ESP32 AT Instruction Set and Examples

Version 1.0

Copyright © 2017

About This Guide

This document introduces the ESP32 AT commands, explains how to use them and provides examples of several common AT commands. The document is structured as follows:

Chapter	Title	Content
Chapter 1	Overview	Provides instructions on user-defined AT commands and downloading of AT firmware.
Chapter 2	Command Description	Gives a basic description of AT commands.
Chapter 3	Basic AT Commands	Lists AT commands of basic functions.
Chapter 4	Wi-Fi AT Commands	Lists Wi-Fi-related AT commands.
Chapter 5	TCP/IP-Related AT Commands	Lists TCP/IP-related AT commands.
Chapter 6	AT Commands with Configuration Saved in the NVS Area	Lists the AT commands whose configuration is saved in the NVS area.
Chapter 7	AT Commands Examples	Gives examples of using ESP32 AT Commands.
Chapter 8	OTA Update	Introduces how to create a device on *iot.espressif.cn* and update the OTA BIN on it.
Chapter 9	Q & A	Provides information on where and how to consult questions about ESP32 AT commands.

Release Notes

Date	Version	Release notes
2017.01	V1.0	Initial release.

Table of Contents

1. Overview

This document introduces the ESP32 AT commands, and explains how to use them.

The AT command set is divided into different categories: Basic AT commands, Wi-Fi AT commands, TCP/IP AT commands, etc.

> 📖 ***Note:***
>
> *For codes related to ESP32 AT instruction set, please refer to https://github.com/espressif/esp32-at .*

1.1. User-Defined AT Commands

Please use only English letters or an underscore (_), when naming user-defined AT commands. The AT command name must NOT contain characters or numbers.

AT firmware is based on the Espressif IoT Development Framework (ESP-IDF). Espressif Systems' AT commands are provided in ***libat_core.a***, which is included in the AT BIN firmware. Examples of customized, user-defined AT commands are provided in ***esp-at***.

The structure, `at_cmd_struct`, is used to define four types of a command. Examples of implementing user-defined AT commands are provided for the users in ***/esp-at/main/at_task.c***.

To compile the AT firmware, users need to configure the environment variable ***IDF_PATH*** in the file ***esp-at***, which is accessed through the terminal.

```
export IDF_PATH=/home/genmisc/software_output/xcg/esp32_at/esp-idf
```

1.2. Downloading AT Firmware into Flash

Please refer to ***esp-at/README.md*** for instructions on how to download AT firmware into flash. Please use Espressif's official Flash Download Tools to download the firmware. Make sure you select the corresponding flash size.

Espressif's official Flash Download Tools:
http://espressif.com/en/support/download/other-tools?keys=&field_type_tid%5B%5D=13.

BIN	Address	Description
bootloader.bin	0x1000	Boot loader area
partitions_at.bin	0x8000	partitions area
phy_init_data.bin	0xF000	phy_init area, RF parameter area
esp-at.bin	0x10000	OTA1 area
blank.bin	0xFA000	Non-volatile storage (NVS) area, used for saving the configuration of Wi-Fi, AT and other parameters.
blank.bin	0x1FD000	Area for OTA configuration

> ⚠ ***Notice:***
>
> *Change the UART interface for AT commands to GPIO16 and GPIO17. Use GPIO16 as ESP32's Rx and GPIO17 as ESP32's Tx.*

2. Command Description

Each command set contains four types of AT commands.

Type	Command Format	Description
Test Command	AT+<x>=?	Queries the Set Commands' internal parameters and their range of values.
Query Command	AT+<x>?	Returns the current value of parameters.
Set Command	AT+<x>=<…>	Sets the value of user-defined parameters in commands, and runs these commands.
Execute Command	AT+<x>	Runs commands with no user-defined parameters.

Notice:

- *Not all AT commands support all four variations mentioned above.*
- *Square brackets [ ] designate the default value; it is either not required or may not appear.*
- *String values need to be included in double quotation marks, for example:* `AT+CWSAP="ESP756290","21030826",1,4`.
- *The default baud rate is 115200. The configuration of serial options is shown in Figure 2-1.*
- *AT commands have to be capitalized, and must end with* `\r\n`*, as Figure 2-2 shows.*

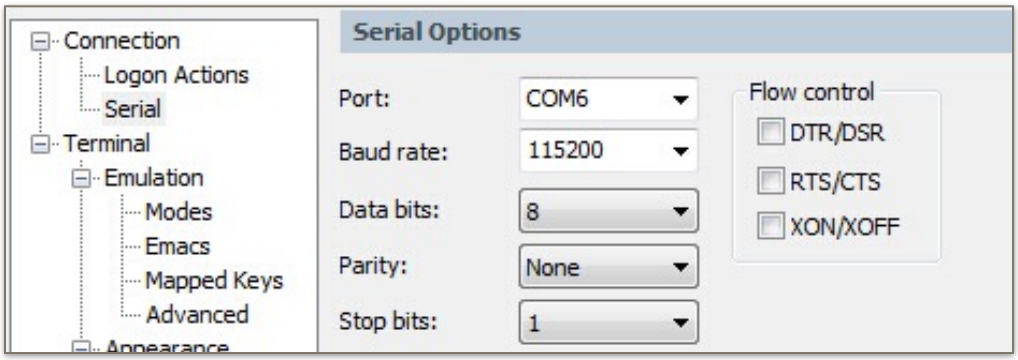


Figure 2-1. Configuration of Serial Options

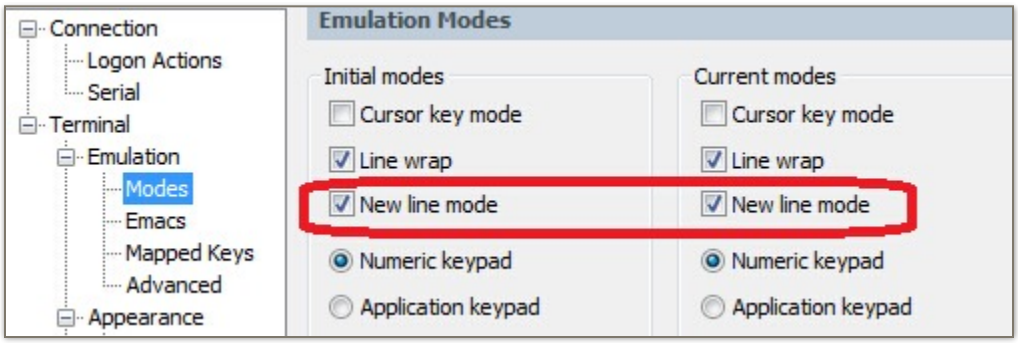


Figure 2-2. New Line Mode

3. Basic AT Commands

3.1. Overview

Commands	Description
AT	Tests AT startup.
AT+RST	Restarts a module.
AT+GMR	Checks version information.
AT+GSLP	Enters Deep-sleep mode.
ATE	Configures echoing of AT commands.
AT+RESTORE	Restores the factory default settings of the module.
AT+UART	UART configuration.
AT+UART_CUR	Current UART configuration.
AT+UART_DEF	Default UART configuration, saved in flash.
AT+SLEEP	Sets the sleep mode.
AT+SYSRAM	Checks the remaining space of RAM.

3.2. Commands

3.2.1. AT—Tests AT Startup

Execute Command	AT
Response	OK
Parameters	-

3.2.2. AT+RST—Restarts the Module

Execute Command	AT+RST
Response	OK
Parameters	-

3.2.3. AT+GMR—Checks Version Information

Execute Command	AT+GMR
Response	<AT version info> <SDK version info> <compile time> OK
Parameters	• <AT version info>: information about the AT version. • <SDK version info>: information about the SDK version. • <compile time>: the duration of time for compiling the BIN.

3.2.4. AT+GSLP—Enters Deep-sleep Mode

Set Command	AT+GSLP=<time>
Response	<time> OK
Parameters	<time>: the duration of ESP32's sleep. Unit: ms. ESP32 will wake up after Deep-sleep for as many milliseconds (ms) as <time> indicates.

3.2.5. ATE—AT Commands Echoing

Execute Command	ATE
Response	OK
Parameters	• ATE0: Switches echo off. • ATE1: Switches echo on.

3.2.6. AT+RESTORE—Restores the Factory Default Settings

Execute Command	AT+RESTORE
Response	OK
Parameters	The execution of this command will reset all parameters saved in flash, and restore the factory default settings of the module. The chip will be restarted when this command is executed.

3.2.7. AT+UART—UART Configuration

Set Command	`AT+UART=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>`
Response	`OK`

Parameters	• `<baudrate>`: UART baud rate
	• `<databits>`: data bits
	  ▸ `5`: 5-bit data
	  ▸ `6`: 6-bit data
	  ▸ `7`: 7-bit data
	  ▸ `8`: 8-bit data
	• `<stopbits>`: stop bits
	  ▸ `1`: 1-bit stop bit
	  ▸ `2`: 1.5-bit stop bit
	  ▸ `3`: 2-bit stop bit
	• `<parity>`: parity bit
	  ▸ `0`: None
	  ▸ `1`: Odd
	  ▸ `2`: Even
	• `<flow control>`: flow control
	  ▸ `0`: flow control is not enabled
	  ▸ `1`: enable RTS
	  ▸ `2`: enable CTS
	  ▸ `3`: enable both RTS and CTS

Notes	1. The configuration changes will be saved in the NVS area, and will still be valid when the chip is powered on again.
	2. The use of flow control requires the support of hardware:
	  ▸ IO15 is UART0 CTS
	  ▸ IO14 is UART0 RTS
	3. The range of baud rates supported: 80 ~ 5000000.

Example	`AT+UART=115200,8,1,0,3`

3.2.8. AT+UART_CUR—Current UART Configuration, Not Saved in Flash

Set Command	`AT+UART_CUR=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>`
Response	`OK`

Parameters	• `<baudrate>`: UART baud rate
	• `<databits>`: data bits
	  ▸ `5`: 5-bit data
	  ▸ `6`: 6-bit data
	  ▸ `7`: 7-bit data
	  ▸ `8`: 8-bit data
	• `<stopbits>`: stop bits
	  ▸ `1`: 1-bit stop bit
	  ▸ `2`: 1.5-bit stop bit
	  ▸ `3`: 2-bit stop bit
	• `<parity>`: parity bit
	  ▸ `0`: None
	  ▸ `1`: Odd
	  ▸ `2`: Even
	• `<flow control>`: flow control
	  ▸ `0`: flow control is not enabled
	  ▸ `1`: enable RTS
	  ▸ `2`: enable CTS
	  ▸ `3`: enable both RTS and CTS

Notes	1. The configuration changes will **NOT** be saved in flash.
	2. The use of flow control requires the support of hardware:
	  ▸ IO15 is UART0 CTS
	  ▸ IO14 is UART0 RTS
	3. The range of baud rates supported: 80 ~ 5000000.

Example	`AT+UART=115200,8,1,0,3`

3.2.9. AT+UART_DEF—Default UART Configuration, Saved in Flash

Set Command	`AT+UART_DEF=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>`
Response	`OK`

Parameters	• `<baudrate>`: UART baud rate
	• `<databits>`: data bits
	  ▸ `5`: 5-bit data
	  ▸ `6`: 6-bit data
	  ▸ `7`: 7-bit data
	  ▸ `8`: 8-bit data
	• `<stopbits>`: stop bits
	  ▸ `1`: 1-bit stop bit
	  ▸ `2`: 1.5-bit stop bit
	  ▸ `3`: 2-bit stop bit
	• `<parity>`: parity bit
	  ▸ `0`: None
	  ▸ `1`: Odd
	  ▸ `2`: Even
	• `<flow control>`: flow control
	  ▸ `0`: flow control is not enabled
	  ▸ `1`: enable RTS
	  ▸ `2`: enable CTS
	  ▸ `3`: enable both RTS and CTS

Notes	1. The configuration changes will be saved in the NVS area, and will still be valid when the chip is powered on again.
	2. The use of flow control requires the support of hardware:
	  ▸ IO15 is UART0 CTS
	  ▸ IO14 is UART0 RTS
	3. The range of baud rates supported: 80 ~ 5000000.

