK AT+WCONN="halow_ap","wpa2-psk","12345678" OK AT+WCONN? +WCONN:”halow_ap”,”8C:0F:FA:00:2B:A1”,”wpa2-psk”,””,”connected” OK **WPA3-OWE :** AT+WSCAN +WSCAN:"8c:0f:fa:00:2b:a1",922.0@4,-13,"[WPA2-OWE-CCMP][ESS]","halow_ap” OK AT+WCONN="halow_ap”,"wpa3-owe" OK AT+WCONN? +WCONN:”halow_ap”,”8C:0F:FA:00:2B:A1”,”wpa3-owe”,””,”connected” OK **WPA3-SAE :** AT+WSCAN +WSCAN:"8c:0f:fa:00:2b:a1",922.0@4,-14,"[WPA2-SAE-CCMP][ESS]","halow_ap" OK AT+WCONN="halow_ap","wpa3-sae","12345678" OK AT+WCONN? +WCONN:”halow_ap”,”8C:0F:FA:00:2B:A1”,”wpa3-sae”,””,”connected” OK

6.18 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

6.19 AT+WSOFTAP

Command	**SET** AT+WSOFTAP=<frequency>[@<bandwidth>],"<ssid>"[,"<security>"[,"<password>"]] **GET** AT+WSOFTAP?
Response	**SET** OK **GET** +WSOFTAP=<bandwidth>,<frequency>,"<ssid>","<security>","<password>"[,"dhcp"] OK
Parameters	**<bandwidth>** S1G channel bandwidth (1/2/4 MHz) **<frequency>** S1G channel frequency (MHz) **<ssid>** The SSID of the AP. **<security>** open*, wpa2-psk (or psk), wpa3-owe (or owe), wpa3-sae (or sae) **<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. NOTE: For security reasons from AT Command Set v1.26.7, we decided to hide the password in the response message to the GET command. However, for compatibility with previous versions, the password field is displayed as "".
Description	Run as the AP mode or retrieves information about the current settings. NOTE: The system should be reset to exit the AP mode. Software Reset is possible with the ATZ command.
Example	AT+WCOUNTRY="JP"

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

6.20 AT+WSOFTAPSSID

Command	**SET** AT+WSOFTAPSSID=<type> **GET** AT+WSOFTAPSSID?
Response	**SET** OK **GET** +WSOFTAPSSID:<type> OK
Parameters	**<type>** 0 : Full SSID* 1 : Empty SSID (length=0) 2 : Clear SSID
Description	Set how to specify the SSID in the beacon frame.

	Empty SSID or Clear SSID is used to hide the SSID on the network. NOTE: Set the SSID type before starting the AP with the AT+WSOFTAP command.
Example	AT+WSOFTAPSSID? +WSOFTAPSSID:0 OK AT+WSOFTAPSSID=1 OK AT+WSOFTAPSSID? +WSOFTAPSSID:1 OK AT+WSOFTAP=925,"halow_atcmd_open" OK AT+WSOFTAPSSID? +WSOFTAPSSID:1 OK AT+WSOFTAPSSID=2 ERROR

6.21 AT+WBSSMAXIDLE

Command	**SET** AT+WBSSMAXIDLE=<period>[,<retry>] **GET** AT+WBSSMAXIDLE?
Response	**SET** OK **GET** +WBSSMAXIDLE:<period>,<retry> OK
Parameters	**<period>** BSS MAX IDLE period in 1000TU (1 ~ 65535, default: 0) *TU : Time Unit (1024 us)

	<retry> retry count for receiving keep alive packet from STA (3 ~ 100, default: 3)
Description	Configure the BSS MAX IDLE service for SoftAP. SoftAP disconnects STA that is inactive for BSS MAX IDLE time. If the AP does not receive a keep alive packet from the STA for BSS MAX IDLE time, it is determined that the STA is in an inactive state. The listen interval time should be less than BSS MAX IDLE time to avoid association-reject. Example: - period = 1800 TU, retry count = 5 - BSS MAX IDLE time = 1800 x (1000 x 1024) = 1843.2 secs - Total BSS MAX IDLE time = 5 x 1843.2 = 9216 secs If the period is set 0, the service is disabled.
Example	AT+WBSSMAXIDLE? +WBSSMAXIDLE:0,3 OK AT+WBSSMAXIDLE=1800 OK AT+WBSSMAXIDLE? +WBSSMAXIDLE:1800,3 OK AT+WSOFTAP=918.5,"halow_softap_wpa2","wpa2-psk","12345678" OK AT+WDHCPS +WDHCPS:"192.168.50.1","255.255.255.0","192.168.50.1" OK AT+WBSSMAXIDLE=1800,5 OK AT+WBSSMAXIDLE? +WBSSMAXIDLE:1800,5

	OK AT+WBSSMAXIDLE=0 OK AT+WBSSMAXIDLE? +WBSSMAXIDLE:0,3 OK

6.22 AT+WSTAINFO

Command	**SET** AT+WSTAINFO=<aid>[,<time>] **GET** AT+WSTAINFO?
Response	+WSTAINFO=<aid>,"<mac_address>",<rssi>,<snr>,<tx_mcs>,<rx_mcs> OK
Parameters	**<aid>** Association ID **<time>** Monitoring duration in seconds. **<mac_address>** Hardware address of associated station **<rssi>** Received Signal Strength Indication **<snr>** Signal to Noise Ratio **<tx_mcs> , <rx_mcs>** Modulation Coding Scheme index
Description	Get information of associated STAs in SoftAP mode.
Example	AT+WSOFTAP=918.5,"halow_softap","wpa2-psk","12345678" OK AT+WIPADDR="192.168.1.1","255.255.255.0","192.168.1.1"

	OK AT+WDHCPS +WDHCPS:”192.168.1.1”,”255.255.255.0”,”192.168.1.1” OK Wait for one or more stations to be associated … AT+WSTAINFO? +WSTAINFO:1,"8c:0f:fa:00:2b:a1",-34,31,7,7 +WSTAINFO:2,"8c:0f:fa:00:2b:a2",-45,34,7,7 +WSTAINFO:3,"8c:0f:fa:00:2b:a3",-16,21,7,7 OK AT+WSTAINFO=1 +WSTAINFO:1,"8c:0f:fa:00:2b:a1",-33,34,7,7 OK AT+WSTAINFO=3,5 +WSTAINFO:3,"8c:0f:fa:00:2b:a3",-16,22,7,7 +WSTAINFO:3,"8c:0f:fa:00:2b:a3",-18,21,7,7 +WSTAINFO:3,"8c:0f:fa:00:2b:a3",-16,21,7,7 +WSTAINFO:3,"8c:0f:fa:00:2b:a3",-16,22,7,7 +WSTAINFO:3,"8c:0f:fa:00:2b:a3",-17,21,7,7 OK

6.23 AT+WMAXSTA

Command	**SET** AT+WMAXSTA=<max_num_sta> **GET** AT+WMAXSTA?
Response	**SET** OK **GET** +WMAXSTA=<max_num_sta> OK
Parameters	**<max_num_sta>** maximum number of STAs
Description	Set the maximum number of STAs allowed in AP mode.

	NOTE: The maximum number of STAs must be set before starting AP mode with the AT+WSOFTAP SET command.
Example	AT+WMAXSTA? +WMAXSTA:10 OK AT+WMAXSTA=1 OK AT+WSOFTAP=925,"halow_softap_psk","psk","12345678" OK AT+WMAXSTA? +WMAXSTA:1 OK

6.24 AT+WIPADDR

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

6.25 AT+WDNS

Command	**SET** AT+WDNS=”<DNS1>”[,”<DNS2>”] **GET** AT+WDNS?
Response	**SET** OK **GET** +WDNS=”<DNS1>”,”<DNS2>” OK
Parameters	**<DNS1>,<DNS2>** IPv4 address
Description	Configure the IP address of the DNS server.
Example	AT+WDNS? +WDNS=”192.168.200.1”,”0.0.0.0” OK AT+WDNS=”8.8.8.8” OK AT+WDNS? +WDNS=”8.8.8.8”,”0.0.0.0” OK AT+WDNS=”8.8.8.8”,”8.8.4.4” OK AT+WDNS? +WDNS=”8.8.8.8”,”8.8.4.4” OK

