ESP8266

AT Instruction Set

Version 3.0.2

Espressif Systems

About This Guide

This document provides AT commands list based on ESP8266_NONOS_SDK.

Release Notes

Date	Version	Release notes
2016.04	V1.5.3	First Release.
2016.05	V1.5.4	Updated Section 5.2.16 and Section 5.2.19
2016.07	V2.0.0	Added Section 3.2.11, updated Section 1.2
2017.05	V2.1.0	Updated Section 3.2, Section 4.1 and Section 5.2.
2017.08	V2.1.1	Added Appendix B.
2018.02	V2.2	Added Section 3.2.22, 3.2.23, 4.2.39, 4.2.40, 5.2.15 Updated 4.2.7, 4.2.8, and Appendix B.
2018.05	V2.2.1	Updated 4.2.10, 4.2.11 and 4.2.12
2018.05	V2.2.2	Added Section 5.2.11, 5.2.23, 5.2.24 Updated Section 3.2.10.
2018.08	V3.0	Added Section 5.2.5, 5.2.25, 5.2.26, 5.2.27 Updated Chapter 1, and Section 4.2.3, 4.2.4, 4.2.11, 4.2.12, Appendix A Remove AT+RFAUTOTRACE command.
2019.02	V3.0.1	Updated Section 5.2.5, 5.2.30, 5.2.31.
2019.06	V3.0.2	Corrected a typo in Section 4.2.5.

Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe *here*.

Certifications

Please download the product certification(s) *here*.

Table of Contents

1. Overview

ESP8266 *AT Firmware*, officially launched by Espressif, is available for download and can be used directly. Also, users may find *AT Project* that Espressif specially created for users to customize AT firmware.

This document introduces how to customize AT firmware and download AT firmware into flash. It also provides detailed information about the AT instruction set.

> ⚠ ***Notes:***
>
> - *Please make sure that correct BIN (**/ESP8266_NONOS_SDK/bin/at**) is already in the ESP8266 module before using the AT commands listed in this document.*
> - *AT firmware uses priority levels 0 and 1 of `system_os_task`, so only one task of priority 2 is allowed to be set up by the user application.*

1.1. Customize AT Firmware

1.1.1. Compiling AT project

If users want to customize AT source code, or add customized AT commands, please copy the folder *at* in examples to the root directory of the corresponding *ESP8266_NONOS_SDK* , and then enter ***ESP8266_NONOS_SDK/at*** folder to develop and compile AT project. For details, please refer to *ESP8266 Getting Started Guide*.

1.1.2. Customize AT Functions

- **OTA**:
  - The official AT firmware launched by Espressif supports the command AT+CIUPDATE by default, which helps update AT firmware to the latest version from Espressif Cloud.
  - For the customized AT firmware, users have to implement this function by themselves to update the firmware from their own cloud. Please refer to the OTA example that Espressif introduced in *at_upgrade.c*.
- **SmartConfig**:
  - The official AT firmware launched by Espressif supports the commands AT+CWSTARTSMART and AT+CWSTOPSMART.
  - If users don't need SmartConfig, you can compile AT Project and disable CONFIG_AT_SMARTCONFIG_COMMAND_ENABLE in *user_config.h* for smaller bin size and more memory.

1.1.3. Add User-Defined AT Commands

Please use only English letters when naming user-defined AT commands. The AT command name must NOT contain characters or numbers.

AT firmware is based on ESP8266_NONOS_SDK. Espressif Systems' AT commands are provided in ***libat.a***, which is included in the AT BIN firmware. Examples of customized, user-defined AT commands are provided in ***ESP8266_NONOS_SDK/example/at***.

Examples of implementing user-defined AT commands are provided in ***/ESP8266_NONOS_SDK/examples/at/user/user_main.c***. The structure, `at_funcationType`, is used to define four types of a command, for details of which please refer to the following table.

Definition	Type	Description	
at_testCmd	Test	AT Command	AT+TEST=?
		Registered Callback In Example	at_testCmdTest
		Function Design	Return the value range of parameters
		If at_testCmd is registered as NULL, there will be no testing command.	
at_queryCmd	Query	AT Command	AT+TEST?
		Registered Callback In Example	at_queryCmdTest
		Function Design	Return the current value
		If at_queryCmd is registered as NULL, there will be no Query Command.	
at_setupCmd	Set	AT Command	AT+TEST=parameter1, parameter2, …
		Registered Callback In Example	at_setupCmdTest
		Function Design	Set configuration
		If at_setupCmd is registered as NULL, there will be no setup command.	
at_exeCmd	Execute	AT Command	AT+TEST
		Registered Callback In Example	at_exeCmdTest
		Function Design	Execute an action
		If at_exeCmd is registered as NULL, there will be no execution command.	

1.2. Downloading AT Firmware into the Flash

Please refer to ***ESP8266_NONOS_SDK/bin/at/readme.txt*** 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=14.

Since ESP8266_NONOS_SDK_V3.0.0, AT_V1.7, limited by the size of the AT bin files, only 1024 KB +1024 KB flash map is supported by default.

1.2.1. 16 Mbit Flash, Map: 1024 KB + 1024 KB

Use Espressif Flash download tool and select flash size: 16 Mbit-C1.

BIN	Address	Description
blank.bin	0x1FB000	Initializes RF_CAL parameter area.
esp_init_data_default.bin	0x1FC000	Stores default RF parameter values, has to be downloaded into flash at least once. If the RF_CAL parameter area is initialized, this bin has to be downloaded too.
blank.bin	0xFE000	Initializes Flash user parameter area, more details in ***Appendix***.
blank.bin	0x1FE000	Initializes Flash system parameter area, more details in ***Appendix***.
boot.bin	0x00000	In ***/bin/at***.
user1.2048.new.5.bin	0x01000	In ***/bin/at/1024+1024***.

1.2.2. 32 Mbit Flash, Map: 1024 KB + 1024 KB

Use Espressif Flash download tool and select flash size: 32 Mbit-C1.

BIN	Address	Description
blank.bin	0x3FB000	Initializes RF_CAL parameter area
esp_init_data_default.bin	0x3FC000	Stores default RF parameter values, has to be downloaded into flash at least once. If the RF_CAL parameter area is initialized, this bin has to be downloaded too.
blank.bin	0xFE000	Initializes Flash user parameter area, more details in ***Appendix***.
blank.bin	0x3FE000	Initializes Flash system parameter area, more details in ***Appendix***.
boot.bin	0x00000	In ***/bin/at***.
user1.2048.new.5.bin	0x01000	In ***/bin/at/1024+1024***.

1.2.3. 4 Mbit Flash

With the release of ESP8266_NONOS_SDK_V2.0.0, AT_V1.3, AT firmware can use 4-Mbit flash but does not supports FOTA (upgrade AT firmware through Wi-Fi) function.

BIN	Address	Description
blank.bin	0x78000	Initializes the RF_CAL parameter area.

BIN	Address	Description
esp_init_data_default.bin	0x7C000	Stores the default RF parameter values; the BIN has to be downloaded into flash at least once. If the RF_CAL parameter area is initialized, this BIN has to be downloaded too.
blank.bin	0x7A000	Initializes the flash user parameter area; for more details please see ***Appendix***.
blank.bin	0x7E000	Initializes Flash system parameter area; for more details please see ***Appendix***.
eagle.flash.bin	0x00000	In ***/bin/at/noboot***.
eagle.irom0text.bin	0x10000	In ***/bin/at/noboot***.

1.2.4. 8 Mbit Flash

If the flash size is 8 Mbit or larger, users can use boot mode which supports AT firmware upgrade feature through Wi-Fi by command `AT+CIUPDATE`. Use Espressif Flash download tool and select flash size: 8 Mbit.

BIN	Address	Description
blank.bin	0xFB000	Initializes the RF_CAL parameter area.
esp_init_data_default.bin	0xFC000	Stores the default RF parameter values; the BIN has to be downloaded into flash at least once. If the RF_CAL parameter area is initialized, this BIN has to be downloaded too.
blank.bin	0x7E000	Initializes the flash user parameter area; for more details please see ***Appendix***.
blank.bin	0xFE000	Initializes the flash system parameter area; for more details please see ***Appendix***.
boot.bin	0x00000	In ***/bin/at***
user1.1024.new.2.bin	0x01000	In ***/bin/at/512+512***

1.2.5. 16 Mbit Flash, Map: 512 KB + 512 KB

Use Espressif Flash download tool and select flash size: 16 Mbit.

BIN	Address	Description
blank.bin	0x1FB000	Initializes RF_CAL parameter area.
esp_init_data_default.bin	0x1FC000	Stores default RF parameter values, has to be downloaded into flash at least once. If the RF_CAL parameter area is initialized, this bin has to be downloaded too.

BIN	Address	Description
blank.bin	0x7E000	Initializes Flash user parameter area, more details in ***Appendix***.
blank.bin	0x1FE000	Initializes Flash system parameter area, more details in ***Appendix***.
boot.bin	0x00000	In ***/bin/at***.
user1.1024.new.2.bin	0x01000	In ***/bin/at/512+512***.

1.2.6. 32 Mbit Flash, Map: 512 KB + 512 KB

Use Espressif Flash download tool and select flash size: 32 Mbit.

BIN	Address	Description
blank.bin	0x3FB000	Initializes RF_CAL parameter area.
esp_init_data_default.bin	0x3FC000	Stores default RF parameter values, has to be downloaded into flash at least once. If the RF_CAL parameter area is initialized, this bin has to be downloaded too.
blank.bin	0x7E000	Initializes Flash user parameter area, more details in ***Appendix***.
blank.bin	0x3FE000	Initializes Flash system parameter area, more details in ***Appendix***.
boot.bin	0x00000	In ***/bin/at***.
user1.1024.new.2.bin	0x01000	In ***/bin/at/512+512***.