Example	`AT+UART=115200,8,1,0,3`

3.2.10. AT+SLEEP—Sets the Sleep Mode

Set Command	AT+SLEEP=<sleep mode>
Response	OK or ERROR
Parameters	<sleep mode>: ▸ 0: disable the sleep mode. ▸ 1: Modem-sleep mode.
Example	AT+SLEEP=0

3.2.11. AT+SYSRAM—Checks the Remaining Space of RAM

Query Command	AT+SYSRAM?
Response	+SYSRAM:<remaining RAM size> OK
Parameters	<remaining RAM size>: remaining space of RAM, unit: byte
Example	AT+SYSRAM? +SYSRAM:148408 OK

4. Wi-Fi AT Commands

4.1. Overview

Commands	Description
AT+CWMODE	Sets the Wi-Fi mode (STA/AP/STA+AP).
AT+CWJAP	Connects to an AP.
AT+CWLAPOPT	Sets the configuration of command AT+CWLAP.
AT+CWLAP	Lists available APs.
AT+CWQAP	Disconnects from the AP.
AT+CWSAP	Sets the configuration of the ESP32 SoftAP.
AT+CWLIF	Gets the Station IP to which the ESP32 SoftAP is connected.
AT+CWDHCP	Enables/disables DHCP.
AT+CWDHCPS	Sets the IP range of the ESP32 SoftAP DHCP server. Saves the setting in flash.
AT+CWAUTOCONN	Connects to the AP automatically on power-up.
AT+CIPSTAMAC	Sets the MAC address of ESP32 Station.
AT+CIPAPMAC	Sets the MAC address of ESP32 SoftAP.
AT+CIPSTA	Sets the IP address of ESP32 Station.
AT+CIPAP	Sets the IP address of ESP32 SoftAP.
AT+CWSTARTSMART	Starts SmartConfig.
AT+CWSTOPSMART	Stops SmartConfig.
AT+WPS	Enables the WPS function.

4.2. Commands

4.2.1. AT+CWMODE—Sets the Wi-Fi Mode (Station/SoftAP/Station+SoftAP)

Commands	Test Command: `AT+CWMODE=?`	Query Command: `AT+CWMODE?` Function: to query the current Wi-Fi mode of ESP32.	Set Command: `AT+CWMODE=<mode>` Function: to set the current Wi-Fi mode of ESP32.
Response	`+CWMODE:<mode>` `OK`	`+CWMODE:<mode>` `OK`	`OK`
Parameters	`<mode>`: ▸ 1: Station mode ▸ 2: SoftAP mode ▸ 3: SoftAP+Station mode		
Note	The configuration changes will be saved in the NVS area.		
Example	`AT+CWMODE=3`		

4.2.2. AT+CWJAP—Connects to an AP

Commands	Query Command: `AT+CWJAP?` Function: to query the AP to which the ESP32 Station is already connected.	Set Command: `AT+CWJAP=<ssid>,<pwd>[,<bssid>]` Function: to set the AP to which the ESP32 Station needs to be connected.

Response	`+CWJAP:<ssid>,<bssid>,<channel>,<rssi>`
	`OK`
		`OK`
		or
		`+CWJAP:<error code>`
		`ERROR`

Parameters	`<ssid>`: a string parameter showing the SSID of the target AP.	• `<ssid>`: the SSID of the target AP. • `<pwd>`: password, MAX: 64-byte ASCII. • `[<bssid>]`: the target AP's MAC address, used when multiple APs have the same SSID. • `<error code>`: (for reference only) ▸ 1: connection timeout. ▸ 2: wrong password. ▸ 3: cannot find the target AP. ▸ 4: connection failed. This command requires Station mode to be active. Escape character syntax is needed if SSID or password contains any special characters, such as , or " or \.
Note	The configuration changes will be saved in the NVS area.	

Examples	`AT+CWJAP="abc","0123456789"`
	For example, if the target AP's SSID is `"ab\,c"` and the password is `"0123456789"\"`, the command is as follows:
	`AT+CWJAP="ab\\\,c","0123456789\"\\"`
	If multiple APs have the same SSID as `"abc"`, the target AP can be found by BSSID:
	`AT+CWJAP="abc","0123456789","ca:d7:19:d8:a6:44"`	

4.2.3. AT+CWLAPOPT—Sets the Configuration for the Command AT+CWLAP

Set Command	`AT+CWLAPOPT=<sort_enable>,<mask>`

Response	`OK`
	or
	`ERROR`

Parameters	• `<sort_enable>`: determines whether the result of command `AT+CWLAP` will be listed according to RSSI: ▸ `0`: the result is ordered according to RSSI. ▸ `1`: the result is not ordered according to RSSI. • `<mask>`: determines the parameters shown in the result of `AT+CWLAP`; 0 means not showing the parameter corresponding to the bit, and 1 means showing it. ▸ `bit 0`: determines whether `<ecn>` will be shown in the result of AT+CWLAP. ▸ `bit 1`: determines whether `<ssid>` will be shown in the result of AT+CWLAP. ▸ `bit 2`: determines whether `<rssi>` will be shown in the result of AT+CWLAP. ▸ `bit 3`: determines whether `<mac>` will be shown in the result of AT+CWLAP. ▸ `bit 4`: determines whether `<ch>` will be shown in the result of AT+CWLAP.
Example	`AT+CWLAPOPT=1,31` The first parameter is 1, meaning that the result of the command `AT+CWLAP` will be ordered according to RSSI; The second parameter is `31`, namely `0x1F`, meaning that the corresponding bits of `<mask>` are set to 1. All parameters will be shown in the result of `AT+CWLAP`.

4.2.4. AT+CWLAP—Lists the Available APs

Commands	Set Command: `AT+CWLAP=<ssid>[,<mac>,<ch>]` Function: to query the APs with specific SSID and MAC on a specific channel.	Execute Command: `AT+CWLAP` Function: to list all available APs.

Response	`+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<ch>`
	`OK`
	or
	`ERROR`
		`+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<ch>`
		`OK`

Parameters	• `<ecn>`: encryption method. ▸ `0`: OPEN ▸ `1`: WEP ▸ `2`: WPA_PSK ▸ `3`: WPA2_PSK ▸ `4`: WPA_WPA2_PSK ▸ `5`: WPA2_Enterprise (AT can NOT connect to WPA2_Enterprise AP for now.) • `<ssid>`: string parameter, SSID of the AP. • `<rssi>`: signal strength. • `<mac>`: string parameter, MAC address of the AP.	

Examples	`AT+CWLAP="Wi-Fi","ca:d7:19:d8:a6:44",6`
	or search for APs with a designated SSID:
	`AT+CWLAP="Wi-Fi"`	

4.2.5. AT+CWQAP—Disconnects from the AP

Execute Command	`AT+CWQAP`
Response	`OK`
Parameters	-

4.2.6. AT+CWSAP—Configuration of the ESP32 SoftAP

Commands	Query Command: `AT+CWSAP?` Function: to obtain the configuration parameters of the ESP32 SoftAP.	Set Command: `AT+CWSAP=<ssid>,<pwd>,<chl>,<ecn>[,<max conn>][,<ssid hidden>]` Function: to list all available APs.

Response	`+CWSAP:<ssid>,<pwd>,<chl>,<ecn>,<max conn>,<ssid hidden>`	`OK`
		or
		`ERROR`

Parameters	• `<ssid>`: string parameter, SSID of AP.
	• `<pwd>`: string parameter, length of password: 8 ~ 64 bytes ASCII.
	• `<chl>`: channel ID.
	• `<ecn>`: encryption method; WEP is not supported.
	  ▸ `0`: OPEN
	  ▸ `2`: WPA_PSK
	  ▸ `3`: WPA2_PSK
	  ▸ `4`: WPA_WPA2_PSK
	• `[<max conn>]` (optional): maximum number of Stations to which ESP32 SoftAP can be connected; within the range of [1, 10].
	• `[<ssid hidden>]` (optional):
	  ▸ `0`: SSID is broadcast. (the default setting)
	  ▸ `1`: SSID is not broadcast.
		The same as above.
		⚠ ***Notice:***
		This command is only available when SoftAP is active.

Note	The configuration changes will be saved in the NVS area.	
Example	`AT+CWSAP="ESP32","1234567890",5,3`	

4.2.7. AT+CWLIF—IP of Stations to Which the ESP32 SoftAP is Connected

Execute Command	`AT+CWLIF`
Response	`<ip addr>,<mac>` `OK`
Parameters	• `<ip addr>`: IP address of Stations to which ESP32 SoftAP is connected. • `<mac>`: MAC address of Stations to which ESP32 SoftAP is connected.
Note	This command cannot get a static IP. It only works when both DHCPs of the ESP32 SoftAP, and of the Station to which ESP32 is connected, are enabled.

4.2.8. AT+CWDHCP—Enables/Disables DHCP

Commands	Query Command: `AT+CWDHCP?`	Set Command: `AT+CWDHCP=<operate>,<mode>` Function: to enable/disable DHCP.
Response	`DHCP disabled or enabled now?`	`OK`
Parameters	• `Bit0`: ▸ `0`: Station DHCP is disabled. ▸ `1`: Station DHCP is enabled. • `Bit1`: ▸ `0`: SoftAP DHCP is disabled. ▸ `1`: SoftAP DHCP is enabled.	• `<operate>`: ▸ `0`: disable ▸ `1`: enable • `<mode>`: ▸ `Bit0`: Station DHCP ▸ `Bit1`: SoftAP DHCP
Notes	• The configuration changes will be stored in the NVS area. • This set command interacts with static-IP-related AT commands (`AT+CIPSTA`-related and `AT+CIPA`-related commands): ▸ If DHCP is enabled, static IP will be disabled; ▸ If static IP is enabled, DHCP will be disabled; ▸ Whether it is DHCP or static IP that is enabled depends on the last configuration.	

Examples	`AT+CWDHCP=1,1`
	Enable Station DHCP. If the last DHCP mode is 2, then the current DHCP mode will be 3.
	`AT+CWDHCP=0,2`
	Disable SoftAP DHCP. If the last DHCP mode is 3, then the current DHCP mode will be 1.	

4.2.9. AT+CWDHCPS—Sets the IP Address Allocated by ESP32 SoftAP DHCP (The configuration is saved in Flash.)

Commands	Query Command: `AT+CWDHCPS?`	Set Command: `AT+CWDHCPS=<enable>,<lease time>,<start IP>,<end IP>` Function: sets the IP address range of the ESP32 SoftAP DHCP server.
Response	`+CWDHCPS=<lease time>,<start IP>,<end IP>`	`OK`
Parameters	• `<enable>`: ▸ `0`: Disable the settings and use the default IP range. ▸ `1`: Enable setting the IP range, and the parameters below have to be set. • `<lease time>`: lease time, unit: minute, range [1, 2880]. • `<start IP>`: start IP of the IP range that can be obtained from ESP32 SoftAP DHCP server. • `<end IP>`: end IP of the IP range that can be obtained from ESP32 SoftAP DHCP server.	
Notes	• The configuration changes will be saved in the NVS area. • This AT command is enabled when ESP8266 runs as SoftAP, and when DHCP is enabled. The IP address should be in the same network segment as the IP address of ESP32 SoftAP.	

Examples	`AT+CWDHCPS=1,3,"192.168.4.10","192.168.4.15"`
	or
	`AT+CWDHCPS=0 //Disable the settings and use the default IP range.`	

4.2.10. AT+CWAUTOCONN—Auto-Connects to the AP or Not

Set Command	`AT+CWAUTOCONN=<enable>`
Response	`OK`
Parameters	`<enable>`: ▸ `0`: does NOT auto-connect to AP on power-up. ▸ `1`: connects to AP automatically on power-up. The ESP32 Station connects to the AP automatically on power-up by default.
Note	The configuration changes will be saved in the NVS area.
Example	`AT+CWAUTOCONN=1`

4.2.11. AT+CIPSTAMAC—Sets the MAC Address of the ESP32 Station

Commands	Query Command: `AT+CIPSTAMAC?`	Set Command: `AT+CIPSTAMAC=<mac>` Function: to set the MAC address of the ESP32 Station.
Response	`+CIPSTAMAC:<mac>` `OK`	`OK`
Parameters	`<mac>`: string parameter, MAC address of the ESP8266 Station.	
Notes	• The configuration changes will be saved in the NVS area. • The MAC address of ESP32 SoftAP is different from that of the ESP32 Station. Please make sure that you do not set the same MAC address for both of them. • Bit 0 of the ESP32 MAC address CANNOT be 1. For example, a MAC address can be “1a:…” but not “15:…”. • FF:FF:FF:FF:FF:FF and 00:00:00:00:00:00 are invalid MAC and cannot be set.	
Example	`AT+CIPSTAMAC="18:fe:35:98:d3:7b"`	

4.2.12. AT+CIPAPMAC—Sets the MAC Address of the ESP32 SoftAP

Commands	Query Command: `AT+CIPAPMAC?` Function: to obtain the MAC address of the ESP32 SoftAP.	Set Command: `AT+CIPAPMAC=<mac>` Function: to set the MAC address of the ESP32 SoftAP.
Response	`+CIPAPMAC:<mac>` `OK`	`OK`
Parameters	`<mac>`: string parameter, MAC address of ESP32 SoftAP.	
Notes	• The configuration changes will be saved in the NVS area. • The MAC address of ESP32 SoftAP is different from that of the ESP32 Station. Please make sure you do not set the same MAC address for both of them. • Bit 0 of the ESP32 MAC address CANNOT be 1. For example, a MAC address can be “18:…” but not “15:…”. • FF:FF:FF:FF:FF:FF and 00:00:00:00:00:00 are invalid MAC and cannot be set.	
Example	`AT+CIPAPMAC="1a:fe:36:97:d5:7b"`	

