ESP8266 WiFi extension Design

Virgile Neu

June 3, 2017
version 1.0

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Contents

1 Introduction

The ESP8266 chip is a WiFi module with a AT command mode. It can be used for single connection or multiple connections (server), either TCP or UDP.
It has 4 pins of interest plus `VCC` and `GND` : `Chip Enable`, `RESET`, `Rx` and `Tx`. The picture 1 below show the esp8266 board with it's connectivity.

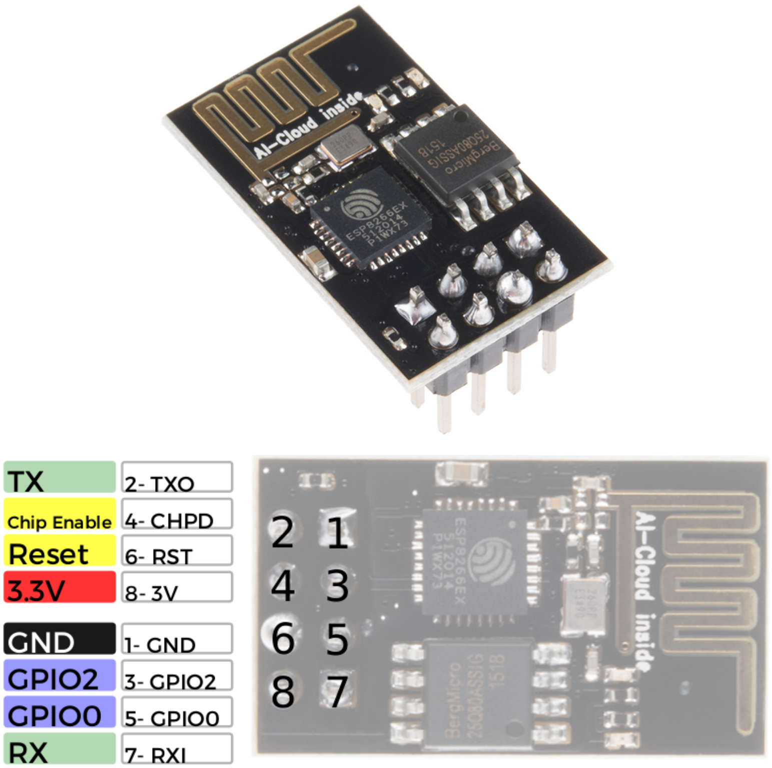


Figure 1: The ESP8266 module

The goal is to make available this WiFi module to use on the FPGA and to make it easily usable.

2 Parameters

2.1 Default configuration

The default configuration depends on the firmware version and can be changed at any time by AT commands with the _DEF modifier (e.g. AT+UART_DEF). Most of the parameters can be set that way, or customise at run time without changing the stored settings with the _CUR command modifier.

2.2 Serial Parameters

The ESP8266 WiFi module uses UART communication to transmit information with the FPGA.

The UART works as described on figure 2. It starts with the start bit, always '0', then comes the data (here 8 bits), least significant bit first, then the parity bit (if set), and then 1 or 2 stop bit, always '1'.

The ESP8266 supports a lot of different settings :

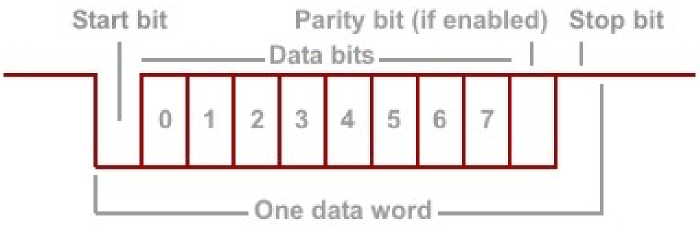


Figure 2: UART data transfert.

2.2.1 Data bits

- 5 bits
- 6 bits
- 7 bits
- 8 bits

2.2.2 Baud rates

Unlike the HC05, it supports a continuous range of baud rates, between 300 to 115200*40 bits/s.

2.2.3 Stop bit

- 1 bit
- 1.5 bit
- 2 bit

2.2.4 Parity bit

- None
- `Odd` parity
- `Event` parity

2.2.5 Flow control

- No flow control
- enable `Request To Send`
- enable `Clear To Send`
- enable both `RTS` and `CTS`

The ESP8266 has on chip `Tx/Rx` FIFO of 128 Bytes.

2.3 WiFi Parameters

2.3.1 Router connection

It can connect to routers with several security :

- `OPEN`,
- `WEP`,
- `WPA_PSK`,
- `WPA2_PSK`,
- `WPA_WPA2_PSK`.

The connection to routers with the `WPA2_Entreprise` security is not available with the current version of the AT command set.

2.3.2 Connection mode

The ESP8266 has three WiFi connection modes :

- `Station` mode : Client only
- `SoftAP` mode : Server only
- `SoftAP` + `Station` mode

3 Design Choices

Here I will show how the extension will look like, see figure 3. It will consist of Four parts:

- `Registers`, to store configuration, status and other things,
- A `FIFO_OUT` to send data from the CPU to the `UART` custom interface,
- A `FIFO_IN` to receive data from the `ESP8266`,
- A custom `UART` interface to communicate to the `ESP8266`.

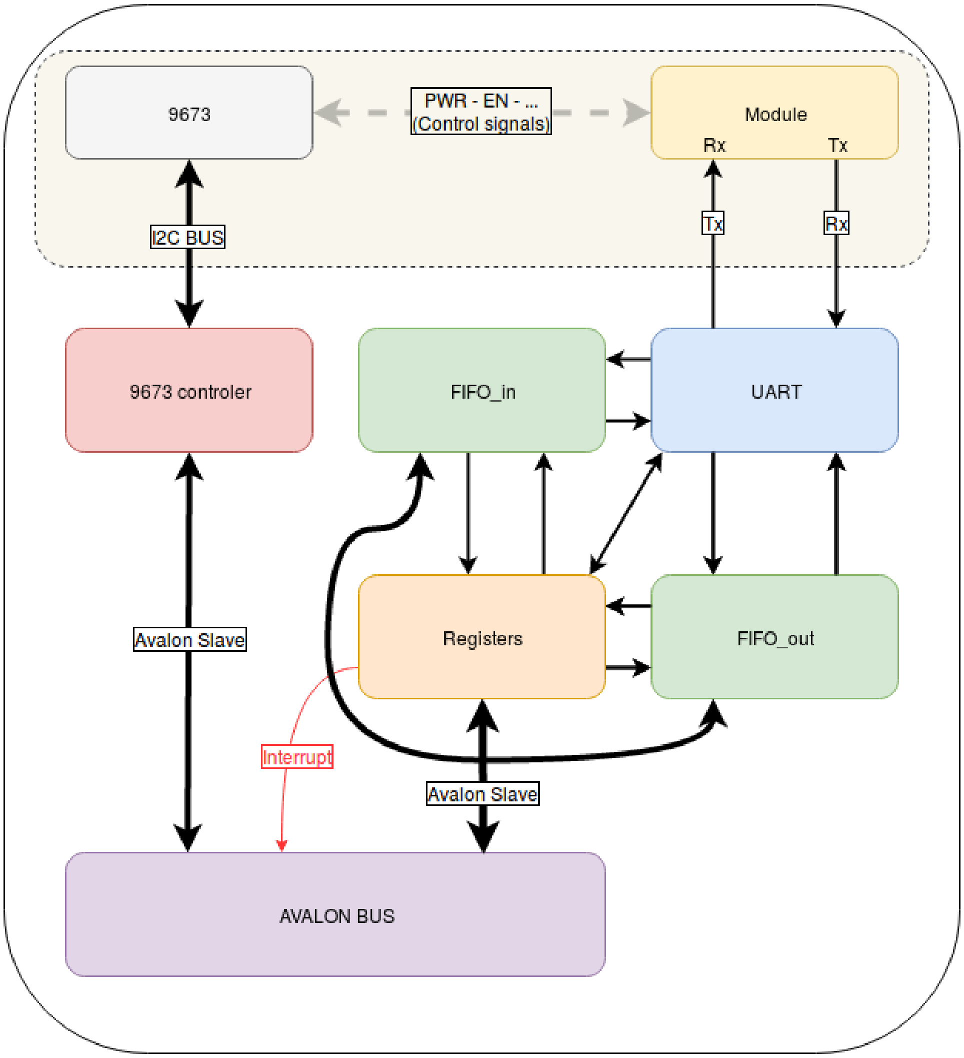


Figure 3: High level block diagram of the ESP8266 extension.

3.1 Registers

The registers will have height registers :

- A control register `CTRL`,
- A status register `STATUS`,
- A register for the `UART` waiting cycles (depends on the `UART rate`),

- The `FIFO_out_data` register,
- The `FIFO_out_free_space` register,
- The `FIFO_in_data` register,
- The `FIFO_in_pending_data` register.
- The `reset_FIFO` register.

Here is the register map in table 1 below.

Table 1: Register map of the Registers component.

#	addr	31..8	7	6	5	4	3	2	1	0	R/W
0	0x00	Unused			UART_CTRL			I_ENABLE		UART_ON	R/W
1	0x04	Unused							i_pending		R/W
2	0x08	UART_wait_cycles									R/W
3	0x0C	ignored	FIFO_out_data								W
4	0x10	FIFO_out_free_space									R
5	0x14	zeros	FIFO_in_data								R
6	0x18	FIFO_in_pending_data									R
7	0x1C	Unused							Reset_out	Reset_in	W

UART_CTRL		
5	4	3
Parity_bit		Stop_bit

I_ENABLE	
2	1
i_dropped	i_received

The role of each bit is described below :

- 0x00 :
  - `UART_ON` : Specifies if the `UART` will capture or send data or if it will stay off.
  - `i_reveived` : Specifies if the device can send interrupts request when receiving data from the `ESP8266`.
  - `i_dropped` : Specifies if the device can send interrupts request when some data is dropped.
  - `stop_bit` : Specifies the number of stop bit, '0' for 1, '1' for 2.
  - `parity_bit` : Specifies the parity bit, "00" for `None`, "10" for `Even` and "11" for `Odd`.
- 0x04 :

  - `i_pending` : Tells if there is an interrupt waiting to be served by the CPU. The CPU must clear it by software when serving the interrupt. Bit 0 is for i_received, bit 1 is for i_dropped. Writing '1' to any of the two bits has no effect.

- 0x08 : `UART_wait_cycles` : Specifies to the `UART` how many cycles it should wait before capturing the values during the transfer. The values to put are described in the table 2 below for a 50MHz clock.

- 0x0C : `FIFO_out_data` : Address to write to send data to the `ESP8266` through the `FIFO_out`. The write must has the `byte_enable` signal equal to "0001".

- 0x10 : `FIFO_out_free_space` : Number of free words (10 bits) in the `FIFO_out`.

- 0x14 : `FIFO_in_data` : Address to read to receive data from the `ESP8266` through the `FIFO_in`.

- 0x18 : `FIFO_out_free_space` : Number of waiting words (11 bits) in the `FIFO_in`.

- 0x1C :

  - `Reset_in` : Write only bit to clear the `FIFO_in`.
  - `Reset_out` : Write only bit to clear the `FIFO_out`.

The value to put in the `UART_wait_cycles` registers depend on the desired `UART baud rate`, and is computed with the following formula.

$$\begin{aligned} wait\_cycles &= \frac{time\_per\_bit}{time\_per\_cycles} \\ &= \frac{\frac{1}{baud\_rate}}{clk\_period} \\ &= \frac{clk\_freq}{baud\_rate} \end{aligned}$$

For 4800 bits/s of baud rate we have.

$$\begin{aligned} wait\_cycles &= \frac{clk\_freq}{baud\_rate} \\ &= \frac{50 \cdot 10^6}{4800} = 10416.667 \quad clk\_cycles \end{aligned}$$

The rounding doesn't matter.

Table 2: UART_wait_cycles values for a given UART.

UART_Rate	wait_cycles value (decimal)
4800 bits/s	10416 clk_cycles
9600 bits/s	5207 clk_cycles
19200 bits/s	2604 clk_cycles
38400 bits/s	1302 clk_cycles
57600 bits/s	868 clk_cycles
115200 bits/s	434 clk_cycles
230400 bits/s	217 clk_cycles
460800 bits/s	109 clk_cycles
921600 bits/s	54 clk_cycles
1382400 bits/s	36 clk_cycles

The ports of the Registers component are described on figure 4 below.

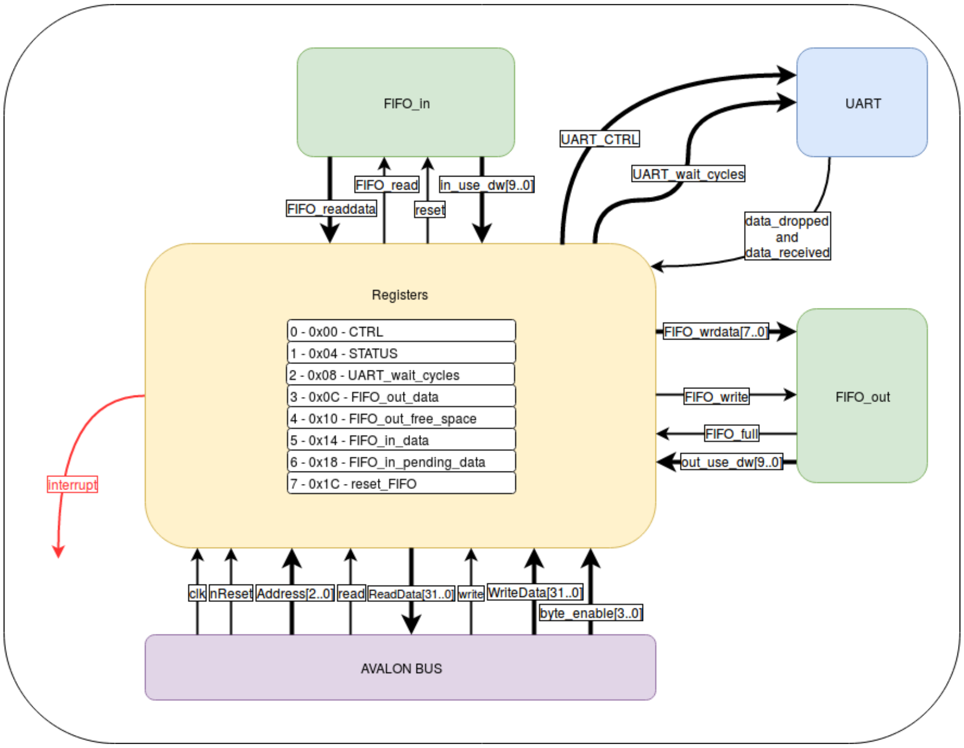


Figure 4: Ports description of the Registers component.

3.2 FIFO_out

For the FIFO_out we will use the FIFO available in the IP catalogue of Quartus with the following configurations :

- Width = 8 bits,
- Depth = 1024 (biggest size with only one M10k element),
- control signals :
  - use_dw[] (10 bits),
  - empty,
  - asynchronous clear;
- Show ahead FIFO mode,
- Auto memory block type,

- No optimisation or circuitry protection.

The ports of the FIFO_out component are described on figure 5.

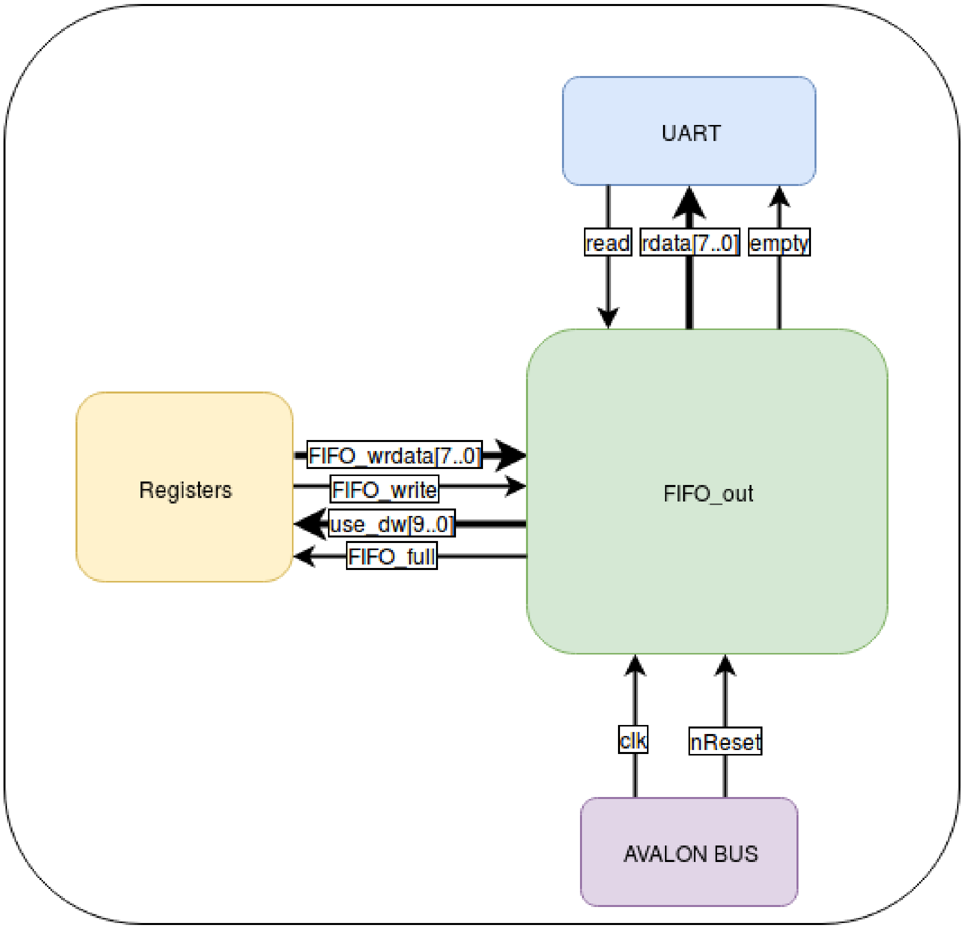


Figure 5: Ports description of the FIFO_out component.

3.3 FIFO_in

For the FIFO_in we will also use the FIFO available in the IP catalogue of Quartus with almost the same configurations :

- Width = 8 bits,
- Depth = 1024 (biggest size with only one M10k element),
- control signals :
    - use_dw[] (10 bits),
    - full,
    - asynchronous clear;
- Normal synchronous FIFO mode,
- Auto memory block type,
- No optimisation or circuitry protection.

The ports of the FIFO_in component are described on figure 6.

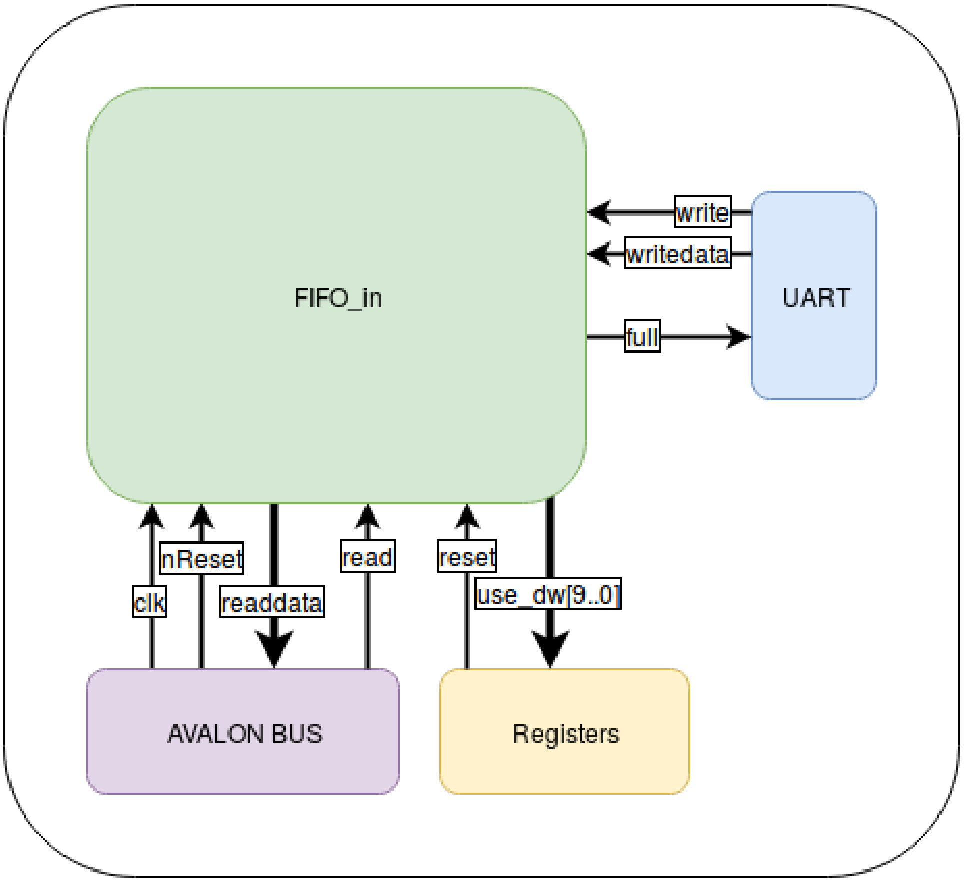


Figure 6: Ports description of the FIFO_in component.