6.26 AT+WDHCP

Command	**RUN** AT+WDHCP **SET** AT+WDHCP=<mode>

	GET AT+WDHCP?
Response	**RUN** +WDHCP:”<address>”,”<netmask>”,”<gateway>”,<lease_time> OK **SET** OK **GET** +WDHCP:{0|1} OK
Parameters	**<mode>** 0 : run manually after connection 1 : run automatically connection or reconnection **<address>, <netmask>** and **<gateway>** IPv4 Address **<lease_time>** Duration of time in seconds that a DHCP server grants a device to use an IP address.
Description	Request dynamic IP allocation from the DHCP server. NOTE: Wi-Fi connection must be established before using this command.
Example	AT+WCONN="halow_ap","wpa3-sae","12345678" OK AT+WDHCP +WDHCP:″192.168.200.20″,″255.255.255.0″,″192.168.200.1″,86400 OK AT+WDISCONN OK AT+WDHCP? +WDHCP:0 OK AT+WDHCP=1 OK AT+WCONN="halow_ap","wpa3-sae","12345678"

	OK +WEVENT:"DHCP_RUN" +WEVENT:"DHCP_SUCCESS","192.168.200.18","255.255.255.0","192.168.200.1" +WEVENT:"DISCONNECT","","halow_ap","wpa3-sae" +WEVENT:"CONNECT_SUCCESS","","halow_ap","wpa3-sae" +WEVENT:"DHCP_RUN" +WEVENT:"DHCP_SUCCESS","192.168.200.18","255.255.255.0","192.168.200.1",86400

6.27 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

6.28 AT+WPING

Command	**SET** AT+WPING=”<remote address>”[,<time>] **GET** AT+WPING?
Response	**SET** +WPING:<size>,”<remote address>”,<sequence number>,<TTL>,<elapsed time> : +WPING:<size>,”<remote address>”,<sequence number>,<TTL>,<elapsed time>

	OK **GET** +WPING:”<remote address>”,<time>
Parameters	**<remote address>** The remote IPv4 address of the recipient. **<time>** Monitoring duration in seconds. (Default: 5) **<sequence number>** ICMP sequence number. **<TTL>** Time to leave (TTL). **<elapsed time>** Time since the start of the session in seconds.
Description	Send ICMP ECHO_REQUEST to network hosts with IPv4 address. - Interval Time : 1 sec - Packet Size : 64-bytes
Example	AT+WPING =”192.168.200.1”,10 +WPING:64,”192.168.200.1”,1,64,4 +WPING:64,”192.168.200.1”,2,64,4 : +WPING:64,”192.168.200.1”,9,64,4 +WPING:64,”192.168.200.1”,10,64,4 OK

6.29 AT+WDEEPSLEEP

Command	**SET** AT+WDEEPSLEEP=<timeout>[,<gpio>]
Response	**SET** OK
Parameters	**<timeout>** Time in milliseconds.

<table>
<tr><td></td><td>0 for TIM mode.

<gpio>
GPIO number to use as external signal input.
<table><tr><th>Host Interface Type</th><th>Available GPIO numbers</th></tr><tr><td>HSPI</td><td>10, 11, 12, 13, 14, 20, 25</td></tr><tr><td>UART</td><td>6, 7, 10, 11, 25, 28, 29, 30</td></tr></table></td></tr>
<tr><td>Description</td><td>Configure deep-sleep mode to save power.

Deep sleep mode powers off most peripherals to use minimal power. The RTC and retention RAM are always powered. The CPU is powered only in TIM mode to run the uCode stored in the retention RAM. And the GPIO may be powered for external signal input.

In TIM mode, the NRC7394 wakes up when there are frames to receive. However, in Non-TIM mode, it cannot be woken up until a timeout.

If there are frames to send, the NRC7394 can only be woken up via the GPIO input. The GPIO input level should be low in active mode. If it is high in deep sleep mode, the NRC7394 wakes up. After waking up, the CPU resets and the firmware reboots. When the firmware reboot is finished, the host application or terminal program will receive a "DEEPSLEEP_WAKEUP" event message. And the AP connection and IP address will also be recovered to the same as before entering deep sleep.</td></tr>
<tr><td>Example</td><td>< Deep Sleep, TIM mode >
AT+WCONN="halow_ap","wpa2-psk","12345678"
OK
AT+WDHCP
+WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1"
OK
AT+WDEEPSLEEP=0,11
OK

+WEVENT:”DEEPSLEEP_WAKEUP”

AT+WCONN?
+WCONN="halow_ap","wpa2-psk","*",”connected”
OK
AT+WIPADDR?</td></tr>
</table>

	+WIPADDR:"192.168.200.18","255.255.255.0","192.168.200.1" OK AT+WPING=”192.168.200.1”,2 +WEVENT:"PING",64,"192.168.200.1",1,64,5 +WEVENT:"PING",64,"192.168.200.1",2,64,4 OK **< Deep Sleep, Non-TIM mode >** AT+WCONN="halow_ap","wpa3-sae","12345678" OK AT+WDHCP +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1" OK AT+WDEEPSLEEP=5000,11 OK +WEVENT:”DEEPSLEEP_WAKEUP” AT+WCONN? +WCONN="halow_ap","wpa3-sae","*",”connected” OK AT+WIPADDR? +WIPADDR:"192.168.200.18","255.255.255.0","192.168.200.1" OK AT+WPING=”192.168.200.1”,2 +WEVENT:"PING",64,"192.168.200.1",1,64,6 +WEVENT:"PING",64,"192.168.200.1",2,64,4 OK

6.30 AT+WFOTA

Command	**SET** AT+WFOTA=<check_time>[,\"<server_url>\"] AT+WFOTA=<check_time>[,\"<server_url>\",\"<bin_name>\",<bin_crc32>] **GET** AT+WFOTA? **RUN**

	AT+WFOTA
Response	**SET** OK **GET** +WFOTA:<check_time>,"<server_url>","<bin_name>",<bin_crc32> OK **RUN** OK
Parameters	**<check_time>** Interval time in seconds to get new firmware information from the server. Set to 0 to stop the getting or get manually. Set to -1 to disable FOTA operation. **<server_url>** HTTP or HTTPS Server URL *AT command firmware for 2MB FLASH does not support HTTPS. **<bin_name>** Firmware binary name with extension .bin. **<bin_crc32>** A 32-bit hexadecimal value, prefixed with '0x' and calculated using the CRC-32 algorithm to detect data corruption. To determine the CRC value of the 'newFW.bin' file, you can use the 'crc.py' script located in the 'package\standalone\atcmd\host\python-http-server\python' directory. Simply run the command 'python crc.py newFW.bin' and add the '0x' prefix to the result. (ex) python crc.py newFW.bin 97cb8611
Description	FOTA(Firmware Over-the-Air) is enabled with the SET command and disabled by AT+WFOTA=-1 command. When FOTA is enabled, the current firmware starts checking for new firmware on the server. The server check interval can be controlled through the <check_time> parameter. To check for new firmware, the current firmware downloads the fota.json file from the server. The server should have a fota.json file as well as firmware binary. The contents of the fota.json file are as follows.

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

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

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

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

After getting information about new firmware from the server, the current firmware sends a FOTA_VERSION event to the terminal or host.

+WEVENT:”FOTA_VERSION”,”<sdk_version>”,”<atcmd_version>”

After receiving the FOTA_VERSION event, the terminal or host can use the RUN command to download new firmware from the server.

If there is no fota.json file on the server, the firmware information to be downloaded can be set with the bin_name and bin_crc32 parameters. And the terminal or host can use the RUN command without receiving the FOTA_VERSION event.

The terminal or host can check the download process through FOTA_BINARY and FOTA_DOWNLOAD events from the current firmware.

+WEVENT: “FOTA_BINARY”,”<binary_name>”

+WEVENT: “FOTA_DOWNLOAD”,<total_size>,<download_size>

When the download is complete and ready to update, the terminal or host will receive a FOTA_UPDATE event from the current firmware.

+WEVENT: “FOTA_UPDATE”

If an error occurs during the above process, the terminal or host will receive a FOTA_FAIL event from the current firmware.

+WEVENT: “FOTA_FAIL”

And FOTA will be automatically disabled.

If there are no errors, the current firmware will be replaced with the new firmware after a software reset. A software reset is possible with the ATZ command. Firmware replacement will take about 10 seconds or more.