4.2.13. AT+CIPSTA—Sets the IP Address of the ESP32 Station

Commands	Query Command: `AT+CIPSTA?` Function: to obtain the IP address of the ESP32 SoftAP.	Set Command: `AT+CIPSTA=<ip>,[,<gateway>,<netmask>]` Function: to set the IP address of the ESP32 SoftAP.
Response	`+CIPSTA:<ip>` `OK`	`OK`
Parameters	⚠ ***Notice:*** Only when the ESP32 Station is connected to an AP can its IP address be queried.	• `<ip>`: string parameter, the IP address of the ESP32 Station. • `[<gateway>]`: gateway. • `[<netmask>]`: netmask.
Notes	• The configuration changes will be saved in the NVS area. • The set command interacts with DHCP-related AT commands (`AT+CWDHCP`-related commands): ▸ If static IP is enabled, DHCP will be disabled; ▸ If DHCP is enabled, static IP will be disabled; ▸ Whether it is DHCP or static IP that is enabled depends on the last configuration.	
Example	`AT+CIPSTA="192.168.6.100","192.168.6.1","255.255.255.0"`	

4.2.14. AT+CIPAP—Sets the IP Address of the ESP32 SoftAP

Commands	Query Command: `AT+CIPAP?` Function: to obtain the IP address of the ESP32 SoftAP.	Set Command: `AT+CIPAP=<ip>,[,<gateway>,<netmask>]` Function: to set the IP address of the ESP32 SoftAP.
Response	`+CIPAP:<ip>,<gateway>,<netmask>` `OK`	`OK`
Parameters	• `<ip>`: string parameter, the IP address of the ESP32 SoftAP. • `[<gateway>]`: gateway. • `[<netmask>]`: netmask.	
Notes	• The configuration changes will be saved in the NVS area. • Currently, ESP32 only supports class C IP addresses. • The set command interacts with DHCP-related AT commands (`AT+CWDHCP`-related commands): ▸ If static IP is enabled, DHCP will be disabled; ▸ If DHCP is enabled, static IP will be disabled; ▸ Whether it is DHCP or static IP that is enabled depends on the last configuration.	
Example	`AT+CIPAP="192.168.5.1","192.168.5.1","255.255.255.0"`	

4.2.15. AT+CWSTARTSMART—Starts SmartConfig

Commands	Execute Command: `AT+CWSTARTSMART` Function: to start SmartConfig. (The type of SmartConfig is ESP-TOUCH + AirKiss.)	Set Command: `AT+CWSTARTSMART=<type>` Function: to start SmartConfig of a designated type.
Response	`OK`	
Parameters	`<type>`: ▸ 1: ESP-TOUCH ▸ 2: AirKiss ▸ 3: ESP-TOUCH+AirKiss	
Notes	• For details on SmartConfig please see *ESP-TOUCH User Guide*. • SmartConfig is only available in the ESP32 Station mode. • The message `Smart get Wi-Fi info` means that SmartConfig has successfully acquired the AP information. ESP32 will try to connect to the target AP. • Message `Smartconfig connected Wi-Fi` is printed if the connection is successful. Use command `AT+CWSTOPSMART` to stop SmartConfig before running other commands. Please make sure that you do not execute other commands during SmartConfig.	
Example	`AT+CWMODE=1` `AT+CWSTARTSMART`	

4.2.16. AT+CWSTOPSMART—Stops SmartConfig

Execute Command	`AT+CWSTOPSMART`
Response	`OK`
Parameters	-
Note	Irrespective of whether SmartConfig succeeds or not, before executing any other AT commands, please always call `AT+CWSTOPSMART` to release the internal memory taken up by SmartConfig.
Example	`AT+CWSTOPSMART`

4.2.17. AT+WPS—Enables the WPS Function

Set Command	AT+WPS=<enable>
Response	OK or ERROR
Parameters	<enable>: ▸ 1: enable WPS/Wi-Fi Protected Setup (implemented by PBC/Push Button Configuration). ▸ 0: disable WPS (implemented by PBC).
Notes	• WPS must be used when the ESP32 Station is enabled. • WPS does not support WEP/Wired-Equivalent Privacy encryption.
Example	AT+CWMODE=1 AT+WPS=1

5. TCP/IP-Related AT Commands

5.1. Overview

Commands	Description
AT+CIPSTATUS	Gets the connection status.
AT+CIPDOMAIN	DNS function.
AT+CIPSTART	Establishes TCP connection, UDP transmission or SSL connection.
AT+CIPSEND	Sends data.
AT+CIPSENDEX	Sends data when length of data is <length>, or when \0 appears in the data.
AT+CIPCLOSE	Closes TCP/UDP/SSL connection.
AT+CIFSR	Gets the local IP address.
AT+CIPMUX	Configures the multiple connections mode.
AT+CIPSERVER	Deletes/Creates TCP server.
AT+CIPMODE	Configures the transmission mode.
AT+SAVETRANSLINK	Saves the transparent transmission link in flash.
AT+CIPSTO	Sets timeout when ESP32 runs as a TCP server.
AT+CIUPDATE	Updates the software through Wi-Fi.
AT+CIPDINFO	Shows remote IP and remote port with +IPD.
AT+CIPSNTPCFG	Configures the time domain and SNTP server.
AT+CIPSNTPTIME	Queries the SNTP time.

5.2. Commands

5.2.1. AT+CIPSTATUS—Gets the Connection Status

Execute Command	`AT+CIPSTATUS`
Response	`STATUS:<stat>` `+CIPSTATUS:<link ID>,<type>,<remote IP>,<remote port>,<local port>,<tetype>`

Parameters	• `<stat>`: status of the ESP32 Station interface.
	  ▸ `2`: The ESP32 Station is connected to an AP and its IP is obtained.
	  ▸ `3`: The ESP32 Station has created a TCP or UDP transmission.
	  ▸ `4`: The TCP or UDP transmission of ESP32 Station is disconnected.
	  ▸ `5`: The ESP32 Station does NOT connect to an AP.
	• `<link ID>`: ID of the connection (0~4), used for multiple connections.
	• `<type>`: string parameter, `"TCP"` or `"UDP"`.
	• `<remote IP>`: string parameter indicating the remote IP address.
	• `<remote port>`: the remote port number.
	• `<local port>`: ESP32 local port number.
	• `<tetype>`:
	  ▸ `0`: ESP32 runs as a client.
	  ▸ `1`: ESP32 runs as a server.

5.2.2. AT+CIPDOMAIN—DNS Function

Execute Command	`AT+CIPDOMAIN=<domain name>`
Response	`+CIPDOMAIN:<IP address>`
Parameter	`<domain name>`: the domain name.

Example	`AT+CWMODE=1 // set Station mode`
	`AT+CWJAP="SSID","password" // access to the internet`
	`AT+CIPDOMAIN="iot.espressif.cn" // DNS function`

5.2.3. AT+CIPSTART—Establishes TCP Connection, UDP Transmission or SSL Connection

Establish TCP Connection

Set Command	Single TCP connection (`AT+CIPMUX=0`): `AT+CIPSTART=<type>,<remote IP>,<remote port>[,<TCP keep alive>]`	Multiple TCP Connections (`AT+CIPMUX=1`): `AT+CIPSTART=<link ID>,<type>,<remote IP>,<remote port>[,<TCP keep alive>]`

Response	`OK`
	or
	`ERROR`	

Parameters	• `<link ID>`: ID of network connection (0~4), used for multiple connections. • `<type>`: string parameter indicating the connection type: `"TCP"` or `"UDP"`. • `<remote IP>`: string parameter indicating the remote IP address. • `<remote port>`: the remote port number. • `[<TCP keep alive>]`: detection time interval when TCP is kept alive; this function is disabled by default. ▸ `0`: disable TCP keep-alive. ▸ `1 ~ 7200`: detection time interval; unit: second (s).	

Examples	`AT+CIPSTART="TCP","iot.espressif.cn",8000`
	`AT+CIPSTART="TCP","192.168.101.110",1000`
	For more information please see Chapter 7: ***AT Command Examples.***	

Establish UDP Transmission

Set Command	Single connection (`AT+CIPMUX=0`): `AT+CIPSTART=<type>,<remote IP>,<remote port>[,(<UDP local port>),(<UDP mode>)]`	Multiple connections (`AT+CIPMUX=1`): `AT+CIPSTART=<link ID>,<type>,<remote IP>,<remote port>[,(<UDP local port>),(<UDP mode>)]`

Response	`OK`
	or
	`ERROR`
	If TCP is already connected, the response is:
	`ALREADY CONNECT`	

Parameters	• <link ID>: ID of network connection (0~4), used for multiple connections. • <type>: string parameter indicating the connection type: "TCP" or "UDP". • <remote IP>: string parameter indicating the remote IP address. • <remote port>: remote port number. • [<TCP keep alive>]: detection time interval when TCP is kept alive; this function is disabled by default. ▸ 0: disable the TCP keep-alive function. ▸ 1 ~ 7200: detection time interval, unit: second (s). ⚠ ***Notice:*** *To use* <UDP mode> , <UDP local port> *must be set first.*
Example	AT+CIPSTART="UDP","192.168.101.110",1000,1002,2 For more information please see Chapter 7: ***AT Command Examples.***

Establish SSL Connection

Set Command	AT+CIPSTART=[<link ID>,]<type>,<remote IP>,<remote port>[,<TCP keep alive>]
Response	OK or ERROR If TCP is already connected, the response is: ALREADY CONNECT
Parameters	• <link ID>: ID of network connection (0~4), used for multiple connections. • <type>: string parameter indicating the connection type: "TCP" or "UDP". • <remote IP>: string parameter indicating the remote IP address. • <remote port>: the remote port number. • [<TCP keep alive>]: detection time interval when TCP is kept alive; this function is disabled by default. ▸ 0: disable the TCP keep-alive function. ▸ 1 ~ 7200: detection time interval, unit: second (s).
Notes	• ESP32 can only set one SSL connection at most. • SSL connection does not support UART-Wi-Fi passthrough mode (transparent transmission). • SSL connection needs a large amount of memory; otherwise, it may cause system reboot.
Example	AT+CIPSTART="SSL","iot.espressif.cn",8443

5.2.4. AT+CIPSEND—Sends Data

Commands	Set Command:
	1. Single connection: (+CIPMUX=0)
	`AT+CIPSEND=<length>`
	2. Multiple connections: (+CIPMUX=1)
	`AT+CIPSEND=<link ID>,<length>`
	3. Remote IP and ports can be set in UDP transmission:
	`AT+CIPSEND=[<link ID>,]<length>[,<remote IP>,<remote port>]`
	Function: to configure the data length in normal transmission mode.
		Execute Command:
		`AT+CIPSEND`
		Function: to start sending data in transparent transmission mode.

Response	Send data of designated length.
	Wrap return > after the set command. Begin receiving serial data. When the requirement of data length is met, the transmission of data starts.
	If the connection cannot be established or gets disrupted during data transmission, the system returns:
	`ERROR`
	If data is transmitted successfully, the system returns:
	`SEND OK`
		Wrap return > after executing this command.
		Enter transparent transmission, with a 20-ms interval between each packet, and a maximum of 2048 bytes per packet.
		When a single packet containing +++ is received, ESP32 returns to normal command mode. Please wait for at least one second before sending the next AT command.
		This command can only be used in transparent transmission mode which requires single connection.
		For UDP transparent transmission, the value of `<UDP mode>` has to be `0` when using `AT+CIPSTART`.

Parameters	• `<link ID>`: ID of the connection (0~4), for multiple connections. • `<length>`: data length, MAX: 2048 bytes. • `[<remote IP>]`: remote IP can be set in UDP transmission. • `[<remote port>]`: remote port can be set in UDP transmission.	-
Example	For more information please see Chapter 7: ***AT Command Examples.***	

5.2.5. AT+CIPSENDEX—Sends Data

Commands	Set Command:
	1. Single connection: (+CIPMUX=0)
	`AT+CIPSENDEX=<length>`
	2. Multiple connections: (+CIPMUX=1)
	`AT+CIPSENDEX=<link ID>,<length>`
	3. Remote IP and ports can be set in UDP transmission:
	`AT+CIPSENDEX=[<link ID>,]<length>[,<remote IP>,<remote port>]`
	Function: to configure the data length in normal transmission mode.

Response	Send data of designated length. Wrap return > after the set command. Begin receiving serial data. When the requirement of data length, determined by <length>, is met, or when \0 appears in the data, the transmission starts. If connection cannot be established or gets disconnected during transmission, the system returns: ERROR If data are successfully transmitted, the system returns: SEND OK
Parameters	• <link ID>: ID of the connection (0~4), for multiple connections. • <length>: data length, MAX: 2048 bytes. • When the requirement of data length, determined by <length>, is met, or when \0 appears, the transmission of data starts. Go back to the normal command mode and wait for the next AT command. • When sending \0, please send it as \\0.