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.*
- *AT commands have to be capitalized, and must end with a new line (CR LF).*

3. Basic AT Commands

3.1. Overview

Commands	Description
AT	Tests AT startup.
AT+RST	Restarts the 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_CUR	The current UART configuration.
AT+UART_DEF	The default UART configuration, saved in flash.
AT+SLEEP	Configures the sleep modes.
AT+WAKEUPGPIO	Configures a GPIO to wake ESP8266 up from Light-sleep mode.
AT+RFPOWER	Sets the maximum value of the RF TX Power.
AT+RFVDD	Sets the RF TX Power according to VDD33.
AT+SYSRAM	Checks the available RAM size.
AT+SYSADC	Checks the ADC value.
AT+SYSIOSETCFG	Sets configuration of IO pins.
AT+SYSIOGETCFG	Gets configuration of IO pins.
AT+SYSGPIODIR	Configures the direction of GPIO.
AT+SYSGPIOWRITE	Configures the GPIO output level
AT+SYSGPIOREAD	Checks the GPIO input level.
AT+SYSMSG_CUR	Sets current system messages.
AT+SYSMSG_DEF	Sets default system messages.

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 ESP8266's sleep. Unit: ms. ESP8266 will wake up after Deep-sleep for as many milliseconds (ms) as <time> indicates.
Note	A minor adjustment has to be made before the module enter the Deep-sleep mode, i.e., connecting XPD_DCDC to EXT_RSTB via a 0-ohm resistor.

3.2.5. ATE—AT Commands Echoing

Execute Command	`ATE`
Response	`OK`
Parameters	• `ATE0`: Switches echo off. • `ATE1`: Switches echo on.
Note	This command ATE is used to trigger command echo. It means that entered commands can be echoed back to the sender when ATE command is used. Two parameters are possible. The command returns `OK` in normal cases and `ERROR` when a parameter other than 0 or 1 was specified.

3.2.6. AT+RESTORE—Restores the Factory Default Settings

Execute Command	`AT+RESTORE`
Response	`OK`
Note	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_CUR—Current UART Configuration; Not Saved in the Flash

Command	Query Command: `AT+UART_CUR?`	Set Command: `AT+UART_CUR=<baudrate>,<databits>,<stop bits>,<parity>,<flow control>`
Response	`+UART_CUR:<baudrate>,<databits>,<stop bits>,<parity>,<flow control>` `OK`	`OK`
Note	Command `AT+UART_CUR?` will return the actual value of UART configuration parameters, which may have allowable errors compared with the set value because of the clock division. For example, if the UART baud rate is set as 115200, the baud rate returned by using command `AT+UART_CUR?` could be 115273.	-

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 the flash.
	2. The use of flow control requires the support of hardware:
	  ▸ MTCK is UART0 CTS
	  ▸ MTDO is UART0 RTS
	3. The range of baud rates supported: 110~115200*40.

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

3.2.8. AT+UART_DEF—Default UART Configuration; Saved in the Flash

Command	Query Command: `AT+UART_DEF?`	Set Command: `AT+UART_DEF=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>`
Response	`+UART_DEF:<baudrate>,<databits>,<stopbits>,<parity>,<flow control>` `OK`	`OK`
Parameter	• `<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 user parameter area in the flash, and will still be valid when the chip is powered on again. 2. The use of flow control requires the support of hardware: ▸ MTCK is UART0 CTS ▸ MTDO is UART0 RTS 3. The range of baud rates supported: 110~115200*40.	
Example	`AT+UART_DEF=115200,8,1,0,3`	

3.2.9. AT+SLEEP—Configures the Sleep Modes

Command	Query Command: `AT+SLEEP?`	Set Command: `AT+SLEEP=<sleep mode>`

Response	`+SLEEP:<sleep mode>`
	
	`OK`
		
		`OK`

Parameter	`<sleep mode>`: ▸ `0`: disables sleep mode ▸ `1`: Light-sleep mode ▸ `2`: Modem-sleep mode	
Notes	This command can only be used in Station mode. Modem-sleep is the default sleep mode.	
Example	`AT+SLEEP=0`	

3.2.10. AT+WAKEUPGPIO—Configures a GPIO to Wake ESP8266 up from Light-sleep Mode

Command	`AT+WAKEUPGPIO=<enable>,<trigger_GPIO>,<trigger_level>[,<awake_GPIO>,<awake_level>]`
Response	`OK`
Parameter	• `<enable>` ▸ 0: ESP8266 can NOT be woken up from light-sleep by GPIO. ▸ 1: ESP8266 can be woken up from light-sleep by GPIO. • `<trigger_GPIO>` ▸ Sets the GPIO to wake ESP8266 up; range of value: [0, 15]. • `<trigger_level>` ▸ 0: The GPIO wakes up ESP8266 on low level. ▸ 1: The GPIO wakes up ESP8266 on high level. • `[<awake_GPIO>]` ▸ Optional; this parameter is used to set a GPIO as a flag of ESP8266's being awoken form Light-sleep; range of value: [0, 15]. • `[<awake_level>]` ▸ Optional; ▸ 0: The GPIO is set to be low level after the wakeup process. ▸ 1: The GPIO is set to be high level after the wakeup process.

Notes	• Since the system needs some time to wake up from light sleep, it is suggested that wait at least 5ms before sending next AT command. • The value of <trigger_GPIO> and <awake_GPIO> in the command should not be the same. • After being woken up by <trigger_GPIO> from Light-sleep, when the ESP8266 attempts to sleep again, it will check the status of the <trigger_GPIO>: ‣ if it is still in the wakeup status, the EP8266 will enter Modem-sleep mode instead; ‣ if it is NOT in the wakeup status, the ESP8266 will enter Light-sleep mode.

Example	• Set ESP8266 to be woken from Light-sleep, when GPIO0 is on low level:
	`AT+WAKEUPGPIO=1,0,0`
	• Set ESP8266 to be woken from Light-sleep, when GPIO0 is on high level. After the waking-up, GPIO13 is set to high level.
	`AT+WAKEUPGPIO=1,0,1,13,1`
	• Disable the function that ESP8266 can be woken up from Light-sleep by a GPIO.
	`AT+WAKEUPGPIO=0`

3.2.11. AT+RFPOWER—Sets the Maximum Value of RF TX Power

Set Command	`AT+RFPOWER=<TX Power>`
Response	`OK`
Parameter	<TX Power>: the maximum value of RF TX power; range: [0, 82]; unit: 0.25 dBm.
Note	This command sets the maximum value of ESP8266 RF TX power; it is not precise. The actual value could be smaller than the set value.
Example	`AT+RFPOWER=50`

3.2.12. AT+RFVDD—Sets RF TX Power According to VDD33

Command	Query Command: `AT+RFVDD?` Function: Checks the value of ESP8266 VDD33.	Set Command: `AT+RFVDD=<VDD33>` Function: Sets the RF TX Power according to <VDD33>.	Execute Command: `AT+RFVDD` Function: Automatically sets the RF TX Power.
Response	`+RFVDD:<VDD33>` `OK`	`OK`	`OK`
Parameter	<VDD33>: power voltage of ESP8266 VDD33; unit: 1/1024 V.	<VDD33>: power voltage of ESP8266 VDD33 ; range: [1900, 3300].	-

Note	The command should only be used when TOUT pin has to be suspended, or else the returned value would be invalid.	-	TOUT pin has to be suspended in order to measure VDD33.
Example	AT+RFVDD=2800		

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

Query Command	AT+SYSRAM?
Response	+SYSRAM:<remaining RAM size> OK
Parameter	<remaining RAM size>: remaining space of RAM, unit: byte.

3.2.14. AT+SYSADC—Checks the Value of ADC

Query Command	AT+SYSADC?
Response	+SYSADC:<ADC> OK
Parameter	<ADC>: the value of ADC; unit: 1/1024V.

3.2.15. AT+SYSIOSETCFG—Configures IO Working Mode

Set Command	AT+SYSIOSETCFG=<pin>,<mode>,<pull-up>
Response	OK
Parameter	• <pin>: number of an IO pin • <mode>: the working mode of the IO pin • <pull-up> ▸ 0: disable the pull-up ▸ 1: enable the pull-up of the IO pin
Note	Please refer to *ESP8266 Pin List* for uses of AT+SYSIO-related commands.
Example	AT+SYSIOSETCFG=12,3,1 //Set GPIO12 to work as a GPIO

3.2.16. AT+SYSIOGETCFG—Checks the Working Modes of IO Pins

Set Command	AT+SYSIOGETCFG=<pin>

Response	+SYSIOGETCFG:<pin>,<mode>,<pull-up> OK
Parameter	• <pin>: number of an IO pin • <mode>: the working mode of the IO pin • <pull-up> ▸ 0: disable the pull-up ▸ 1: enable the pull-up of the IO pin
Note	Please refer to *ESP8266 Pin List* for uses of AT+SYSIO-related commands.