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

NOTE:

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

EVENT:

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

TEST:

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

Path : atcmd/host/python-http-server

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

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

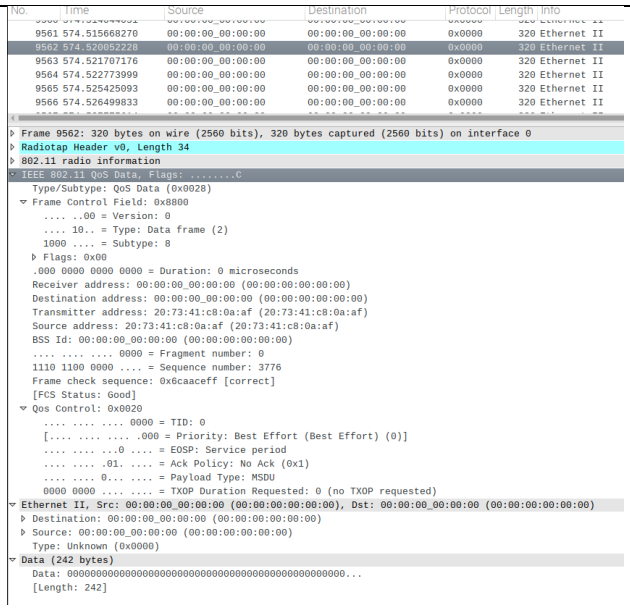
Example	AT+VER? +VER:”1.0.0”,”1.23.5”

	OK AT+WFOTA? +WFOTA:0,"","",0x0 OK **< Get new firmware information from fota.json file >** AT+WFOTA=10,"https://192.168.200.1:4443" AT+WFOTA=10,"https://192.168.200.1:4443" OK AT+WFOTA? +WFOTA:10,"https://192.168.200.1:4443","",0x0 OK +WEVENT:"FOTA_VERSION","10.10.10","10.10.10" +WEVENT:"FOTA_VERSION","10.10.10","10.10.10" +WEVENT:"FOTA_VERSION","10.10.10","10.10.10" *Stop the getting to switch manually. AT+WFOTA=0 OK AT+WFOTA=0 OK +WEVENT:"FOTA_VERSION","10.10.10","10.10.10" **< Set new firmware information without fota.json file >** AT+WFOTA=0,"https://192.168.200.1:4443","nrc7394_atcmd_hspi.bin",0x3e47cf92 OK AT+WFOTA? +WEVENT:0,"https://192.168.200.1:4443","nrc7394_atcmd_hspi.bin",0x3E47CF92 OK **< Download the firmware binary >** AT+WFOTA OK +WEVENT:"FOTA_BINARY","nrc7394_atcmd_hspi.bin" +WEVENT:"FOTA_DOWNLOAD",897632,90112 +WEVENT:"FOTA_DOWNLOAD",897632,180224

	+WEVENT:"FOTA_DOWNLOAD",897632,270336 : +WEVENT:"FOTA_DOWNLOAD",897632,720896 +WEVENT:"FOTA_DOWNLOAD",897632,811008 +WEVENT:"FOTA_DOWNLOAD",897632,897632 +WEVENT:"FOTA_UPDATE" **< Reset and update >** ATZ

6.31 AT+WCTX

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



NOTE:
This command is for testing purposes only.

Example

AT+WCOUNTRY=”US”
OK

< Set parameters for continuous TX >
AT+WCTX=9180,4,7,17
OK
AT+WCTX?
+WCTX:9180,4,7,17
OK

< Start continuous TX >
AT+WCTX
OK

< Stop continuous TX >
AT+WCTX=0
OK

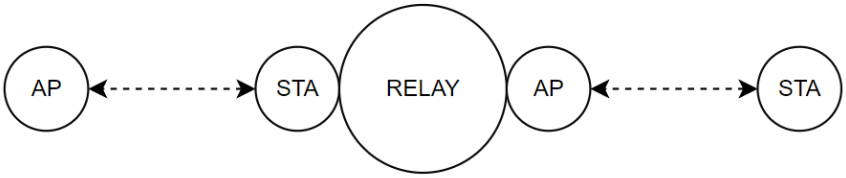
6.32 AT+WSTX

Command	**RUN**

	AT+WSTX **SET** AT+WSTX=<frequency>,<bandwidth>,<txpower> **GET** AT+WSTX?
Response	**RUN/SET** OK **GET** +WSTX: <frequency>,<bandwidth>,<txpower> OK
Parameters	**<frequency>** Channel frequency in units of 100 KHz **<bandwidth>** S1G channel bandwidth (1, 2 and 4 MHz) **<txpower>** Transmission Power Level (1 ~ 30 dBm)
Description	Start or stop sine wave TX. NOTE: This command is for testing purposes only.
Example	**< Set parameters for sine wave TX >** AT+WSTX=9180,4,17 OK AT+WSTX? +WSTX:9180,4,17 OK **< Start sine wave TX >** AT+WSTX OK **< Stop sine wave TX >** AT+WSTX=0 OK

6.33 AT+WRELAY

Command	**SET** AT+WRELAY=”<ap_ssid>”,”<sta_ssid>”[,”<sta_security>”[,”<sta_password>”]] **GET** AT+WRELAY?
Response	**SET** OK **GET** +WRELAY=<bandwidth>,<frequency>,”<ap_ssid>”,”<sta_ssid>”,”<security>”,”<password>” OK
Parameters	**<bandwidth>** S1G channel bandwidth (1, 2 and 4 MHz) **<frequency>** S1G Channel frequency in units of 100 KHz **<ap_ssid>** SSID used for RELAY AP (wlan0) **<sta_ssid>** SSID used by RELAY STA (wlan1) **<sta_security> , <security>** open*, wpa2-psk (or psk) **<sta_password> , <password>** Password for wpa2-psk (length : 8 ~ 63) NOTE: For security reasons from AT Command Set v1.26.7, we decided to hide the password in the response message to the GET command. However, for compatibility with previous versions, the password field is displayed as "".
Description	Run as the RELAY mode or retrieves information about the current settings.



RELAY is an AP that supports STA mode and is located between the AP and STA for STAs outside the AP's coverage range.

In RELAY mode, two WLAN interfaces are activated, and the interface names are wlan0 and wlan1, respectively. wlan0 is used as the RELAY AP and wlan1 is used as the RELAY STA. And wlan0 and wlan1 are bridged through the br interface.

```
br        HWaddr 8C:0F:FA:00:FF:7A     MTU:1500
          inet:192.168.200.2          netmask:255.255.255.0     gateway:192.168.200.1

wlan0     HWaddr 8C:0F:FA:FF:FF:7A     MTU:1500
          inet:0.0.0.0        netmask:0.0.0.0 gateway:0.0.0.0

wlan1     HWaddr 8C:0F:FA:FF:FF:D1     MTU:1500
          inet:0.0.0.0        netmask:0.0.0.0 gateway:0.0.0.0
```

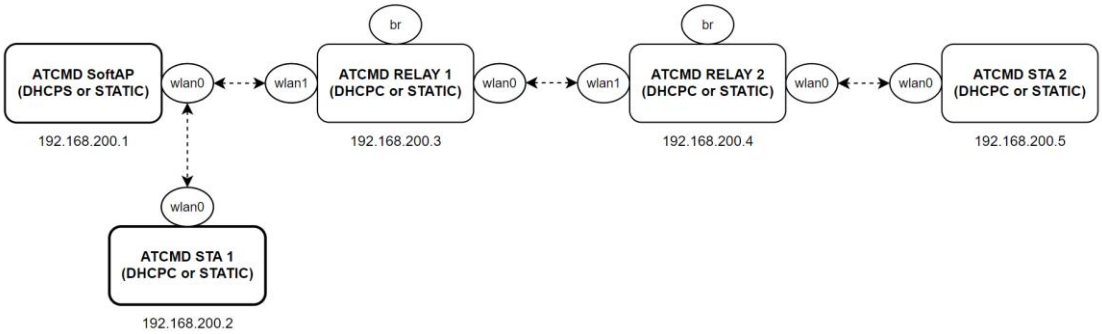
Once the RELAY STA on wlan1 is successfully connected to the AP, the RELAY AP on wlan0 will run with the same channel and security settings as the AP. The SSID of the RELAY AP may be the same or different from the AP.

NOTE:

The system should be reset to exit the RELAY mode.

Software Reset is possible with the ATZ command.