5.2.6. AT+CIPCLOSE—Closes TCP/UDP/SSL Connection

Commands	Set Command (for multiple connections): `AT+CIPCLOSE=<link ID>` Function: to close TCP/UDP connection.	Execute Command (for single connection): `AT+CIPCLOSE`
Response	OK	
Parameters	<link ID>: ID number of connections to be closed; when ID=5, all connections will be closed.	-

5.2.7. AT+CIFSR—Gets the Local IP Address

Execute Command	`AT+CIFSR`

Response	`+CIFSR:<SoftAP IP address>`
	`+CIFSR:<Station IP address>`
	`OK`

Parameters	`<IP address>`: IP address of the ESP32 SoftAP; IP address of the ESP32 Station.
Notes	Only when the ESP32 Station is connected to an AP can the Station IP can be queried.

5.2.8. AT+CIPMUX—Enables/Disables Multiple Connections

Commands	Query Command: `AT+CIPMUX?`	Set Command: `AT+CIPMUX=<mode>` Function: to set the connection type.
Response	`+CIPMUX:<mode>` `OK`	`OK`
Parameters	`<mode>`: ▸ `0`: single connection ▸ `1`: multiple connections	
Notes	• The default mode is single connection mode. • Multiple connections can only be set when transparent transmission is disabled (`AT+CIPMODE=0`). • This mode can only be changed after all connections are disconnected. • If the TCP server is running, it must be deleted (`AT+CIPSERVER=0`) before the single connection mode is activated.	
Example	`AT+CIPMUX=1`	

5.2.9. AT+CIPSERVER—Deletes/Creates TCP Server

Set Command	`AT+CIPSERVER=<mode>[,<port>]`
Response	`OK`
Parameters	• `<mode>`: ▸ `0`: delete server. ▸ `1`: create server. • `<port>`: port number; 333 by default.
Notes	• A TCP server can only be created when multiple connections are activated (`AT+CIPMUX=1`). • A server monitor will automatically be created when the TCP server is created. • When a client is connected to the server, it will take up one connection and be assigned an ID.

5.2.10. AT+CIPMODE—Configures the Transmission Mode

Commands	Query Command: `AT+CIPMODE?` Function: to obtain information about transmission mode.	Set Command: `AT+CIPMODE=<mode>` Function: to set the transmission mode.
Response	`+CIPMODE:<mode>` `OK`	`OK`
Parameters	`<mode>`: ▸ `0`: normal transmission mode. ▸ `1`: UART-Wi-Fi passthrough mode (transparent transmission), which can only be enabled in TCP single connection mode or in UDP mode when the remote IP and port do not change.	
Notes	• The configuration changes will NOT be saved in flash. • During the UART-Wi-Fi passthrough transmission, if the TCP connection breaks, ESP32 will keep trying to reconnect until +++ is input to exit the transmission. If it is a normal TCP transmission and the TCP connection breaks, ESP32 will give a prompt and will not attempt to reconnect.	
Example	`AT+CIPMODE=1`	

5.2.11. AT+SAVETRANSLINK—Saves the Transparent Transmission Link in Flash

Save TCP Single Connection in Flash

Set Command	`AT+SAVETRANSLINK=<mode>,<remote IP or domain name>,<remote port>[,<type>,<TCP keep alive>]`

Response	`OK`
	or
	`ERROR`

Parameters	• `<mode>`: ▸ `0`: normal mode, ESP32 will NOT enter UART-Wi-Fi passthrough mode on power-up. ▸ `1`: ESP32 will enter UART-Wi-Fi passthrough mode on power-up. • `<remote IP>`: remote IP or domain name. • `<remote port>`: remote port. • `[<type>]` (optional): `TCP` or `UDP`, `TCP` by default. • `[<TCP keep alive>]` (optional): TCP is kept alive. This function is disabled by default. ▸ `0`: disables the TCP keep-alive function. ▸ `1 ~ 7200`: keep-alive detection time interval; unit: second (s).
Notes	• This command will save the UART-Wi-Fi passthrough mode and its link in the NVS area. ESP32 will enter the UART-Wi-Fi passthrough mode on any subsequent power cycles. • As long as the remote IP (or domain name) and port are valid, the configuration will be saved in flash.
Example	`AT+SAVETRANSLINK=1,"192.168.6.110",1002,"TCP"`

Save UDP Transmission in Flash

Set Command	`AT+SAVETRANSLINK=<mode>,<remote IP>,<remote port>,<type>[,<UDP local port>]`

Response	`OK`
	or
	`ERROR`

Parameters	• `<mode>`: ▸ `0`: normal mode; ESP32 will NOT enter UART-Wi-Fi passthrough mode on power-up. ▸ `1`: ESP32 enters UART-Wi-Fi passthrough mode on power-up. • `<remote IP>`: remote IP or domain name. • `<remote port>`: remote port. • `[<type>]` (optional): `UDP`, `TCP` by default. • `[<UDP local port>]` (optional): local port when UDP transparent transmission is enabled on power-up.

Notes	• This command will save the UART-Wi-Fi passthrough mode and its link in the NVS area. ESP32 will enter the UART-Wi-Fi passthrough mode on any subsequent power cycles. • As long as the remote IP (or domain name) and port are valid, the configuration will be saved in flash.
Example	`AT+SAVETRANSLINK=1,"192.168.6.110",1002,"UDP",1005`

5.2.12. AT+CIPSTO—Sets the TCP Server Timeout

Commands	Query Command: `AT+CIPSTO?` Function: to check the TCP server timeout.	Set Command: `AT+CIPSTO=<time>` Function: to set the TCP server timeout.
Response	`+CIPSTO:<time>` `OK`	`OK`
Parameter	`<time>`: TCP server timeout within the range of 0 ~ 7200s.	
Notes	• ESP32 configured as a TCP server will disconnect from the TCP client that does not communicate with it until timeout. • If `AT+CIPSTO=0`, the connection will never time out. This configuration is not recommended.	

Example	`AT+CIPMUX=1`
	`AT+CIPSERVER=1,1001`
	`AT+CIPSTO=10`	

5.2.13. AT+CIPSNTPCFG—Sets the Time Zone and the SNTP Server

Commands	Query Command: `AT+CIPSNTPCFG?`	Execute Command: `AT+CIPSNTPCFG`	Set Command: `AT+CIPSNTPCFG=<timezone>[,<SNTP server1>,<SNTP server2>,<SNTP server3>]`
Response	`+CIPSNTPCFG:<enable>,<timezone>,<SNTP server1>[,<SNTP server2>,<SNTP server3>]` `OK`	`OK`	`OK`
Parameters	• `<enable>`: ▸ 1: the SNTP server is configured. ▸ 0: the SNTP server is not configured. • `<timezone>`: time zone, range: [-11,13]. • `<SNTP server1>`: the first SNTP server. • `<SNTP server2>`: the second SNTP server. • `<SNTP server3>`: the third SNTP server.	Clear the SNTP server information.	• `<timezone>`: time zone, range: [-11,13]. • `<SNTP server1>`: the first SNTP server. • `<SNTP server2>`: the second SNTP server. • `<SNTP server3>`: the third SNTP server. If the three SNTP servers are not configured, the following default configuration is used: `"cn.ntp.org.cn"`, `"ntp.sjtu.edu.cn"`, `"us.pool.ntp.org"`.

Example	AT+CIPSNTPCFG=8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"

5.2.14. AT+CIPSNTPTIME—Queries the SNTP Time

Query Command	AT+CIPSNTPTIME?
Response	+CIPSNTPTIME:SNTP time OK
Parameters	-
Example	AT+CIPSNTPCFG=8,"cn.ntp.org.cn","ntp.sjtu.edu.cn" OK AT+CIPSNTPTIME? +CIPSNTPTIME:Mon Dec 12 02:33:32 2016 OK

5.2.15. AT+CIUPDATE—Updates the Software Through Wi-Fi

Execute Command	AT+CIUPDATE
Response	+CIPUPDATE:<n> OK
Parameters	• <n>: ▸ 1: find the server. ▸ 2: connect to server. ▸ 3: get the software version. ▸ 4: start updating.
Notes	• The speed of the upgrade is susceptible to the connectivity of the network. • ERROR will be returned if the upgrade fails due to unfavourable network conditions. Please wait for some time before retrying.
Notice	• If using Espressif's AT BIN (***/esp-idf/bin/at***), AT+CIUPDATE will download a new AT BIN from the Espressif Cloud. • If using a user-compiled AT BIN, users need to make their own AT+CIUPDATE upgrade. Espressif provides a demo as a reference for local upgrade (***/esp-idf/example/at)***. • It is suggested that users call AT+RESTORE to restore the factory default settings after upgrading the AT firmware.

5.2.16. AT+CIPDINFO—Shows the Remote IP and Port with "+IPD"

Set Command	AT+CIPDINFO=<mode>
Response	OK

Parameters	<mode>: ▸ 0: does not show the remote IP and port with "+IPD". ▸ 1: shows the remote IP and port with "+IPD".
Example	AT+CIPDINFO=1

5.2.17. +IPD—Receives Network Data

Command	Single connection: (+CIPMUX=0)+IPD,<len>[,<remote IP>,<remote port>]:<data>	multiple connections: (+CIPMUX=1)+IPD,<link ID>,<len>[,<remote IP>,<remote port>]:<data>
Parameters	The command is valid in normal command mode. When the module receives network data, it will send the data through the serial port using the +IPD command. • [<remote IP>]: remote IP, enabled by command AT+CIPDINFO=1. • [<remote port>]: remote port, enabled by command AT+CIPDINFO=1. • <link ID>: ID number of connection. • <len>: data length. • <data>: data received.	

6. AT Commands with Configuration Saved in the NVS Area

Commands	Examples
AT+UART	AT+UART=115200,8,1,0,3
AT+UART_DEF	AT+UART_DEF=115200,8,1,0,3
AT+CWDHCP	AT+CWDHCP=1,1
AT+CIPSTAMAC	AT+CIPSTAMAC="18:fe:35:98:d3:7b"
AT+CIPAPMAC	AT+CIPAPMAC="1a:fe:36:97:d5:7b"
AT+CIPSTA	AT+CIPSTA="192.168.6.100"
AT+CIPAP	AT+CIPAP="192.168.5.1"
AT+CWDHCPS	AT+CWDHCPS=1,3,"192.168.4.10","192.168.4.15"
AT+SAVETRANSLINK	AT+SAVETRANSLINK=1,"192.168.6.10",1001
AT+CWMODE	AT+CWMODE=3
AT+CWJAP	AT+CWJAP="abc","0123456789"
AT+CWSAP	AT+CWSAP="ESP32","12345678",5,3
AT+CWAUTOCONN	AT+CWAUTOCONN=1

> ⚠ ***Notice:***
>
> *NVS parameter area is 0xFA000 ~ 0x110000, and it is 88 KB in size.*

7. AT Commands Examples

Herein we introduce some examples of how to use Espressif's AT Commands.

7.1. ESP32 as a TCP Client in Single Connection

1. Set the Wi-Fi mode:

```
AT+CWMODE=3                          // SoftAP+Station mode
```

Response:

```
OK
```

2. Connect to the router:

```
AT+CWJAP="SSID","password"           // SSID and password of router
```

Response:

```
OK
```

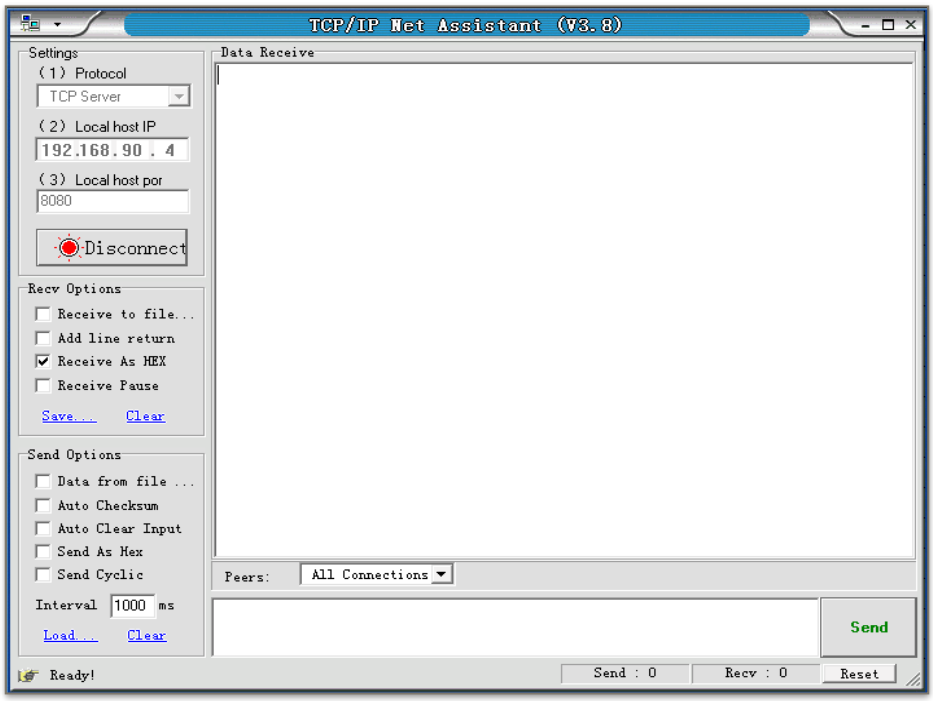
3. Query the device's IP:

```
AT+CIFSR
```

Response:

```
192.168.3.106                        // device got an IP from router
```

4. Connect the PC to the same router which ESP32 is connected to. Use a network tool on the PC to create a TCP server.



5. ESP32 is connected to the TCP server as a client:

```
AT+CIPSTART="TCP","192.168.3.116",8080   // protocol、server IP & port
```

6. Send data:

```
AT+CIPSEND=4                             // set date, such as 4 bytes
>DGFY                                    // enter the data, no CR
```

Response:

```
SEND OK
```

> ⚠ ***Notice:***
>
> *If the number of bytes sent is bigger than the size defined (n), the response will be* `busy`*. After sending the first n number of bytes,* `SEND OK` *will be returned.*

7. Receive data:

```
+IPD,n:xxxxxxxxxx                        // received n bytes, data=xxxxxxxxxx
```

7.2. UDP Transmission

UDP transmission is established via `AT+CIPSTART`. There is no such distinction between UDP server and UDP client.