3.4 UART

The UART will be the part communicating with the ESP8266 module. It will send whenever it can while the FIFO_out isn't empty, and whenever it receives information, it will recompose the words, perform the parity check (if set) and send the correct words to the FIFO_in.
The ports of the UART component are described on figure 7.

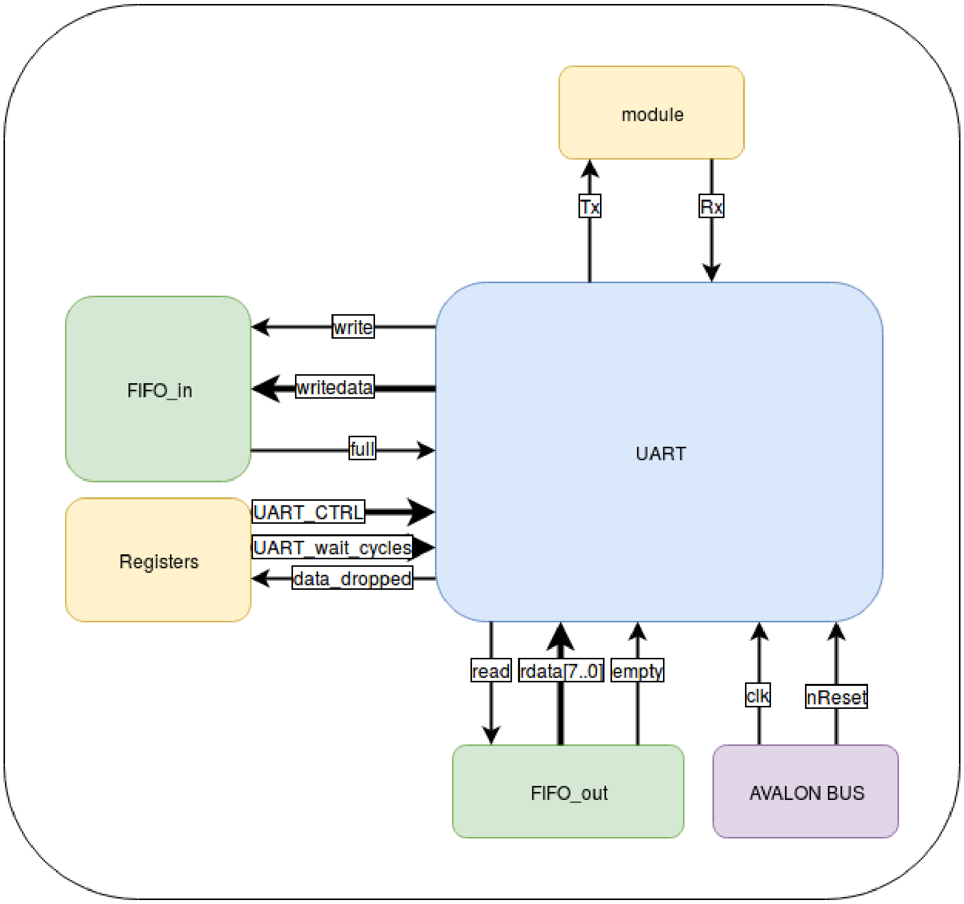


Figure 7: Ports description of the UART component.

4 Pinout

The external connectivity of the device is described on table 3.

Table 3: Pinout table of the device.

signal name	connectivity
BLT_RxD	GPIO_0 23 -- FPGA PIN_T11
BLT_TxD	GPIO_0 25 -- FPGA PIN_AF6
BLT_State	PCA9673 via Avalon Bus
BLT_EN	
BLT_ATSel	

5 States Machines

This section describes the several states machines used in the extension.

5.1 UART

5.1.1 Transmitting State Machine

The figure 8 below describe the state machine used for transmitting data. It consists of 5 states : WAITING, START, SENDING, PARITY and STOP states. It starts at the WAITING states, and wait for data to be available in the FIFO_out. Once data is available, it issue a read to the FIFO_out and go to the start states. During the start state, it outputs the '0' value, as specified in the UART protocol, and store the data from the FIFO_out_readdata during the first cycle in this state. Once it has waited enough, it goes to sending. During sending state, it will send bit after bit, every time waiting the good amount of time. Oncei all the 8 bit of data are sent, it will either go to STOP if the parity is disabled (Parity_bit = "00") or to PARITY if it is enable. In the PARITY state, it will output the parity value (odd or even) for the right amount of time, and then go to the STOP state. In the STOP state, it will output 1 or 2 bit at '1', depending on the settings of the Stop_bit, and then go to the WAITING state, ready to transfer again.

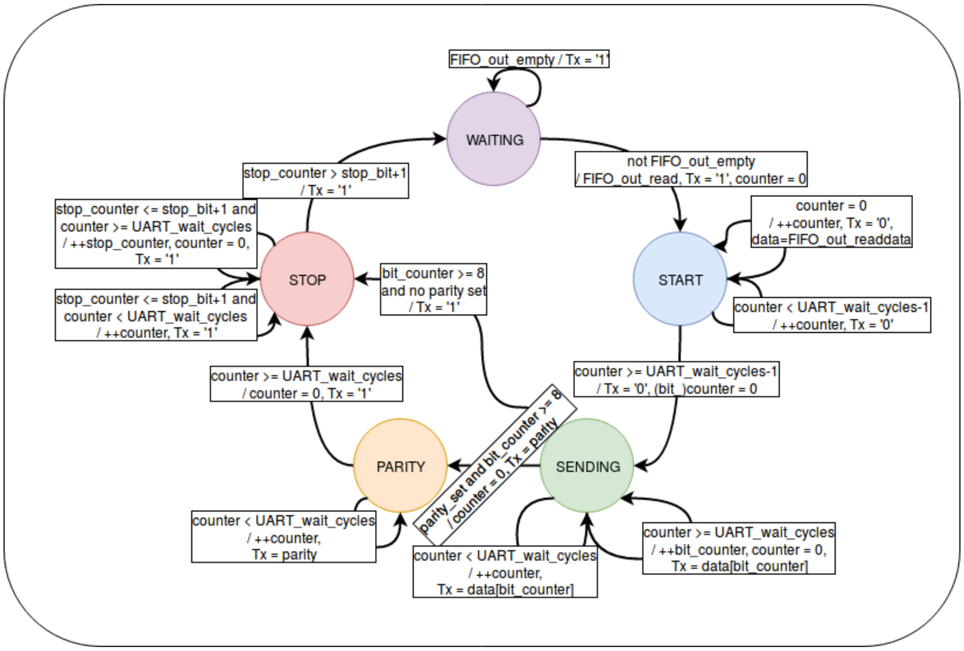

FIFO_out_empty / Tx = '1'
WAITING
stop_counter > stop_bit+1
/ Tx = '1'
not FIFO_out_empty
/ FIFO_out_read, Tx = '1', counter = 0
stop_counter <= stop_bit+1 and
counter >= UART_wait_cycles
/ ++stop_counter, counter = 0,
Tx = '1'
counter = 0
/ ++counter, Tx = '0',
data=FIFO_out_readdata
STOP
bit_counter >= 8
and no parity set
/ Tx = '1'
START
stop_counter <= stop_bit+1 and
counter < UART_wait_cycles
/ ++counter, Tx = '1'
counter < UART_wait_cycles-1
/ ++counter, Tx = '0'
counter >= UART_wait_cycles
/ counter = 0, Tx = '1'
counter >= UART_wait_cycles-1
/ Tx = '0', (bit_)counter = 0
parity_set and bit_counter >= 8
/ counter = 0, Tx = parity
PARITY
SENDING
counter < UART_wait_cycles
/ ++counter,
Tx = parity
counter < UART_wait_cycles
/ ++counter,
Tx = data[bit_counter]
counter >= UART_wait_cycles
/ ++bit_counter, counter = 0,
Tx = data[bit_counter]

Figure 8: State machine used for sending one word (8 bits) to the ESP8266.

5.1.2 Receiving State Machine

The figure 9 below describe the state machine used for receiving data. It has 4 states : WAITING, START, RECEIVING and PARITY. It starts at the WAITING states, and wait until the BLT_Tx is '0' (start bit). Then we wait for half the cycles to wait in the START state in order to capture each bit in correctly and not just when they are supposed to go up (in order to avoid wrong bits), continuously checking that the start bit is still on (BLT_Tx = '0'). Then we go to the RECEIVING state, where we wait for a full wait before capturing each bit. There is a transition back to the WAITING state with a big condition, it is to catch an error in the start bit during the first half of the first wait round. Once we received all the bits, we either go to the parity check in the PARITY state if enable or directly to the WAITING state and writing the data to the FIFO_in if it is not full. If we go to the PARITY state, we check if the parity of the data we received is correct, and if it is we write it to the FIFO_in if it is not full, and else we discard it. Then we go back to WAITING.

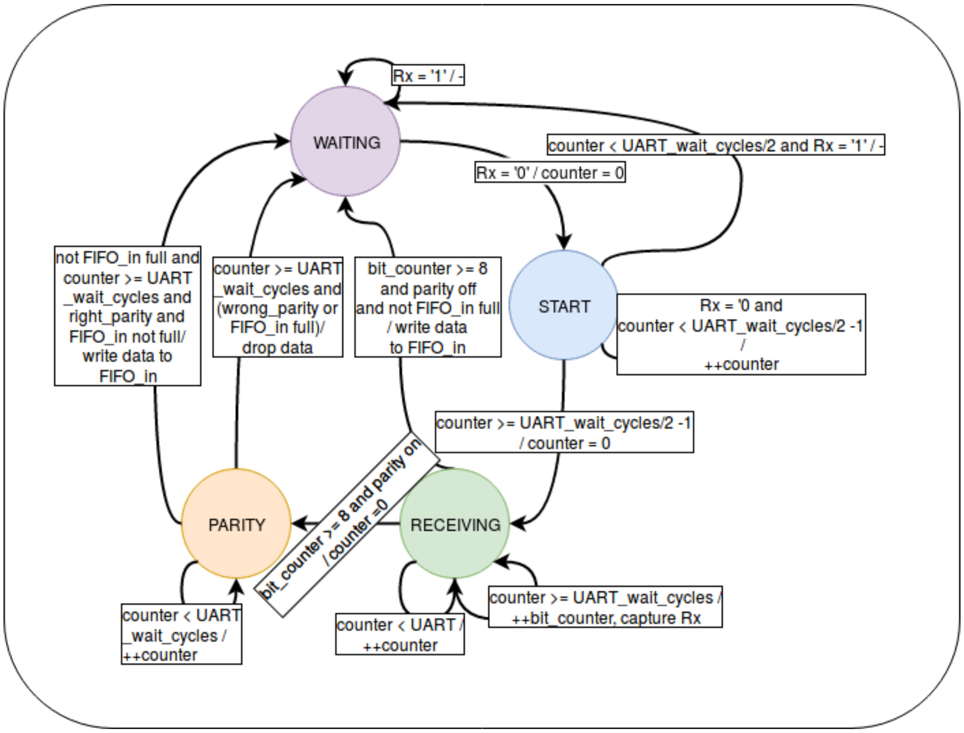
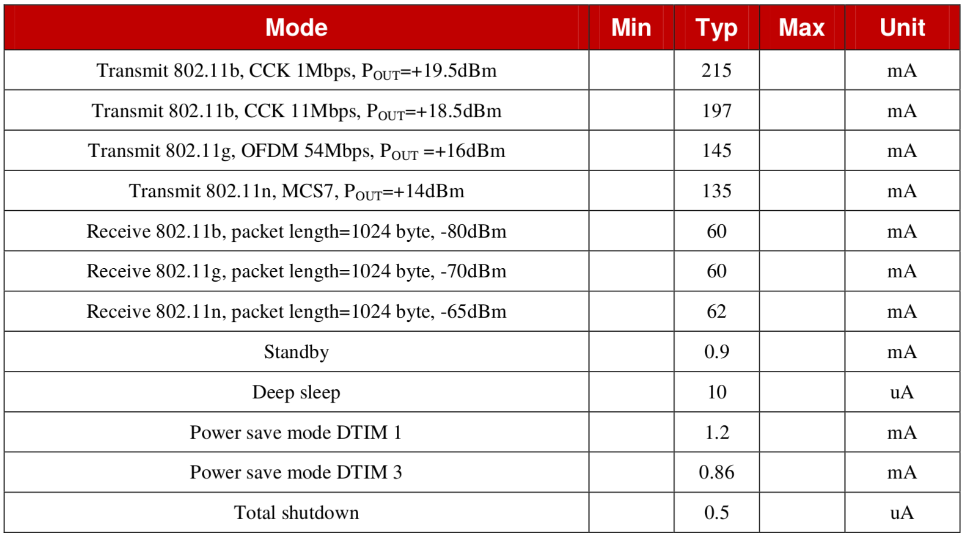

Rx = '1' / -
WAITING
counter < UART_wait_cycles/2 and Rx = '1' / -
Rx = '0' / counter = 0
not FIFO_in full and counter >= UART _wait_cycles and right_parity and FIFO_in not full/ write data to FIFO_in
counter >= UART _wait_cycles and (wrong_parity or FIFO_in full)/ drop data
bit_counter >= 8 and parity off and not FIFO_in full / write data to FIFO_in
START
Rx = '0 and counter < UART_wait_cycles/2 -1 / ++counter
counter >= UART_wait_cycles/2 -1 / counter = 0
bit_counter >= 8 and parity on / counter =0
PARITY
RECEIVING
counter < UART _wait_cycles / ++counter
counter < UART / ++counter
counter >= UART_wait_cycles / ++bit_counter, capture Rx

Figure 9: State machine used for receiving one word (8 bits) from the ESP8266.

6 Power consumption

The ESP8266 module specification document has the following table regarding power consumption:



Mode	Min	Typ	Max	Unit
Transmit 802.11b, CCK 1Mbps, P_{OUT}=+19.5dBm		215		mA
Transmit 802.11b, CCK 11Mbps, P_{OUT}=+18.5dBm		197		mA
Transmit 802.11g, OFDM 54Mbps, P_{OUT} =+16dBm		145		mA
Transmit 802.11n, MCS7, P_{OUT}=+14dBm		135		mA
Receive 802.11b, packet length=1024 byte, -80dBm		60		mA
Receive 802.11g, packet length=1024 byte, -70dBm		60		mA
Receive 802.11n, packet length=1024 byte, -65dBm		62		mA
Standby		0.9		mA
Deep sleep		10		uA
Power save mode DTIM 1		1.2		mA
Power save mode DTIM 3		0.86		mA
Total shutdown		0.5		uA

Figure 10: Table with the power consumption of the module.

7 Throughput

For the ESP8266, I was not able to have some good measure for the throughput. The module can't connect to the WPA2_Entreprise network of EPFL, so I used my smartphone as a router, which can't be trusted regarding performance. More over, to make measurement, I needed to build a server on another machine also connected to my smartphone, which was too much work to do.
According to the Espressif team[1] (the ones who made the module), the maximum speed they measure was :

- TCP Throughput: AT TCP passthrough Tx 303 Kbps; Rx 302Kbps @ baudrate 420000
- UDP Throughput: AT UDP passthrough Tx 250 Kbps; Rx 250 Kbps @ baudrate 420000

[1] http://bbs.espressif.com/viewtopic.php?t=2187

8 Other

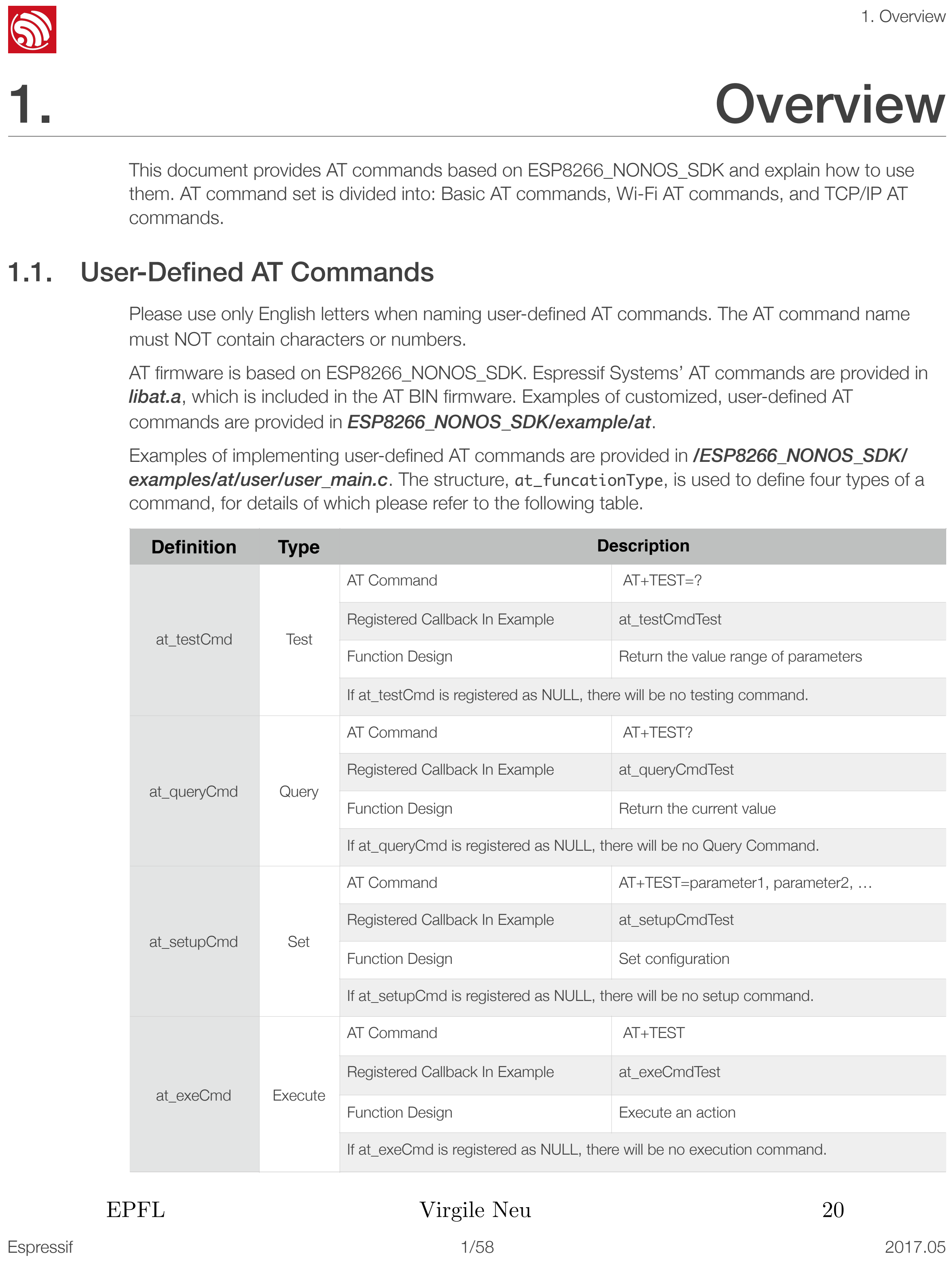
The ESP8266 module has a lot more to offer, since there is a real ARM 32 bit CPU inside with some GPIOs and ADCs for instance. All of that is not accessible as be only use it by UART without any SDKs, only with the AT commands. That way to use the module is sufficient to do some inter FPGA communication, or to access a remote server, but if one wants to use the SPI, ADC or many other things, he will have to flash another firmware with an SDK.