Example



< ATCMD SoftAP >

```
AT+WCOUTNRY=”US”
OK
AT+WSOFTAP=918,”halow_softap”,”wpa2-psk”,”12345678”
OK
AT+WSOFTAP?
+WSOFTAP:4,918.0,"halow_softap","wpa2-psk",""
OK
AT+WIPADDR=”192.168.200.1”,”255.255.255.0”,”192.168.200.1”
OK
AT+WDHCPS
+WDHCPS:192.168.200.1,255.255.255.0,192.168.200.1
OK
```

< ATCMD STA 1 >

```
AT+WCOUNTRY=”US”
OK
AT+WSCAN
+WSCAN:"8c:0f:fa:00:28:16",918.0@4,-44," [WPA2-PSK-CCMP][ESS]","halow_softap"
OK
AT+WCONN="halow_softap","wpa2-psk","12345678"
OK
AT+WCONN?
+WCONN:"halow_softap","8c:0f:fa:00:28:16","wpa2-psk","","connected"
OK
AT+WDHCP
+WDHCP:"192.168.200.2","255.255.255.0","192.168.200.1"
OK
```

< ATCMD RELAY 1 >

```
AT+WCOUNTRY=”US”
OK
AT+WSCAN
+WSCAN:"8c:0f:fa:00:28:16",918.0@4,-41," [WPA2-PSK-CCMP][ESS]","halow_softap"
OK
AT+WRELAY="halow_relay_1","halow_softap","wpa2-psk","12345678"
```

OK
AT+WRELAY?
+WRELAY:4,918.0,"halow_relay_1","halow_softap","wpa2-psk",""
OK
AT+WDHCP
+WDHCP:"192.168.200.3","255.255.255.0","192.168.200.1"
OK

< ATCMD RELAY 2 >

AT+WCOUNTRY=”US”
OK
AT+WSCAN
+WSCAN:"8c:0f:fa:00:28:16",918.0@4,-63," [WPA2-PSK-CCMP][ESS]","halow_softap"
+WSCAN:"8c:0f:fa:ff:ff:7a",918.0@4,-45," [WPA2-PSK-CCMP][ESS]","halow_relay_1"
OK
AT+WRELAY="halow_relay_2","halow_relay_1","wpa2-psk","12345678"
OK
AT+WRELAY?
+WRELAY:4,918.0,"halow_relay_2","halow_relay_1","wpa2-psk",""
OK
AT+WDHCP
+WDHCP:"192.168.200.4","255.255.255.0","192.168.200.1"
OK

< ATCMD STA 2 >

AT+WCOUNTRY=”US”
OK
AT+WSCAN
+WSCAN:"8c:0f:fa:00:28:16",918.0@4,-74,"[ESS]","halow_softap"
+WSCAN:"8c:0f:fa:ff:ff:7a",918.0@4,-55,"[ESS]","halow_relay_1"
+WSCAN:"8c:0f:fa:00:0d:3c",918.0@4,-42,"[ESS]","halow_relay_2"
OK
AT+WCONN="halow_relay_2","wpa2-psk","12345678"
OK
AT+WCONN?
+WCONN:"halow_relay_2","8c:0f:fa:00:0d:3c","wpa2-psk","","connected"
OK

	AT+WDHCP +WDHCP:"192.168.200.5","255.255.255.0","192.168.200.1" OK AT+WPING=”192.168.200.2” +WPING:64,"192.168.200.2",1,255,11 +WPING:64,"192.168.200.2",2,255,9 +WPING:64,"192.168.200.2",3,255,25 +WPING:64,"192.168.200.2",4,255,9 +WPING:64,"192.168.200.2",5,255,9 OK

6.34 AT+WWPS

Command	**RUN** AT+WWPS **SET** AT+WWPS=”<bssid>”
Response	**RUN** OK **SET** OK
Parameters	**<bssid>** The BSSID of the AP.
Description	Enable WPS-PBC mode and start WPS negotiation. WPS-PBC (Push Button Configuration) is a method within Wi-Fi Protected Setup (WPS). The result of the WPS Negotiation can be identified by one of the following events. +WEVENT:"WPS_SUCCESS" +WEVENT:"WPS_TIMEOUT" +WEVENT:"WPS_FAIL" NOTE: This command is not supported in RELAY mode. The SET command with the BSSID of the AP is supported only in STA mode. The AT+WWPS=0 command can be used to cancel a pending WPS operation.
Example	**[ATCMD SOFTAP]**

AT+WMACADDR?
+WMACADDR:"88:57:1d:f1:e1:ba","88:57:1d:f1:e1:bb"
OK
AT+WSOFTAP=918,"halow_atcmd_softap","wpa2-psk","12345678"
OK
AT+WSOFTAP?
+WSOFTAP:4,918.0,"halow_atcmd_softap","wpa2-psk",""
OK
AT+WIPADDR="192.168.100.1","255.255.255.0","192.168.100.1"
OK
AT+WDHCPS
+WDHCPS:192.168.100.1,255.255.255.0,192.168.100.1
OK
AT+WWPS
OK

+WEVENT:"STA_CONNECT","88:57:1D:F1:E1:70"
+WEVENT:"WPS_SUCCESS"
+WEVENT:"STA_DISCONNECT","88:57:1D:F1:E1:70"
+WEVENT:"STA_CONNECT","88:57:1D:F1:E1:70"

[ATCMD STA]

AT+WMACADDR?
+WMACADDR:"88:57:1d:f1:e1:70","88:57:1d:f1:e1:71"
OK
AT+WDHCP=1
OK
AT+WWPS
OK

+WEVENT:"WPS_SUCCESS"
+WEVENT:"CONNECT_SUCCESS","88:57:1d:f1:e1:ba","halow_atcmd_softap","wpa2-psk"
+WEVENT:"DHCP_START"
+WEVENT:"DHCP_SUCCESS","192.168.100.2","255.255.255.0","192.168.100.1"

6.35 AT+WTIMEOUT

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

	OK

6.36 +WEVENT

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

	+WEVENT:"DHCP_STOP" +WEVENT:"DHCP_BUSY" +WEVENT:"DHCP_FAIL" +WEVENT:"DHCP_SUCCESS","192.168.200.18","255.255.255.0","192.168.200.1",86400 +WEVENT:"DHCP_TIMEOUT",60 +WEVENT:"STA_CONNECT","8C:0F:FA:00:39:0D" +WEVENT:"STA_DISCONNECT","8C:0F:FA:00:39:0D" +WEVENT:"FOTA_VERSION","10.10.10","10.10.10" +WEVENT:"FOTA_BINARY","nrc7394_atcmd_hspi.bin" +WEVENT:"FOTA_DOWNLOAD",897632,90112 +WEVENT:"FOTA_UPDATE" +WEVENT:"FOTA_FAIL" +WEVENT:”DEEPSLEEP_WAKEUP” +WEVENT:"WPS_SUCCESS" +WEVENT:"WPS_TIMEOUT" +WEVENT:"WPS_FAIL"

7 Socket AT Commands

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

7.1 AT+SOPEN

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

	OK

7.2 AT+SCLOSE

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

7.3 AT+SLIST

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

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

7.4 AT+SSEND

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

	SEND_DONE event. (0:disable, 1:enable)
Description	Send data through a socket. Data can be sent in one of the following modes when the return message is OK. ① Synchronous Send Synchronous send mode is set when the length parameter has a positive number. The length parameter indicates the length of data sent with one AT+SSEND command. Data can be sent up to 4096 bytes at a time. ② (Buffered) Passthrough Send Data can be continuously sent with one AT+SSEND command. Passthrough send mode is set when the length parameter is 0 or omitted. Data is copied to the TCP/IP stack by the socket send function without buffering, and the length of the copied data is variable. Buffered passthrough send mode is set when the length parameter has a negative number. The length parameter indicates the length of the buffer. The maximum length of the buffer is 4096 bytes. If the length parameter is -2048, data is buffered up to 2048 bytes. The maximum length of data copied to the TCP/IP stack by the socket send function is equal to the buffer length. To exit (buffered) passthrough send mode and send a new AT command, the following is required: 1. Wait at least 1 second after sending the last data. 2. Send the EXIT command "AT\r\n" when SEND_IDLE event is raised. 3. Send a new AT command after SEND_EXIT event is raised. If an error occurs before the data is copied to the TCP/IP stack, SEND_ERROR event is raised. If the done_event parameter is set to 1, SEND_DONE event is raised when data is successfully copied to the TCP/IP stack. NOTE: If the host interface is UART and hardware flow control is disabled, the (buffered) passthrough send mode is not available. Data can only be sent in synchronous

	send mode, and it is recommended to set the done_event parameter to 1 and send the next data after checking the SEND_DONE event.
Example	**[Synchronous Send : done_event=0]** AT+SSEND=0,6 OK Hello! **[Synchronous Send : done_event=1]** AT+SSEND=0,6,1 OK Hello! +SEVENT:"SEND_DONE",6 **[Passthrough Send : done_event=0]** AT+SSEND=0 OK Hello, World! Goodbye, World! */* If no data is sent for more than 1 second, the SEND_IDLE event is raised. */* +SEVENT:"SEND_IDLE",0,28,0,0 */* Send the EXIT command "AT\r\n" to exit the passthrough send mode. */* AT OK +SEVENT:"SEND_EXIT",0,28,0 **[Buffered Passthrough Send : done_event=1]** AT+SSEND=0,-8,1 OK TEST0001 +SEVENT:"SEND_DONE",8 TEST0002 +SEVENT:"SEND_DONE",8 TEST0003