1. Set the Wi-Fi mode:

```
AT+CWMODE=3                              // SoftAP+Station mode
```

Response:

```
OK
```

2. Connect to the router:

```
AT+CWJAP="SSID","password"               // SSID and password of router
```

Response:

```
OK
```

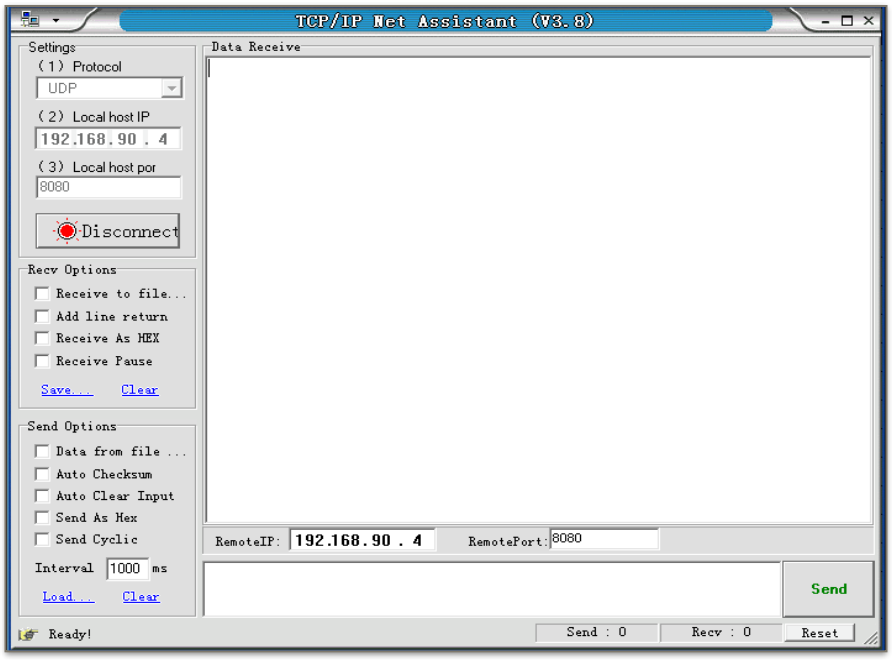
3. Query the device's IP:

```
AT+CIFSR
```

Response:

```
+CIFSR:STAIP,"192.168.101.104"           // IP address of ESP32 Station
```

4. Connect the PC to the same router which ESP32 is connected to. Use a network tool on the PC to create UDP transmission.



Below are two examples of UDP transmission:

7.2.1. UDP (with Fixed Remote IP and Port)

In UDP transmission, whether the remote IP and port are fixed or not is determined by the last parameter of `AT+CIPSTART`, namely `0`. `0` means that the remote IP and port are fixed and cannot be changed. A specific ID is given to such a connection, ensuring that the data sender and receiver will not be replaced by other devices.

1. Enable multiple connections:

```
AT+CIPMUX=1
```

Response:

```
OK
```

2. Create a UDP transmission, with the ID being 4, for example.

```
AT+CIPSTART=4,"UDP","192.168.101.110",8080,1112,0
```

Response:

```
4,CONNECT
OK
```

> 📖 ***Notes:***
>
> - `"192.168.101.110"` *and* `8080` *are the remote IP and port of UDP transmission on the remote side, i.e., the UDP configuration set by PC.*
> - `1112` *is the local port number of ESP32. Users can define this port number. The value of this parameter will be random if it is not defined beforehand.*
> - `0` *means that the remote IP and port are fixed and cannot be changed. For example, if another PC also creates a UDP entity and sends data to ESP32 port 1112, ESP32 can receive the data sent from UDP port 1112. But when data are sent using AT command* `AT+CIPSEND=4,X`*, it will still be sent to the first PC end. If parameter* `0` *is not used, the data will be sent to the new PC.*

3. Send data:

```
AT+CIPSEND=4,5              // send 5 bytes to transmission NO.4
>DGFYQ                     // enter the data, no CR
```

Response:

```
SEND OK
```

> ⚠ ***Notice:***
>
> *If the number of bytes sent is bigger than the size defined as n, the response would be* `busy`*. After sending the first n number of bytes,* `SEND OK` *will be returned.*

4. Receive data:

```
+IPD,4,n:xxxxxxxxxx          // received n bytes, data=xxxxxxxxxx
```

5. Close UDP transmission No.4:

```
AT+CIPCLOSE=4
```

Response:

```
4,CLOSED
OK
```

7.2.2. UDP (with Changeable Remote IP and Port)

1. Create a UDP transmission with the last parameter being 2.

```
AT+CIPSTART="UDP","192.168.101.110",8080,1112,2
```

Response:

```
CONNECT
OK
```

> 📖 ***Notes:***
>
> - `"192.168.101.110"` *and* `8080` *here refer to the IP and port of the remote UDP transmission terminal which is created on a PC in* ***Section 7.2.1.***
> - `1112` *is the local port of ESP32. Users can define this port. The value of this parameter will be random if it is not defined beforehand.*
> - `2` *means the means the opposite terminal of UDP transmission can be changed. The remote IP and port will be automatically changed to those of the last UDP connection to ESP32.*

2. Send data:

```
AT+CIPSEND=5                // send 5 bytes
>DGFYQ                      // enter the data, no CR
```

Response:

```
SEND OK
```

> ⚠ ***Notice:***
> *If the number of bytes sent is bigger than the size defined as n, the response would be* `busy`*. After sending the first n number of bytes,* `SEND OK` *will be returned.*

3. If you want to send data to any other UDP terminals, please designate the IP and port of the target terminal in the command.

```
AT+CIPSEND=6,"192.168.101.111",1000          // send six bytes
>abcdef                                       // enter the data, no CR
```

Response:

```
SEND OK
```

4. Receive data:

```
+IPD,n:xxxxxxxxxx                            // received n bytes, data=xxxxxxxxxx
```

5. Close UDP transmission:

```
AT+CIPCLOSE
```

Response:

```
CLOSED
OK
```

7.3. Transparent Transmission

AT Demo supports transparent transmission only when ESP32 works as a TCP client in single connection or UDP transmission.

7.3.1. ESP32 as a TCP Client in UART-Wi-Fi Passthrough (Single Connection Mode)

Here is an example of the ESP32 Station working as a TCP client in single connection mode of transparent transmission.

1. Set the Wi-Fi mode:

```
AT+CWMODE=3                                    // SoftAP+Station mode
```

Response:

```
OK
```

2. Connect to the router:

```
AT+CWJAP="SSID","password"                     // SSID and password of router
```

Response:

```
OK
```

3. Query the device's IP:

```
AT+CIFSR
```

Response:

```
192.168.101.105                                // device's IP that got from router
```

4. Connect the PC to the same router to which ESP32 is connected. Use a network tool on the PC to create a TCP server.

5. Connect the device to the TCP server as a TCP client:

```
AT+CIPSTART="TCP","192.168.101.110",8080     // protocol, server IP & port
```

Response:

```
OK
```

6. Enable the transparent transmission mode:

```
AT+CIPMODE=1
```

Response:

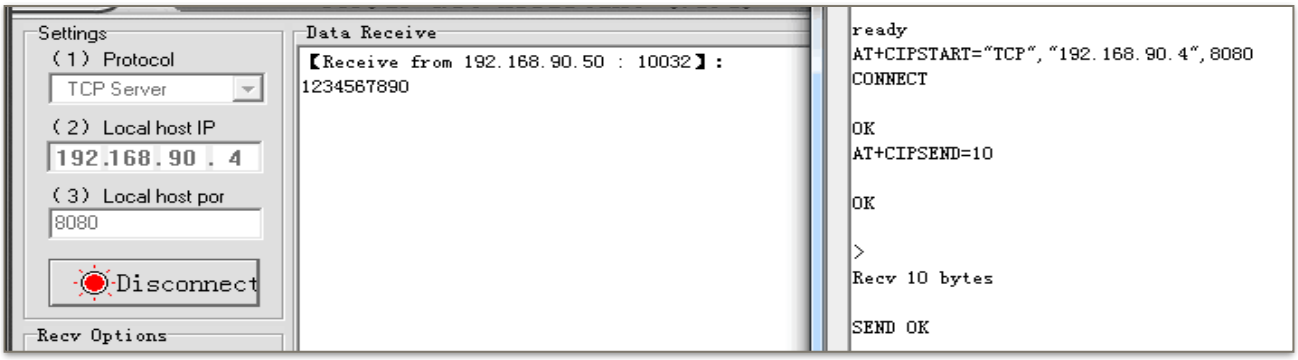
```
OK
```

7. Send data:

```
AT+CIPSEND
```

Response:

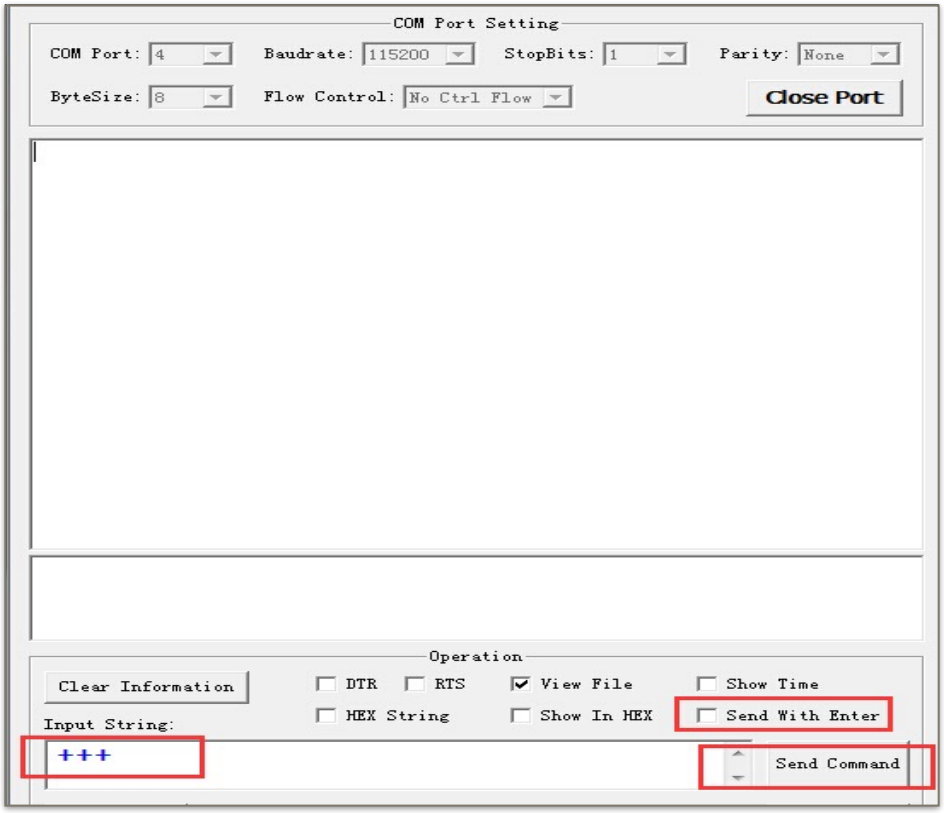
```
>                    // From now on, data received from UART will be transparent transmitted to server
```



8. Stop sending data:

If a packet of data that contains only +++ is received, the transparent transmission process will be stopped. Please wait for at least one second before sending the next AT command.

Please note that if you input +++ directly by typing, the +++, may not be recognized as three consecutive + because of the time needed for typing it. It is recommended that users deploy the following software:



- Input String: +++;
- Do NOT select Send With Enter;
- Click Send Command.

> ⚠ ***Notice:***
>
> *The aim of ending the packet with +++ is to exit transparent transmission and to accept normal AT commands, while TCP still remains connected. However, users can also deploy command* `AT+CIPSEND` *to go back into transparent transmission.*

9. Exit the transparent transmission mode:

```
AT+CIPMODE=0
```

Response:

```
OK
```

10. Close the TCP connection:

```
AT+CIPCLOSE
```

Response:

```
CLOSED
OK
```

7.3.2. UDP Transmission (UART-Wi-Fi PassthroughTransmission)

Here is an example of the ESP32 working as a SoftAP in UDP transparent transmission.