A Flashing the ROM

In order to use the AT commands, you need to flash the ROM to update the software. This document was written for the version of the software given in archive with this file.
In order to correctly flash your module, please refer to the other pdf ”`Flash_ESP8266_WiFi_module.pdf`”.

B AT Commands

Below you can find the section from the `ESP8266` instruction manual regarding the AT commands.



1. Overview

This document provides AT commands based on ESP8266_NONOS_SDK and explain how to use them. AT command set is divided into: Basic AT commands, Wi-Fi AT commands, and TCP/IP AT commands.

1.1. User-Defined AT Commands

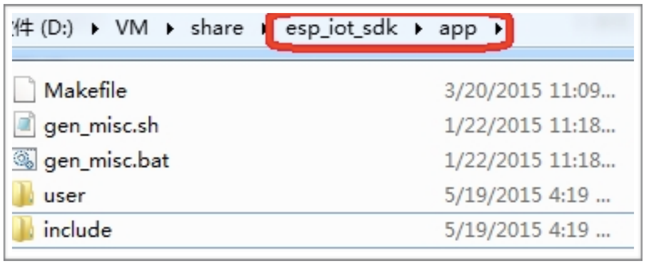
Please use only English letters when naming user-defined AT commands. The AT command name must NOT contain characters or numbers.

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.	

All the files in folder ***at*** should be copied to the folder ***app*** in ***ESP8266_NONOS_SDK*** if users need to compile the AT firmware.



件 (D:) ▸ VM ▸ share ▸ esp_iot_sdk ▸ app ▸

Makefile	3/20/2015 11:09...
gen_misc.sh	1/22/2015 11:18...
gen_misc.bat	1/22/2015 11:18...
user	5/19/2015 4:19 ...
include	5/19/2015 4:19 ...

For details please refer to ESP8266 Getting Started Guide.

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.

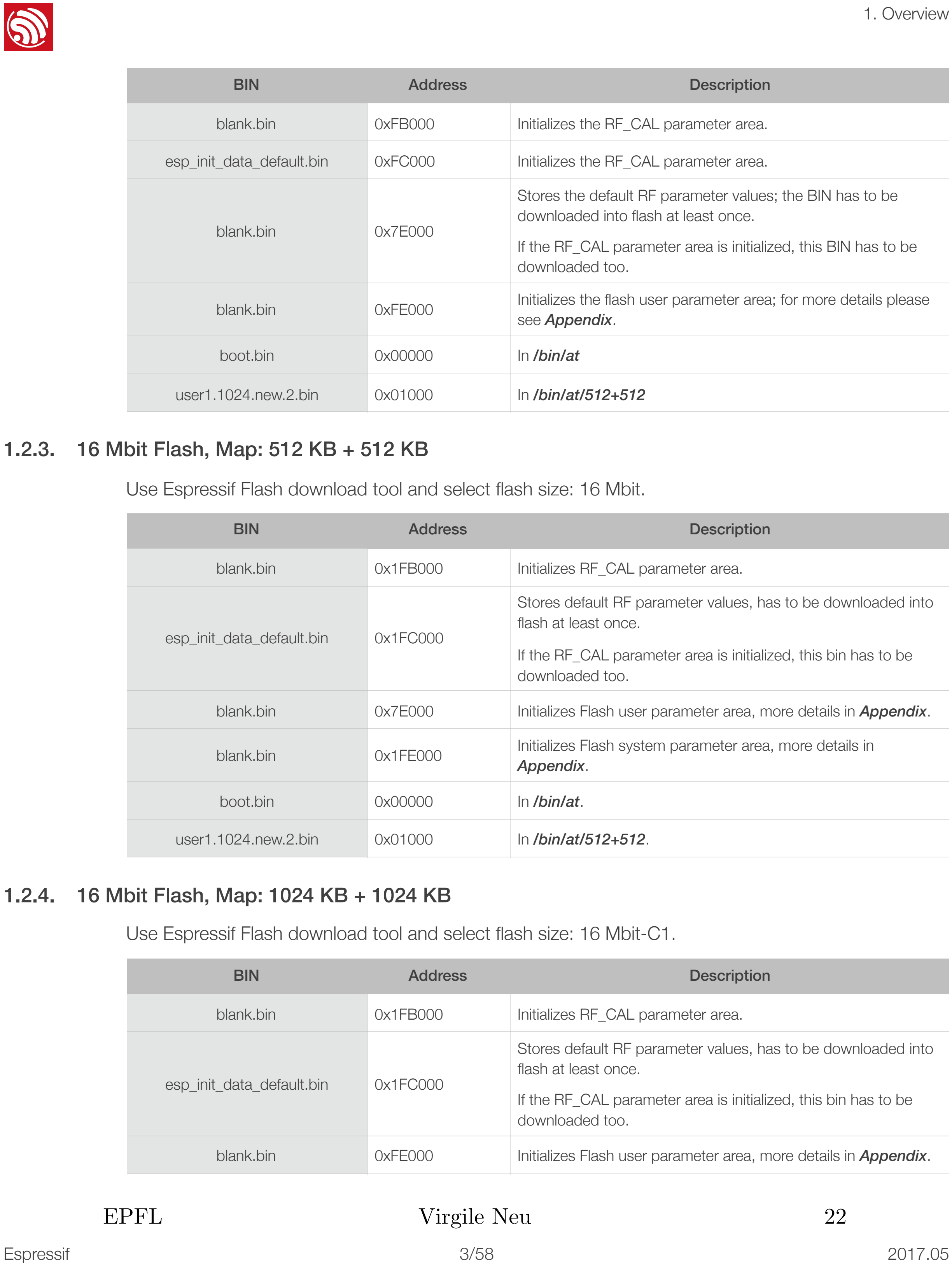
1.2.1. 4 Mbit Flash

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.
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***.

1.2.2. 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	Initializes the RF_CAL parameter area.
blank.bin	0x7E000	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	0xFE000	Initializes the flash user 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***

1.2.3. 16 Mbit Flash, Map: 512 KB + 512 KB

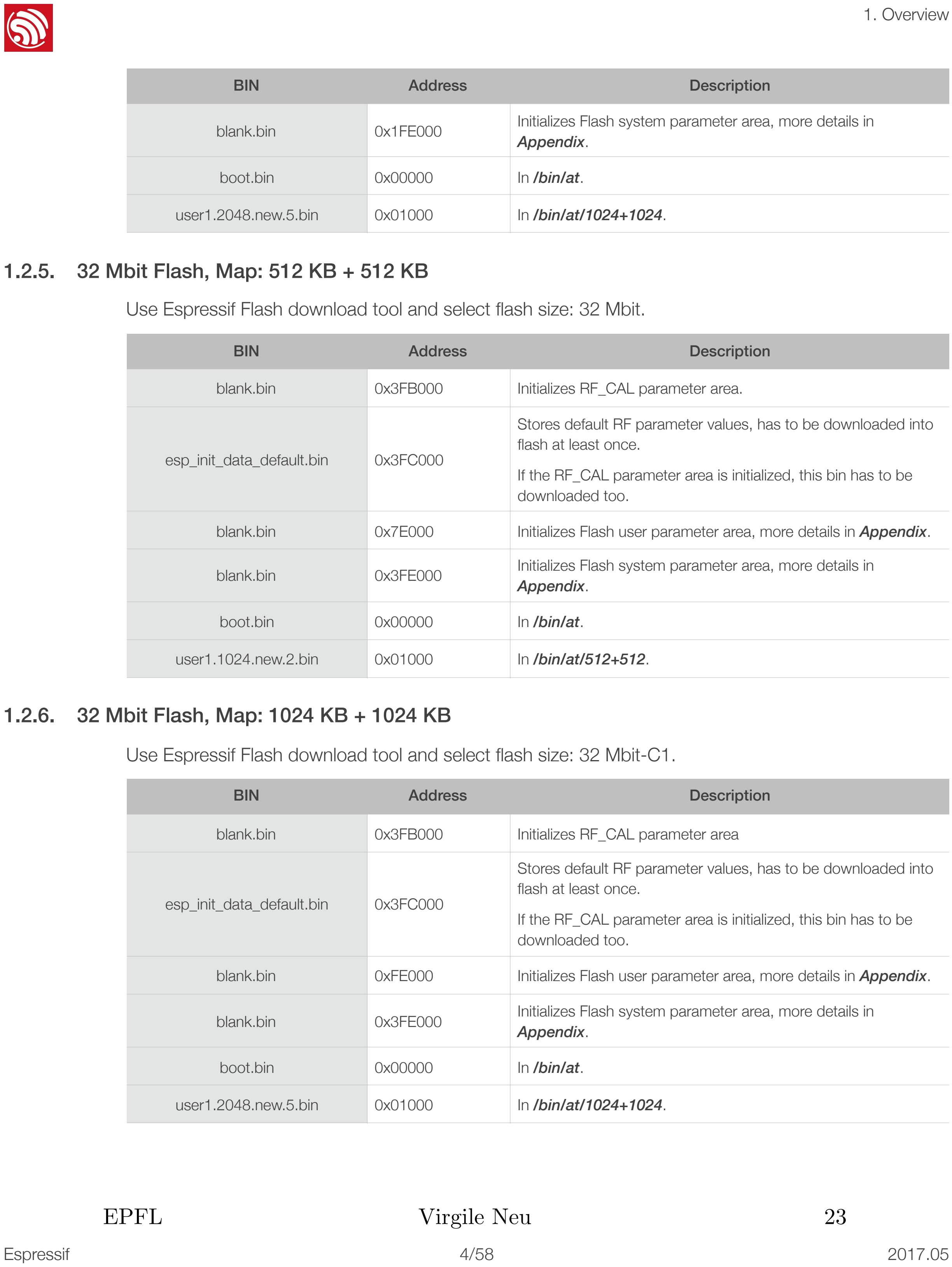
Use Espressif Flash download tool and select flash size: 16 Mbit.

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.
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***.

1.2.4. 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***.



BIN	Address	Description
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***.

1.2.5. 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***.

1.2.6. 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***.

⚠ ***Notes:***

- *Please make sure that correct BIN (**/ESP8266_NONOS_SDK/bin/at**) is already in the chip (ESP8266) 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.*
- *AT returns messages below to show status of the ESP8266 Station's Wi-Fi connection.*
  - `Wi-Fi CONNECTED`*: Wi-Fi is connected.*
  - `Wi-Fi GOT IP`*: the ESP8266 Station has got the IP from the AP.*
  - `Wi-Fi DISCONNECT`*: Wi-Fi is disconnected.*

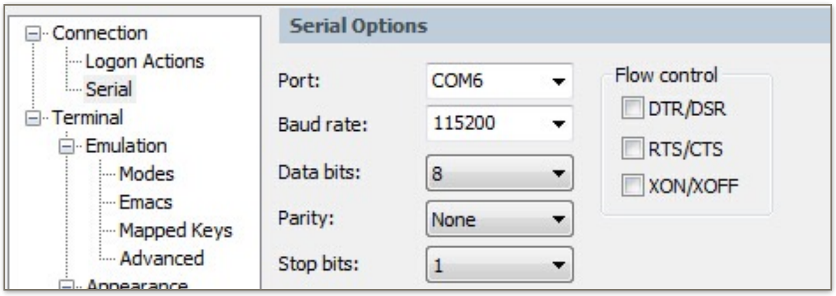
2. Command Description

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:***

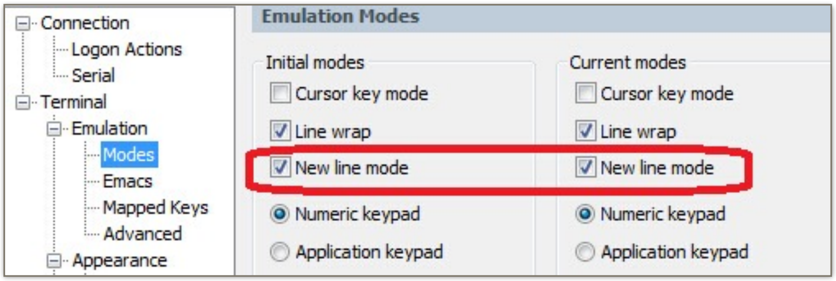
- *Not all AT commands support all four variations mentioned above.*
- *Square brackets [ ] designate the default value; it is either not required or may not appear.*
- *String values need to be included in double quotation marks, for example:* `AT+CWSAP="ESP756290","21030826",1,4`*.*
- *The default 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.*



Port: COM6
Baud rate: 115200
Data bits: 8
Parity: None
Stop bits: 1

Flow control
☐ DTR/DSR
☐ RTS/CTS
☐ XON/XOFF

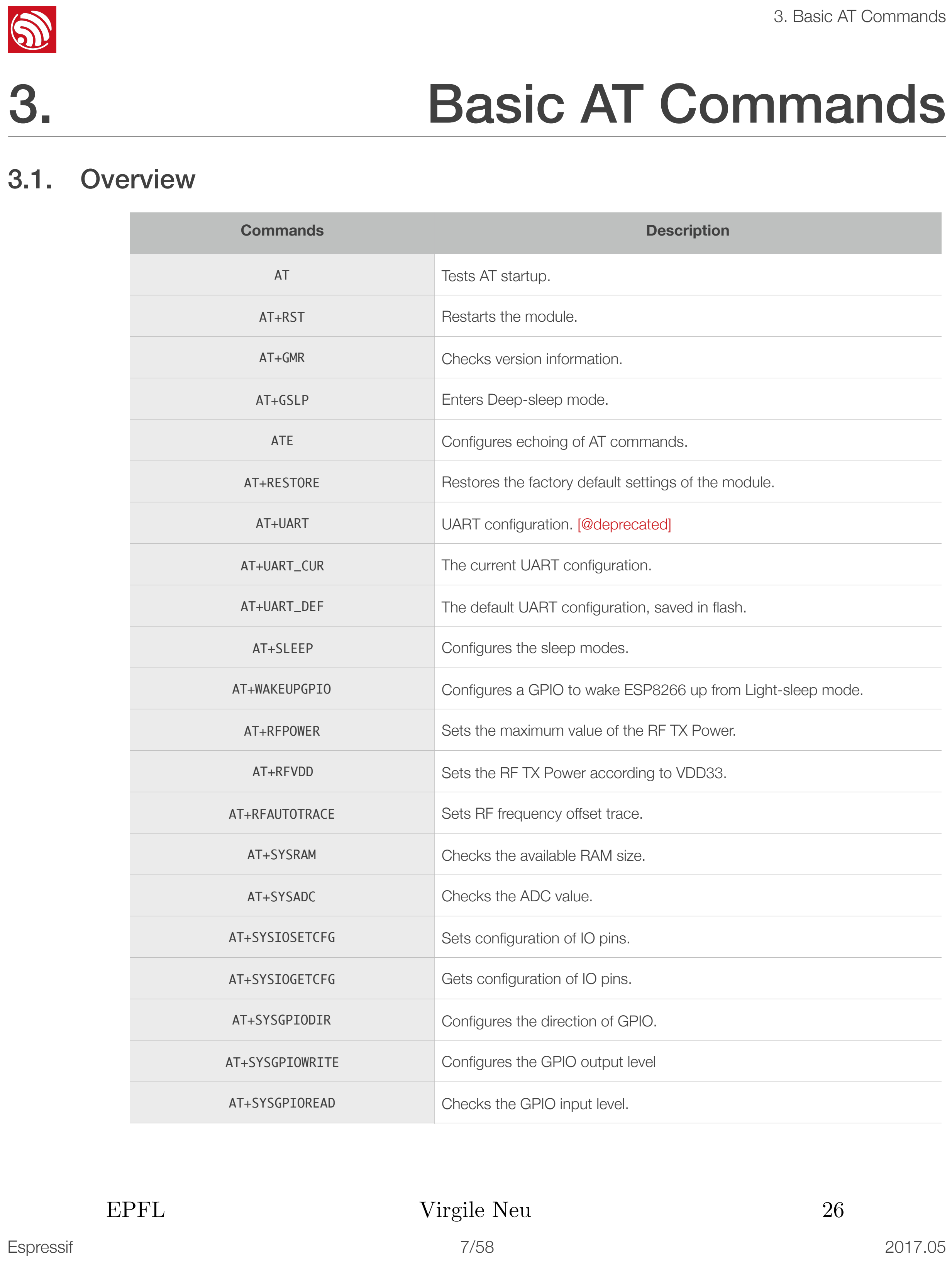
Figure 2-1. Configuration of Serial Options



Initial modes
☐ Cursor key mode
☑ Line wrap
☑ New line mode
● Numeric keypad
○ Application keypad

Current modes
☐ Cursor key mode
☑ Line wrap
☑ New line mode
● Numeric keypad
○ Application keypad

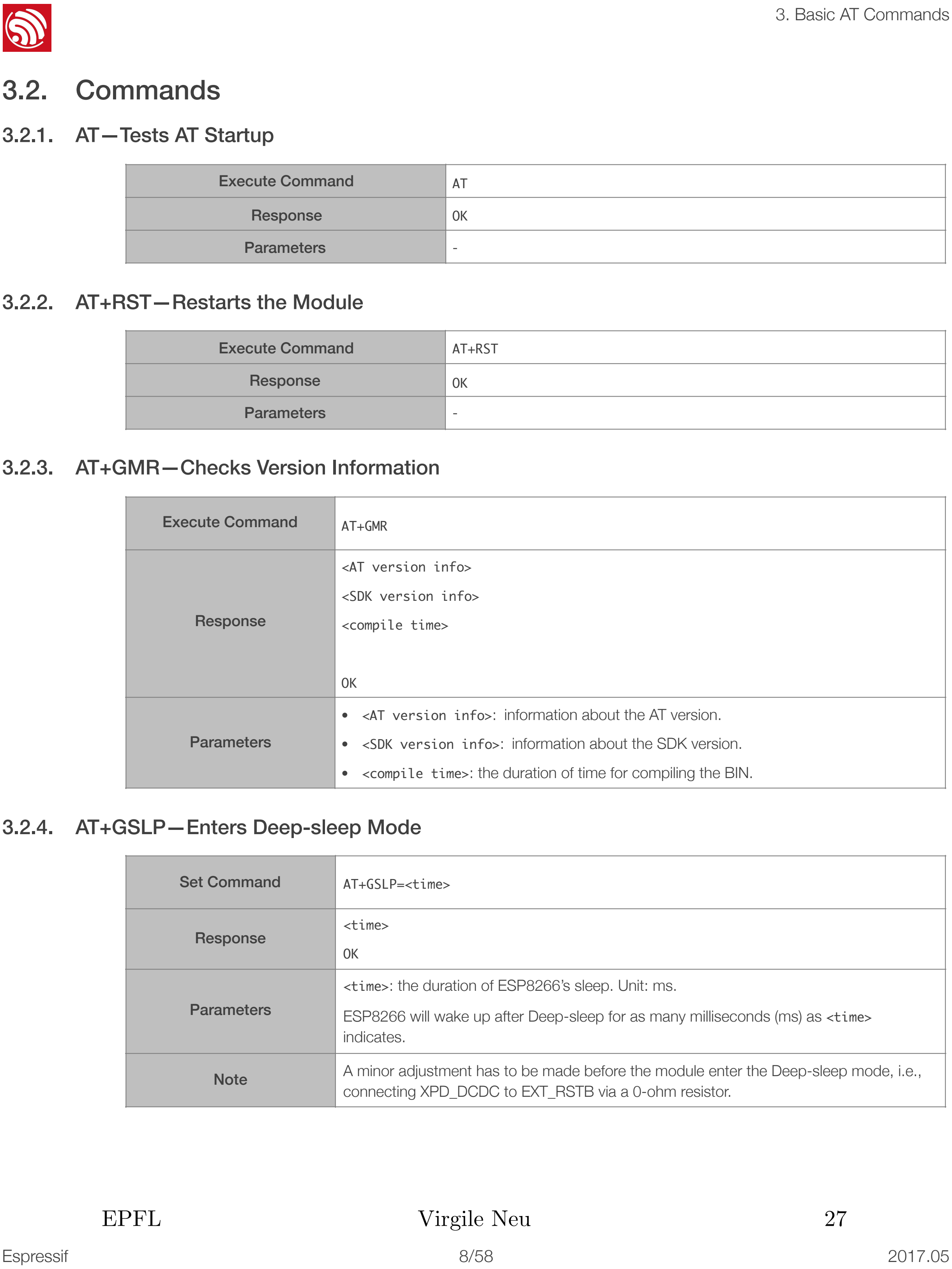
Figure 2-2. New Line Mode



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	UART configuration. [@deprecated]
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+RFAUTOTRACE	Sets RF frequency offset trace.
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.