	+SEVENT:"SEND_DONE",8 */* Wait for the SEND_IDLE event without sending any data to exit the buffered passthrough send mode. */* +SEVENT:"SEND_IDLE",0,24,0,0 AT OK +SEVENT:"SEND_EXIT",0,24,0

7.5 AT+SRECV

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

Example	AT+SLIST? +SLIST:0,"TCP","192.168.200.1",50000,0 +SLIST:1,"UDP","0.0.0.0",0,60001 OK +SEVENT:"RECV_READY",0,1024 +SEVENT:"RECV_READY",1,1024 AT+SRECV? +SRECV:0,7168 +SRECV:1,7168 OK AT+SRECV=0 +RXD:0,4096,"192.168.200.1",50000 OK AT+SRECV=1 +RXD:1,1024,"192.168.200.1",60000 OK +SEVENT:"RECV_READY",0,3072 +SEVENT:"RECV_READY",1,6144 AT+SRECV?=0 +SRECV:0,3072 OK AT+SRECV?=1 +SRECV:1,6144 OK

7.6 AT+SRECVMODE

Command	**SET** AT+SRECVMODE=<mode>[,<event>] **GET** AT+SRECVMODE?
Response	**SET** OK **GET** +SRECVMODE:<mode>,<event>

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

7.7 AT+SRECVINFO

Command	**SET** AT+SRECVINFO=<mode> **GET** AT+SRECVINFO?

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

7.8 AT+SADDRINFO

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

	OK

7.9 AT+STCPKEEPALIVE

Command	**SET** AT+STCPKEEPALIVE=<socket ID>,<keepalive>[,<keepidle>,<keepcnt>,<keepintvl>] **GET** AT+STCPKEEPALIVE? AT+STCPKEEPALIVE?=<socket ID>
Response	**SET** OK **GET** +STCPKEEPALIVE:<socket_ID>,<keepalive>,<keepidle>,<keepcnt>,<keepintvl> : OK
Parameters	**<socket ID>** The ID allocated to the socket for TCP client. **<keepalive>** 0 : disable 1 : enable **<keepidle>** The time to wait before sending out the first probe in seconds. (default : 7200) **<keepcnt>** The number of probes that are sent and unacknowledged. (default : 9) **<keepintvl>** The interval between subsequent keepalive probes in seconds. (default : 75)
Description	Enable or disable TCP keepalive.
Example	**< TCP Server >** AT+SOPEN=”TCP”,50000 +SOPEN=0 OK +SEVENT:”CONNECT”,1 AT+SLIST? +SLIST:0,”TCP”,”0.0.0.0”,0,50000 +SLIST:1,”TCP”,”192.168.200.2”,52432,0 OK AT+STCPKEEPALIVE?

	+STCPKEEPALIVE:1,0,7200,9,75 OK AT+STCPKEEPALIVE=1,0,60,5,30 OK AT+STCPKEEPALIVE? +STCPKEEPALIVE:1,0,60,5,30 OK AT+STCPKEEPALIVE=1,1 OK AT+STCPKEEPALIVE? +STCPKEEPALIVE:1,1,60,5,30 OK **< TCP Client >** AT+SOPEN=”TCP”,”192.168.200.1”,50000 +SOPEN:0 OK AT+SLIST? +SLIST:0,”TCP”,”192.168.200.1”,50000,0 OK AT+STCPKEEPALIVE? +STCPKEEPALIVE:0,0,7200,9,75 OK AT+STCPKEEPALIVE=0,1,60,5,30 OK AT+STCPKEEPALIVE?=0 +STCPKEEPALIVE:0,1,60,5,30 OK

7.10 AT+STCPNODELAY

Command	**SET** AT+STCPNODELAY=<socket ID>,{0|1} **GET** AT+STCPNODELAY?
Response	**SET** OK

	GET +STCPNODELAY:<socket_ID>,<status> OK
Parameters	**<socket ID>** The ID allocated to the socket. **<status>** 0 : disable 1 : enable
Description	Enable or disable TCP Nagle's algorithm.
Example	**< TCP Server >** AT+SOPEN="TCP",50000 +SOPEN=0 OK +SEVENT:"CONNECT",1 AT+SLIST? +SLIST:0,"TCP","0.0.0.0",0,50000 +SLIST:1,"TCP","192.168.200.2",52432,0 OK AT+STCPNODELAY? +STCPNODELAY:1,0 OK AT+STCPNODELAY=1,1 OK AT+STCPNODELAY? +STCPNODELAY:1,1 OK **< TCP Client >** AT+SOPEN="TCP","192.168.200.1",50000 +SOPEN:0 OK AT+SLIST? +SLIST:0,"TCP","192.168.200.1",50000,0 OK AT+STCPNODELAY? +STCPNODELAY:0,0 OK AT+STCPNODELAY=0,1

	OK AT+STCPNODELAY? +STCPNODELAY:0,1 OK

7.11 AT+STIMEOUT

Command	**SET** AT+STIMEOUT=”<command>”,<timeout> **GET** AT+STIMEOUT?
Response	**SET** OK **GET** +STIMEOUT:”<command>”,<timeout> … OK
Parameters	**<command>** “SOPEN”, “SSEND” **<timeout>** Timeout in seconds. (0 : no timeout)
Description	Configure the response timeout for the specified socket command. Default timeout : - SOPEN : 30 secs - SSEND : 0 (blocking mode)
Example	AT+STIMEOUT? +STIMEOUT:″SOPEN″,30 +STIMEOUT:″SSEND″,0 OK AT+STIMEOUT=″SOPEN″,60 OK AT+STIMEOUT=″SSEND″,3 OK AT+STIMEOUT? +STIMEOUT:″SOPEN″,60

	+STIMEOUT:"SSEND",3 OK

7.12 +SEVENT

Response	+SEVENT:<event>,<socket ID>[,<parameter 1>,…,<parameter N>]
Parameters	**<event>** "CONNECT",<socket ID> "CLOSE",<socket ID>,<error>,"<description>" "SEND_DONE",<socket ID>,<done> "SEND_DROP",<socket ID>,<drop> "SEND_IDLE",<socket ID>,<done>,<drop>,<wait> "SEND_EXIT",<socket ID>,<done>,<drop> "SEND_ERROR",<socket ID>,<error>,"<description>" "RECV_READY",<socket ID>,<length> "RECV_ERROR",<socket ID>,<error>,"<description>" **<socket ID>** Socket ID **<done>** The length of the sent payload. **<drop>** The length of the dropped payload. **<wait>** The length of the buffered payload. **<length>** The length of the receivable payload. **<error>** error code

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

7.13 +RXD

Response	**RX Log Level (Terse)** +RXD:<socket ID>,<actual read length> <raw bytes> **RX Log Level (Verbose)** +RXD:<socket ID>,<actual read length>,”<remote IP>”,<remote port> <raw bytes>
Parameters	**<socket ID>** The ID allocated to the socket. **<max read length>** The maximum number of bytes to read. (Max: 2048) **<actual read length>** Actual number of bytes read.

	<remote IP>,<remote port> The remote IP and port. **<raw bytes>** The received raw bytes (0x00~0xFF) payload.
Description	An event log for a received packet with payload. Upon receiving packets, +RXD event logs will automatically appear on the terminal output. Note that there will be no 'OK' message following the event log.
Example	**RX Log Level (Terse)** +RXD=0,15 ABCDE12345,.?=+ **RX Log Level (Verbose)** +RXD=0,12,"192.168.200.1",5025 HELLO,WORLD!

8 Test Application

8.1 Command Line Interface (raspi-atcmd-cli)

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

8.1.1 Source files

File	Description
common.h	Common header file
main.c	CLI related functions.
Makefile	Make file for building.
nrc-atcmd.c nrc-atcmd.h	AT command handler
nrc-hspi.c nrc-hspi.h	HSPI protocol driver ***The HSPI protocol driver is required to communicate with the ATCMD HSPI firmware. (Appendix A.2 HSPI protocol driver)**
nrc-iperf.c nrc-iperf.h	Iperf server/client
raspi-hif.c raspi-hif.h	Wrapper for user mode driver.
raspi-eirq.c	User mode driver for GPIO EIRQ.
raspi-spi.c	User mode driver for SPI.
raspi-uart.c	User mode driver for UART.
scripts/	Script files

Table 8.1 raspi-atcmd-cli source files

8.1.2 Build

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

```
$ cd $HOME
```

$ cd raspi-atcmd-cli

$ make clean

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

$ make

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

8.1.3 Run

- **Help**

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

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

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

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

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

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

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

- **SPI**

  The maximum clock frequency is 20MHz.