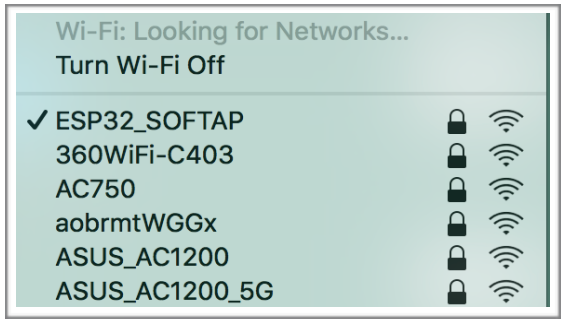
1. Set the Wi-Fi mode:

```
AT+CWMODE=3                         // SoftAP+Station mode
```

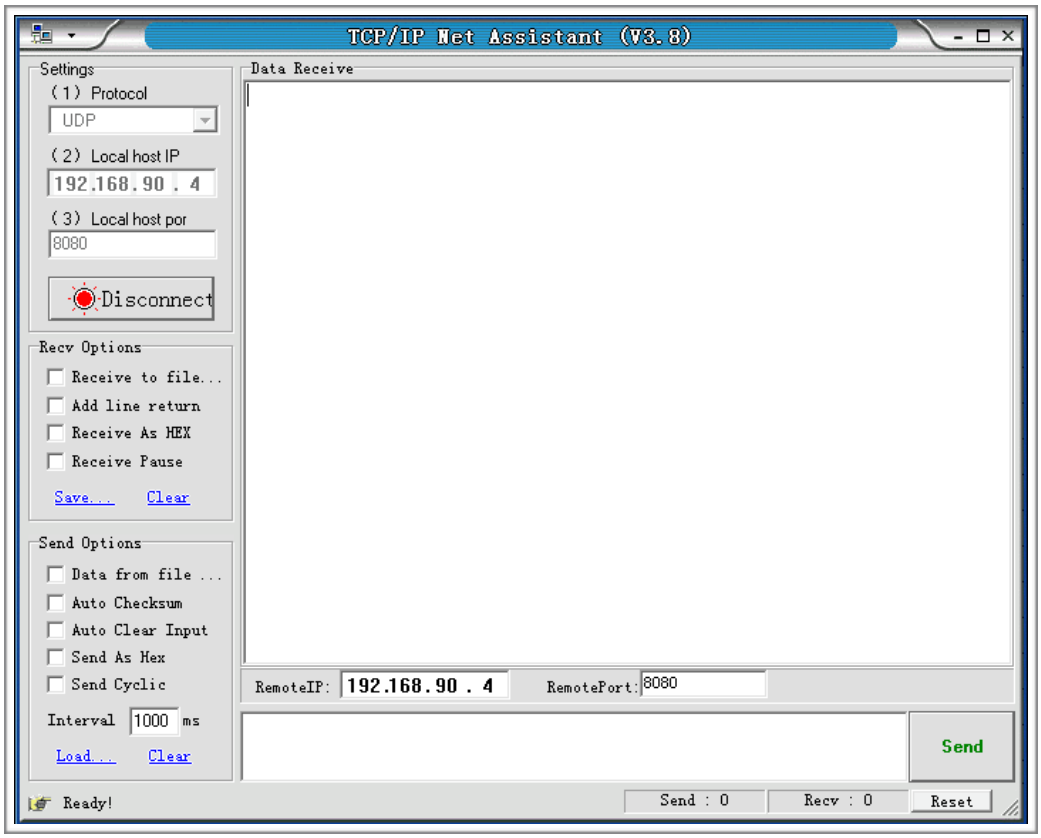
Response:

```
OK
```

2. Connect the PC to the ESP32 SoftAP:



3. Use a network tool on PC to create a UDP.



4. Create a UDP transmission between ESP32 and the PC with a fixed remote IP and port.

```
AT+CIPSTART="UDP","192.168.4.2",1001,2233,0
```

Response:

```
OK
```

5. Enable the transparent transmission mode:

```
AT+CIPMODE=1
```

Response:

```
OK
```

6. Send data:

```
AT+CIPSEND
```

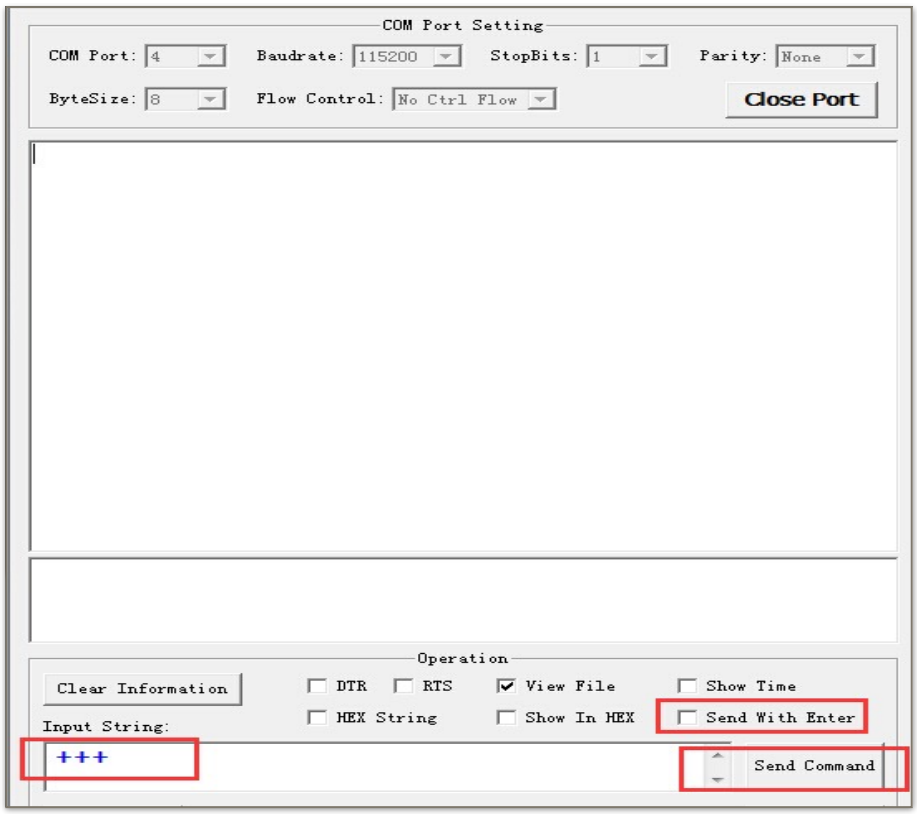
Response:

```
>                    // from now on, data received from UART will be transparent transmitted to server
```

7. Stop sending data:

If a packet of data that contains only +++ is received, the transparent transmission process will be stopped. Please wait for at least one second before sending the next AT command.

Please note that if you input +++ directly by typing, the +++ may not be recognized as three consecutive + because of the prolonged time needed for typing this. It is recommended that users deploy the following software:



- Input String: +++;
- Do NOT select Send With Enter;
- Click Send Command.

> ⚠ ***Notice:***
>
> *The aim of ending the packet with +++ is to exit transparent transmission and to accept normal AT commands, while TCP still remains connected. However, users can also use command* `AT+CIPSEND` *to go back into transparent transmission.*

9. Exit the transparent transmission mode:

```
AT+CIPMODE=0
```

Response:

```
OK
```

10. Close the UDP transmission:

```
AT+CIPCLOSE
```

Response:

```
CLOSED
OK
```

7.4. ESP32 as a TCP Server in Multiple Connections

When ESP32 works as a TCP server, multiple connections should be enabled; that is to say, there should be more than one client connecting to ESP32.

Below is an example showing how a TCP server is established when ESP32 works in the SoftAP mode. If ESP32 works as a Station, set up a server in the same way after connecting ESP32 to the router.

1. Set the Wi-Fi mode:

```
AT+CWMODE=3                         // SoftAP+Station mode
```

Response:

```
OK
```

2. Enable multiple connections:

```
AT+CIPMUX=1
```

Response:

```
OK
```

3. Set up a TCP server:

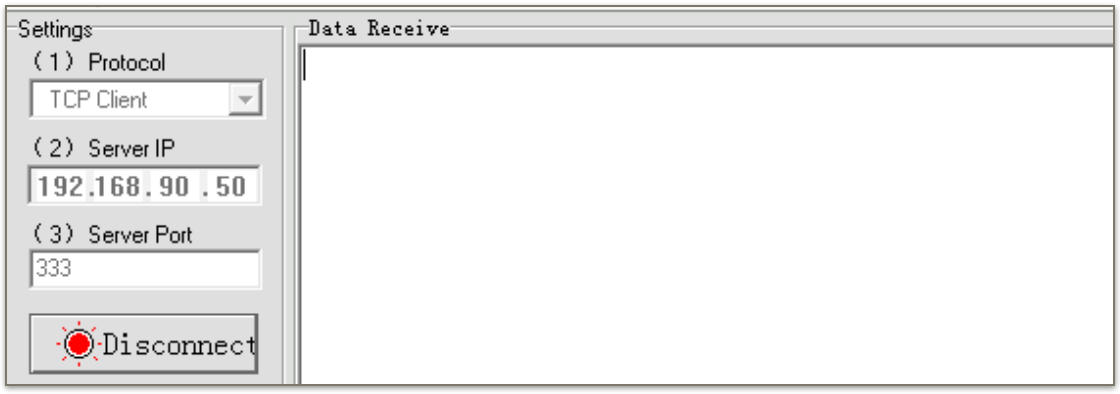
```
AT+CIPSERVER=1                      // default port = 333
```

Response:

```
OK
```

4. Connect the PC to the ESP32 SoftAP:

5. Connect the device to the PC, with the PC working as a TCP client:



> **Notice:**
>
> *When ESP32 works as a TCP server, there is a timeout mechanism. If the TCP client is connected to the ESP32 TCP server, while there is no data transmission for a period of time, the server will disconnect from the client. To avoid such a problem, please set up a data transmission cycle every two seconds.*

6. Send data:

```
                              // ID number of connection is defaulted to be 0
AT+CIPSEND=0,4                // send 4 bytes to connection NO.0
>iopd                         // enter the data, no CR
```

Response:

```
SEND OK
```

> **Notice:**
>
> *If the number of sent bytes is bigger than the size defined as n, the response will be* `busy`*. After sending the first n number of bytes,* `SEND OK` *will be returned.*