3.2.17. AT+SYSGPIODIR—Configures the Direction of a GPIO

Set Command	AT+SYSGPIODIR=<pin>,<dir>
Response	• If the configuration is successful, the command will return: OK • If the IO pin is not in GPIO mode, the command will return: NOT GPIO MODE! ERROR
Parameter	• <pin>: GPIO pin number • <dir>: ▸ 0: sets the GPIO as an input ▸ 1: sets the GPIO as an output
Note	Please refer to *ESP8266 Pin List* for uses of AT+SYSGPIO-related commands.
Example	AT+SYSIOSETCFG=12,3,1 //Set GPIO12 to work as a GPIO AT+SYSGPIODIR=12,0 //Set GPIO12 to work as an input

3.2.18. AT+SYSGPIOWRITE—Configures the Output Level of a GPIO

Set Command	AT+SYSGPIOWRITE=<pin>,<level>
Response	• If the configuration is successful, the command will return: OK • If the IO pin is not in output mode, the command will return: NOT OUTPUT! ERROR
Parameter	• <pin>: GPIO pin number • <level>: ▸ 0: low level ▸ 1: high level

Note	Please refer to *ESP8266 Pin List* for uses of AT+SYSGPIO-related commands.
Example	AT+SYSIOSETCFG=12,3,1 //Set GPIO12 to work as a GPIO AT+SYSGPIODIR=12,1 //Set GPIO12 to work as an output AT+SYSGPIOWRITE=12,1 //Set GPIO12 to output high level

3.2.19. AT+SYSGPIOREAD—Reads the GPIO Input Level

Set Command	AT+SYSGPIOREAD=<pin>
Response	• If the configuration is successful, the command returns: +SYSGPIOREAD:<pin>,<dir>,<level> OK • If the IO pin is not in GPIO mode, the command will return: NOT GPIO MODE! ERROR
Parameter	• <pin>: GPIO pin number • <dir>: ▸ 0: sets the GPIO as an input ▸ 1: sets the GPIO as an output • <level>: ▸ 0: low level ▸ 1: high level
Note	Please refer to *ESP8266 Pin List* for uses of AT+SYSGPIO-related commands.
Example	AT+SYSIOSETCFG=12,3,1 //Set GPIO12 to work as a GPIO AT+SYSGPIODIR=12,0 //Set GPIO12 to work as an input AT+SYSGPIOREAD=12

3.2.20. AT+SYSMSG_CUR—Set Current System Messages

Set Command	AT+SYSMSG_CUR=<n>
Response	OK

Parameter	<n>: • bit0: configure the message of quitting WiFi-UART passthrough transmission ▸ if the bit0 is 0, there is no message when quitting WiFi-UART passthrough transmission; default is 0 ▸ if the bit0 is 1, when quitting WiFi-UART passthrough transmission, it will prompt the message +QUITT // Quit transparent transmission • bit1: configure the message of establishing a network transmission ▸ if the bit1 is 0, when a network connection is established, it will prompt the message <Link_ID>,CONNECT; default is 0 ▸ if the bit1 is 1, when establishing a network connection, it will prompt the message +LINK_CONN:<status_type>,<link_id>,"UDP/TCP/SSL",<c/s>,<remote_ip>,<remote_port>,<local_port>; - <status_type> : 0 - the connection is established successfully; 1 - fail to establish the connection - <c/s> : 0 - the ESP works as a client; 1 - the ESP works as a server
Note	The configuration changes will NOT be saved in the flash.
Example	AT+SYSMSG_CUR=3

3.2.21. AT+SYSMSG_DEF—Set Default System Messages

Set Command	`AT+SYSMSG_DEF=<n>`
Response	`OK`
Parameter	`<n>`: • bit0: configure the message of quitting WiFi-UART passthrough transmission ▸ if the bit0 is 0, there is no message when quitting WiFi-UART passthrough transmission; default is 0 ▸ if the bit0 is 1, when quitting WiFi-UART passthrough transmission, it will prompt the message `+QUITT` // Quit transparent transmission • bit1: configure the message of establishing a network transmission ▸ if the bit1 is 0, when a network connection is established, it will prompt the message `<Link_ID>,CONNECT`; default is 0 ▸ if the bit1 is 1, when establishing a network connection, it will prompt the message `+LINK_CONN:<status_type>,<link_id>,"UDP/TCP/SSL",<c/s>,<remote_ip>,<remote_port>,<local_port>`; - <status_type> : 0 - the connection is established successfully; 1 - fail to establish the connection - <c/s> : 0 - the ESP works as a client; 1 - the ESP works as a server
Note	The configuration changes will be saved in the flash user parameter area.
Example	`AT+SYSMSG_DEF=3`

4. Wi-Fi AT Commands

4.1. Overview

Commands	Description
AT+CWMODE_CUR	Sets the Wi-Fi mode (Station/AP/Station+AP); configuration not saved in the flash.
AT+CWMODE_DEF	Sets the default Wi-Fi mode (Station/AP/Station+AP); configuration saved in the flash.
AT+CWJAP_CUR	Connects to an AP; configuration not saved in the flash.
AT+CWJAP_DEF	Connects to an AP; configuration saved in the flash.
AT+CWLAPOPT	Sets the configuration of command `AT+CWLAP`.
AT+CWLAP	Lists available APs.
AT+CWQAP	Disconnects from an AP.
AT+CWSAP_CUR	Sets the current configuration of the ESP8266 SoftAP; configuration not saved in the flash.
AT+CWSAP_DEF	Sets the configuration of the ESP8266 SoftAP; configuration saved in the flash.
AT+CWLIF	Gets the Station IP to which the ESP8266 SoftAP is connected.
AT+CWDHCP_CUR	Enables/Disables DHCP; configuration not saved in the flash.
AT+CWDHCP_DEF	Enable/Disable DHCP; configuration saved in the flash.
AT+CWDHCPS_CUR	Sets the IP range of the DHCP server; configuration not saved in the flash.
AT+CWDHCPS_DEF	Sets the IP range of the DHCP server; configuration saved in the flash.
AT+CWAUTOCONN	Connects to an AP automatically on power-up.
AT+CIPSTAMAC_CUR	Sets the MAC address of the ESP8266 Station; configuration not saved in the flash.
AT+CIPSTAMAC_DEF	Sets the MAC address of ESP8266 station; configuration saved in the flash.
AT+CIPAPMAC_CUR	Sets the MAC address of the ESP8266 SoftAP; configuration not saved in the flash.
AT+CIPAPMAC_DEF	Sets the MAC address of the ESP8266 SoftAP; configuration saved in the flash.
AT+CIPSTA_CUR	Sets the IP address of the ESP8266 Station; configuration not saved in the flash.
AT+CIPSTA_DEF	Sets the IP address of the ESP8266 Station; configuration saved in the flash.
AT+CIPAP_CUR	Sets the IP address of ESP8266 SoftAP; configuration not saved in the flash.

AT+CIPAP_DEF	Sets the IP address of ESP8266 SoftAP; configuration saved in the flash.
AT+CWSTARTSMART	Starts SmartConfig.
AT+CWSTOPSMART	Stops SmartConfig.
AT+CWSTARTDISCOVER	Enables the mode that ESP8266 can be found by WeChat.
AT+CWSTOPDISCOVER	Disables the mode that ESP8266 can be found by WeChat.
AT+WPS	Sets the WPS function.
AT+MDNS	Sets the MDNS function.
AT+CWHOSTNAME	Sets the host name of the ESP8266 Station.
AT+CWCOUNTRY_CUR	Sets current WiFi country code
AT+CWCOUNTRY_DEF	Sets default WiFi country code

4.2. Commands

4.2.1. AT+CWMODE_CUR—Sets the Current Wi-Fi mode; Configuration Not Saved in the Flash

Commands	Test Command: `AT+CWMODE_CUR=?`	Query Command: `AT+CWMODE_CUR?` Function: to query the current Wi-Fi mode of ESP8266.	Set Command: `AT+CWMODE_CUR=<mode>` Function: to set the current Wi-Fi mode of ESP8266.
Response	`+CWMODE_CUR:<mode>` `OK`	`+CWMODE_CUR:<mode>` `OK`	`OK`
Parameters	`<mode>`: ▸ 1: Station mode ▸ 2: SoftAP mode ▸ 3: SoftAP+Station mode		
Note	The configuration changes will NOT be saved in the flash.		
Example	`AT+CWMODE_CUR=3`		

4.2.2. AT+CWMODE_DEF—Sets the Default Wi-Fi mode; Configuration Saved in the Flash

Commands	Test Command: `AT+CWMODE_DEF=?`	Query Command: `AT+CWMODE_DEF?` Function: to query the current Wi-Fi mode of ESP8266.	Set Command: `AT+CWMODE_DEF=<mode>` Function: to set the current Wi-Fi mode of ESP8266.
Response	`+CWMODE_DEF:<mode>` `OK`	`+CWMODE_DEF:<mode>` `OK`	`OK`
Parameters	`<mode>`: ▸ 1: Station mode ▸ 2: SoftAP mode ▸ 3: SoftAP+Station mode		
Note	The configuration changes will be saved in the system parameter area in the flash.		
Example	`AT+CWMODE_DEF=3`		

4.2.3. AT+CWJAP_CUR—Connects to an AP; Configuration Not Saved in the Flash

Commands	Query Command: `AT+CWJAP_CUR?` Function: to query the AP to which the ESP8266 Station is already connected.	Set Command: `AT+CWJAP_CUR=<ssid>,<pwd>,[<bssid>][,<pci_en>]` Function: to set the AP to which the ESP8266 Station needs to be connected.

Response	`+CWJAP_CUR:<ssid>,<bssid>,<channel>,<rssi>`
	`OK`
		`OK`
		or
		`+CWJAP_CUR:<error code>`
		`FAIL`

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>]`: optional parameter, the target AP's MAC address, used when multiple APs have the same SSID.
		• `[<pci_en>]`: optional parameter, disable the connection to WEP or OPEN AP, and can be used for PCI authentication.
		• `<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 enabled. Escape character syntax is needed if SSID or password contains any special characters, such as , or " or \.

Note	The configuration changes will NOT be saved in the flash.	

Examples	`AT+CWJAP_CUR="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_CUR="ab\\\,c","0123456789\"\\"`
	If multiple APs have the same SSID as `"abc"`, the target AP can be found by BSSID:
	`AT+CWJAP_CUR="abc","0123456789","ca:d7:19:d8:a6:44"`	

4.2.4. AT+CWJAP_DEF—Connects to an AP; Configuration Saved in the Flash

Commands	Query Command: `AT+CWJAP_DEF?` Function: to query the AP to which the ESP8266 Station is already connected.	Set Command: `AT+CWJAP_DEF=<ssid>,<pwd>,[<bssid>] [,<pci_en>]` Function: to set the AP to which the ESP8266 Station needs to be connected.