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`

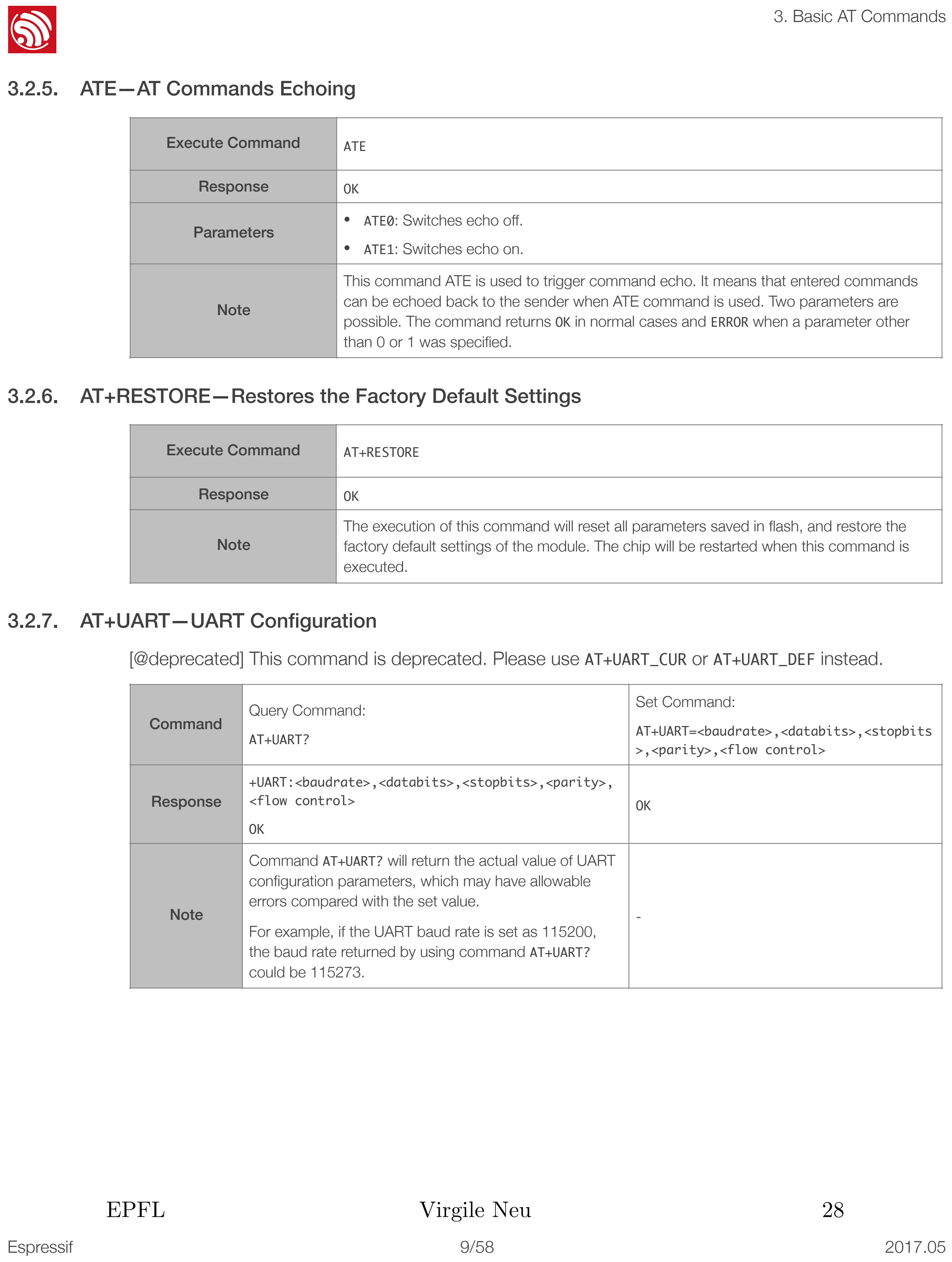
Parameters	• `<AT version info>`: information about the AT version. • `<SDK version info>`: information about the SDK version. • `<compile time>`: the duration of time for compiling the BIN.

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.

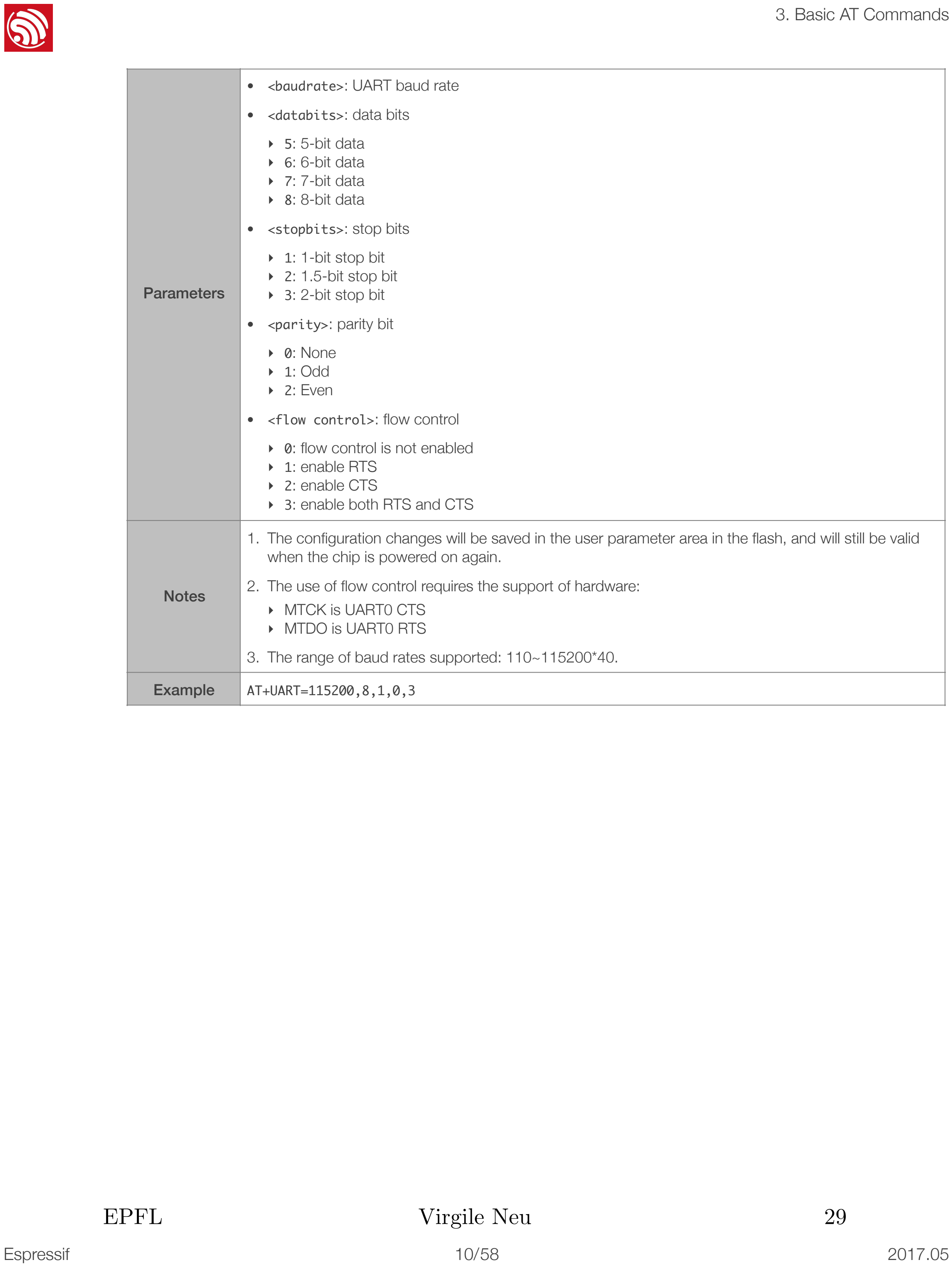
3.2.7. AT+UART—UART Configuration

[@deprecated] This command is deprecated. Please use `AT+UART_CUR` or `AT+UART_DEF` instead.

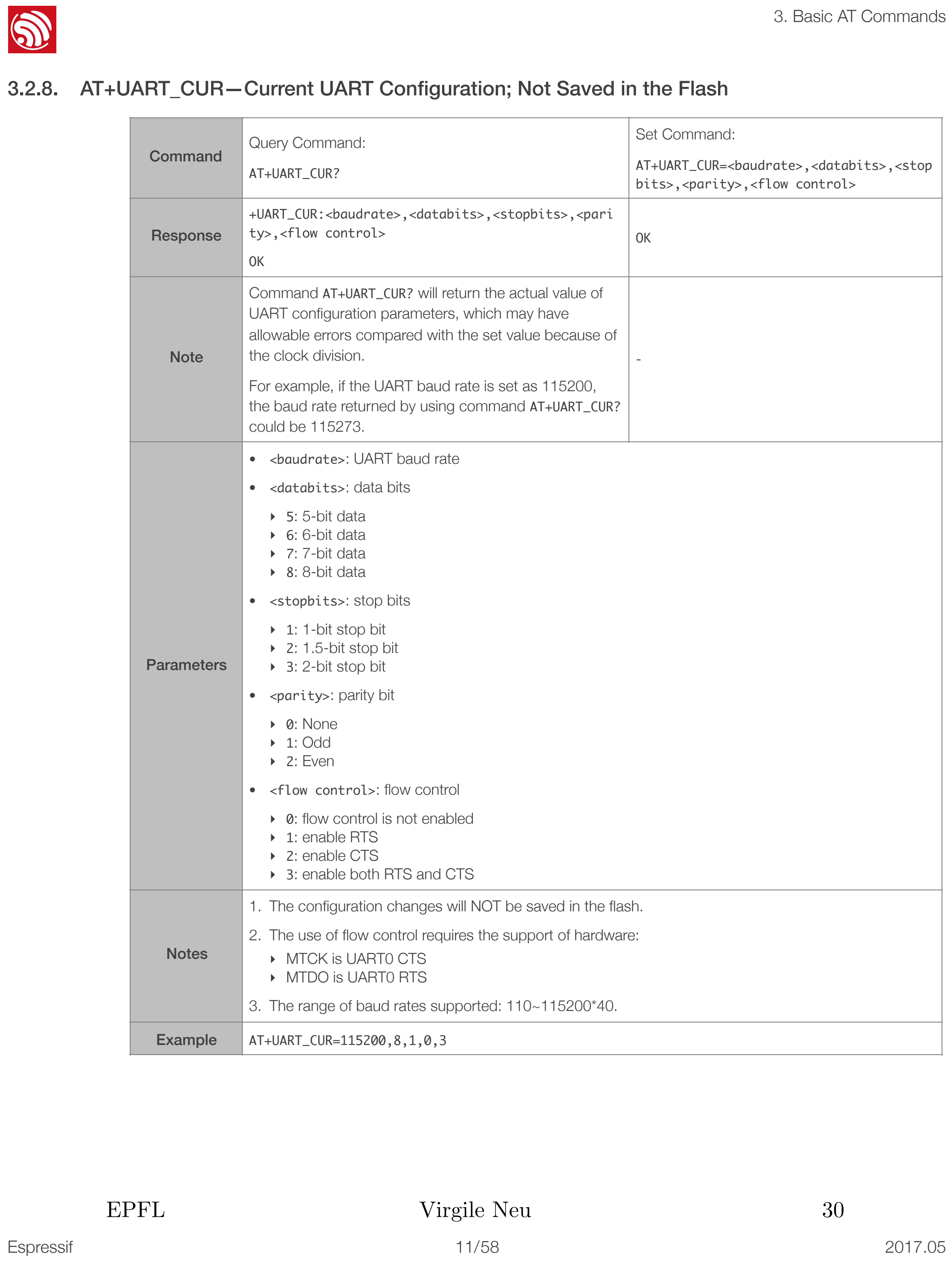
Command	Query Command: `AT+UART?`	Set Command: `AT+UART=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>`

Response	`+UART:<baudrate>,<databits>,<stopbits>,<parity>,<flow control>`
	`OK`
		`OK`

Note	Command `AT+UART?` will return the actual value of UART configuration parameters, which may have allowable errors compared with the set value. For example, if the UART baud rate is set as 115200, the baud rate returned by using command `AT+UART?` 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 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=115200,8,1,0,3`

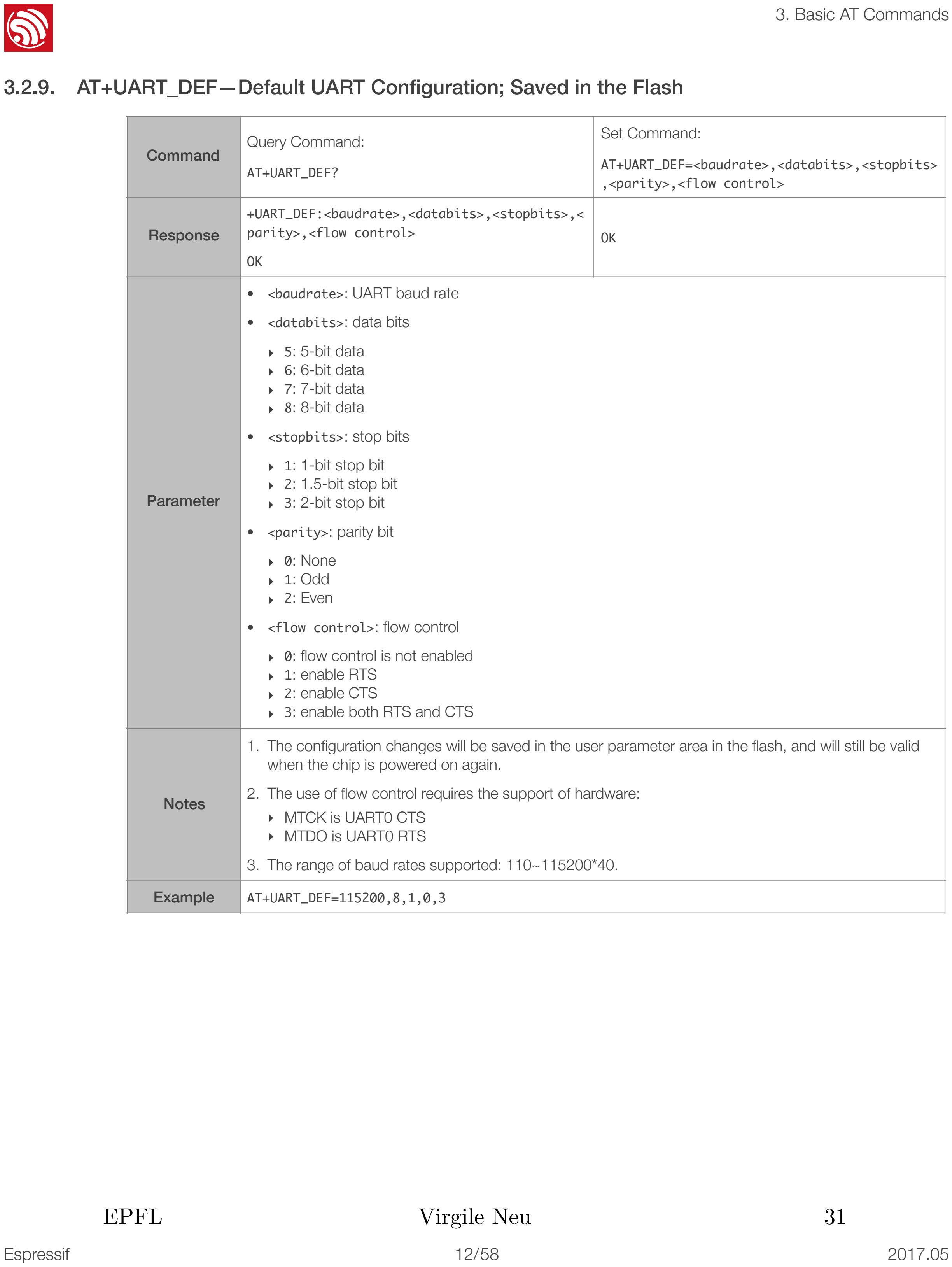


3.2.8. AT+UART_CUR—Current UART Configuration; Not Saved in the Flash

Command	Query Command: `AT+UART_CUR?`	Set Command: `AT+UART_CUR=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>`

Response	`+UART_CUR:<baudrate>,<databits>,<stopbits>,<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`	

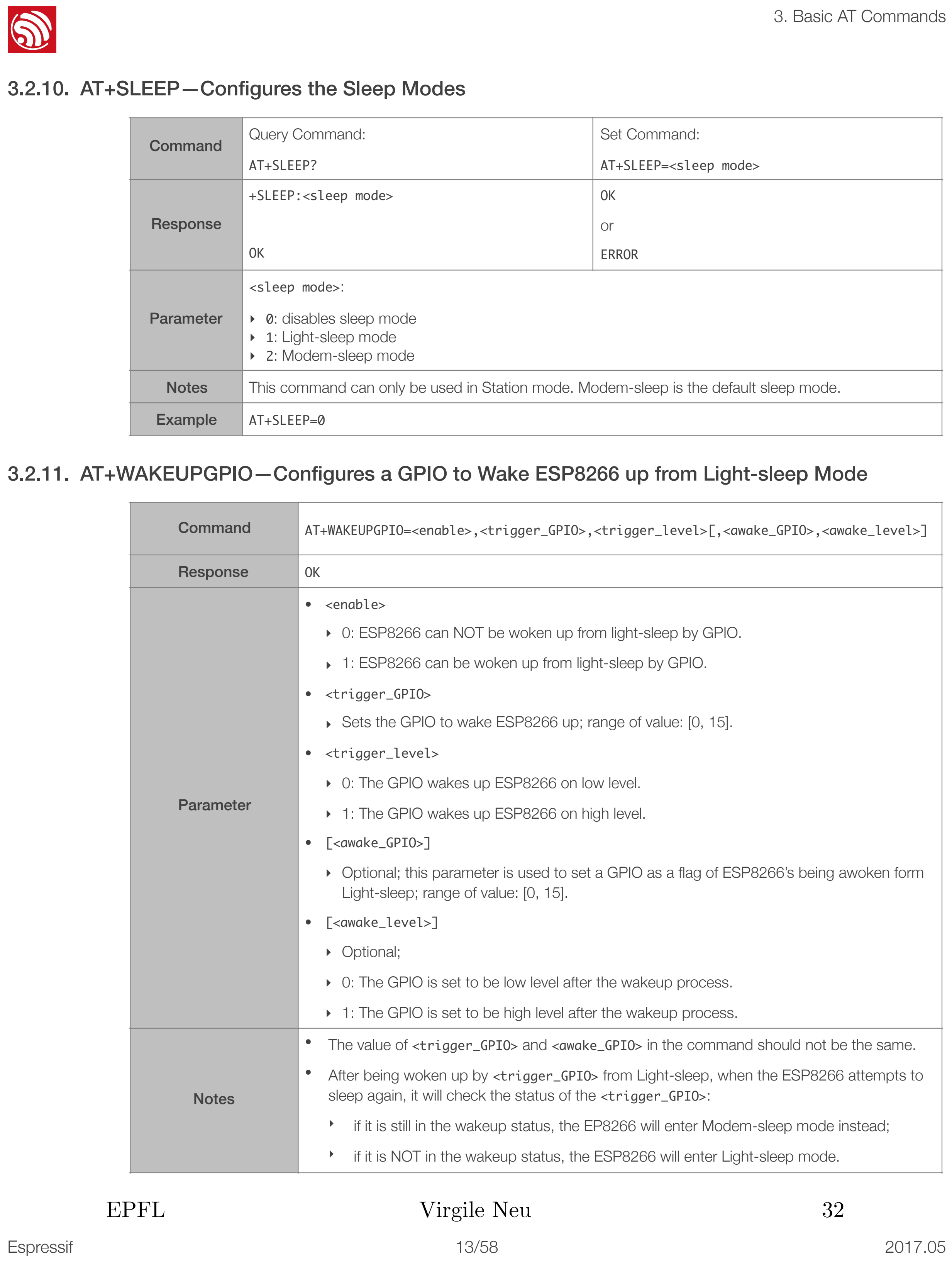


3.2.9. AT+UART_DEF—Default UART Configuration; Saved in the Flash

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`	



3.2.10. AT+SLEEP—Configures the Sleep Modes

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

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

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`	

3.2.11. 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	• 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`