7. Receive data:

```
+IPD,0,n:xxxxxxxxxx           // received n bytes, data = xxxxxxxxxx
```

8. Close the TCP connection:

```
AT+CIPCLOSE=0                // delete NO.0 connection
```

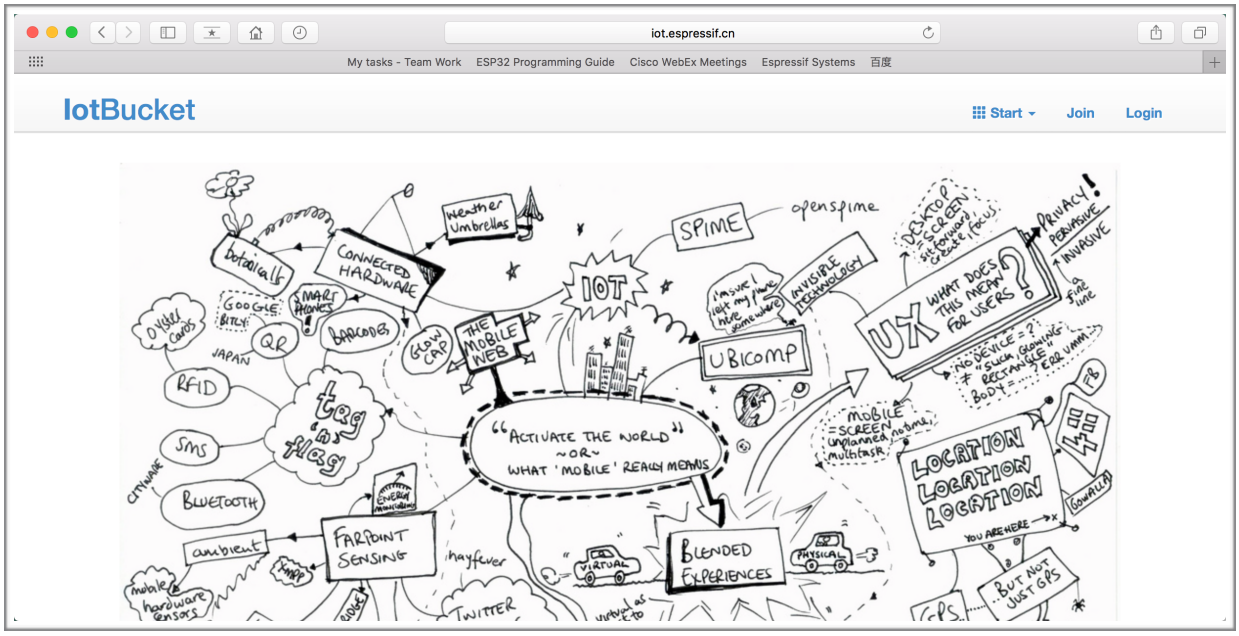
Response:

```
0,CLOSED
OK
```

8. OTA Update

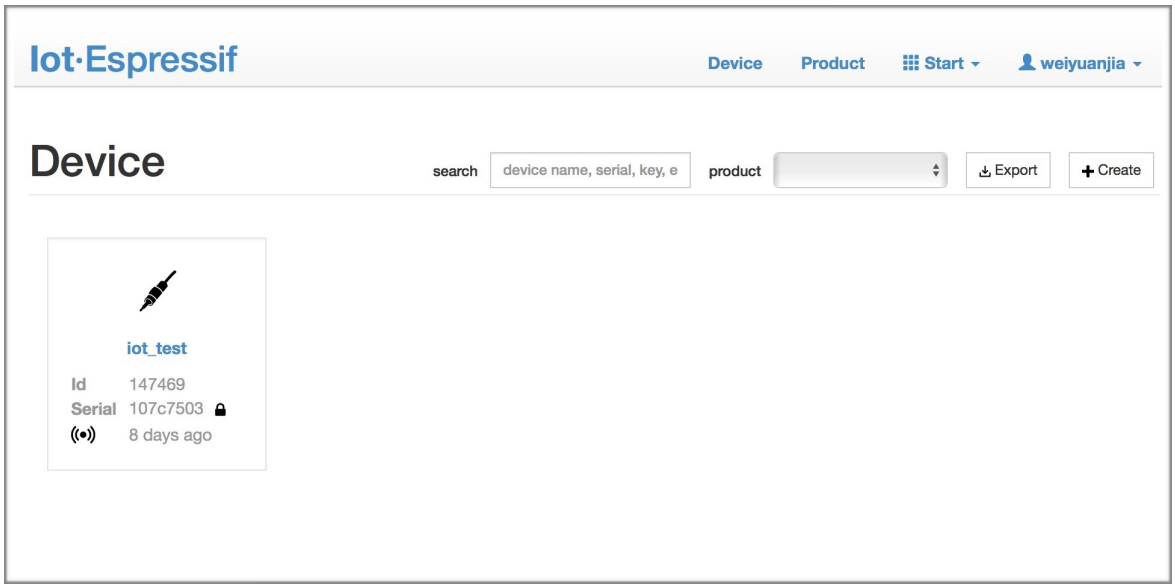
The following steps guide the users in creating a device on *iot.espressif.cn* and updating the OTA BIN on it.

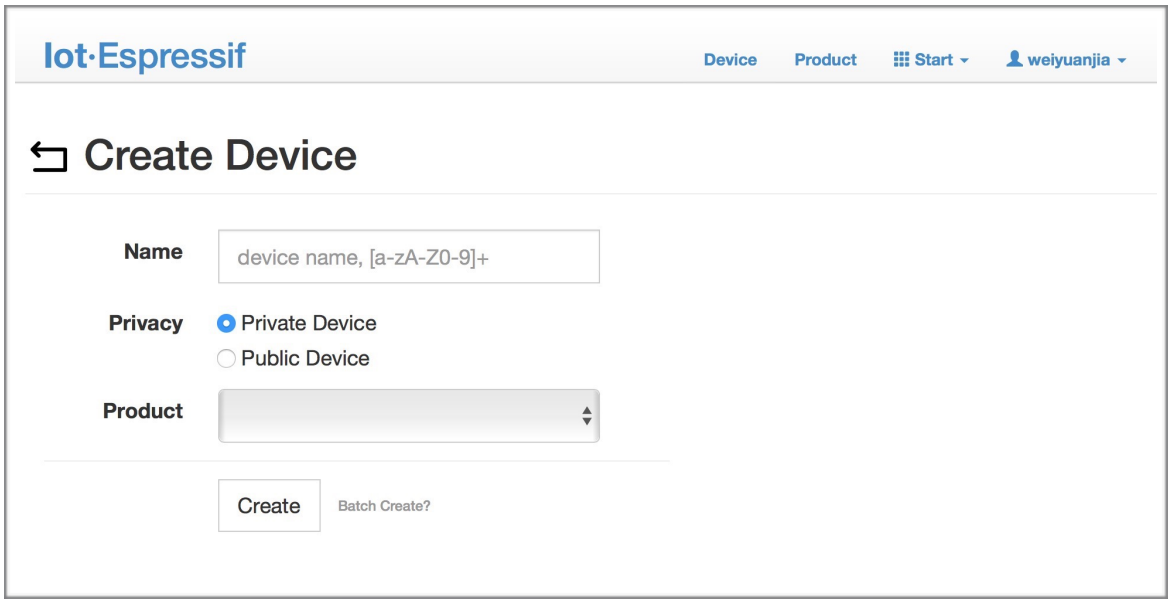
1. Open the website *iot.espressif.cn*.



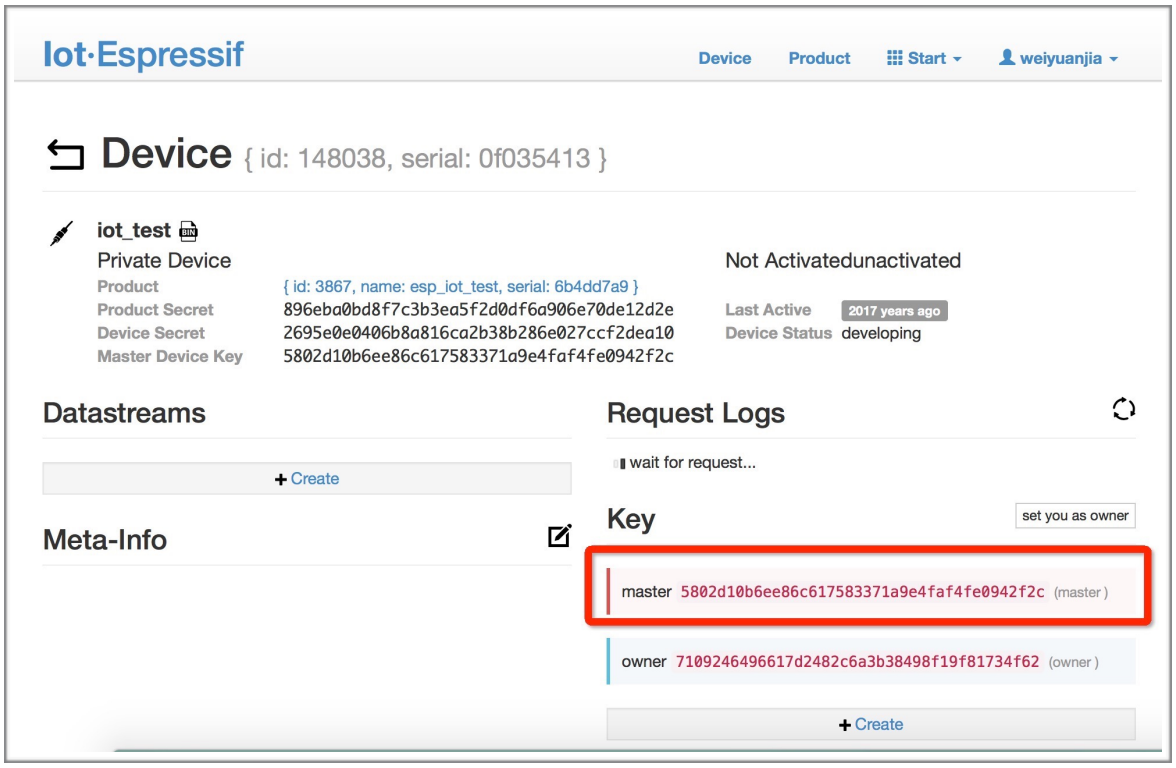
2. Click “Join” in the upper right corner of the webpage, and enter your name, email address, and password.

3. Click on “Device” in the upper right corner of the webpage, and click on “Create” to create a device.

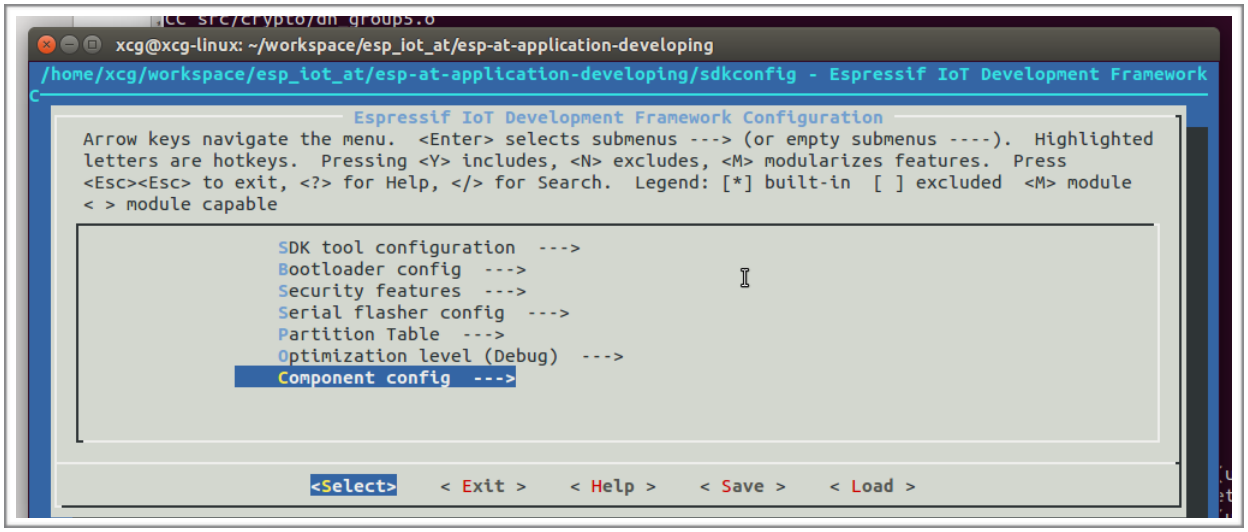


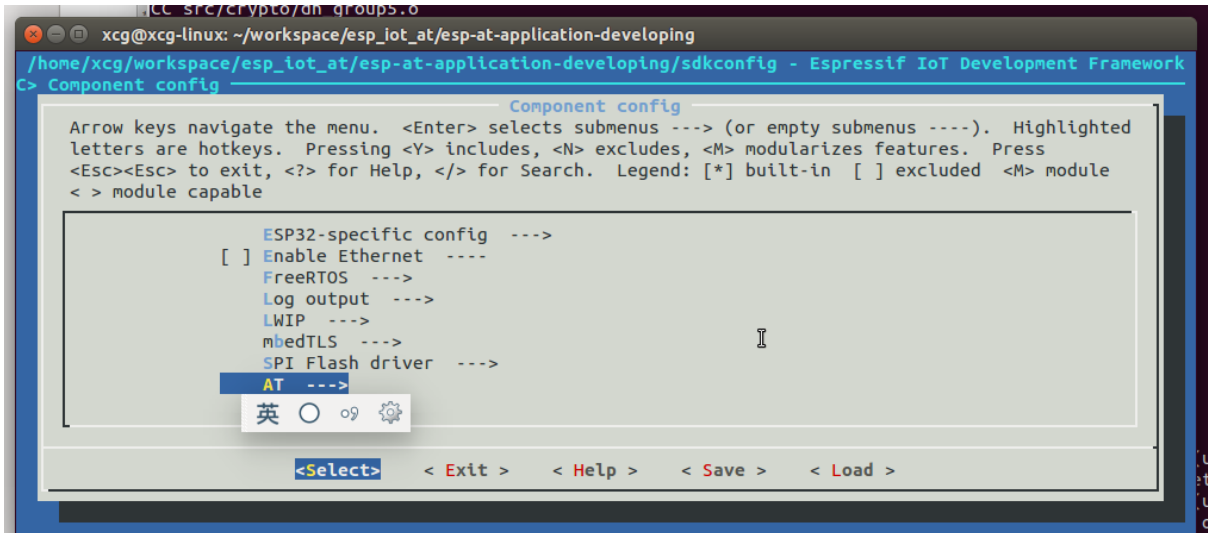


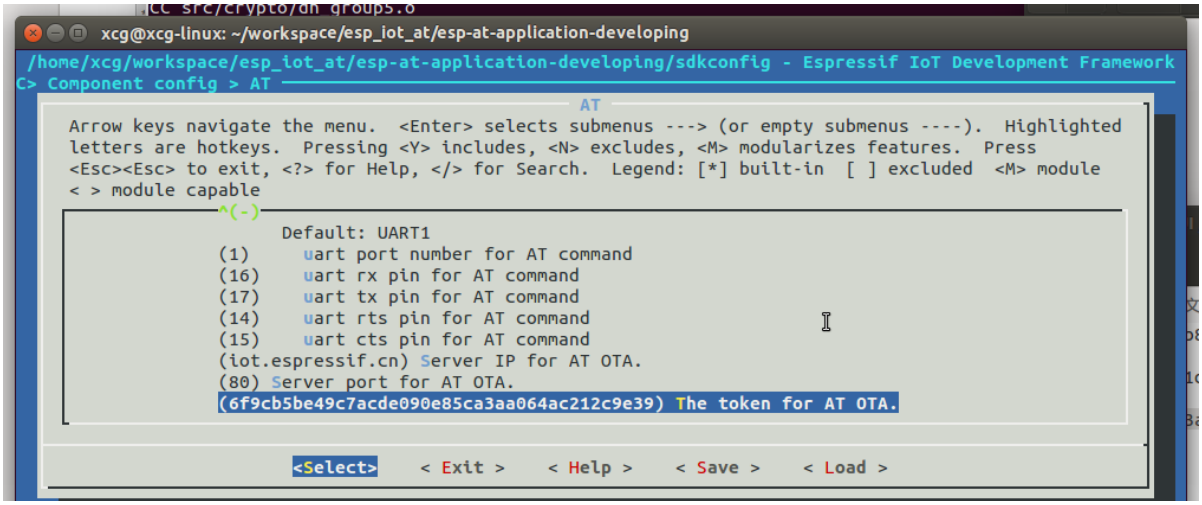
4. A key is generated when the device is successfully created, as the figure below shows.