Response	`+CWJAP_DEF:<ssid>,<bssid>,<channel>,<rssi>`
	`OK`
		`OK`
		or
		`+CWJAP_DEF:<error code>`
		`FAIL`

Parameters	`<ssid>`: a string parameter showing the SSID of the target AP.	• `<ssid>`: the SSID of the target AP, MAX: 32 bytes.
		• `<pwd>`: password, MAX: 64-byte ASCII.
		• `[<bssid>]`: optional parameter, the target AP's MAC address, used when multiple APs have the same SSID.
		• `[<pci_en>]`: optional parameter, disable the connection to WEP or OPEN AP, and can be used for PCI authentication.
		• `<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 enabled. 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 system parameter area in the flash.	

Examples	`AT+CWJAP_DEF="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_DEF="ab\\\,c","0123456789\"\\"`
	If multiple APs have the same SSID as `"abc"`, the target AP can be found by BSSID:
	`AT+CWJAP_DEF="abc","0123456789","ca:d7:19:d8:a6:44"`	

4.2.5. 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 not ordered according to RSSI. ▸ `1`: the result is 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`. ▸ `bit 5`: determines whether `<freq offset>` will be shown in the result of `AT+CWLAP`. ▸ `bit 6`: determines whether `<freq calibration>` will be shown in the result of `AT+CWLAP`. ▸ `bit 7`: determines whether `<pairwise_cipher>` will be shown in the result of `AT+CWLAP`. ▸ `bit 8`: determines whether `<group_cipher>` will be shown in the result of `AT+CWLAP`. ▸ `bit 9`: determines whether `<bgn>` will be shown in the result of `AT+CWLAP`. ▸ `bit 10`: determines whether `<wps>` will be shown in the result of `AT+CWLAP`.
Example	`AT+CWLAPOPT=1,2047` The first parameter is 1, meaning that the result of the command `AT+CWLAP` will be ordered according to RSSI; The second parameter is `2047`, namely `0x7FF`, meaning that the corresponding bits of `<mask>` are set to 1. All parameters will be shown in the result of `AT+CWLAP`.

4.2.6. AT+CWLAP—Lists Available APs

Commands	Set Command: `AT+CWLAP[=<ssid>,<mac>,<channel>,<scan_type>,<scan_time_min>,<scan_time_max>]` 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>,<channel>,<freq offset>,<freq cali>,<pairwise_cipher>,<group_cipher>,<bgn>,<wps>` `OK`	`+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<channel>,<freq offset>, <freq cali>,<pairwise_cipher>,<group_cipher>,<bgn>,<wps>` `OK`

Parameters	• [<scan_type>]: optional parameter
	  ‣ 0: active scan
	  ‣ 1: passive scan
	• [<scan_time_min>] : optional parameter, unit: ms, range: [0,1500]
	  ‣ For active scan mode, <scan_time_min> is the minimum scan time for each channel, default is 0
	  ‣ For passive scan mode, <scan_time_min> is meaningless, can be omitted
	• [<scan_time_max>] : optional parameter, unit: ms, range: [0,1500]
	  ‣ For active scan mode, <scan_time_max> is the maximum scan time for each channel; if it is set to be 0, a default value, 120ms will be used.
	  ‣ For passive scan mode, <scan_time_max> is the scan time for each channel, default is 360ms
	• <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.
	• <channel>: channel number
	• <freq offset>: frequency offset of AP; unit: KHz. The value of ppm is <freq offset>/2.4.
	• <freq calibration>: calibration for frequency offset.
	• <pairwise_cipher>:
	  ‣ 0: CIPHER_NONE
	  ‣ 1: CIPHER_WEP40
	  ‣ 2: CIPHER_WEP104
	  ‣ 3: CIPHER_TKIP
	  ‣ 4: CIPHER_CCMP
	  ‣ 5: CIPHER_TKIP_CCMP
	  ‣ 6: CIPHER_UNKNOWN
	• <group_cipher>: definitions of cipher types are same as <pairwise_cipher>
	• <bgn>:
	  ‣ bit0 is for 802.11b mode; bit1 is for 802.11g mode; bit2 is for 802.11n mode;
	  ‣ if the value of the bit is 1, the corresponding 802.11 mode is enabled; if the bit value is 0, the mode is disabled.
	• <wps>: 0, WPS is disabled; 1, WPS is enabled

Examples	`AT+CWLAP="Wi-Fi","ca:d7:19:d8:a6:44",6`
	or search for APs with a designated SSID:
	`AT+CWLAP="Wi-Fi"`
	or enable passive scan:
	`AT+CWLAP=,,,1,,`

4.2.7. AT+CWQAP—Disconnects from the AP

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

4.2.8. AT+CWSAP_CUR—Configures the ESP8266 SoftAP; Configuration Not Saved in the Flash

Commands	Query Command: `AT+CWSAP_CUR?` Function: to obtain the configuration parameters of the ESP8266 SoftAP.	Set Command: `AT+CWSAP_CUR=<ssid>,<pwd>,<chl>,<ecn>[,<max conn>][,<ssid hidden>]` Function: to configure the ESP8266 SoftAP.

Response	`+CWSAP_CUR:<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 ESP8266 SoftAP can be connected; within the range of [1, 8].
	• `[<ssid hidden>]` (optional):
	  ▸ `0`: SSID is broadcasted. (the default setting)
	  ▸ `1`: SSID is not broadcasted.
		⚠ ***Notice:***
		This command is only available when SoftAP is active.

Note	The configuration changes will NOT be saved in the flash.	
Example	`AT+CWSAP_CUR="ESP8266","1234567890",5,3`	

4.2.9. AT+CWSAP_DEF—Configures the ESP8266 SoftAP; Configuration Saved in the Flash

Commands	Query Command: `AT+CWSAP_DEF?` Function: to obtain the configuration parameters of the ESP8266 SoftAP.	Set Command: `AT+CWSAP_DEF=<ssid>,<pwd>,<chl>,<ecn>[,<max conn>][,<ssid hidden>]` Function: to list all available APs.

Response	`+CWSAP_DEF:<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 ESP8266 SoftAP can be connected; within the range of [1, 8].
	• `[<ssid hidden>]` (optional):
	  ▸ `0`: SSID is broadcasted. (the default setting)
	  ▸ `1`: SSID is not broadcasted.
		The same as above.
		⚠ ***Notice:***
		This command is only available when SoftAP is active.

Note	The configuration changes will be saved in the flash system parameter area.	
Example	`AT+CWSAP_DEF="ESP8266","1234567890",5,3`	

4.2.10. AT+CWLIF—IP of Stations to Which the ESP8266 SoftAP is Connected

Execute Command	`AT+CWLIF`
Response	`<ip addr>,<mac>` `OK`
Parameters	• `<ip addr>`: IP address of Stations to which ESP8266 SoftAP is connected. • `<mac>`: MAC address of Stations to which ESP8266 SoftAP is connected.
Note	This command cannot get a static IP. It only works when both DHCPs of the ESP8266 SoftAP, and of the Station to which ESP8266 is connected, are enabled.

4.2.11. AT+CWDHCP_CUR—Enables/Disables DHCP; Configuration Not Saved in the Flash

Commands	Query Command: `AT+CWDHCP_CUR?`	Set Command: `AT+CWDHCP_CUR=<<mode>,<en>` Function: to enable/disable DHCP.
Response	`DHCP disabled or enabled now?`	`OK`
Parameters	• `Bit0`: ▸ `0`: SoftAP DHCP is disabled. ▸ `1`: SoftAP DHCP is enabled. • `Bit1`: ▸ `0`: Station DHCP is disabled. ▸ `1`: Station DHCP is enabled.	• `<mode>`: ▸ `0`: Sets ESP8266 SoftAP ▸ `1`: Sets ESP8266 Station ▸ `2`: Sets both SoftAP and Station • `<en>`: ▸ `0`: Disables DHCP ▸ `1`: Enables DHCP
Notes	• The configuration changes will not be saved in flash. • 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.	
Example	`AT+CWDHCP_CUR=0,1`	

4.2.12. AT+CWDHCP_DEF—Enables/Disables DHCP; Configuration Saved in the Flash

Commands	Query Command: `AT+CWDHCP_DEF?`	Set Command: `AT+CWDHCP_DEF=<<mode>,<en>` Function: to enable/disable DHCP.
Response	`DHCP disabled or enabled now?`	`OK`
Parameters	• `Bit0`: ▸ `0`: SoftAP DHCP is disabled. ▸ `1`: SoftAP DHCP is enabled. • `Bit1`: ▸ `0`: Station DHCP is disabled. ▸ `1`: Station DHCP is enabled.	• `<mode>`: ▸ `0`: Sets ESP8266 SoftAP ▸ `1`: Sets ESP8266 Station ▸ `2`: Sets both SoftAP and Station • `<en>`: ▸ `0`: Disables DHCP ▸ `1`: Enables DHCP
Notes	• The configuration changes will be stored in the user parameter area in the flash. • 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.	
Example	`AT+CWDHCP_DEF=0,1`	

4.2.13. AT+CWDHCPS_CUR—Sets the IP Address Allocated by ESP8266 SoftAP DHCP; Configuration Not Saved in Flash

Commands	Query Command: `AT+CWDHCPS_CUR?`	Set Command: `AT+CWDHCPS_CUR=<enable>,<lease time>,<start IP>,<end IP>` Function: sets the IP address range of the ESP8266 SoftAP DHCP server.
Response	`+CWDHCPS_CUR=<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 ESP8266 SoftAP DHCP server.
	• `<end IP>`: end IP of the IP range that can be obtained from ESP8266 SoftAP DHCP server.	

Notes	• The configuration changes will NOT be saved in the flash. • 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 ESP8266 SoftAP.	

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

4.2.14. AT+CWDHCPS_DEF—Sets the IP Address Allocated by ESP8266 SoftAP DHCP; Configuration Saved in Flash

Commands	Query Command: `AT+CWDHCPS_DEF?`	Set Command: `AT+CWDHCPS_DEF=<enable>,<lease time>,<start IP>,<end IP>` Function: sets the IP address range of the ESP8266 SoftAP DHCP server.
Response	`+CWDHCPS_DEF=<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 ESP8266 SoftAP DHCP server.
	• `<end IP>`: end IP of the IP range that can be obtained from ESP8266 SoftAP DHCP server.	