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

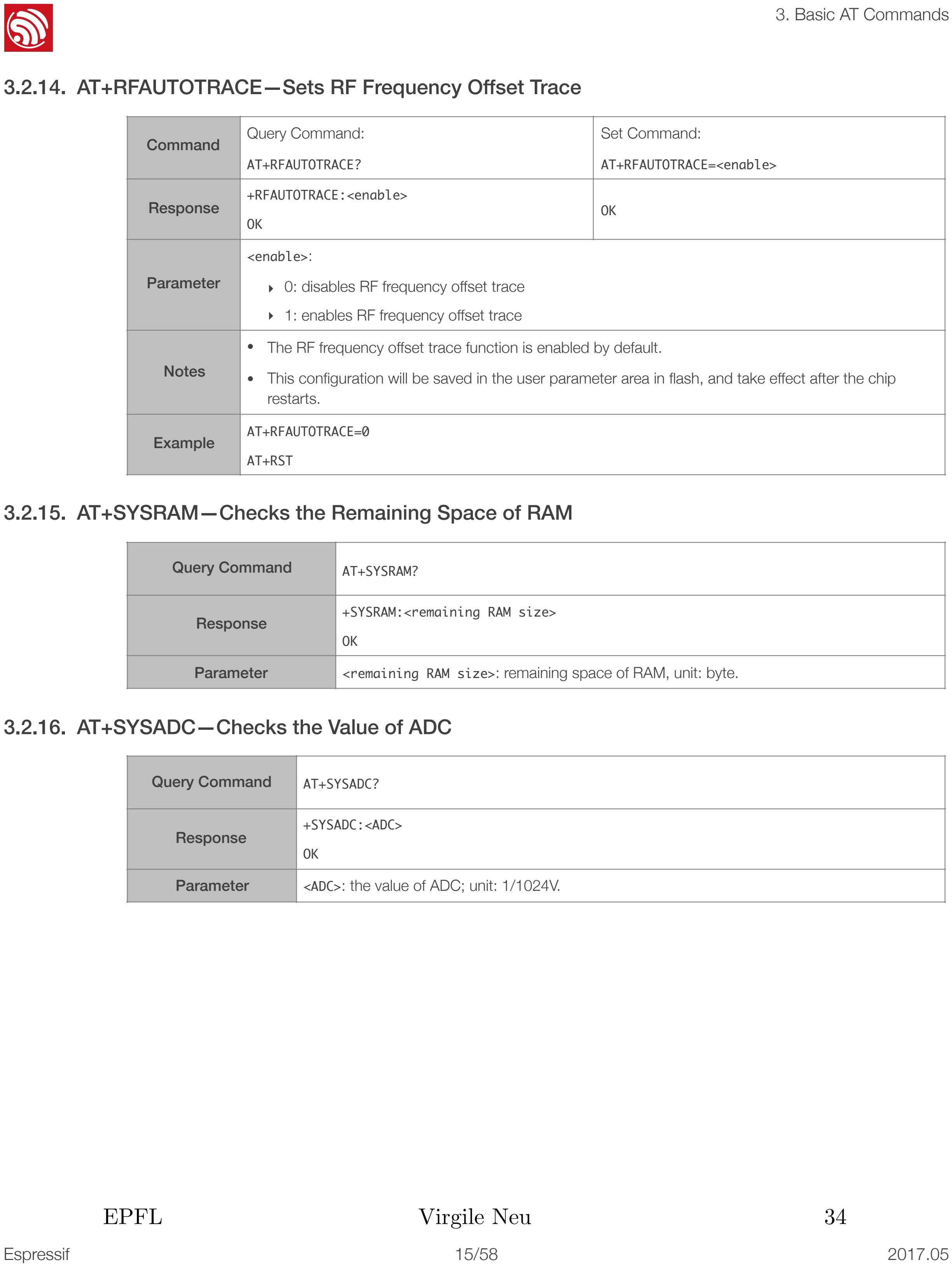
Set Command	`AT+RFPOWER=<TX Power>`
Response	`OK`
Parameter	`<TX Power>`: the maximum value of RF TX power; range: [0, 82]; unit: 0.25 dBm.
Note	This command sets the maximum value of ESP8266 RF TX power; it is not precise. The actual value could be smaller than the set value.
Example	`AT+RFPOWER=50`

3.2.13. 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`		



3.2.14. AT+RFAUTOTRACE—Sets RF Frequency Offset Trace

Command	Query Command: `AT+RFAUTOTRACE?`	Set Command: `AT+RFAUTOTRACE=<enable>`

Response	`+RFAUTOTRACE:<enable>`
	`OK`
		`OK`

Parameter	`<enable>`: ▸ 0: disables RF frequency offset trace ▸ 1: enables RF frequency offset trace	
Notes	• The RF frequency offset trace function is enabled by default. • This configuration will be saved in the user parameter area in flash, and take effect after the chip restarts.	

Example	`AT+RFAUTOTRACE=0`
	`AT+RST`	

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

Query Command	`AT+SYSRAM?`

Response	`+SYSRAM:<remaining RAM size>`
	`OK`

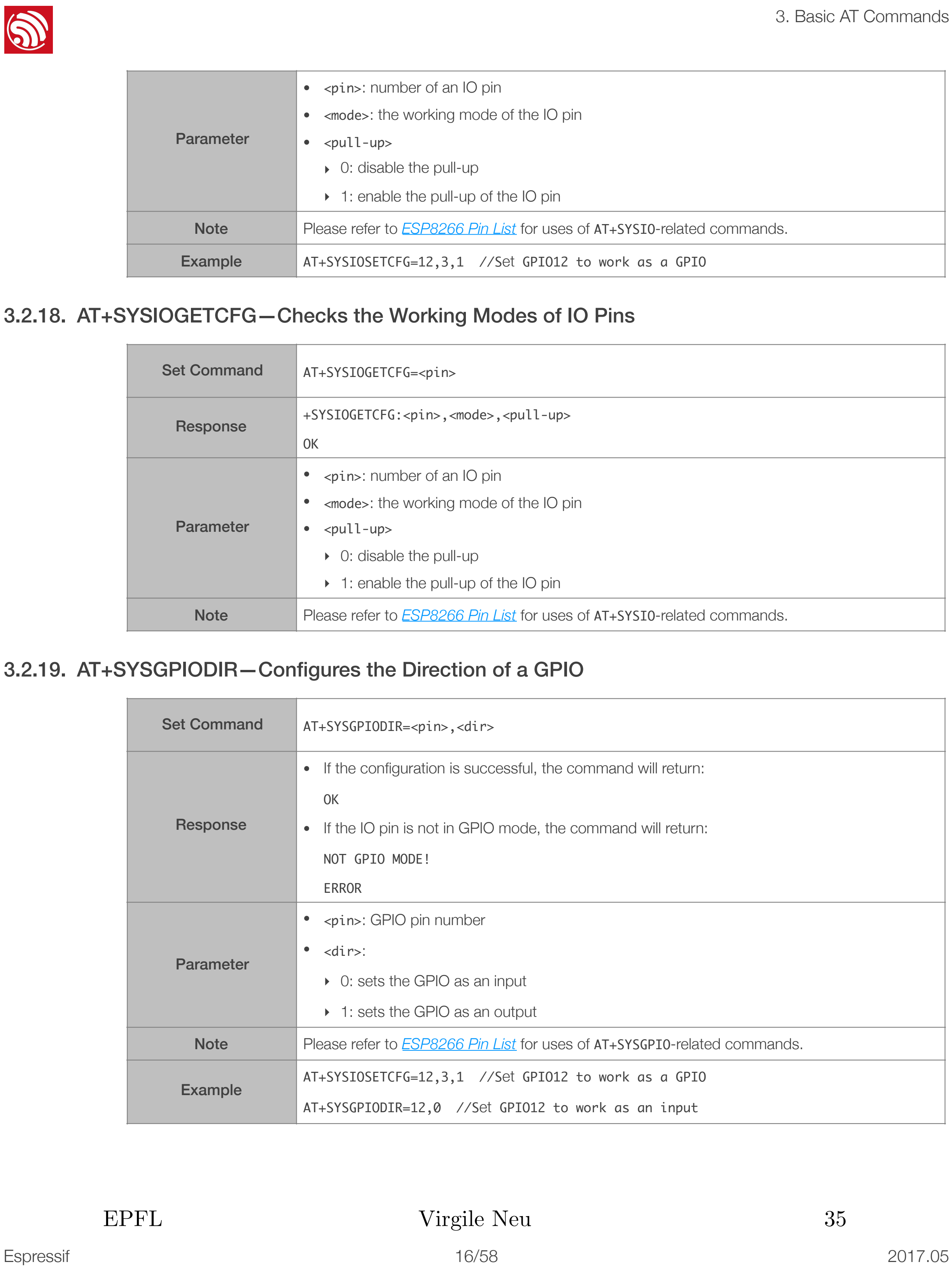
Parameter	`<remaining RAM size>`: remaining space of RAM, unit: byte.

3.2.16. AT+SYSADC—Checks the Value of ADC

Query Command	`AT+SYSADC?`

Response	`+SYSADC:<ADC>`
	`OK`

Parameter	`<ADC>`: the value of ADC; unit: 1/1024V.



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`

3.2.18. 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.

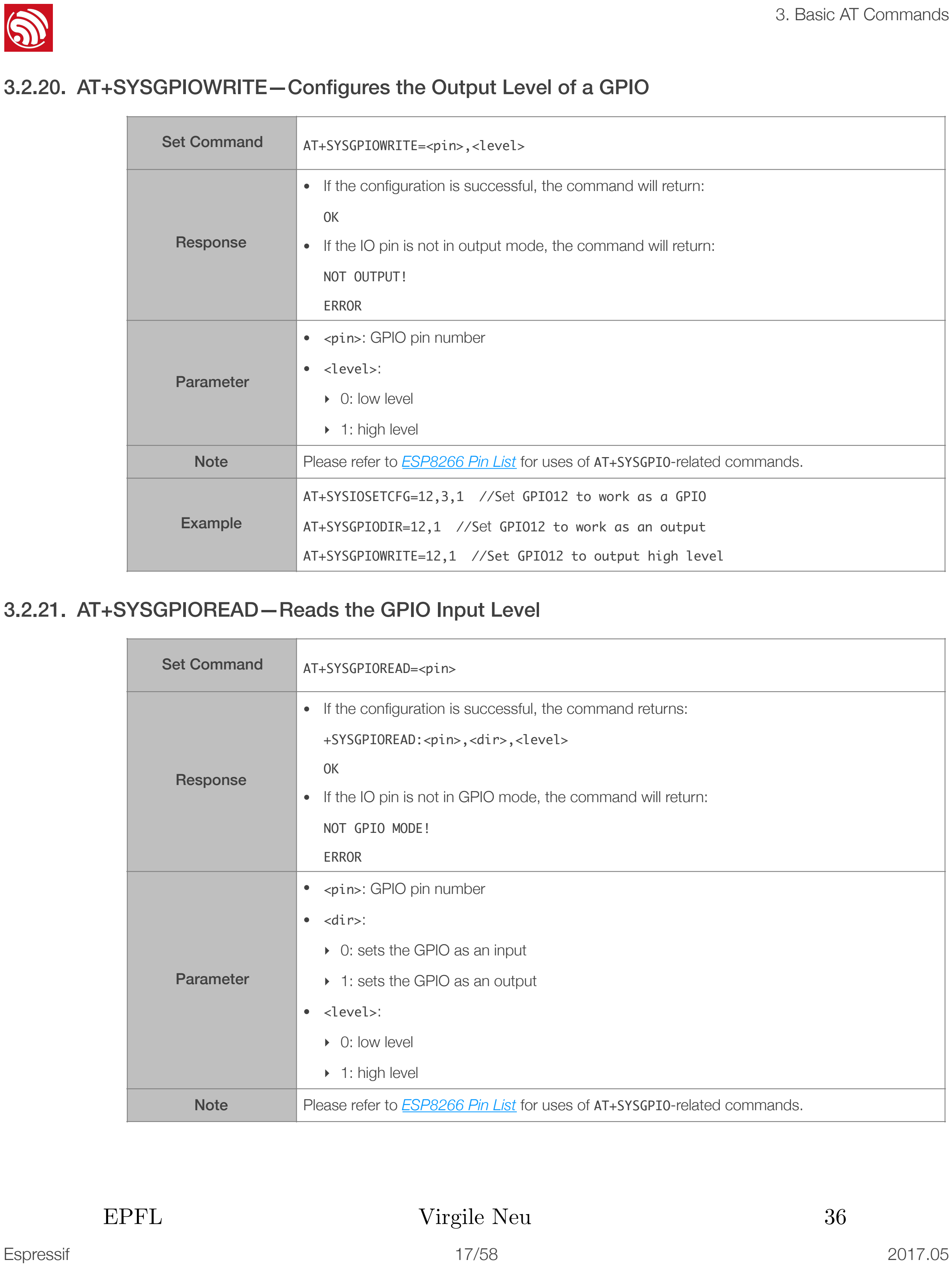
3.2.19. AT+SYSGPIODIR—Configures the Direction of a GPIO

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`



3.2.20. 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`

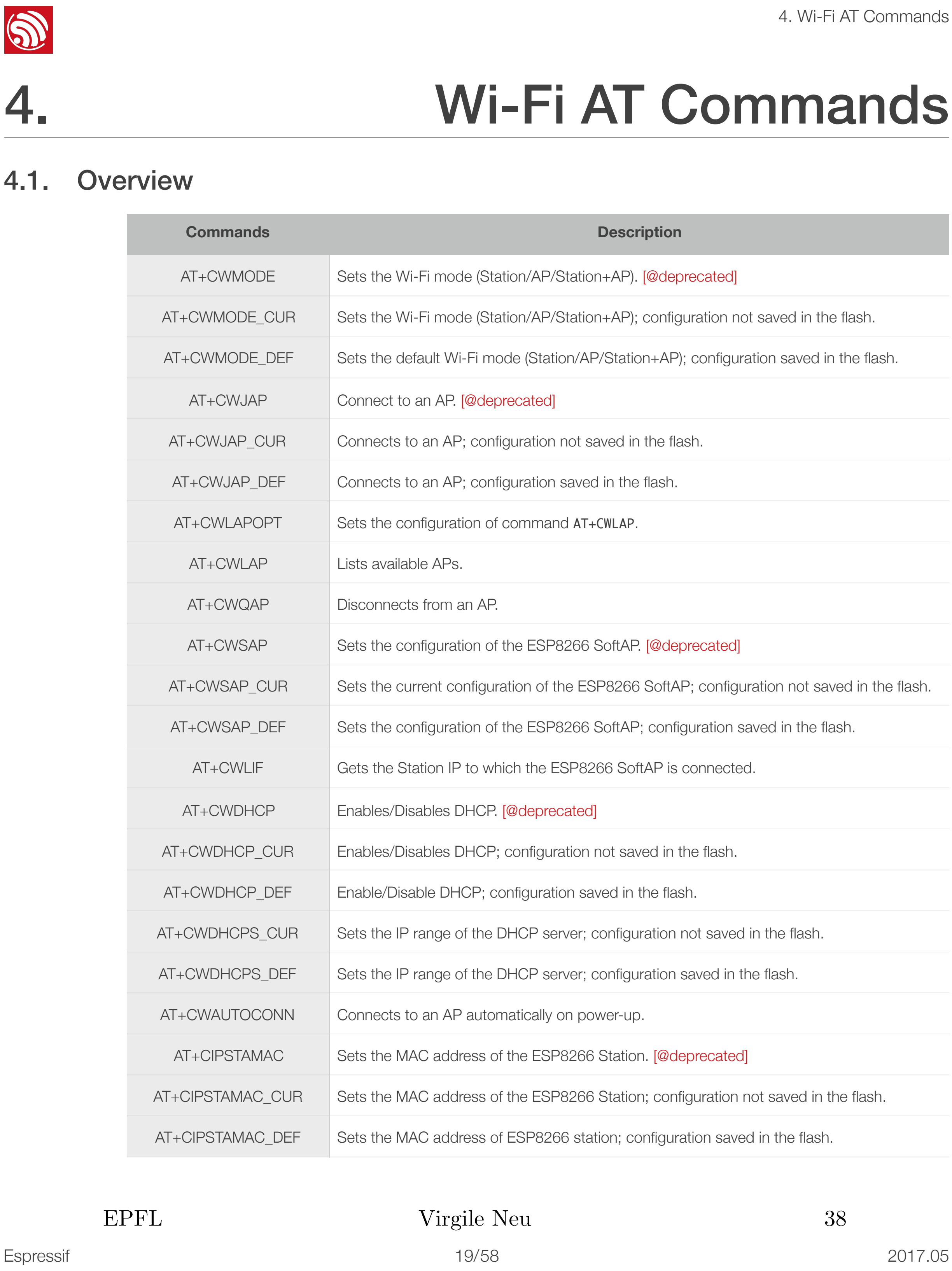
3.2.21. AT+SYSGPIOREAD—Reads the GPIO Input Level

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.