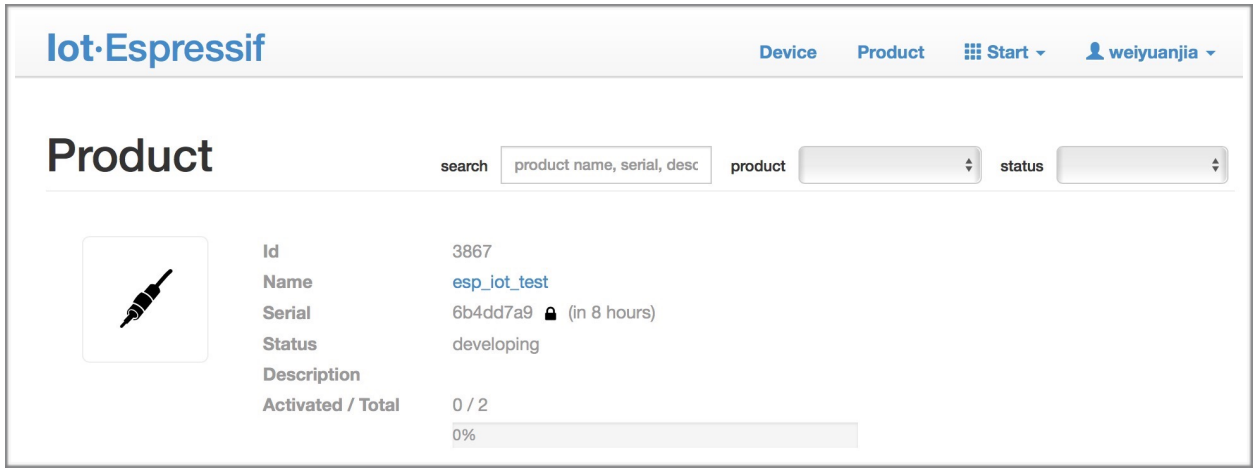
5. Use the key to compile your own OTA BIN. The process of configuring the AT OTA token key is as follows:

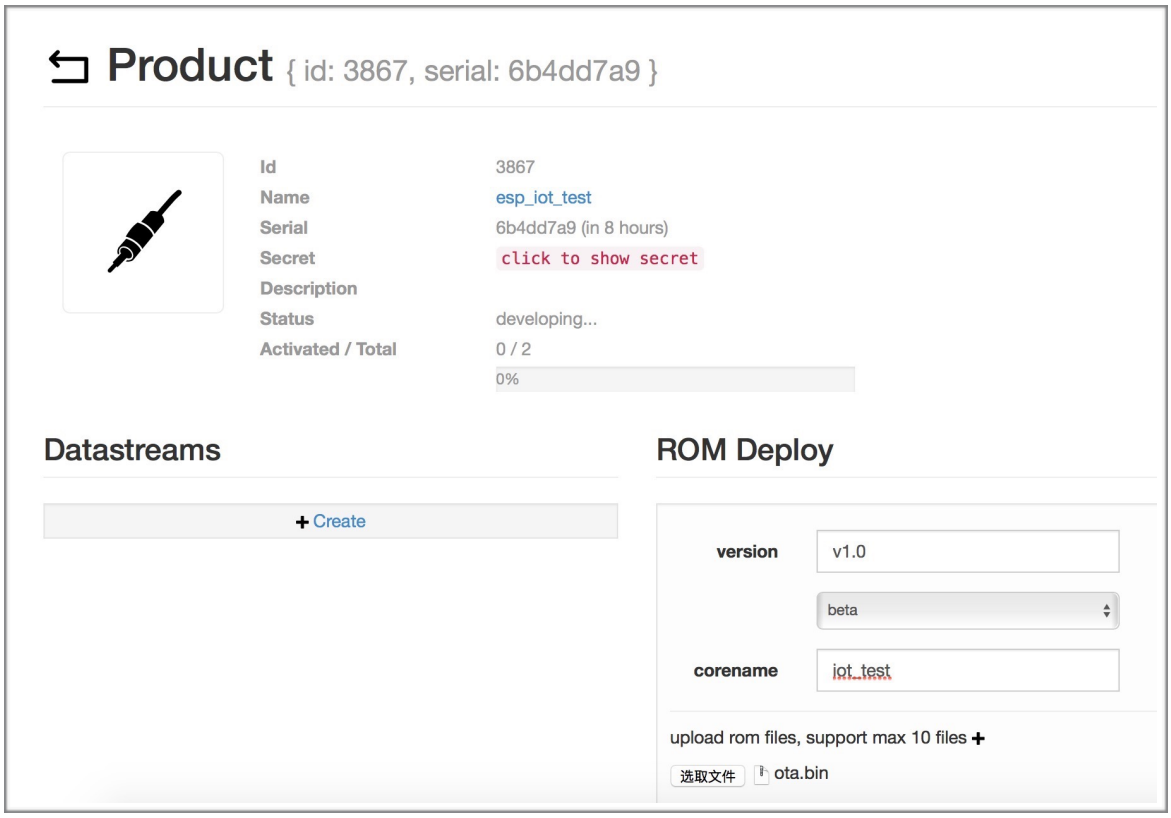




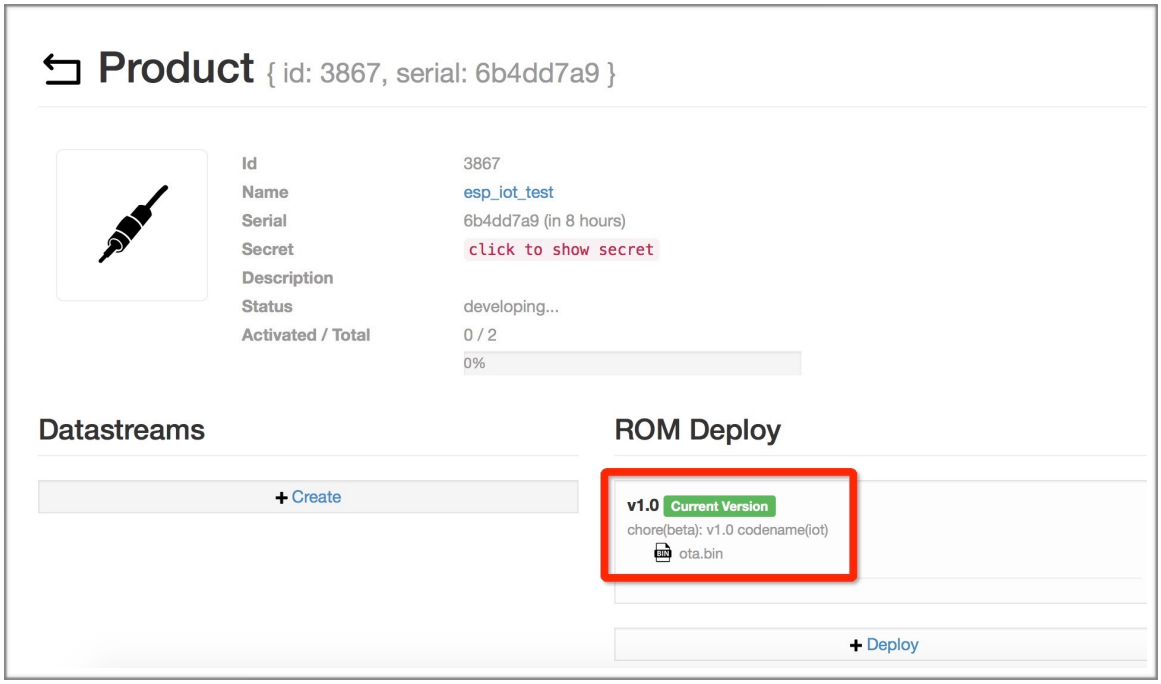


6. Click on “Product” to enter the webpage, as shown below. Click on the device created. Enter version and corename under “ROM Deploy”. Rename the BIN compiled in Step 5 as “ota.bin” and save the configuration.





7. Click on the ***ota.bin*** to save it as the current version.

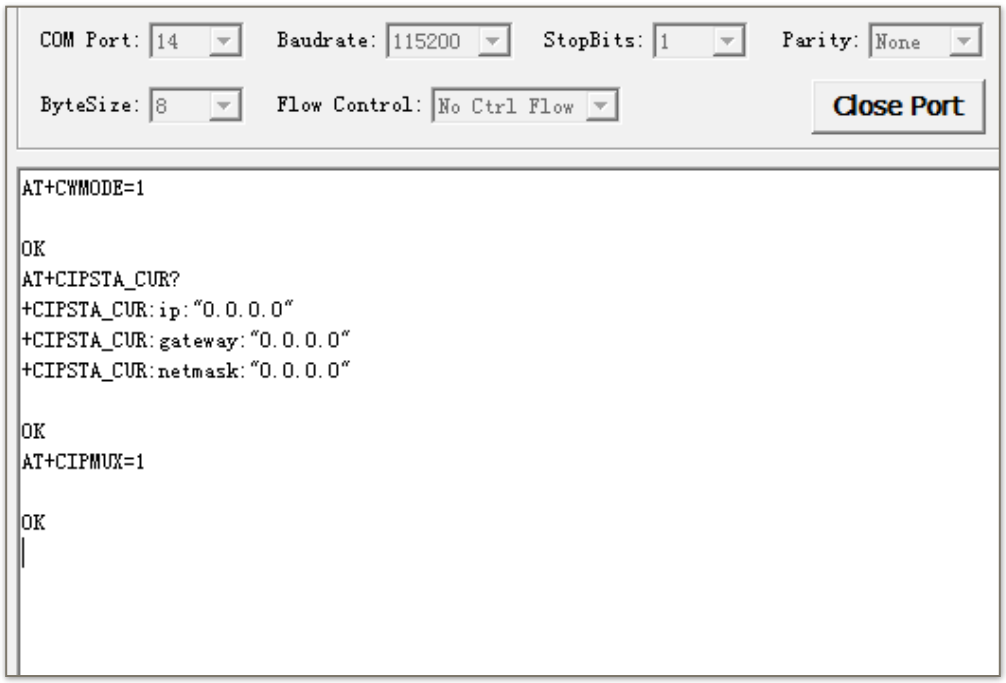


8. Run the command `AT+CIUPDATE`. If the network is connected, OTA update w.

9. Q & A

If you have any questions about the execution of AT commands, please contact us via *Espressif Technical Inquiries*. Please describe the issues that you might encounter, including any relevant details, as follows:

- AT Version information or AT Command: You can use command `AT+GMR` to acquire information on your current AT command version.
- Hardware Module information: for example, ESP-WROOM-32.
- Screenshot of the test steps, for example:



- If possible, please provide the printed log information, such as:

```
Guru Meditation Error of type StoreProhibited occurred on core   0. Exception was unhandled.
Register dump:
PC      :  40135735  PS      :  00060f30  A0      :  800f913b  A1      :  3ffd66c0
A2      :  00000000  A3      :  3ffd6828  A4      :  00000b68  A5      :  b33f0000
A6      :  b33fffff  A7      :  3ffb004c  A8      :  00000003  A9      :  3ffd66a0
A10     :  3ffd6828  A11     :  00000b69  A12     :  00060020  A13     :  3ffc2d30
A14     :  00000003  A15     :  00060023  SAR     :  00000000  EXCCAUSE:  0000001d
EXCVADDR:  00000038  LBEG    :  00000000  LEND    :  00000000  LCOUNT :  00000000
Rebooting...
```

Espressif IOT Team

www.espressif.com

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

All liability, including liability for infringement of any proprietary rights, relating to the use of information in this document, is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member logo is a trademark of the Wi-Fi Alliance. The Bluetooth logo is a registered trademark of Bluetooth SIG.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Copyright © 2017 Espressif Inc. All rights reserved.