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

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

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


#
```

- **UART**

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

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

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

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


#
```

- **UART_HFC**

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

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

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

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


#
```

- **Examples**

  Getting the informations.

```

```

```
# AT
  SEND: AT
  RECV: OK

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

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

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

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

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

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

Connecting to an AP.

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

# AT+WSCAN
  SEND: AT+WSCAN
  RECV: +WSCAN:"8c:0f:fa:00:28:1f",906.0@4,-39,"[WPA3-SAE-CCMP][ESS]","halow_atcmd_sae"
  RECV: +WSCAN:"8c:0f:fa:00:28:11",925.0@2,-68,"[WPA3-OWE-CCMP][ESS]","halow_fota"
  RECV: +WSCAN:"8c:0f:fa:00:28:1e",903.5@1,-93,"[ESS]","halow_s1g_demo_open"
  RECV: OK

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

```
  RECV: OK

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

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

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

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

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

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

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

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

# ABCDEFGHIJKLMNOPQRSTUVWXYZ
  SEND: DATA 10

#    RECV: +RXD:0,10

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

```
  RECV: OK

# DAJFKDAJFKDAJFDKAJFAKFJDK
  SEND: DATA 25

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


# DKAJFKDAJFEKJAFKDJFADKJFAKDJFAKEJFKADJFAKEJFKAJDFKDJAFDKJFADK
  SEND: DATA 61

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


# AT
  SEND: AT
  RECV: OK

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

Closing all sockets.

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

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

# EXIT
```

8.1.4 Run with a script

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

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

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

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

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

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

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

8.1.5 Iperf

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

iperf {-h|--help}

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

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

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

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


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

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

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

- Enter AT command in the CLI application.

```
# AT+WSCAN
  SEND: AT+WSCAN
  RECV: +WSCAN:"8c:0f:fa:00:28:1f",914.0@4,-38,"[WPA3-SAE-CCMP][ESS]","halow_atcmd_sae"
  RECV: OK

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

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

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

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

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

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

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

  SEND: AT+WSCAN
  RECV: +WSCAN:"8c:0f:fa:00:28:1f",906.0@4,-39,"[WPA3-SAE-CCMP][ESS]","halow_atcmd_sae"
  RECV: OK
  SEND: AT+WDISCONN
  RECV: OK
  SEND: AT+WCONN="halow_atcmd_sae","wpa3-sae","12345678"
  RECV: OK
  SEND: AT+WCONN?
  RECV: +WCONN:"halow_atcmd_sae","8c:0f:fa:00:28:1f","wpa3-sae","","connected"
  RECV: OK
  SEND: AT+WDHCP
  RECV: +WDHCP:"192.168.200.18","255.255.255.0","192.168.200.1"
  RECV: OK


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

- **Iperf TCP Client/Server**

```
# iperf -c 192.168.200.1

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

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


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


# iperf -c 192.168.200.1 -P

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


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

```
  Sent 2911 datagrams
  Done


# iperf -c 192.168.200.1 -P -N

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


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


# iperf -s

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


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

```
      3.0 ~   4.0 sec   435.53 KBytes     3.57 Mbits/sec
      4.0 ~   5.0 sec   425.39 KBytes     3.48 Mbits/sec
      5.0 ~   6.0 sec   424.46 KBytes     3.48 Mbits/sec
      6.0 ~   7.0 sec   439.77 KBytes     3.60 Mbits/sec
      7.0 ~   8.0 sec   418.56 KBytes     3.43 Mbits/sec
      8.0 ~   9.0 sec   425.63 KBytes     3.49 Mbits/sec
      9.0 ~ 10.0 sec   416.91 KBytes     3.42 Mbits/sec
      0.0 ~ 10.0 sec    4.15 MBytes     3.49 Mbits/sec
   Done

Press ENTER to continue or type "quit" : quit


#
```

Remote Iperf TCP Server/Client

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

```
[  4]  0.0- 1.0 sec   336 KBytes  2.75 Mbits/sec
[  4]  1.0- 2.0 sec   340 KBytes  2.78 Mbits/sec
[  4]  2.0- 3.0 sec   339 KBytes  2.78 Mbits/sec
[  4]  3.0- 4.0 sec   333 KBytes  2.73 Mbits/sec
[  4]  4.0- 5.0 sec   338 KBytes  2.77 Mbits/sec
[  4]  5.0- 6.0 sec   333 KBytes  2.72 Mbits/sec
[  4]  6.0- 7.0 sec   334 KBytes  2.73 Mbits/sec
[  4]  7.0- 8.0 sec   337 KBytes  2.76 Mbits/sec
[  4]  8.0- 9.0 sec   339 KBytes  2.78 Mbits/sec
[  4]  9.0-10.0 sec   338 KBytes  2.77 Mbits/sec
[  4] 10.0-11.0 sec  15.2 KBytes   124 Kbits/sec
[  4]  0.0-11.3 sec  3.30 MBytes  2.46 Mbits/sec


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

NOTE:

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

- **Iperf UDP Client/Server**

```
# iperf -c 192.168.200.1 -u

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

```
  - send_length: 1470
  - send_time: 10
  - send_passthrough: off
  - send_done_event: 0
  - report_interval: 1


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


# iperf -c 192.168.200.1 -u -P

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


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

```
    9.0 ~ 10.0 sec    462.25 KBytes     3.84 Mbits/sec
    0.0 ~ 10.0 sec     4.58 MBytes     3.84 Mbits/sec
  Sent 3268 datagrams
  Done


# iperf -c 192.168.200.1 -u -P -N

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


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


# iperf -s -u

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


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

```
      1.0 ~  2.0 sec   490.96 KBytes   4.02 Mbits/sec   0.393 ms    0/  342 (0%)
      2.0 ~  3.0 sec   490.96 KBytes   4.02 Mbits/sec   0.276 ms    0/  342 (0%)
      3.0 ~  4.0 sec   489.52 KBytes   4.01 Mbits/sec   0.509 ms    0/  341 (0%)
      4.0 ~  5.0 sec   486.65 KBytes   3.98 Mbits/sec   0.280 ms    0/  339 (0%)
      5.0 ~  6.0 sec   486.65 KBytes   3.99 Mbits/sec   0.544 ms    0/  339 (0%)
      6.0 ~  7.0 sec   490.96 KBytes   4.02 Mbits/sec   0.454 ms    0/  342 (0%)
      7.0 ~  8.0 sec   489.52 KBytes   4.01 Mbits/sec   0.301 ms    0/  341 (0%)
      8.0 ~  9.0 sec   488.09 KBytes   3.99 Mbits/sec   0.607 ms    0/  340 (0%)
      9.0 ~ 10.0 sec   489.52 KBytes   4.01 Mbits/sec   0.807 ms    0/  341 (0%)
      0.0 ~ 10.0 sec   4.77 MBytes     4.00 Mbits/sec   0.807 ms    0/ 3403 (0%)
   Done: 3403/3403

Press ENTER to continue or type "quit" :

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

Press ENTER to continue or type "quit" :

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

Press ENTER to continue or type "quit" : quit

#
```

Remote Iperf UDP Server/Client

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


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

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

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

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

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

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

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

8.2.1 Source files

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

Table 8.2 raspi-atcmd-remote source files

8.2.2 Build

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

```
$ cd $HOME

$ cd raspi-atcmd-remote

$ make clean
```

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

$ make

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

8.2.3 Run

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

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

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

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

Examples:

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

9 Revision History

Revision No	Date	Comments	AT Command Set
1.0	08/04/2023	Initial version	v1.23.5
1.1	08/16/2023	Added commands: AT+WCTX	v1.23.6
1.2	11/29/2023	Added commands: AT+UART AT+WTXPOWER AT+WBSSMAXIDLE AT+WDEEPSLEEP AT+SSEND AT+FWUPDATE AT+FWBINDL AT+WBI AT+WLI AT+WMAXSTA Added events: FWBINDL_IDLE FWBINDL_DROP FWBINDL_DONE	v1.24.1
		Added commands: AT+WSCANSSID AT+WSOFTAPSSID AT+SRECV	v1.24.2
		Added commands: AT+WMACADDR0 AT+WMACADDR1	v1.25.0
1.3	11/22/2024	Added command: AT+XTAL?	v1.25.1
		Added command: AT+WRELAY Updated commands: AT+WMACADDR AT+WTXPOWER AT+WRXSIG AT+WMCS AT+WTSF AT+WSOFTAP AT+WSTAINFO Removed commands:	v1.26.0