Notes	• The configuration changes will be stored in the user parameter area in the flash. • 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 ESP8266 SoftAP.

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

4.2.15. 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 ESP8266 Station connects to the AP automatically on power-up by default.
Note	The configuration changes will be saved in the system parameter area in the flash.
Example	`AT+CWAUTOCONN=1`

4.2.16. AT+CIPSTAMAC_CUR—Sets the MAC Address of the ESP8266 Station; Configuration Not Saved in the Flash

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

4.2.17. AT+CIPSTAMAC_DEF—Sets the MAC Address of the ESP8266 Station; Configuration Saved in the Flash

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

4.2.18. AT+CIPAPMAC_CUR—Sets the MAC Address of the ESP8266 SoftAP; Configuration Not Saved in the Flash

Commands	Query Command: `AT+CIPAPMAC_CUR?` Function: to obtain the MAC address of the ESP8266 SoftAP.	Set Command: `AT+CIPAPMAC_CUR=<mac>` Function: to set the MAC address of the ESP8266 SoftAP.
Response	`+CIPAPMAC_CUR:<mac>` `OK`	`OK`
Parameters	`<mac>`: string parameter, MAC address of ESP8266 SoftAP.	
Notes	• The configuration changes will NOT be saved the flash. • The MAC address of ESP8266 SoftAP is different from that of the ESP8266 Station. Please make sure you do not set the same MAC address for both of them. • Bit 0 of the ESP8266 MAC address CANNOT be 1. For example, a MAC address can be "18: …" but not "15:…".	
Example	`AT+CIPAPMAC_CUR="1a:fe:36:97:d5:7b"`	

4.2.19. AT+CIPAPMAC_DEF—Sets the MAC Address of the ESP8266 SoftAP; Configuration Saved in Flash

Commands	Query Command: `AT+CIPAPMAC_DEF?` Function: to obtain the MAC address of the ESP8266 SoftAP.	Set Command: `AT+CIPAPMAC_DEF=<mac>` Function: to set the MAC address of the ESP8266 SoftAP.

Response	+CIPAPMAC_DEF:<mac> OK	OK
Parameters	<mac>: string parameter, MAC address of ESP8266 SoftAP.	
Notes	• The configuration changes will be saved in the user parameter area in the flash. • The MAC address of ESP8266 SoftAP is different from that of the ESP8266 Station. Please make sure you do not set the same MAC address for both of them. • Bit 0 of the ESP8266 MAC address CANNOT be 1. For example, a MAC address can be "18: …" but not "15:…".	
Example	AT+CIPAPMAC_DEF="1a:fe:36:97:d5:7b"	

4.2.20. AT+CIPSTA_CUR—Sets the Current IP Address of the ESP8266 Station; Configuration Not Saved in the Flash

Commands	Query Command: AT+CIPSTA_CUR? Function: to obtain the current IP address of the ESP8266 Station.	Set Command: AT+CIPSTA_CUR=<ip>[,<gateway>,<netmask>] Function: to set the current IP address of the ESP8266 Station.
Response	+CIPSTA_CUR:<ip> +CIPSTA_CUR:<gateway> +CIPSTA_CUR:<netmask> OK	OK
Parameters	⚠ ***Notice:*** Only when the ESP8266 Station is connected to an AP can its IP address be queried.	• <ip>: string parameter, the IP address of the ESP8266 Station. • [<gateway>]: gateway. • [<netmask>]: netmask.
Notes	• The configuration changes will NOT be saved in the flash. • 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_CUR="192.168.6.100","192.168.6.1","255.255.255.0"	

4.2.21. AT+CIPSTA_DEF—Sets the Default IP Address of the ESP8266 Station; Configuration Saved in the Flash

Commands	Query Command: AT+CIPSTA_DEF? Function: to obtain the default IP address of the ESP8266 Station.	Set Command: AT+CIPSTA_DEF=<ip>[,<gateway>,<netmask>] Function: to set the default IP address of the ESP8266 Station.

Response	+CIPSTA_DEF:<ip> +CIPSTA_DEF:<gateway> +CIPSTA_DEF:<netmask> OK	OK
Parameters	⚠ ***Notice:*** Only when the ESP8266 Station is connected to an AP can its IP address be queried.	• <ip>: string parameter, the IP address of the ESP8266 Station. • [<gateway>]: gateway. • [<netmask>]: netmask.
Notes	• The configuration changes will be saved in the user parameter area in the flash. • 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_DEF="192.168.6.100","192.168.6.1","255.255.255.0"	

4.2.22. AT+CIPAP_CUR—Sets the IP Address of the ESP8266 SoftAP; Configuration Not Saved in the Flash

Commands	Query Command: AT+CIPAP_CUR? Function: to obtain the current IP address of the ESP8266 SoftAP.	Set Command: AT+CIPAP_CUR=<ip>[,<gateway>,<netmask>] Function: to set the current IP address of the ESP8266 SoftAP.
Response	+CIPAP_CUR:<ip> +CIPAP_CUR:<gateway> +CIPAP_CUR:<netmask> OK	OK
Parameters	• <ip>: string parameter, the IP address of the ESP8266 SoftAP. • [<gateway>]: gateway. • [<netmask>]: netmask.	
Notes	• The configuration changes will not be saved in the flash. • Currently, ESP8266 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_CUR="192.168.5.1","192.168.5.1","255.255.255.0"	

4.2.23. AT+CIPAP_DEF—Sets the Default IP Address of the ESP8266 SoftAP; Configuration Saved in the Flash

Commands	Query Command: `AT+CIPAP_DEF?` Function: to obtain the default IP address of the ESP8266 SoftAP.	Set Command: `AT+CIPAP_DEF=<ip>[,<gateway>,<netmask>]` Function: to set the default IP address of the ESP8266 SoftAP.

Response	`+CIPAP_DEF:<ip>`
	`+CIPAP_DEF:<gateway>`
	`+CIPAP_DEF:<netmask>`
	`OK`
		`OK`

Parameters	• `<ip>`: string parameter, the IP address of the ESP8266 SoftAP. • `[<gateway>]`: gateway. • `[<netmask>]`: netmask.	
Notes	• The configuration changes will be saved in the user parameter area in the flash. • Currently, ESP8266 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_DEF="192.168.5.1","192.168.5.1","255.255.255.0"`	

4.2.24. 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	

Messages	When smartconfig starts, it will prompt messages as below: smartconfig type: <type> // AIRKISS or ESPTOUCH smart get wifi info // got SSID and password ssid:<AP's SSID> password:<AP's password> // ESP8266 will try to connect to the AP WIFI CONNECTED WIFI GOT IP smartconfig connected wifi // if the connection failed, it will prompt "smartconfig connect fail"
Notes	• For details on SmartConfig please see *ESP-TOUCH User Guide*. • SmartConfig is only available in the ESP8266 Station mode. • The message `smart get wifi info` means that SmartConfig has successfully acquired the AP information. ESP8266 will try to connect to the target AP. • Message `smartconfig connected wifi` 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. • Starting from AT_v1.0, SmartConfig can get protocol type (AirKiss or ESP-TOUCH) automatically by command `AT+CWSTARTSMART`. • Users can remove this function to reduce bin size and save memory by recompiling the at project, refer to **Section 1.1**, and disable the `#define CONFIG_AT_SMARTCONFIG_COMMAND_ENABLE` in *user_config.h*.
Example	`AT+CWMODE=1` `AT+CWSTARTSMART`

4.2.25. 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`

4.2.26. AT+CWSTARTDISCOVER—Enables the Mode that ESP8266 can be Found by WeChat

Set Command	`AT+CWSTARTDISCOVER=<WeChat number>,<dev_type>,<time>`
Response	`OK`

Parameters	• <WeChat number>: WeChat official account, which is to be obtained from WeChat. • <dev_type>: the device type, which is to be obtained from WeChat. • <time>: the interval of time for ESP8266 to send packets; range: 0 ~ 24x3600; unit: second. ▸ 0: ESP8266 will not take the initiative to send packets; it only makes response to queries from WeChat. ▸ Otherwise: the time interval for ESP8266 to send packets regularly in order to be detected by WeChat on the same LAN.
Note	• For details on detection function of WeChat, please refer to http://iot.weixin.qq.com. • ESP8266 Station should connect to an AP and obtain an IP address first before this command is used.
Example	AT+CWSTARTDISCOVER="gh_9e2cff3dfa51","122475",10

4.2.27. AT+CWSTOPDISCOVER—Disables the Mode that ESP8266 can be Found by WeChat

Execute Command	AT+CWSTOPDISCOVER
Response	OK
Example	AT+CWSTOPDISCOVER

4.2.28. AT+WPS—Enables the WPS Function

Set Command	AT+WPS=<enable>
Response	OK
Parameters	<enable>: ▸ 1: enables WPS/Wi-Fi Protected Setup ▸ 0: disables WPS
Notes	• WPS must be used when the ESP8266 Station is enabled. • WPS does not support WEP/Wired-Equivalent Privacy encryption.
Example	AT+CWMODE=1 AT+WPS=1

4.2.29. AT+MDNS—Configures the MDNS Function

Set Command	AT+MDNS=<enable>,<hostname>,<server_name>,<server_port>
Response	OK or opmode mismatch when mdns ERROR

Parameters	• <enable>:
	  ▸ 1: enables the MDNS function; the following three parameters need to be set.
	  ▸ 0: disables the MDNS function; the following three parameters need not to be set.
	• <hostname>: MDNS host name
	• <server_name>: MDNS server name
	• <server_port>: MDNS server port

Notes	• Please do not use special characters (such as .) or a protocol name (for example, http) for <hostname> and <server_name>. • ESP8266 SoftAP mode does not support the MDNS function for now.
Example	AT+MDNS=1,"espressif","iot",8080

4.2.30. AT+CWHOSTNAME—Configures the Name of ESP8266 Station

Commands	Query Command: AT+CWHOSTNAME? Function: Checks the host name of ESP8266 Station.	Set Command: AT+CWHOSTNAME=<hostname> Function: Sets the host name of ESP8266 Station.
Response	+CWHOSTNAME:<host name> OK If the station mode is not enabled, the command will return: +CWHOSTNAME:<null> OK	OK If the Station mode is not enabled, the command will return: ERROR
Parameters	<hostname>: the host name of the ESP8266 Station, the maximum length is 32 bytes.	
Notes	• The configuration changes are not saved in the flash. • The default host name of the ESP8266 Station is ESP_XXXXXX; XXXXXX is the lower 3 bytes of the MAC address, for example, +CWHOSTNAME:<ESP_A378DA>.	
Example	AT+CWMODE=3 AT+CWHOSTNAME="my_test"	

4.2.31. AT+CWCOUNTRY_CUR—Set ESP8266 WiFi Country Code; Configuration Not Saved in the Flash

Commands	Query Command: AT+CWCOUNTRY_CUR? Function: Check the current WiFi country code of ESP8266.	Set Command: AT+CWCOUNTRY_CUR=<country_policy>,<country_code>,<start_channel>,<total_channel_count> Function: Set the current WiFi country code of ESP8266.