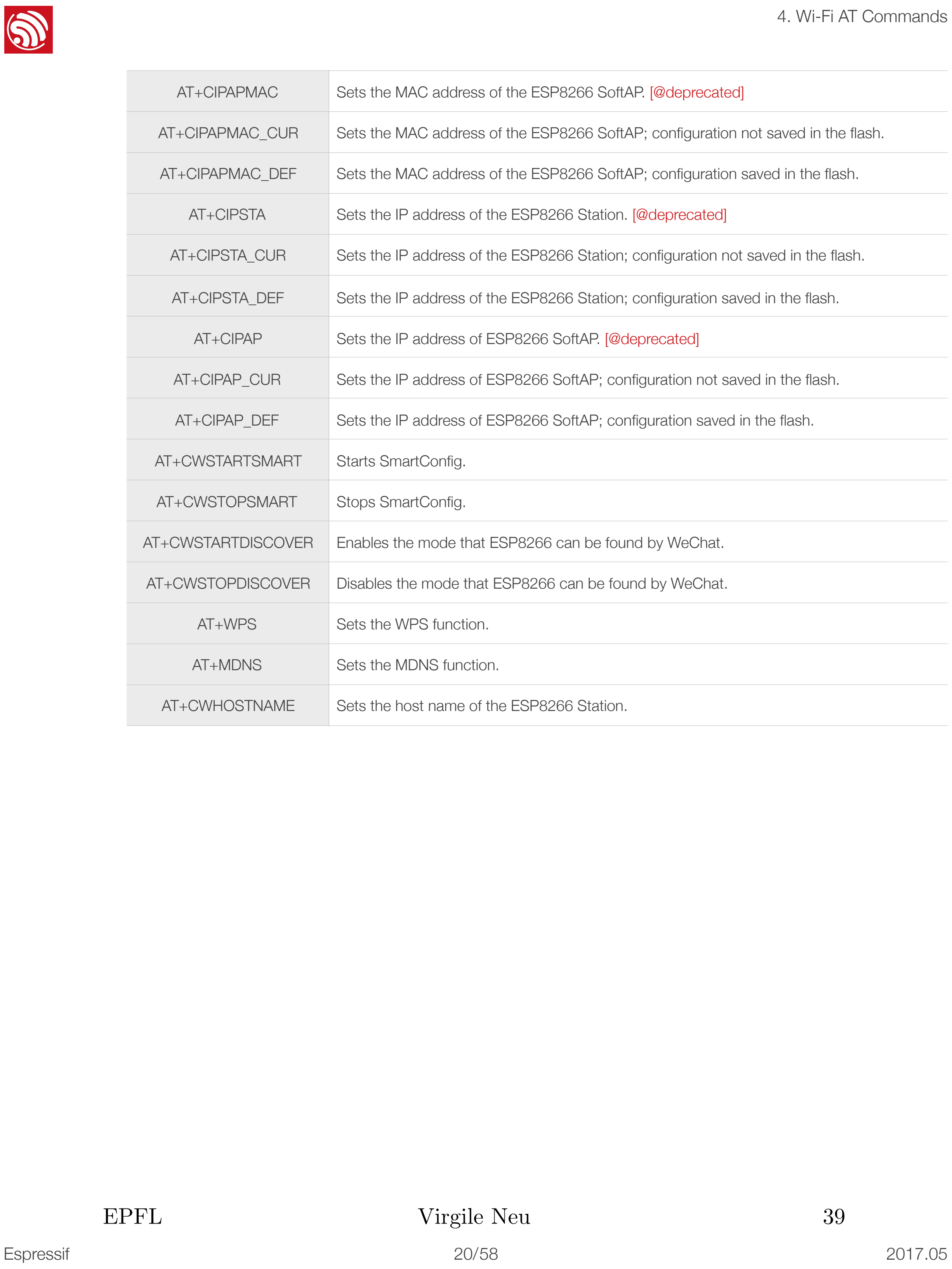
Example	`AT+SYSIOSETCFG=12,3,1  //Set GPIO12 to work as a GPIO`
	`AT+SYSGPIODIR=12,0  //Set GPIO12 to work as an input`
	`AT+SYSGPIOREAD=12`



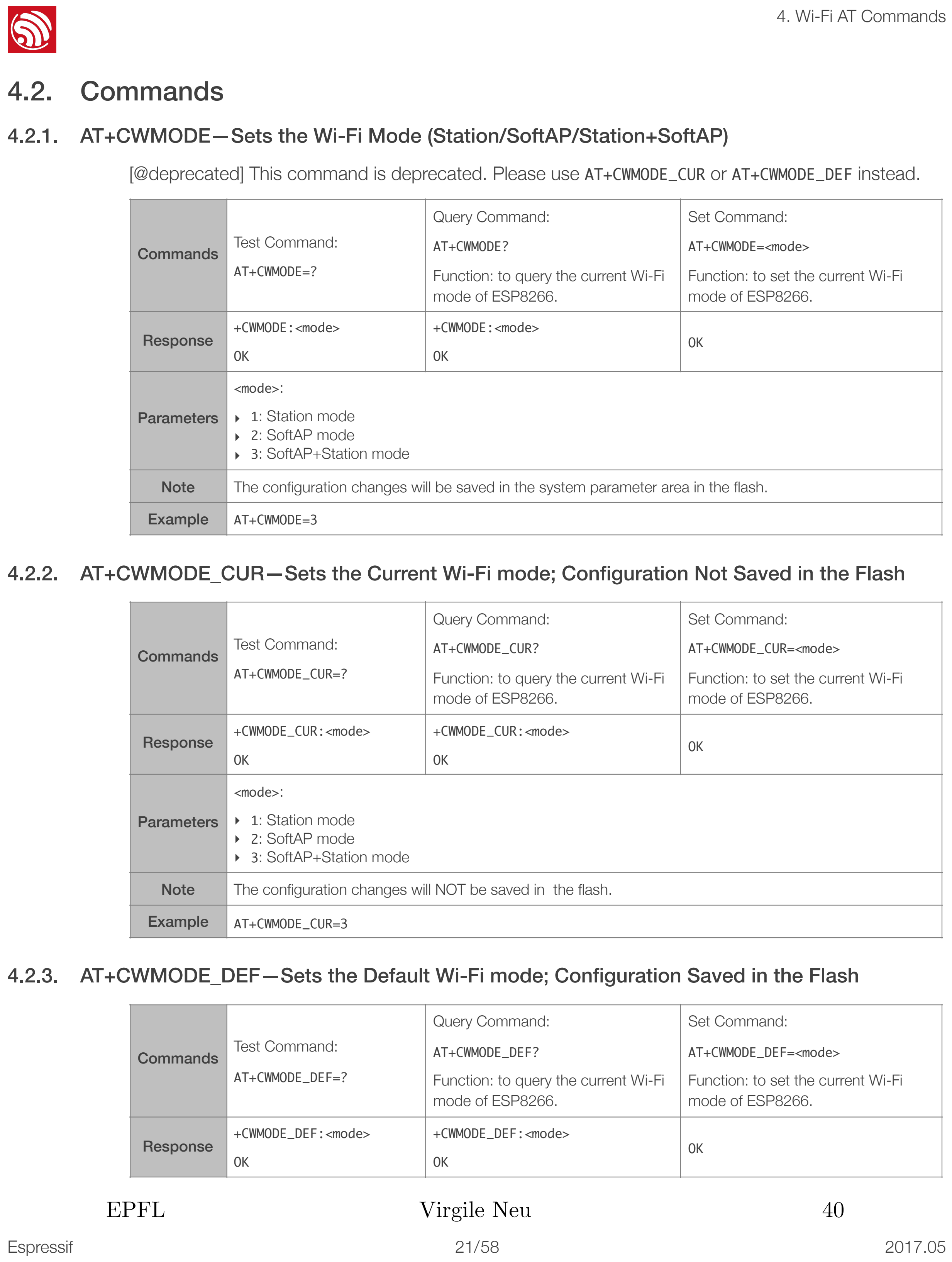
4. Wi-Fi AT Commands

4.1. Overview

Commands	Description
AT+CWMODE	Sets the Wi-Fi mode (Station/AP/Station+AP). [@deprecated]
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	Connect to an AP. [@deprecated]
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	Sets the configuration of the ESP8266 SoftAP. [@deprecated]
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	Enables/Disables DHCP. [@deprecated]
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	Sets the MAC address of the ESP8266 Station. [@deprecated]
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	Sets the MAC address of the ESP8266 SoftAP. [@deprecated]
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	Sets the IP address of the ESP8266 Station. [@deprecated]
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	Sets the IP address of ESP8266 SoftAP. [@deprecated]
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.



4.2. Commands

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

[@deprecated] This command is deprecated. Please use `AT+CWMODE_CUR` or `AT+CWMODE_DEF` instead.

Commands	Test Command: `AT+CWMODE=?`	Query Command: `AT+CWMODE?` Function: to query the current Wi-Fi mode of ESP8266.	Set Command: `AT+CWMODE=<mode>` Function: to set the current Wi-Fi mode of ESP8266.

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 system parameter area in the flash.		
Example	`AT+CWMODE=3`		

4.2.2. 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`		

4.2.3. 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`

4.2.4. AT+CWJAP—Connects to an AP

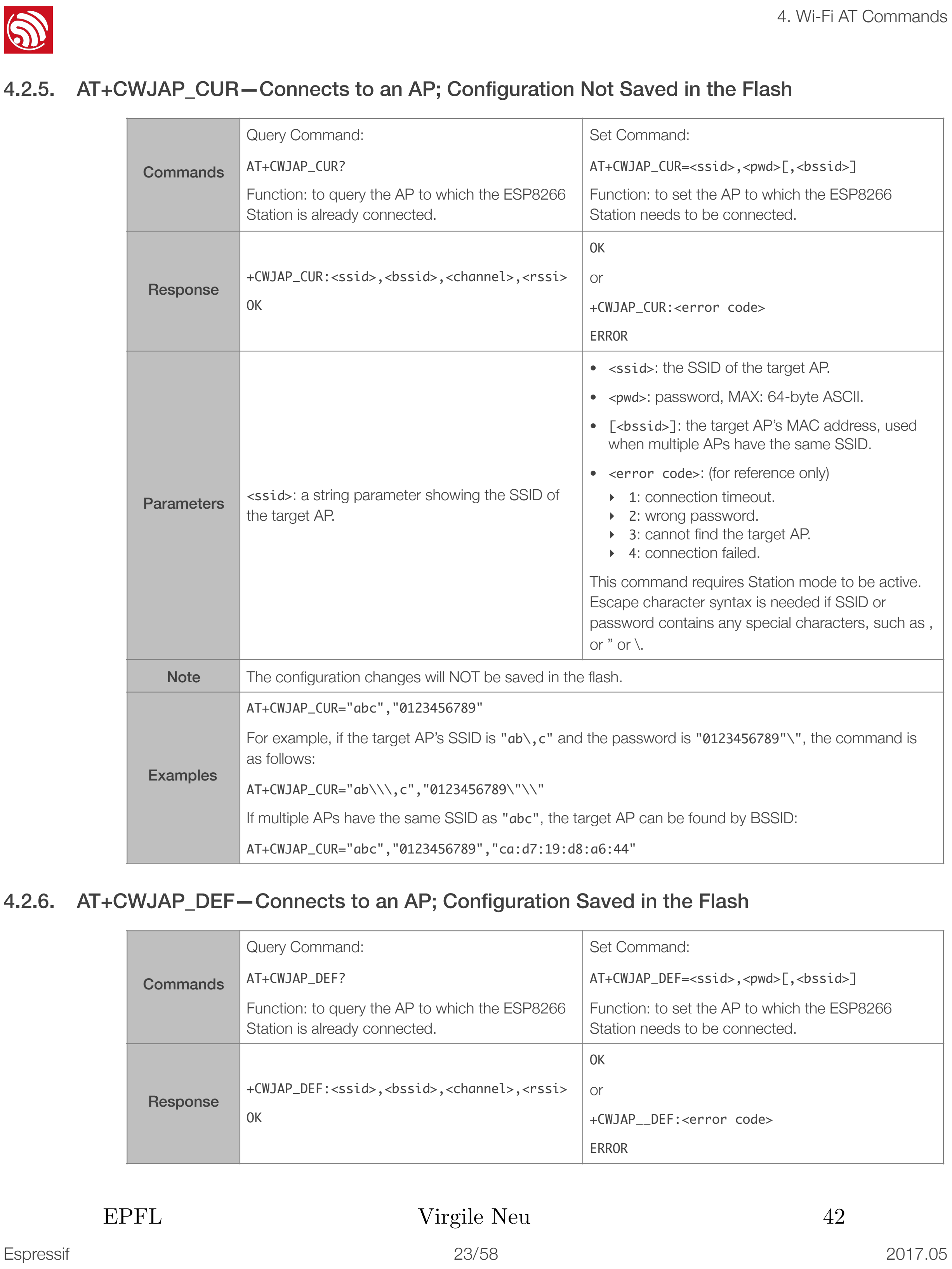
[@deprecated] This command is deprecated. Please use `AT+CWJAP_CUR` or `AT+CWJAP_DEF` instead.

Commands	Query Command: `AT+CWJAP?` Function: to query the AP to which the ESP8266 Station is already connected.	Set Command: `AT+CWJAP=<ssid>,<pwd>[,<bssid>]` Function: to set the AP to which the ESP8266 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 system parameter area in the flash.	

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"`	



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

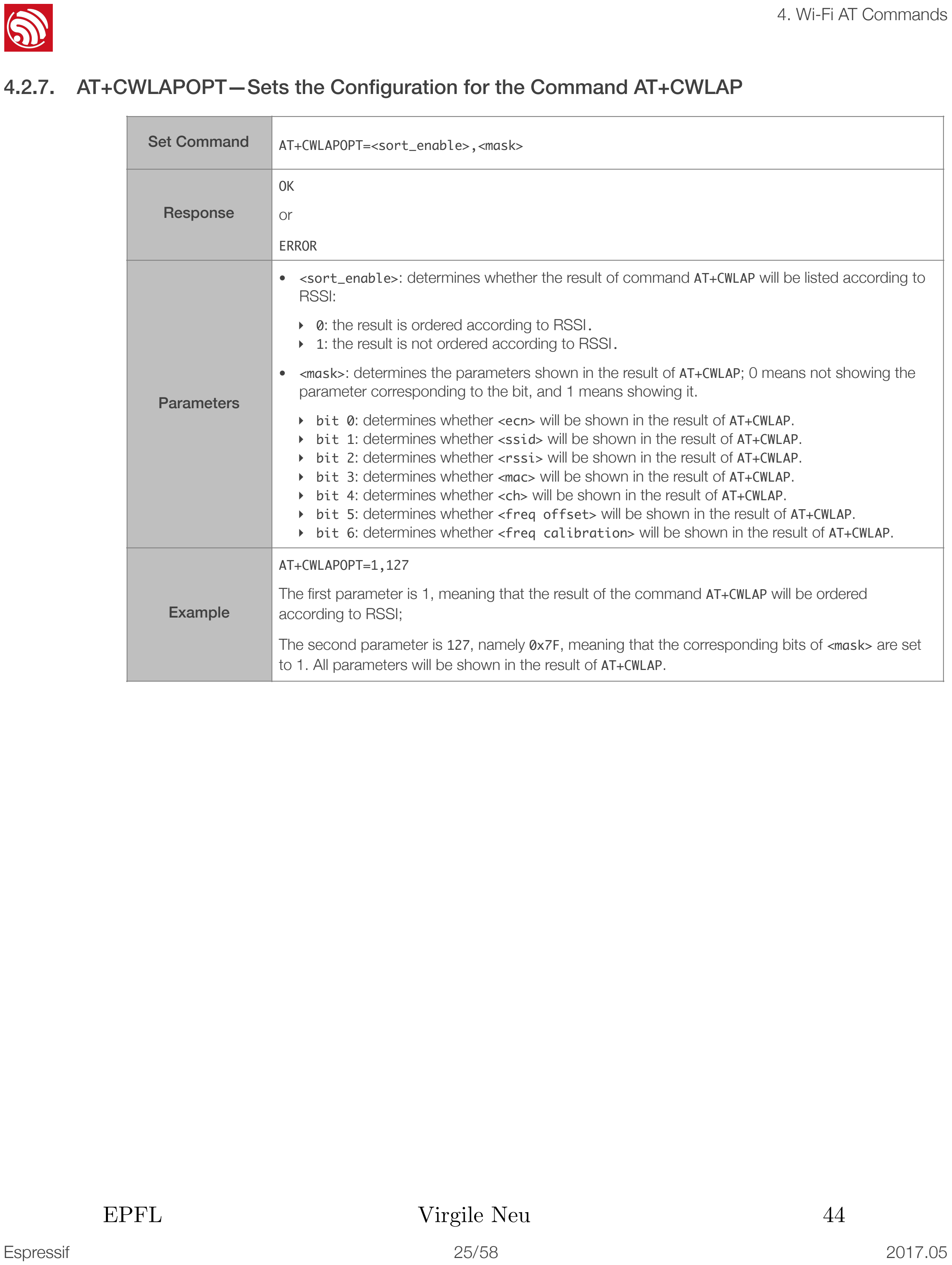
Commands	Query Command: AT+CWJAP_CUR? Function: to query the AP to which the ESP8266 Station is already connected.	Set Command: AT+CWJAP_CUR=<ssid>,<pwd>[,<bssid>] Function: to set the AP to which the ESP8266 Station needs to be connected.
Response	+CWJAP_CUR:<ssid>,<bssid>,<channel>,<rssi> OK	OK or +CWJAP_CUR:<error code> 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 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"	

4.2.6. 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>] 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> 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 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"`	

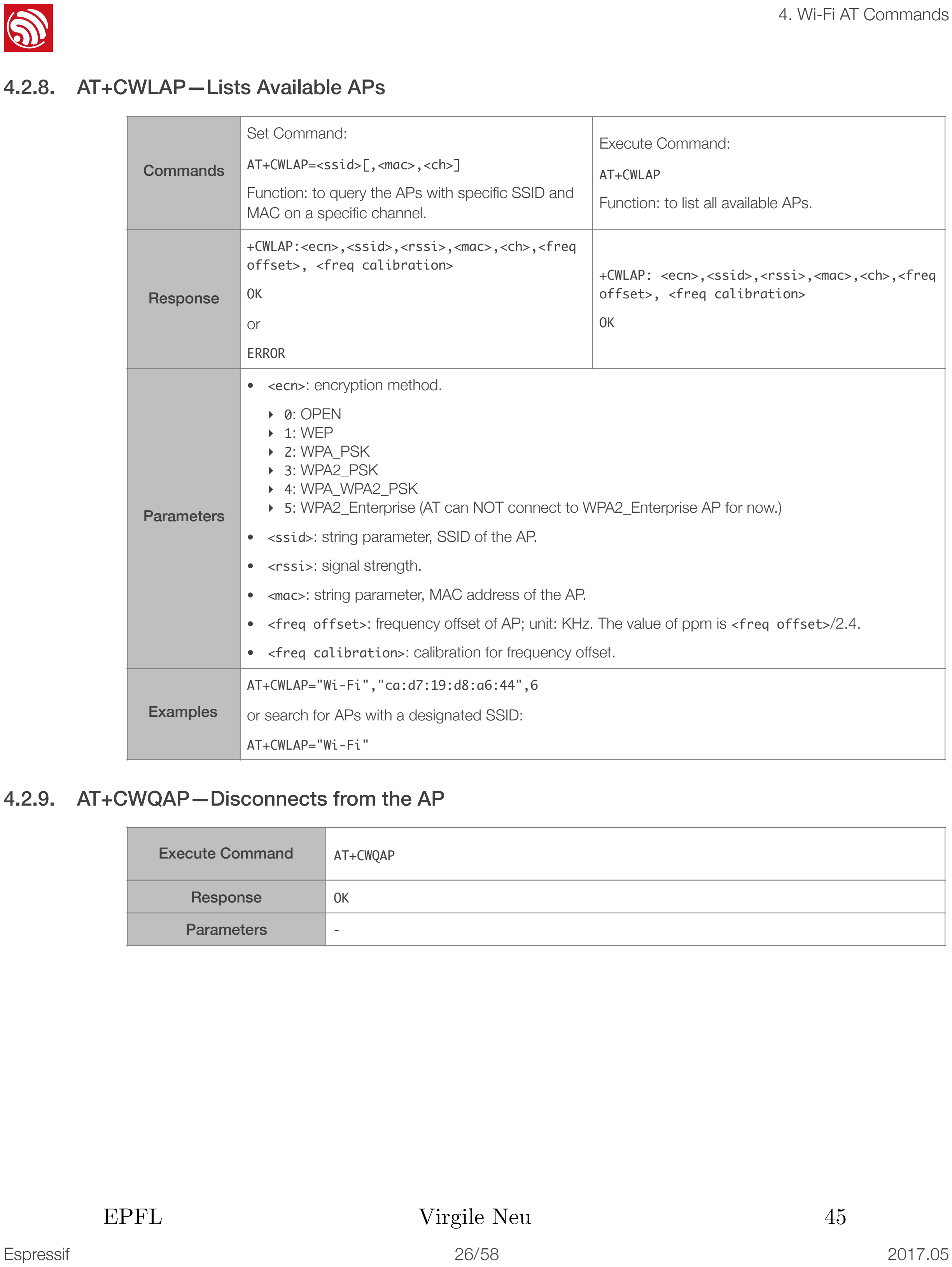


4.2.7. AT+CWLAPOPT—Sets the Configuration for the Command AT+CWLAP

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`. ‣ `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`.
Example	`AT+CWLAPOPT=1,127` The first parameter is 1, meaning that the result of the command `AT+CWLAP` will be ordered according to RSSI; The second parameter is `127`, namely `0x7F`, meaning that the corresponding bits of `<mask>` are set to 1. All parameters will be shown in the result of `AT+CWLAP`.