		AT+WMACADDR0 AT+WMACADDR1	
		Updated commands: AT+FWUPDATE Added events: +BEVENT:”FWBINDL_FAIL”	v1.26.1
		Updated command: AT+WFOTA Added command: AT+WWPS	v1.26.2
		Added command: AT+WBGSCAN	v1.26.3
		Updated command: AT+WCONN	v1.26.3
		Added commands: AT+BOOT AT+SFUSER Updated commands: AT+STIMEOUT AT+WSOFTAP (WPA3 support) Added events: +BOOT +BEVENT:”SFUSER_IDLE” +BEVENT:”SFUSER_DROP” +BEVENT:”SFUSER_FAIL” +BEVENT:”SFUSER_DONE”	v1.26.4
		Added commands: AT+SFSYSUSER AT+WSAEPWE AT+WSTX	v1.26.5
		Updated commands: AT+WTXPOWER AT+WMCS	v1.26.6
		Added Appendix A.	
		Updated commands: AT+WCOUNTRY AT+WCONN AT+WDEEPSLEEP	v1.26.7
1.3.1	12/10/2024	Updated commands: AT+WSCAN AT+WCONN AT+WDHCP	v1.26.8

		AT+WSOFTAP AT+WRELAY	

Appendix A. HSPI Protocol Driver

The host application must have an HSPI protocol driver to communicate with the ATCMD HSPI firmware. The source files for the HSPI protocol driver can be found in the nrc7394_sdk.

nrc7394_sdk/package/atcmd/host/raspi-atcmd-cli/nrc-hspi.c
nrc7394_sdk/package/atcmd/host/raspi-atcmd-cli/nrc-hspi.h

The host application must be built with these files included. If the build succeeds without errors, the host application can call the following functions:

<table>
<tr><th colspan="2">int nrc_hspi_open (hspi_ops_t *ops, enum HSPI_EIRQ_MODE mode)</th></tr>
<tr><td>Arguments</td><td>

ops

A pointer to a structure containing function pointers for operations used by HSPI protocol driver.

```
294 typedef struct
295 {
296     /* Output formatted text (optional) */
297     int (*printf) (const char *fmt, ...);
298
299     /* Delay for a specified number of seconds (optional) */
300     void (*delay) (uint32_t seconds);
301
302     /* Perform an SPI data transfer of len bytes,              *
303      * sending data from tx_buf and receiving data into rx_buf */
304     int (*spi_transfer) (char *tx_buf, char *rx_buf, int len);
305 } hspi_ops_t;
```

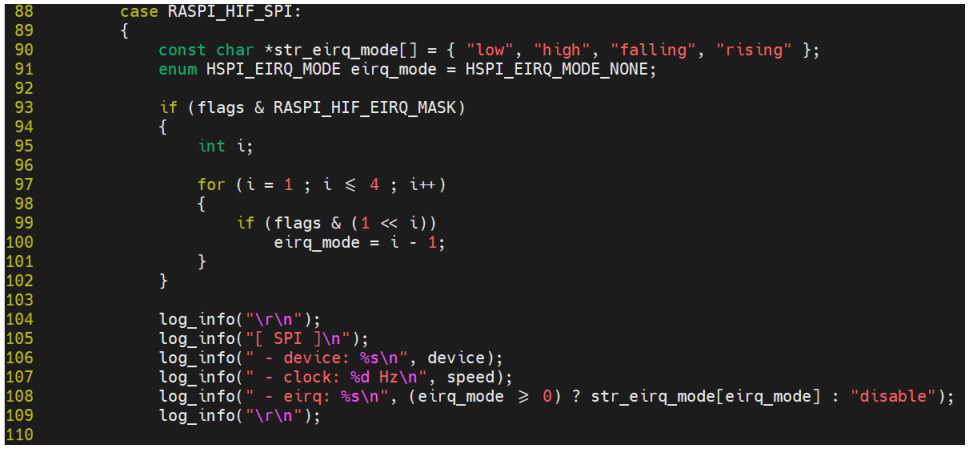
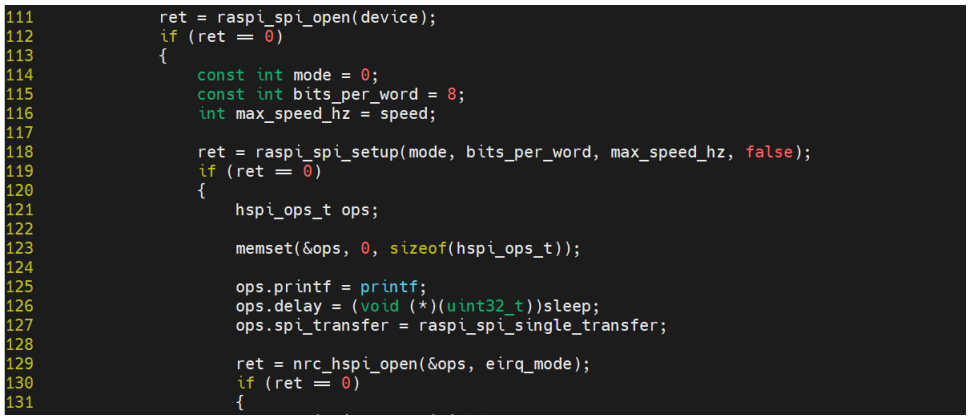
mode

The HSPI EIRQ interrupt mode.

```
234 enum HSPI_EIRQ_MODE
235 {
236     /* No interrupt */
237     HSPI_EIRQ_MODE_NONE = -1,
238
239     /* Low-level inerrupt */
240     HSPI_EIRQ_MODE_LOW = HSPI_EIRQ_LEVEL | HSPI_EIRQ_LOW,
241
242     /* High-level interrupt */
243     HSPI_EIRQ_MODE_HIGH = HSPI_EIRQ_LEVEL | HSPI_EIRQ_HIGH,
244
245     /* Faling-edge interrupt */
246     HSPI_EIRQ_MODE_FALLING = HSPI_EIRQ_EDGE | HSPI_EIRQ_LOW,
247
248     /* Rising-edge interrupt */
249     HSPI_EIRQ_MODE_RISING = HSPI_EIRQ_EDGE | HSPI_EIRQ_HIGH,
250 };
```

If enabled, the interrupt triggers when there is data to send to the host.

</td></tr>
<tr><td>Return</td><td>Returns 0 if successful, or -1 if there is an error.</td></tr>
</table>

Description	The host application must call this function before reading or writing data.
Related Code	**raspi-hif.c, raspi_hif_open(), Lines 121 – 131**   The raspi-atcmd-cli application does not enable the HSPI EIRQ interrupt by default. **raspi-spi.c, raspi_spi_single_transfer()**

```
88          case RASPI_HIF_SPI:
89          {
90              const char *str_eirq_mode[] = { "low", "high", "falling", "rising" };
91              enum HSPI_EIRQ_MODE eirq_mode = HSPI_EIRQ_MODE_NONE;
92
93              if (flags & RASPI_HIF_EIRQ_MASK)
94              {
95                  int i;
96
97                  for (i = 1 ; i <= 4 ; i++)
98                  {
99                      if (flags & (1 << i))
100                         eirq_mode = i - 1;
101                 }
102             }
103
104             log_info("\r\n");
105             log_info("[ SPI ]\n");
106             log_info(" - device: %s\n", device);
107             log_info(" - clock: %d Hz\n", speed);
108             log_info(" - eirq: %s\n", (eirq_mode >= 0) ? str_eirq_mode[eirq_mode] : "disable");
109             log_info("\r\n");
110
```

```
111             ret = raspi_spi_open(device);
112             if (ret == 0)
113             {
114                 const int mode = 0;
115                 const int bits_per_word = 8;
116                 int max_speed_hz = speed;
117
118                 ret = raspi_spi_setup(mode, bits_per_word, max_speed_hz, false);
119                 if (ret == 0)
120                 {
121                     hspi_ops_t ops;
122
123                     memset(&ops, 0, sizeof(hspi_ops_t));
124
125                     ops.printf = printf;
126                     ops.delay = (void (*)(uint32_t))sleep;
127                     ops.spi_transfer = raspi_spi_single_transfer;
128
129                     ret = nrc_hspi_open(&ops, eirq_mode);
130                     if (ret == 0)
131                     {
```

```c
int raspi_spi_single_transfer (char *tx_buf, char *rx_buf, int len)
{
    struct spi_ioc_transfer xfer;
    int ret;

    if ((!tx_buf && !rx_buf) || !len)
    {
        raspi_spi_error("%s\n", strerror(EINVAL));
        return -EINVAL;
    }

    memset(&xfer, 0, sizeof(struct spi_ioc_transfer));

#if __BITS_PER_LONG == 64
    xfer.tx_buf = (__u64)tx_buf;
    xfer.rx_buf = (__u64)rx_buf;
#else
    xfer.tx_buf = (__u32)tx_buf;
    xfer.rx_buf = (__u32)rx_buf;
#endif
    xfer.len = len;

/*  xfer.delay_usecs = 1; */
/*  xfer.cs_change = 1; */

    ret = raspi_spi_transfer(&xfer, 1);
    if (ret < 0)
        return ret;
    else if (ret != len)
        raspi_spi_error("not completed. (%d/%d)\n", ret, len);

    return 0;
}
```

void nrc_hspi_close (void)	
Arguments	None
Return	None
Description	The host application must call this function when it terminates.