Response	+CWCOUNTRY_CUR:<country_policy>,<country_code>,<start_channel>,<total_channel_count> OK Command AT+CWCOUNTRY_CUR? will return the actual value of WiFi country code, which may be changed to the same as the AP it connected to.	OK
Parameters	<country_policy>: • 0: will change the county code to be the same as the AP that ESP is connected to • 1: the country code will not change, always be the one set by command. <country_code>: country code, the length can be 3 characters at most; but the third one is a special character which will not be shown when querying by command AT+CWCOUNTRY_CUR? <start_channel> : the channel number to start at <total_channel_count> : channel count	
Notes	The configuration changes will NOT be saved in the flash.	
Example	AT+CWMODE=3 AT+CWCOUNTRY_CUR=1,"CN",1,13	

4.2.32. AT+CWCOUNTRY_DEF—Set the default WiFi Country Code of ESP8266; Configuration Saved in the Flash

Commands	Query Command: AT+CWCOUNTRY_DEF? Function: Check the default WiFi country code of ESP8266 which is saved in the flash.	Set Command: AT+CWCOUNTRY_DEF=<country_policy>,<country_code>,<start_channel>,<total_channel_count> Function: Set the default WiFi country code of ESP8266, and save in the flash.
Response	+CWCOUNTRY_DEF:<country_policy>,<country_code>,<start_channel>,<total_channel_count> OK Command AT+CWCOUNTRY_DEF? will return the default WiFi country code which is stored in the flash.	OK
Parameters	<country_policy>: • 0: will change the county code to be the same as the AP that ESP is connected to • 1: the country code will not change, always be the one set by command. <country_code>: country code, the length can be 3 characters at most; but the third one is a special character which will not be shown when querying by command AT+CWCOUNTRY_DEF? <start_channel> : the channel number to start at <total_channel_count> : channel count	
Notes	The configuration changes will be saved in the flash user parameter area.	
Example	AT+CWMODE=3 AT+CWCOUNTRY_DEF=1,"CN",1,13	

5. TCP/IP-Related AT Commands

5.1. Overview

Command	Description
AT+CIPSTATUS	Gets the connection status
AT+CIPDOMAIN	DNS function
AT+CIPSTART	Establishes TCP connection, UDP transmission or SSL connection
AT+CIPSSLSIZE	Sets the size of SSL buffer
AT+CIPSSLCONF	Set configuration of ESP SSL client
AT+CIPSEND	Sends data
AT+CIPSENDEX	Sends data when length of data is <length>, or when \0 appears in the data
AT+CIPSENDBUF	Writes data into TCP-send-buffer
AT+CIPBUFRESET	Resets the segment ID count
AT+CIPBUFSTATUS	Checks the status of TCP-send-buffer
AT+CIPCHECKSEQ	Checks if a specific segment is sent or not
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 a TCP server
AT+CIPSERVERMAXCONN	Set the maximum connections that server allows
AT+CIPMODE	Configures the transmission mode
AT+SAVETRANSLINK	Saves the transparent transmission link in the flash
AT+CIPSTO	Sets timeout when ESP8266 runs as TCP server
AT+PING	Ping packets
AT+CIUPDATE	Upgrades the software through network
AT+CIPDINFO	Shows remote IP and remote port with +IPD
+IPD	ESP receives network data
AT+CIPRECVMODE	Set TCP Receive Mode
AT+CIPRECVDATA	Get TCP Data in Passive Receive Mode
AT+CIPRECVLEN	Get TCP Data Length in Passive Receive Mode
AT+CIPSNTPCFG	Configures the time domain and SNTP server.
AT+CIPSNTPTIME	Queries the SNTP time.

AT+CIPDNS_CUR	Sets user-defined DNS servers; configuration not saved in the flash
AT+CIPDNS_DEF	Sets user-defined DNS servers; configuration saved in the flash

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 ESP8266 Station interface. ▸ 2: The ESP8266 Station is connected to an AP and its IP is obtained. ▸ 3: The ESP8266 Station has created a TCP or UDP transmission. ▸ 4: The TCP or UDP transmission of ESP8266 Station is disconnected. ▸ 5: The ESP8266 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>: ESP8266 local port number. • <tetype>: ▸ 0: ESP8266 runs as a client. ▸ 1: ESP8266 runs as a server.

5.2.2. AT+CIPDOMAIN—DNS Function

Execute Command	AT+CIPDOMAIN=<domain name>
Response	+CIPDOMAIN:<IP address> OK Or DNS Fail ERROR
Parameter	<domain name>: the domain name, length should be less than 64 bytes.
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`
	If the TCP connection is already established, the response is:
	`ALREADY CONNECTED`	

Parameters	• `<link ID>`: ID of network connection (0~4), used for multiple connections. • `<type>`: string parameter indicating the connection type: `"TCP"`, `"UDP"` or `"SSL"`. • `<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: *ESP8266 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 the UDP transmission is already established, the response is:
	`ALREADY CONNECTED`	

Parameters	• <link ID>: ID of network connection (0~4), used for multiple connections. • <type>: string parameter indicating the connection type: "TCP", "UDP" or "SSL". • <remote IP>: string parameter indicating the remote IP address. • <remote port>: remote port number. • [<UDP local port>]: optional; UDP port of ESP8266. • [<UDP mode>]: optional. In the UDP transparent transmission, the value of this parameter has to be 0. ▸ 0: the destination peer entity of UDP will not change; this is the default setting. ▸ 1: the destination peer entity of UDP can change once. ▸ 2: the destination peer entity of UDP is allowed to change. ⚠ ***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: *ESP8266 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 the TCP connection is already established, the response is: ALREADY CONNECTED
Parameters	• <link ID>: ID of network connection (0~4), used for multiple connections. • <type>: string parameter indicating the connection type: "TCP", "UDP" or "SSL". • <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	• ESP8266 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. The command AT+CIPSSLSIZE=<size> can be used to enlarge the SSL buffer size.
Example	AT+CIPSTART="SSL","iot.espressif.cn",8443

5.2.4. AT+CIPSSLSIZE—Sets the Size of SSL Buffer

Set Command	`AT+CIPSSLSIZE=<size>`
Response	`OK`
Parameters	`<size>`: the size of the SSL buffer; range of value: [2048, 4096].
Example	`AT+CIPSSLSIZE=4096`

5.2.5. AT+CIPSSLCCONF - Sets Configuration of ESP SSL Client

Commands	Query Command: `AT+CIPSSLCCONF?` Function: Get configuration of the ESP8266 SSL client.	Set Command: `AT+CIPSSLCCONF=<SSL mode>` Function: Set configuration of the ESP8266 SSL client.
Response	`+CIPSSLCCONF:<SSL mode>` `OK`	`OK`
Parameters	`<SSL mode>`: ▸ `bit0`: if set to be 1, certificate and private key will be enabled, so SSL server can verify ESP8266; if 0, then will not. ▸ `bit1`: if set to be 1, CA will be enabled, so ESP8266 can verify SSL server; if 0, then will not.	
Notes	• If certificates need to be enabled, please call this command before SSL connection is established. • If certificates need to be enabled, please refer to the *ESP8266 SSL User Guide* to generate certificates. - esp_ca_cert.bin downloads to 0xFB000 by default - esp_cert_private_key.bin downloads to 0xFC000 by default - Users can revise the SYSTEM_PARTITION_SSL_CLIENT_CA_ADDR and SYSTEM_PARTITION_SSL_CLIENT_CERT_PRIVKEY_ADDR in *user_main.c* to change the downloading addresses. • This configuration will be saved in the flash user parameter area.	

Example	`AT+CWMODE=1` // enable sta mode
	`AT+CWJAP="SSID","PASSWORD"` // connect to an AP
	`AT+CIPSNTPCFG=1,8` // set SNTP timezone
	`AT+CIPSNTPTIME?` // get SNTP time
	`AT+CIPSSLCCONF=2`
	`AT+CIPSTART="SSL","192.168.3.38",8443`	

5.2.6. 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 data length defined by `<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`
	If it failed, the system returns:
	`SEND FAIL`
		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, ESP8266 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: *ESP8266 AT Command Examples*.	

5.2.7. AT+CIPSENDEX—Sends Data

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 If it failed, the system returns: SEND FAIL
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.