4.2.8. AT+CWLAP—Lists 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.

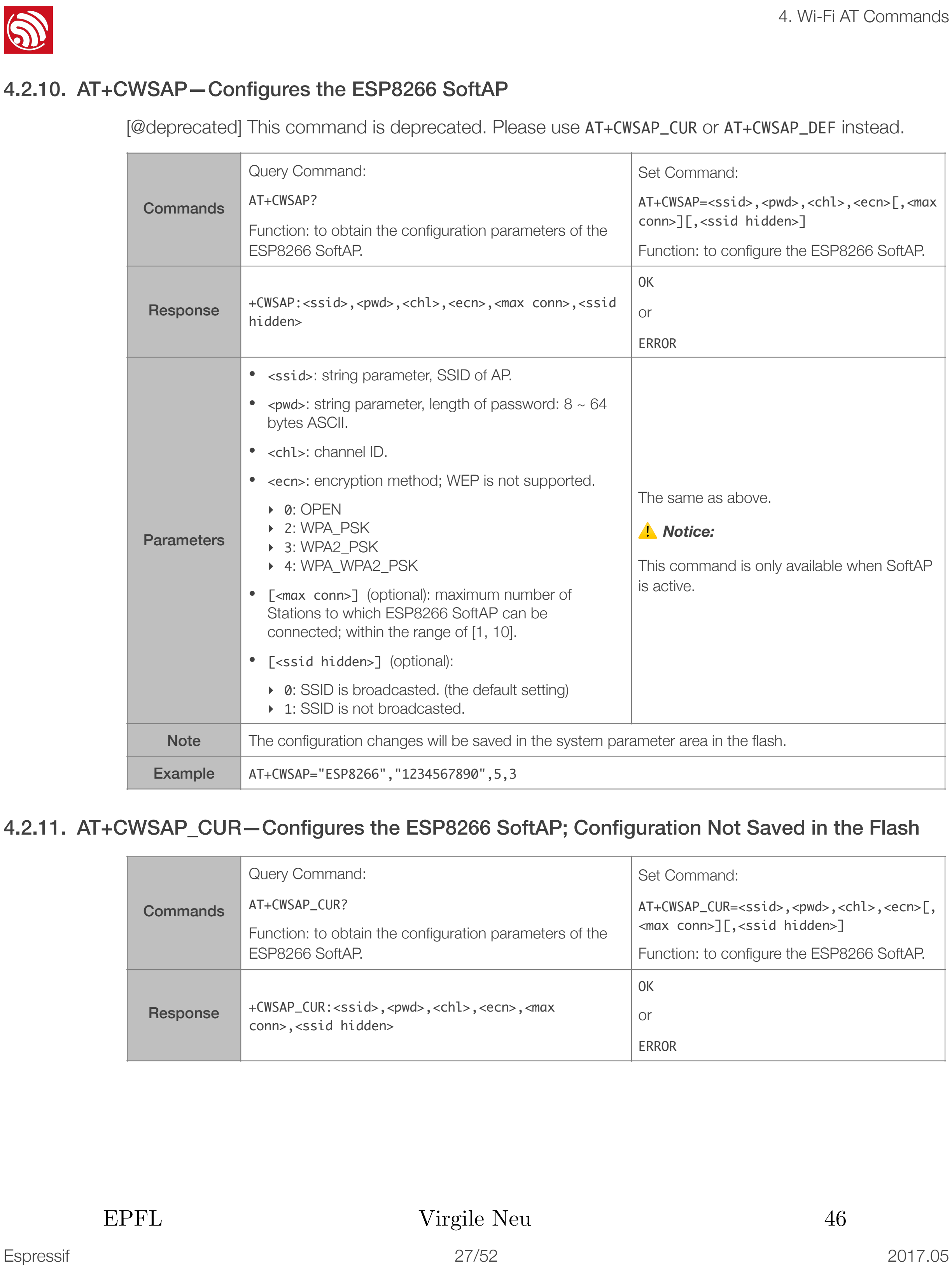
Response	`+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<ch>,<freq offset>, <freq calibration>`
	`OK`
	or
	`ERROR`
		`+CWLAP: <ecn>,<ssid>,<rssi>,<mac>,<ch>,<freq offset>, <freq calibration>`
		`OK`

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. • `<freq offset>`: frequency offset of AP; unit: KHz. The value of ppm is `<freq offset>`/2.4. • `<freq calibration>`: calibration for frequency offset.	

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

4.2.9. AT+CWQAP—Disconnects from the AP

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



4.2.10. AT+CWSAP—Configures the ESP8266 SoftAP

[@deprecated] This command is deprecated. Please use `AT+CWSAP_CUR` or `AT+CWSAP_DEF` instead.

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

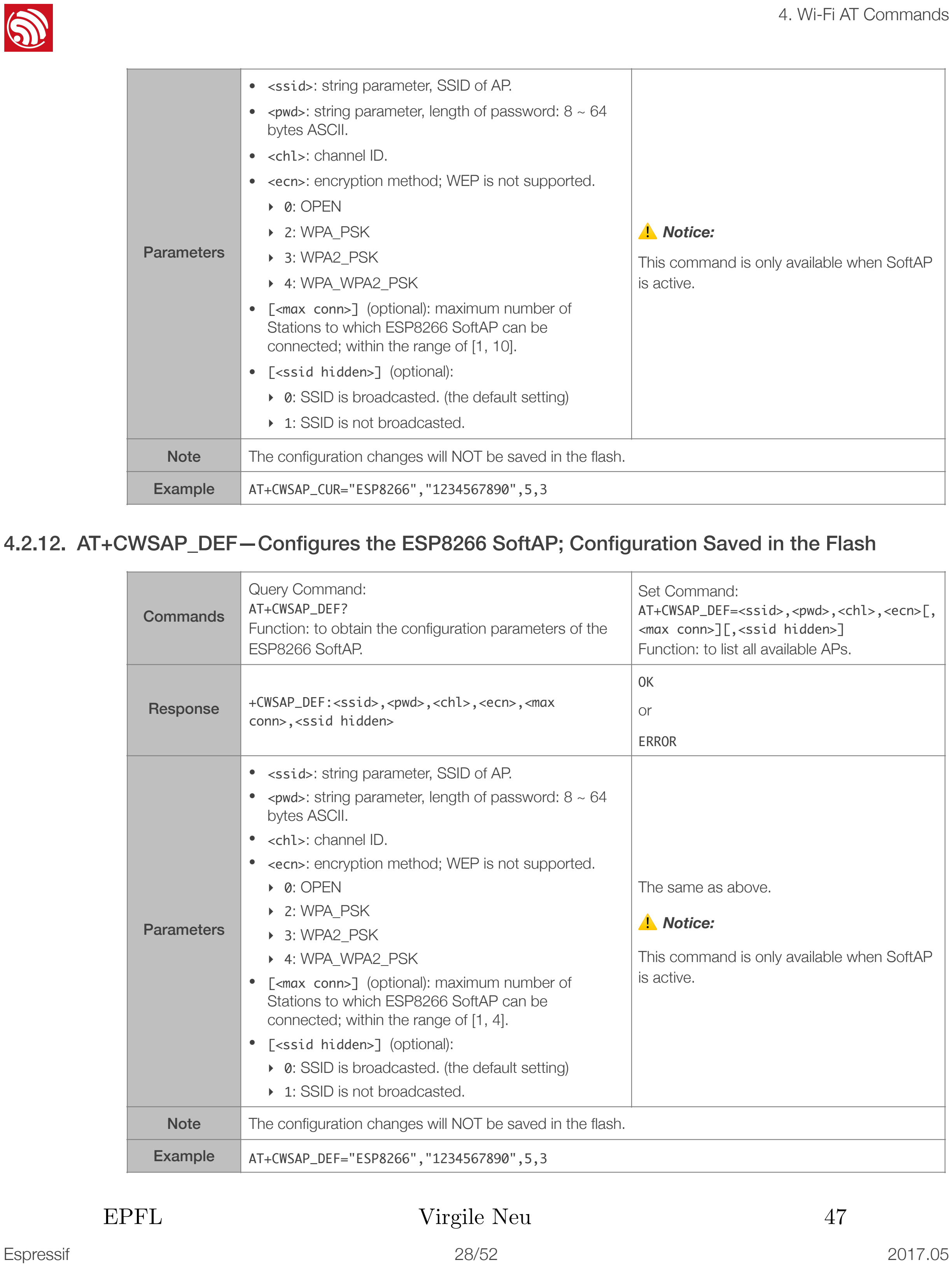
Response	`+CWSAP:<ssid>,<pwd>,<chl>,<ecn>,<max conn>,<ssid hidden>`	`OK`
		or
		`ERROR`

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 ESP8266 SoftAP can be connected; within the range of [1, 10]. • `[<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 system parameter area in the flash.	
Example	`AT+CWSAP="ESP8266","1234567890",5,3`	

4.2.11. 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`



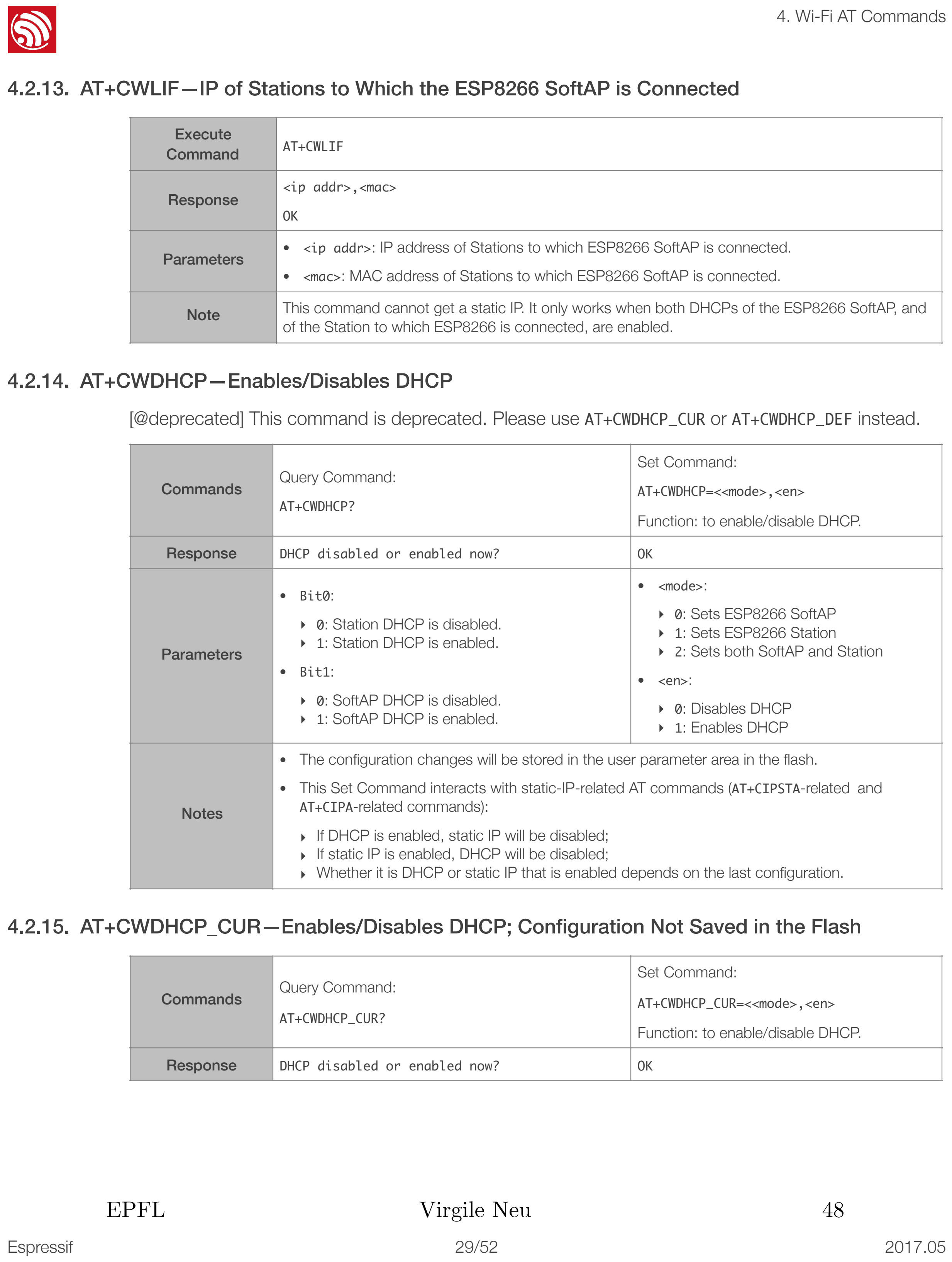
Parameters	• `<ssid>`: string parameter, SSID of AP. • `<pwd>`: string parameter, length of password: 8 ~ 64 bytes ASCII. • `<chl>`: channel ID. • `<ecn>`: encryption method; WEP is not supported. ‣ `0`: OPEN ‣ 2: WPA_PSK ‣ 3: WPA2_PSK ‣ 4: WPA_WPA2_PSK • `[<max conn>]` (optional): maximum number of Stations to which ESP8266 SoftAP can be connected; within the range of [1, 10]. • `[<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`	

4.2.12. 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, 4]. • `[<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 NOT be saved in the flash.	
Example	`AT+CWSAP_DEF="ESP8266","1234567890",5,3`	



4.2.13. 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.

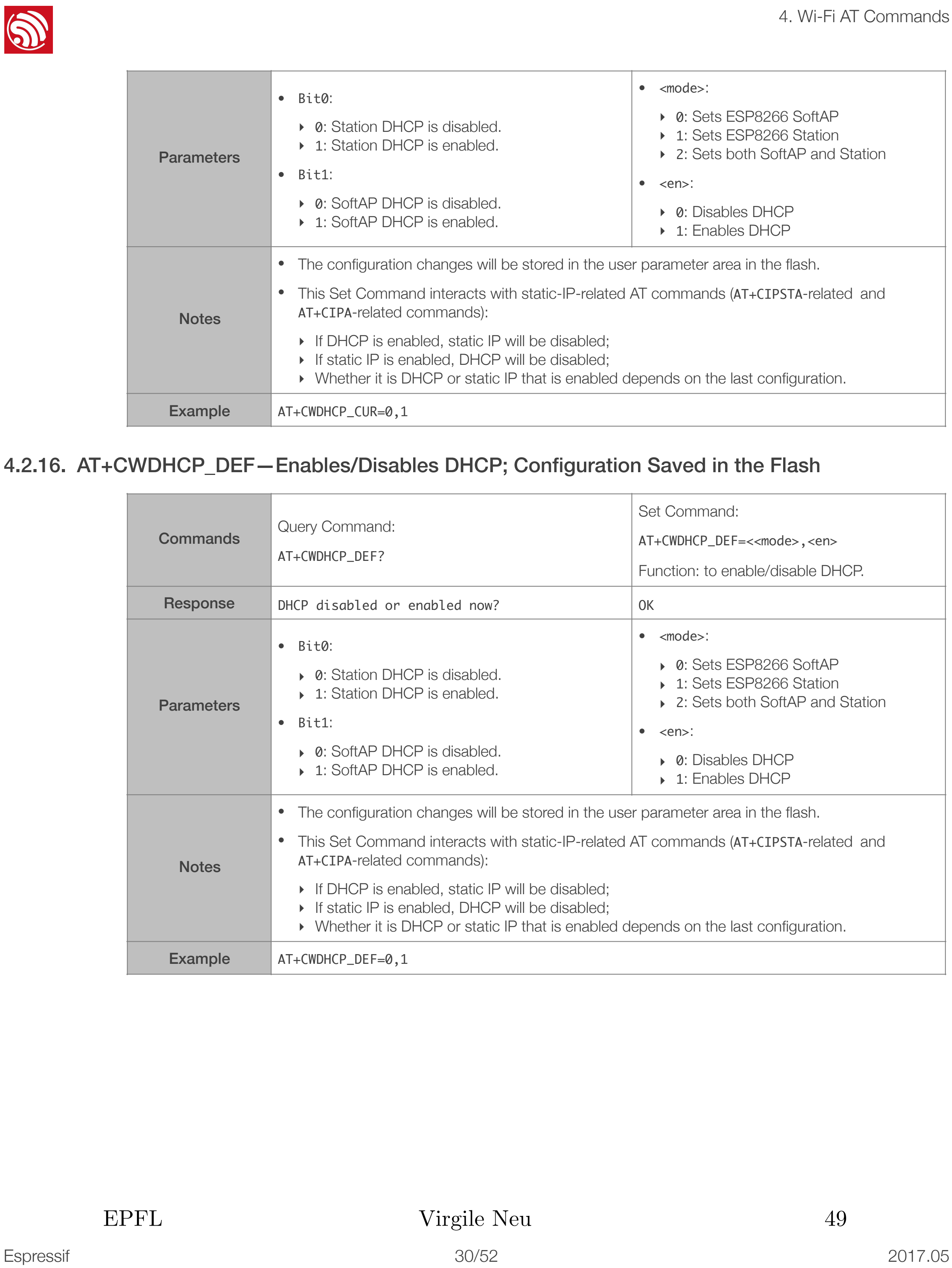
4.2.14. AT+CWDHCP—Enables/Disables DHCP

[@deprecated] This command is deprecated. Please use `AT+CWDHCP_CUR` or `AT+CWDHCP_DEF` instead.

Commands	Query Command: `AT+CWDHCP?`	Set Command: `AT+CWDHCP=<<mode>,<en>` 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.	• `<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.	

4.2.15. 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`: Station DHCP is disabled. ‣ `1`: Station DHCP is enabled. - `Bit1`: ‣ `0`: SoftAP DHCP is disabled. ‣ `1`: SoftAP 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_CUR=0,1`	

4.2.16. 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`: Station DHCP is disabled. ‣ `1`: Station DHCP is enabled. - `Bit1`: ‣ `0`: SoftAP DHCP is disabled. ‣ `1`: SoftAP 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`	

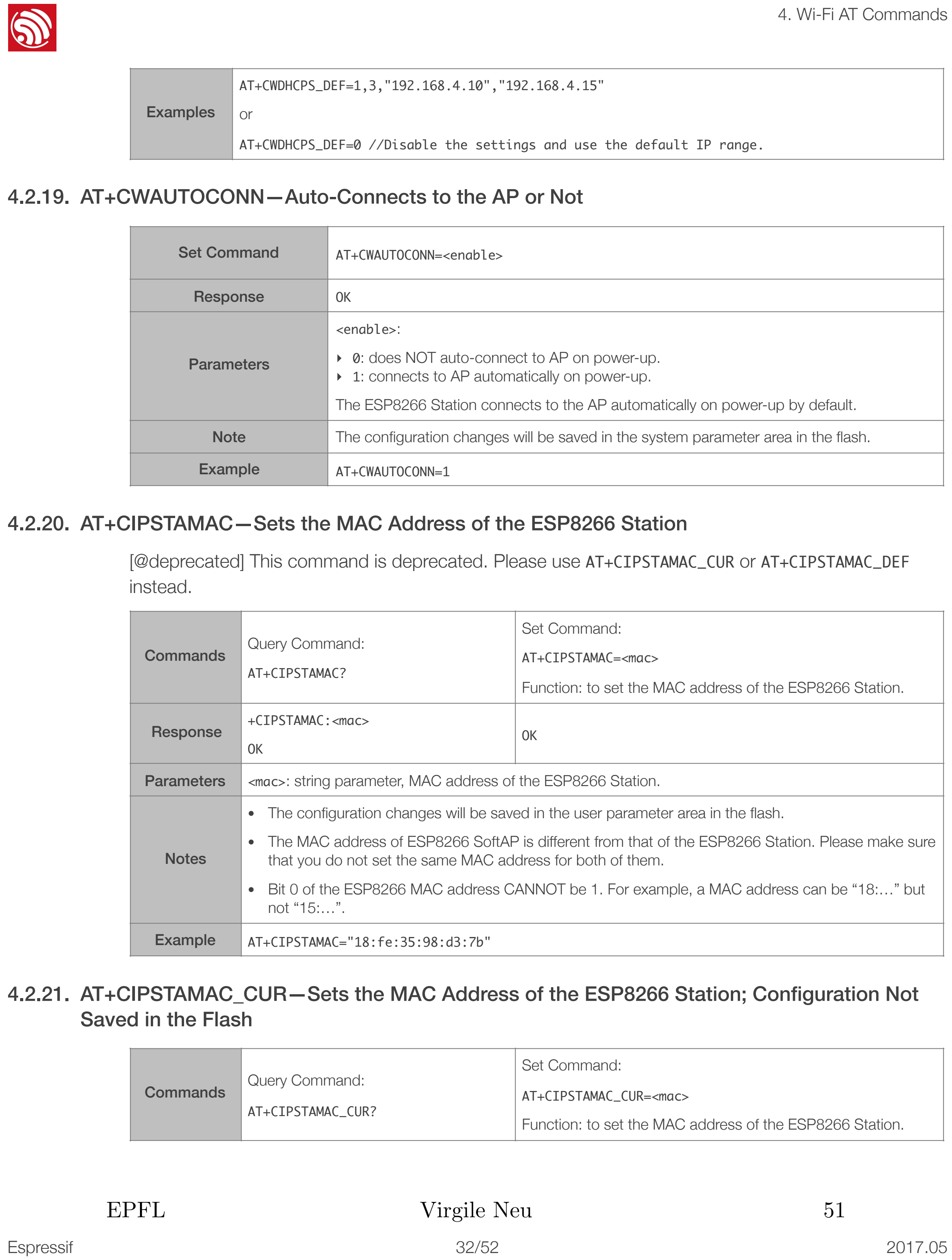
4.2.17. AT+CWDHCPS_CUR—Sets the IP Address Allocated by ESP8266 SoftAP DHCP; Configuration Not Saved in Flash

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.`	

4.2.18. 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.