Related Code	**raspi-hif.c, raspi_hif_close(), Line 196**
	`case RASPI_HIF_SPI:`
	`raspi_eirq_close();`
	`nrc_hspi_close();`
	`raspi_spi_close();`
	`break;`

int nrc_hspi_read (char *data, int len)	
Arguments	**data** A pointer to a buffer where the read data will be stored. **len** The number of bytes to read into the buffer.

Return	Returns the length of the read data, or -1 if there is an error.
Description	The host application must call this function to read data from the ATCMD HSPI firmware. If the HSPI EIRQ interrupt is not enabled, the host application needs to call this function periodically. Otherwise, call this function from the interrupt handler. NOTE: This function cannot be called simultaneously with nrc_hspi_write(). A lock mechanism, such as a mutex, is required to prevent concurrent access.
Related Code	**raspi-hif.c, raspi_hif_open(), Line 136** **raspi-hif.c, raspi_hif_read()** **main.c, raspi_cli_recv_thread()**

```
129                     ret = nrc_hspi_open(&ops, eirq_mode);
130                     if (ret == 0)
131                     {
132                         raspi_hif_mutex_init();
133
134                         hif->type = RASPI_HIF_SPI;
135                         hif->flags = flags;
136                         hif->read = nrc_hspi_read;
137                         hif->write = nrc_hspi_write;
138
```

```
228     if (hif->type == RASPI_HIF_SPI)
229         raspi_hif_mutex_lock();
230
231     ret = hif->read(buf, len);
232
233     if (hif->type == RASPI_HIF_SPI)
234         raspi_hif_mutex_unlock();
235
236     return ret;
237 }
```

```
307     while (1)
308     {
309         raspi_cli_recv_lock();
310         ret = raspi_hif_read(buf, sizeof(buf));
311         raspi_cli_recv_unlock();
312
313         if (ret > 0)
314         {
315             nrc_atcmd_recv(buf, ret);
316             continue;
317         }
318         else if (ret < 0 && ret != -EAGAIN)
319         {
320             log_error("raspi_hif_read(), %s\n", strerror(-ret));
321             sleep(1);
322         }
323
```

int nrc_hspi_write (char *data, int len)	
Arguments	**data** A pointer to a buffer containing the data to be written.

<table>
<tr><td></td><td>len

The number of bytes to write from the buffer.</td></tr>
<tr><td>Return</td><td>Returns the length of the written data, or -1 if there is an error.</td></tr>
<tr><td>Description</td><td>The host application must call this function to write data to the ATCMD HSPI firmware.

NOTE:
This function cannot be called simultaneously with nrc_hspi_read().
A lock mechanism, such as a mutex, is required to prevent concurrent access.</td></tr>
<tr><td>Related Code</td><td>raspi-hif.c, raspi_hif_open(), Line 137


```
129                     ret = nrc_hspi_open(&ops, eirq_mode);
130                     if (ret == 0)
131                     {
132                         raspi_hif_mutex_init();
133
134                         hif->type = RASPI_HIF_SPI;
135                         hif->flags = flags;
136                         hif->read = nrc_hspi_read;
137                         hif->write = nrc_hspi_write;
138
```

raspi-hif.c, raspi_hif_write()


```
256     if (hif->type == RASPI_HIF_SPI)
257         raspi_hif_mutex_lock();
258
259     ret = hif->write(buf, len);
260
261     if (hif->type == RASPI_HIF_SPI)
262         raspi_hif_mutex_unlock();
263
264     return ret;
265 }
```

</td></tr>
</table>

A.1 HSPI data transfer

Data can be transferred in single transfer mode or burst transfer mode. In single transfer mode, only one byte of data can be transferred at a time, while in burst transfer mode, multiple bytes of data can be transferred at a time.

More information about data transfer mode can be found in the document "**NRC7394 Evaluation Kit User Guide (Host Driver Porting)**".

nrc7394_sw_pkg/package/doc/UG-7394-002-Host driver porting.pdf

- Chapter 4 Host SPI
  - Chapter 4.1 Single transfer mode
  - Chapter 4.2 Burst transfer mode
- Chapter 8 HSPI Register Map
- The remaining chapters are about NRC7394 host driver for Linux.

A.2 HSPI slot memory queues

The ATCMD HSPI firmware has two slot memory queues for the HSPI controller, one for TX and one for RX. The slot memory queue for TX is required for data to be sent to the host, and the slot memory queue for RX is required for data to be received from the host. Data is written to and read from slot memory by the TX/RX DMA controller within the HSPI controller.

The ATCMD HSPI firmware writes information about queue depth and slot size to the Device Message register of the HSPI controller, which is printed to the firmware console.

```
[ATCMD] HSPI_OPEN: sw_id=0x00010400 bd_id=0x07030904
[ATCMD] HSPI_FIFO: rx=(0x104446b0, 16384), tx=(0x104486f0, 16384)
[ATCMD] HSPI_MSG:
[ATCMD]  - msg[0]: 0x2D43524E, NRC-
[ATCMD]  - msg[1]: 0x49505348, HSPI
[ATCMD]  - msg[2]: 0x00200200, slot = 32 x 512B (TXQ)
[ATCMD]  - msg[3]: 0x00200200, slot = 32 x 512B (RXQ)
```

- Queue Depth : 32
- Slot Size : 512-bytes
- Total Queue Size : 2 * (32 * 512-bytes) = 32KB

Message register			
Address	**Register**	**R/W**	**Description**
0x20	DEV_MSG_00	R	[07:00] Device message [31:24]
0x21			[07:00] Device message [23:16]
0x22			[07:00] Device message [15:08]
0x23			[07:00] Device message [07:00]
0x24	DEV_MSG_01	R	[07:00] Device message [31:24]
0x25			[07:00] Device message [23:16]
0x26			[07:00] Device message [15:08]
0x27			[07:00] Device message [07:00]
0x28	DEV_MSG_02	R	[07:00] Device message [31:24]
0x29			[07:00] Device message [23:16]
0x2A			[07:00] Device message [15:08]
0x2B			[07:00] Device message [07:00]
0x2C	DEV_MSG_03	R	[07:00] Device message [31:24]
0x2D			[07:00] Device message [23:16]
0x2E			[07:00] Device message [15:08]
0x2F			[07:00] Device message [07:00]

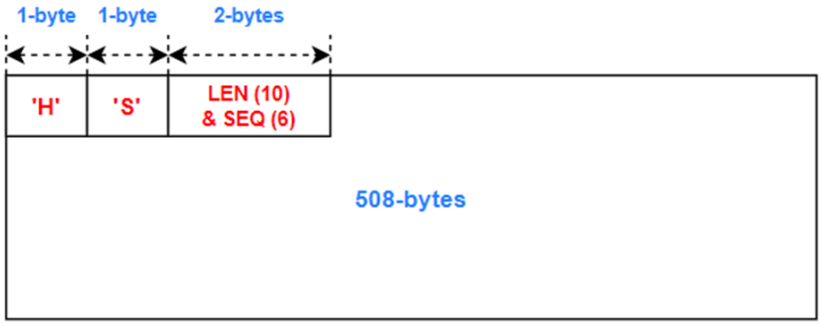
The HSPI protocol driver can determine whether the firmware is ready by reading the information written to the Device Message register. And the queue depth and slot size are stored in global variables and used by the HSPI protocol driver.

All slots have a 4-byte header.

```
279 typedef struct
280 {
281 #define HSPI_SLOT_SIZE_MAX      512
282 #define HSPI_SLOT_HDR_SIZE      4 /* 2 + 1 + 1 */
283 #define HSPI_SLOT_START_SIZE    2
284 #define HSPI_SLOT_START         "HS"
285 #define HSPI_SLOT_SEQ_MAX       0x3F
286
287     uint8_t start[2];
288     uint16_t len:10;
289     uint16_t seq:6;
290
291     uint8_t data[0];
292 } hspi_slot_t;
```

The len field has the length of valid data.
The seq field is incremented by 1 each time data is written to the slot and reset from 63 to 0.

The maximum length of data that can be written to a slot is 508 bytes.



If the data length is less than or equal to 508 bytes, one slot is used. However, if the data length is greater than 508 bytes, two or more slots are used.