5.2.8. AT+CIPSENDBUF—Writes Data into the TCP-Send-Buffer

Set Command	1. Single connection: (+CIPMUX=0)
	`AT+CIPSENDBUF=<length>`
	2. Multiple connections: (+CIPMUX=1)
	`AT+CIPSENDBUF=<link ID>,<length>`

Response	<current segment ID>,<segment ID of which sent successfully> OK > • Wrap return > begins receiving serial data; when the length of data defined by the parameter <length> is met, the data is sent; if the data length over the value of <length>, the data will be discarded, and the command returns busy. • If the connection cannot be established, or if it is not a TCP connection, or if the buffer is full, or some other error occurs, the command returns ERROR • If data is transmitted successfully, ▸ for single connection, the response is: <segment ID>,SEND OK ▸ for multiple connections, the response is: <link ID>,<segment ID>,SEND OK • If it failed, the system returns: SEND FAIL
Parameters	• <link ID>: ID of the connection (0~4), for multiple connections. • <segment ID>: uint32; the ID assigned to each data packet, starting from 1; the ID number increases by 1 every time a data packet is written into the buffer. • <length>: data length; MAX: 2048 bytes.
Notes	• This command only writes data into the TCP-send-buffer, so it can be called continually, and the user need not wait for SEND OK; if a TCP segment is sent successfully, it will return <segment ID>,SEND OK. • Before data length reaches the value defined by <length>, input +++ can switch back from data mode to command mode, and discard the data received before. • This command can NOT be used for SSL connections.

5.2.9. AT+CIPBUFRESET—Resets the Segment ID Count

Set Command	1. Single connection: (+CIPMUX=0) AT+CIPBUFRESET 2. Multiple connections: (+CIPMUX=1) AT+CIPBUFRESET=<link ID>
Response	OK If the connection is not established or there is still TCP data waiting to be sent, the response will be: ERROR
Parameter	<link ID>: ID of the connection (0~4), for multiple connections.
Note	This command can only be used when AT+CIPSENDBUF is used.

5.2.10. AT+CIPBUFSTATUS—Checks the Status of the TCP-Send-Buffer

Set Command	1. Single connection: (+CIPMUX=0) AT+CIPBUFSTATUS 2. Multiple connections: (+CIPMUX=1) AT+CIPBUFSTATUS=<link ID>
Response	<next segment ID>,<segment ID sent >,<segment ID successfully sent>,<remain buffer size>,<queue number> OK
Parameters	• <next segment ID>: the next segment ID obtained by AT+CIPSENDBUF; • <segment ID sent>: the ID of the TCP segment last sent; • Only when <next segment ID> - <segment ID sent> = 1, can AT+CIPBUFRESET be called to reset the counting. • <segment ID successfully sent>: the ID of the last successfully sent TCP segment; • <remain buffer size>: the remaining size of the TCP-send-buffer; • <queue number>: available TCP queue number; it's not reliable and should be used as a reference only.
Notes	This command can not be used for SSL connection.
Example	For example, in single connection, the command AT+CIPBUFSTATUS returns: 20,15,10,200,7 Description: • 20: means that the latest segment ID is 19; so when calling AT+CIPSENDBUF the next time, the segment ID returned is 20; • 15: means that the TCP segment with the ID 15 is the last segment sent, but the segment may not be successfully sent; • 10: means that the TCP segment with the ID 10 was sent successfully; • 200: means that the remaining size of the TCP-send-buffer is 200 bytes; • 7: the available TCP queue number; it is not reliable and should be used as a reference only; when the queue number is 0, no TCP data can be sent.

5.2.11. AT+CIPCHECKSEQ—Checks If a Specific Segment Was Successfully Sent

Set Command	1. Single connection: (+CIPMUX=0) AT+CIPCHECKSEQ=<segment ID> 2. multiple connections: (+CIPMUX=1) AT+CIPCHECKSEQ=<link ID>,<segment ID>
Response	[<link ID>,]<segment ID>,<status> OK

Parameters	• The command can only be used to record the status of the last 32 segments at most. • [<link ID>]: ID of the connection (0~4), for multiple connection; • <segment ID>: the segment ID obtained by calling AT+CIPSENDBUF; • <status>: ▸ FALSE: the segment-sending failed; ▸ TRUE: the segment was sent successfully.
Notes	This command can only be used when AT+CIPSENDBUF is used.

5.2.12. AT+CIPCLOSEMODE—Set the Close Mode of TCP Connection

Commands	1. Single connection: (+CIPMUX=0) AT+CIPCLOSEMODE=<enable_abort> 2. multiple connections: (+CIPMUX=1) AT+CIPCLOSEMODE=<link ID>,<enable_abort>
Response	OK
Parameters	<link ID>: ID of the connection <enable_abort>: 0, normal disconnect; 1, abort to disconnect
Notes	• Default is the normal disconnect mode. Usually the abort mode should not be enabled. • If the abort mode is needed, please use this command after TCP connection is established. And the configuration only takes effect in the current connection. If the connection breaks, you need to set it again after new connection is established. • This configuration only works on normal TCP connection, cannot be used on SSL connection.
Example	AT+CIPSTART=0,"TCP","192.168.3.60",3400 AT+CIPCLOSEMODE=0,1 AT+CIPCLOSE=0

5.2.13. AT+CIPCLOSE—Closes the TCP/UDP/SSL Connection

Commands	Set Command (used in multiple connections): AT+CIPCLOSE=<link ID> Function: close the TCP/UDP Connection.	Execute Command (used in multiple connections): AT+CIPCLOSE
Response	OK	
Parameters	<link ID>: ID of the connection to be closed. When ID is 5, all connections will be closed. (In server mode, the ID 5 has no effect.)	-

5.2.14. AT+CIFSR—Gets the Local IP Address

Execute Command	AT+CIFSR

Response	+CIFSR:APIP,<SoftAP IP address> +CIFSR:APMAC,<SoftAP MAC address> +CIFSR:STAIP,<Station IP address> +CIFSR:STAMAC,<Station MAC address> OK
Parameters	<IP address>: IP address of the ESP8266 SoftAP; IP address of the ESP8266 Station. <MAC address>: MAC address of the ESP8266 SoftAP; MAC address of the ESP8266 Station.
Notes	Only when the ESP8266 Station is connected to an AP can the Station IP can be queried.

5.2.15. AT+CIPMUX—Enable or Disable 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	

5.2.16. AT+CIPSERVER—Deletes/Creates TCP Server

Set Command	AT+CIPSERVER=<mode>[,<port>]
Response	OK
Parameters	• <mode>: ▸ 0: deletes server. ▸ 1: creates 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.
Example	`AT+CIPMUX=1` `AT+CIPSERVER=1,1001`

5.2.17. AT+CIPSERVERMAXCONN—Set the Maximum Connections Allowed by Server

Commands	Query Command: `AT+CIPSERVERMAXCONN?` Function: obtain the maximum number of clients allowed to connect to the TCP or SSL server.	Set Command: `AT+CIPSERVERMAXCONN=<num>` Function: set the maximum number of clients allowed to connect to the TCP or SSL server.
Response	`+CIPSERVERMAXCONN:<num>` `OK`	`OK`
Parameters	`<num>`: the maximum number of clients allowed to connect to the TCP or SSL server, range: [1, 5]	
Notes	To set this configuration, you should call the command `AT+CIPSERVERMAXCONN=<num>` before creating a server.	

Example	`AT+CIPMUX=1`
	`AT+CIPSERVERMAXCONN=2`
	`AT+CIPSERVER=1,80`	

5.2.18. AT+CIPMODE—Sets 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, ESP8266 will keep trying to reconnect until `+++` is input to exit the transmission. If it is a normal TCP transmission and the TCP connection breaks, ESP8266 will give a prompt and will not attempt to reconnect.	
Example	`AT+CIPMODE=1`	

5.2.19. 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`

Parameters	• `<mode>`:
	  ▸ `0`: ESP8266 will NOT enter UART-Wi-Fi passthrough mode on power-up.
	  ▸ `1`: ESP8266 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 flash. ESP8266 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 the 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`

Parameters	• `<mode>`:
	  ▸ `0`: normal mode; ESP8266 will NOT enter UART-Wi-Fi passthrough mode on power-up.
	  ▸ `1`: ESP8266 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 flash. ESP8266 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 the user parameter area in the flash.
Example	`AT+SAVETRANSLINK=1,"192.168.6.110",1002,"UDP",1005`

5.2.20. 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	• ESP8266 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`	

5.2.21. AT+PING—Ping Packets

Set Command	`AT+PING=<IP>` Function: Ping packets.

Response	`+<time>`
	`OK`
	or
	`+timeout`
	`ERROR`

Parameters	• `<IP>`: string; host IP or domain name • `<time>`: the response time of ping
Notes	`AT+PING="192.168.1.1"` `AT+PING="www.baidu.com"`

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

Execute Command	`AT+CIUPDATE` Function: updates software.
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.
Notes	• If using Espressif's AT BIN (***/ESP8266_NONOS_SDK/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 (***/ESP8266_NONOS_SDK/example/at)***. • It is suggested that users call `AT+RESTORE` to restore the factory default settings after upgrading the AT firmware.

5.2.23. 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`

5.2.24. +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.	

5.2.25. AT+CIPRECVMODE—Set TCP Receive Mode

Commands	Set Command: `AT+CIPRECVMODE=<mode>`	Query Command: `AT+CIPRECVMODE?`
Response	`+CIPRECVMODE:<mode>` `OK`	`OK`
Parameters	`<mode>`: the receive mode of TCP data is active mode by default. • 0: active mode - ESP8266 will send all the received TCP data instantly to host MCU through UART with header "+IPD". • 1: passive mode - ESP8266 will keep the received TCP data in an internal buffer (default is 2920 bytes), and wait for host MCU to read the data. If the buffer is full, the TCP transmission will be blocked.	
Example	`AT+CIPRECVMODE=1`	
Note	• The configuration is for normal TCP transmission only, and cannot be used on SSL, UDP or WiFi-UART passthrough mode. • If the passive mode is enabled, when ESP8266 receives TCP data, it will prompt the following message in different scenarios: - for multiple connection mode (AT+CIPMUX=1), the message is: +IPD,<link ID>,<len> - for single connection mode (AT+CIPMUX=0), the message is: +IPD,<len> - <len> is the total length of TCP data in buffer	

5.2.26. AT+CIPRECVDATA—Get TCP Data in Passive Receive Mode

Set Commands	• For single connection mode (AT+CIPMUX=0):
	`AT+CIPRECVDATA=<len>`
	• For multiple connection mode (AT+CIPMUX=1):
	`AT+CIPRECVDATA=<link_id>,<len>`

Response	`+CIPRECVDATA:<actual_len>,<data>` `OK`
Parameters	`<link_id>`: connection ID in multiple connection mode. `<len>`: data length that you want to get, max is 2048 bytes per time. `<actual_len>`: length of the data you actually get `<data>`: the data you get
Notes	• In a case of disconnection, the buffered TCP data will still be there and can be read by MCU until a new connection is established. If the newly established connection happens to use the same link ID, the previously buffered data in the last connection will be lost.

Example	AT+CIPRECVMODE=1
	// For example, if host MCU gets a message of receiving 100 bytes data in connection with No.0, the message will be as following: +IPD,0,100
	// then you can read those 100 bytes by using the command below
	AT+CIPRECVDATA=0,100

5.2.27. AT+CIPRECVLEN—Get TCP Data Length in Passive Receive Mode

Query Commands	AT+CIPRECVLEN?
Response	+CIPRECVLEN:<data length of link0>,<data length of link1>,<data length of link2>,<data length of link3>,<data length of link4> OK
Parameters	<data length of link>: length of the entire data buffered for the link
Example	AT+CIPRECVLEN? +CIPRECVLEN:100,,,,, OK

5.2.28. AT+CIPSNTPCFG—Sets the Configuration of SNTP

Commands	Query Command: AT+CIPSNTPCFG?	Set Command: AT+CIPSNTPCFG=<enable>[,<timezone>][,<SNTP server0>,<SNTP server1>,<SNTP server2>]
Response	+CIPSNTPCFG:<enable>,<timezone>,<SNTP server1>[,<SNTP server2>,<SNTP server3>] OK	OK
Parameters	• <enable>: ▸ 0: SNTP is disabled; ▸ 1: SNTP is enabled. • <timezone>: time zone; range: [-11,13]; if SNTP is enabled, the <timezone> has to be set; • <SNTP server0>: optional parameter indicating the first SNTP server; • <SNTP server1>: optional parameter indicating the second SNTP server; • <SNTP server2>: optional parameter indicating the third SNTP server.	
Example	AT+CIPSNTPCFG=1,8,"cn.ntp.org.cn","ntp.sjtu.edu.cn","us.pool.ntp.org"	
Note	If the <SNTP server> parameters are not set, servers "cn.ntp.org.cn","ntp.sjtu.edu.cn", and "us.pool.ntp.org" will be used by default.	

5.2.29. AT+CIPSNTPTIME—Checks the SNTP Time

Query Command	AT+CIPSNTPTIME?

Response	+CIPSNTPTIME:<time> OK
Parameters	<time>: SNTP time For example, +CIPSNTPTIME:Thu Aug 04 14:48:05 2016 OK
Example	AT+CWMODE=1 //set as station mode AT+CWJAP="DemoAP","password" //connect to router, access the internet AT+CIPSNTPCFG=1,8 //set time zone AT+CIPSNTPTIME? //get time

5.2.30. AT+CIPDNS_CUR—Sets User-defined DNS Servers; Configuration Not Saved in the Flash

Commands	Query Command: AT+CIPDNS_CUR? Function: Get the current DNS server.	Set Command: AT+CIPDNS_CUR=<enable>[,<DNS server0>,<DNS server1>] Function: Set user-defined DNS servers.
Response	[+CIPDNS_CUR:<DNS server0>] [+CIPDNS_CUR:<DNS server1>] OK	OK
Parameters	• <enable>: ▸ 0: disable to use user-defined DNS servers; ▸ 1: enable to use user-defined DNS servers. • <DNS server0>: optional parameter indicating the first DNS server; • <DNS server1>: optional parameter indicating the second DNS serve.	
Example	AT+CIPDNS_CUR=1,"208.67.220.220"	
Note	• For command: AT+CIPDNS_CUR=0 (disable to use user-defined DNS servers), "208.67.222.222" will be used as DNS server by default. And the DNS server may change according to the configuration of the router which the chip connected to. • For command: AT+CIPDNS_CUR=1 (enable to use user-defined DNS servers, but the <DNS server> parameters are not set), servers "208.67.222.222" will be used as DNS server by default. • <DNS server0> and <DNS server1> cannot be set to the same server.	

5.2.31. AT+CIPDNS_DEF—Sets User-defined DNS Servers; Configuration Saved in the Flash

Commands	Query Command: `AT+CIPDNS_DEF?` Function: Get the user-defined DNS servers which saved in flash.	Set Command: `AT+CIPDNS_DEF=<enable>[,<DNS server0>,<DNS server1>]` Function: Set user-defined DNS servers.

Response	`[+CIPDNS_DEF:<DNS server0>]`
	`[+CIPDNS_DEF:<DNS server1>]`
	`OK`
		`OK`

Parameters	• `<enable>`:
	  ▸ `0`: disable to use a user-defined DNS server;
	  ▸ `1`: enable to use a user-defined DNS server.
	• `<DNS server0>`: optional parameter indicating the first DNS server;
	• `<DNS server1>`: optional parameter indicating the second DNS serve.	

Example	`AT+CIPDNS_DEF=1,"208.67.220.220"`	
Note	• This configuration will be saved in the user parameter area of flash. • For command: `AT+CIPDNS_DEF=0` (disable to use user-defined DNS servers), `"208.67.222.222"` will be used as DNS server by default. And the DNS server may change according to the configuration of the router which the chip connected to. • For command: `AT+CIPDNS_DEF=1` (enable to use user-defined DNS servers, but the <DNS server> parameters are not set), servers `"208.67.222.222"` will be used as DNS server by default. • `<DNS server0>` and `<DNS server1>` cannot be set to the same server.	

A. Appendix A

ESP8266 AT commands below will save the configuration changes in flash:

AT Command	Examples
Configuration Saved in the User Parameter Area in the Flash	
AT+UART_DEF	AT+UART_DEF=115200,8,1,0,3
AT+CWDHCP_DEF	AT+CWDHCP_DEF=1,1
AT+CIPSTAMAC_DEF	AT+CIPSTAMAC_DEF="18:fe:35:98:d3:7b"
AT+CIPAPMAC_DEF	AT+CIPAPMAC_DEF="1a:fe:36:97:d5:7b"
AT+CIPSTA_DEF	AT+CIPSTA_DEF="192.168.6.100"
AT+CIPAP_DEF	AT+CIPAP_DEF="192.168.5.1"
AT+CWDHCPS_DEF	AT+CWDHCPS_DEF=1,3,"192.168.4.10","192.168.4.15"
AT+SAVETRANSLINK	AT+SAVETRANSLINK_DEF=1,"192.168.6.10",1001
AT+CIPDNS_DEF	AT+CIPDNS_DEF=1,"208.67.220.220"
AT+SYSMSG_DEF	AT+SYSMSG_DEF=3
AT+CWCOUNTRY_DEF	AT+CWCOUNTRY_DEF=1,"CN",1,13
AT+CIPSSLCCONF	AT+CIPSSLCCONF=2
Configuration Saved in the System Parameter Area in the Flash	
AT+CWMODE_DEF	AT+CWMODE_DEF=3
AT+CWJAP_DEF	AT+CWJAP_DEF="abc","0123456789"
AT+CWSAP_DEF	AT+CWSAP_DEF="ESP8266","12345678",5,3
AT+CWAUTOCONN	AT+CWAUTOCONN=1

⚠ ***Notice:***

- *Only when the configuration changes will the AT firmware write the new configuration into the flash. Therefore, users need not be concerned about wearing out the flash on repeated application of commands that set the same default configurations over and over again.*
- *For 512 KB + 512 KB Flash Map, the user parameter area is 0x7C000 ~ 0x80000, 16 KB;*
- *For 1024 KB + 1024 KB Flash Map: the user parameter area is 0xFC000 ~ 0x100000, 16 KB;*
- *The system parameter area is always the last 16 KB in the flash.*

B. Appendix B

Messages of ESP8266 AT are as below:

Messages	Description
ready	The AT firmware is ready.
ERROR	AT command error, or error occurred during execution.
WIFI CONNECTED	ESP8266 station connected to an AP.
WIFI GOT IP	ESP8266 station got IP address.
WIFI DISCONNECT	ESP8266 station disconnected from an AP.
busy s...	Busy sending. The system is sending data now, cannot accept the newly input.
busy p...	Busy processing. The system is in process of handling the previous command, cannot accept the newly input.
<conn_id>,CONNECT	A network connection of which ID is <conn_id> is established.
<conn_id>,CLOSED	A network connection of which ID is <conn_id> ends.
+IPD	Received network data.
+STA_CONNECTED:<sta_mac>	A station connects to the ESP8266 softAP.
+DIST_STA_IP:<sta_mac>,<sta_ip>	ESP8266 softAP distributes an IP address to the station connected.
+STA_DISCONNECTED:<sta_mac>	A station disconnects from the ESP8266 softAP.

C. 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-02.
- Details of the test steps, for example:

```
AT+CWMODE_CUR=1
OK
AT+GMR
AT version:0.23.0.0(Apr 24 2015 21:11:01)
SDK version:1.0.1
compile time:Apr 24 2015 21:19:31
OK
AT+CIPSTAMAC_DEF="14:CF:11:22:33:05"
OK
```

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

```
ets Jan  8 2013,rst cause: 1, boot mode: (3,3)
load 0x40100000, len 26336, room 16
tail 0
chksum 0xde
load 0x3ffe8000, len 5672, room 8
tail 0
chksum 0x69
load 0x3ffe9630, len 8348, room 8
tail 4
chksum 0xcb
csum 0xcb
SDK version: 0.9.1
addr not ack when tx write cmd
mode : sta(18: fe: 34: 97: d5: 7b) + softAP(1a: fe: 34: 97: d5: 7b)
```

Espressif IOT Team

www.espressif.com