4.2.19. 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

4.2.20. AT+CIPSTAMAC—Sets the MAC Address of the ESP8266 Station

[@deprecated] This command is deprecated. Please use `AT+CIPSTAMAC_CUR` or `AT+CIPSTAMAC_DEF` instead.

Commands	Query Command: AT+CIPSTAMAC?	Set Command: AT+CIPSTAMAC=<mac> Function: to set the MAC address of the ESP8266 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 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="18:fe:35:98:d3:7b"	

4.2.21. 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"	

4.2.22. 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"	

4.2.23. AT+CIPAPMAC—Sets the MAC Address of the ESP8266 SoftAP

[@deprecated] This command is deprecated. Please use `AT+CIPAPMAC_CUR` or `AT+CIPAPMAC_DEF` instead.

Commands	Query Command: AT+CIPAPMAC? Function: to obtain the MAC address of the ESP8266 SoftAP.	Set Command: AT+CIPAPMAC=<mac> Function: to set the MAC address of the ESP8266 SoftAP.
Response	+CIPAPMAC:<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="1a:fe:36:97:d5:7b"`

4.2.24. 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`

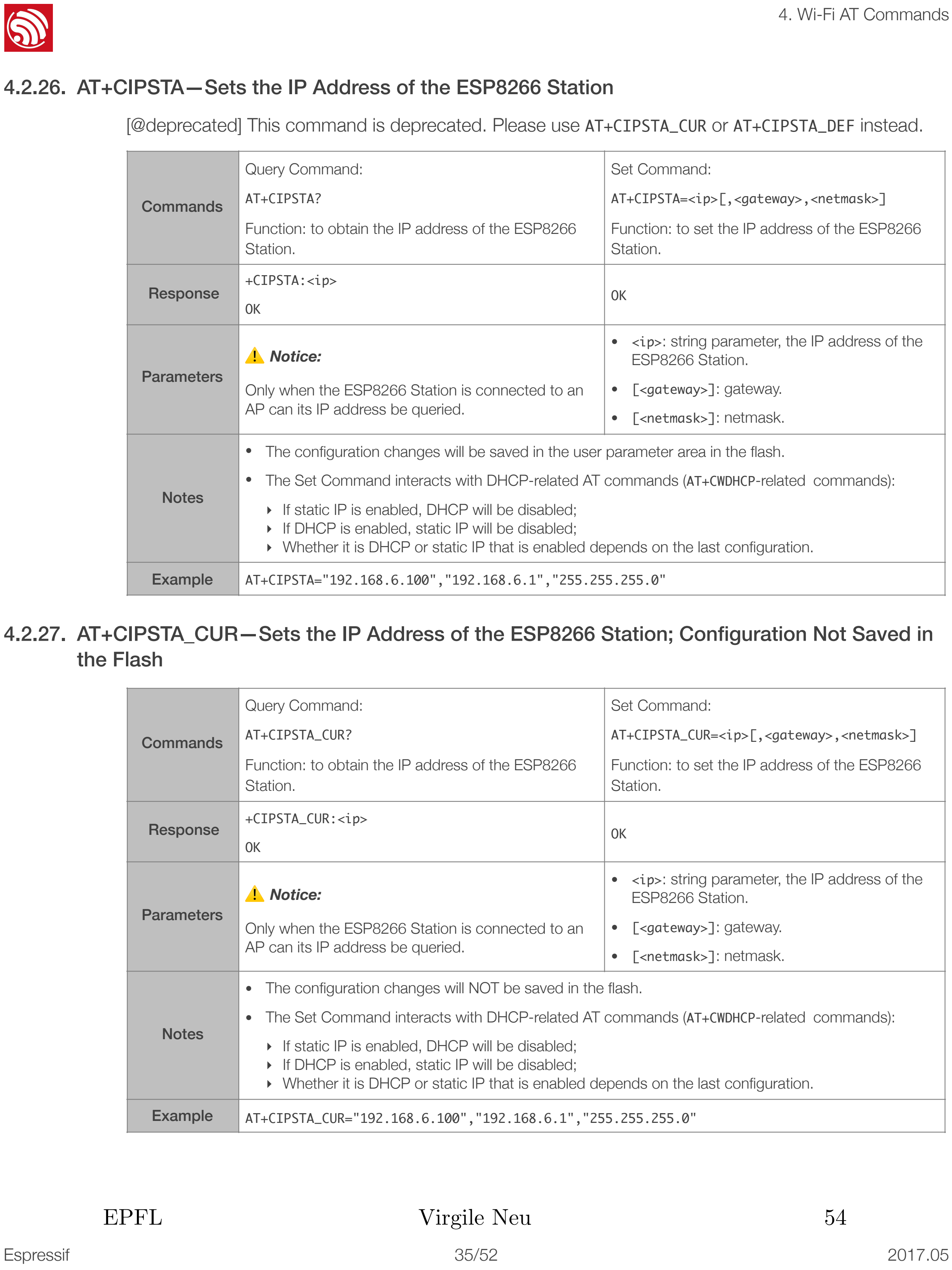
Parameters	`<mac>`: string parameter, MAC address of ESP8266 SoftAP.	
Notes	• The configuration changes will NOT be saved the flash. • The MAC address of ESP8266 SoftAP is different from that of the ESP8266 Station. Please make sure you do not set the same MAC address for both of them. • Bit 0 of the 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"`	

4.2.25. 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"`	



4.2.26. AT+CIPSTA—Sets the IP Address of the ESP8266 Station

[@deprecated] This command is deprecated. Please use `AT+CIPSTA_CUR` or `AT+CIPSTA_DEF` instead.

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

Response	`+CIPSTA:<ip>`
	`OK`
		`OK`

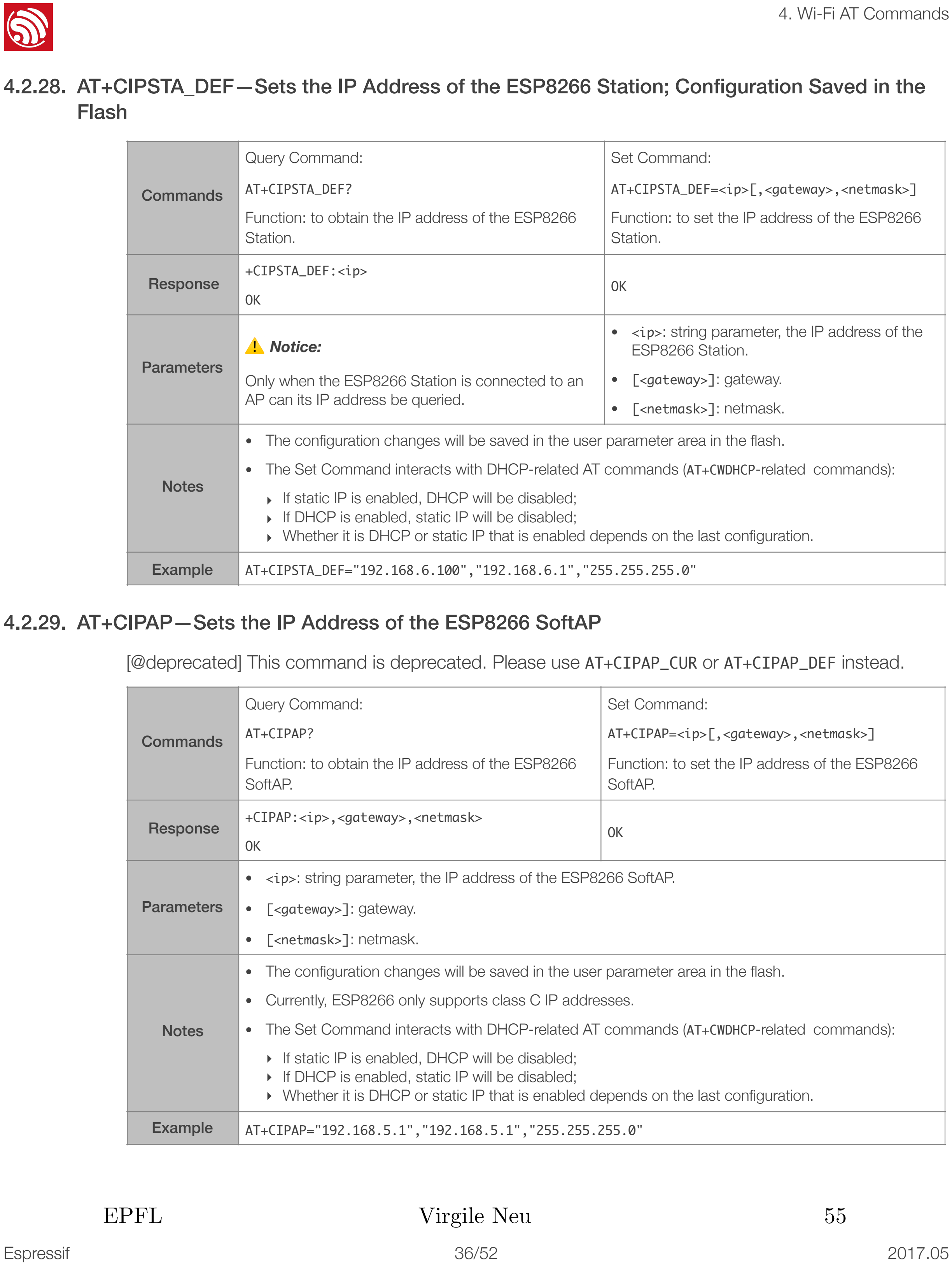
Parameters	⚠ ***Notice:*** Only when the ESP8266 Station is connected to an AP can its IP address be queried.	• `<ip>`: string parameter, the IP address of the ESP8266 Station. • `[<gateway>]`: gateway. • `[<netmask>]`: netmask.
Notes	• The configuration changes will be saved in the user parameter area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

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

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

Response	`+CIPSTA_CUR:<ip>`
	`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"`	



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

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

Response	`+CIPSTA_DEF:<ip>`
	`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"`	

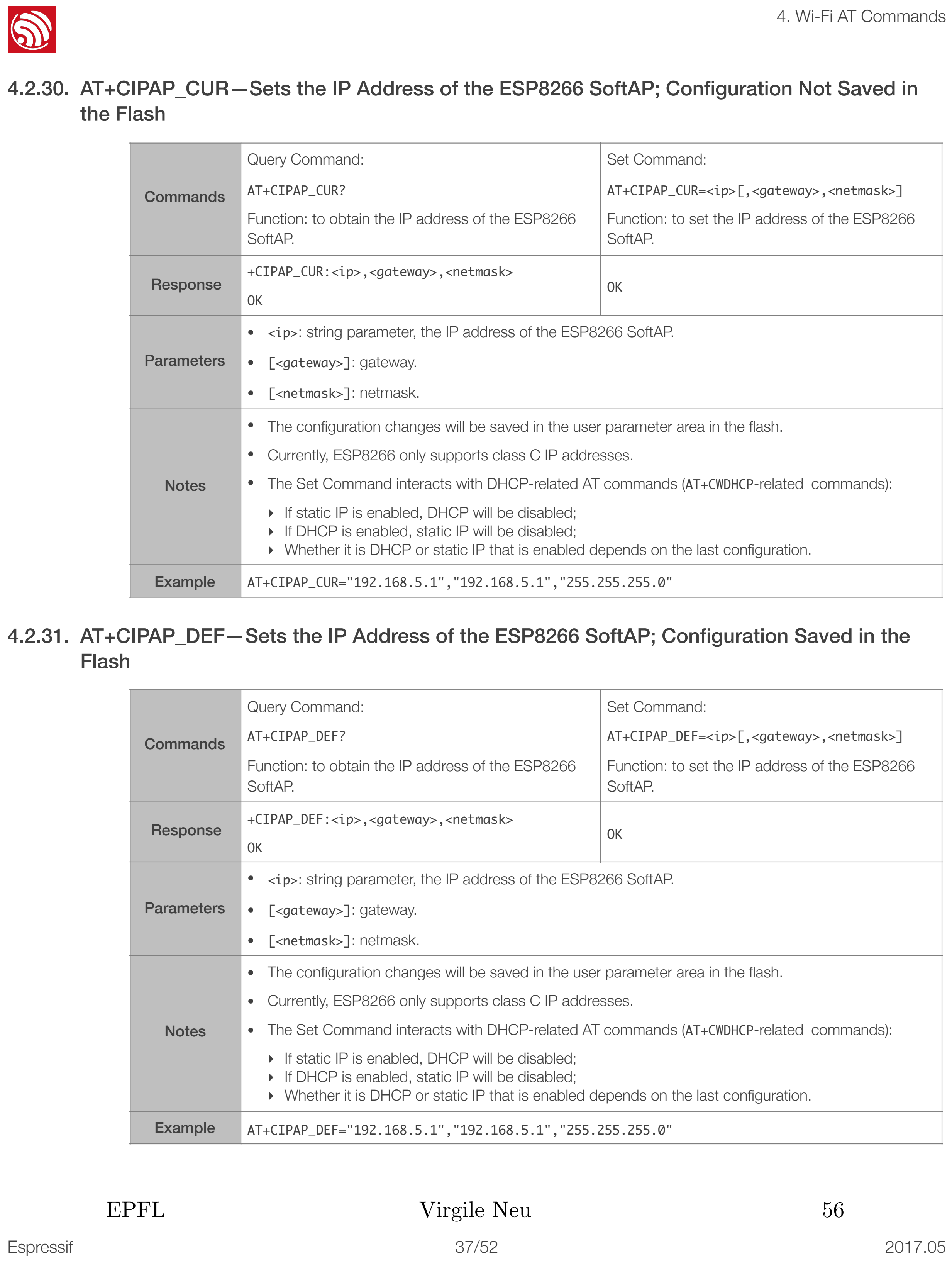
4.2.29. AT+CIPAP—Sets the IP Address of the ESP8266 SoftAP

[@deprecated] This command is deprecated. Please use `AT+CIPAP_CUR` or `AT+CIPAP_DEF` instead.

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

Response	`+CIPAP:<ip>,<gateway>,<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="192.168.5.1","192.168.5.1","255.255.255.0"`	



4.2.30. 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 IP address of the ESP8266 SoftAP.	Set Command: `AT+CIPAP_CUR=<ip>[,<gateway>,<netmask>]` Function: to set the IP address of the ESP8266 SoftAP.

Response	`+CIPAP_CUR:<ip>,<gateway>,<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_CUR="192.168.5.1","192.168.5.1","255.255.255.0"`	

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

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

Response	`+CIPAP_DEF:<ip>,<gateway>,<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"`	

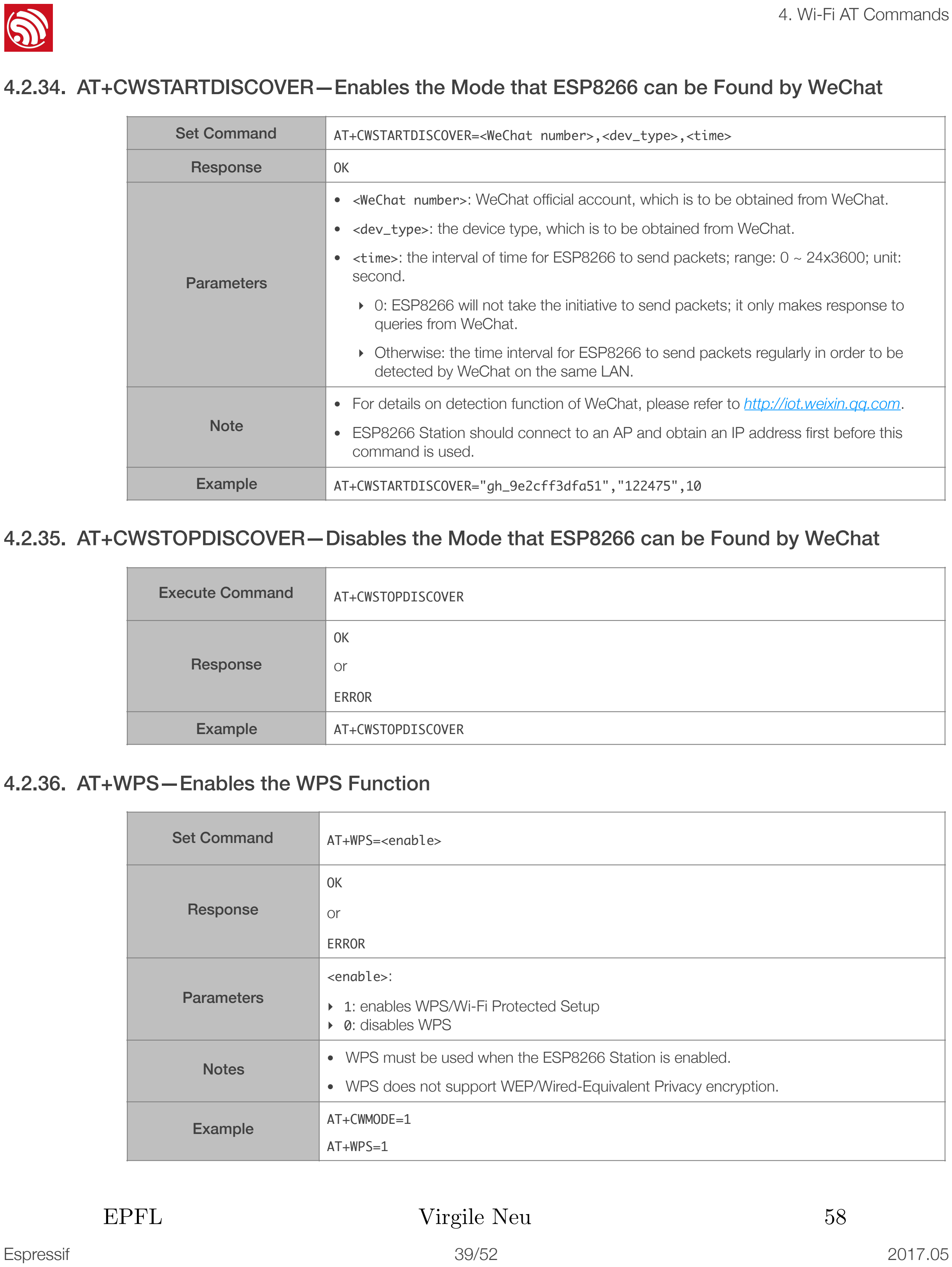
4.2.32. AT+CWSTARTSMART—Starts SmartConfig

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 ESP8266 Station mode. • The message `Smart get Wi-Fi info` means that SmartConfig has successfully acquired the AP information. ESP8266 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. • Starting from AT_v1.0, SmartConfig can get protocol type (AirKiss or ESP-TOUCH) automatically by command `AT+CWSTARTSMART`.	

Example	`AT+CWMODE=1`
	`AT+CWSTARTSMART`	

4.2.33. 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`



4.2.34. 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`

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

Execute Command	`AT+CWSTOPDISCOVER`

Response	`OK`
	or
	`ERROR`

Example	`AT+CWSTOPDISCOVER`

4.2.36. AT+WPS—Enables the WPS Function

Set Command	`AT+WPS=<enable>`

Response	`OK`
	or
	`ERROR`

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`

4.2.37. AT+MDNS—Configures the MDNS Function

Set Command	`AT+MDNS=<enable>,<hostname>,<server_name>,<server_port>`

Response	`OK`
	or
	`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`

4.2.38. 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.	
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"`	

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+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+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`
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`

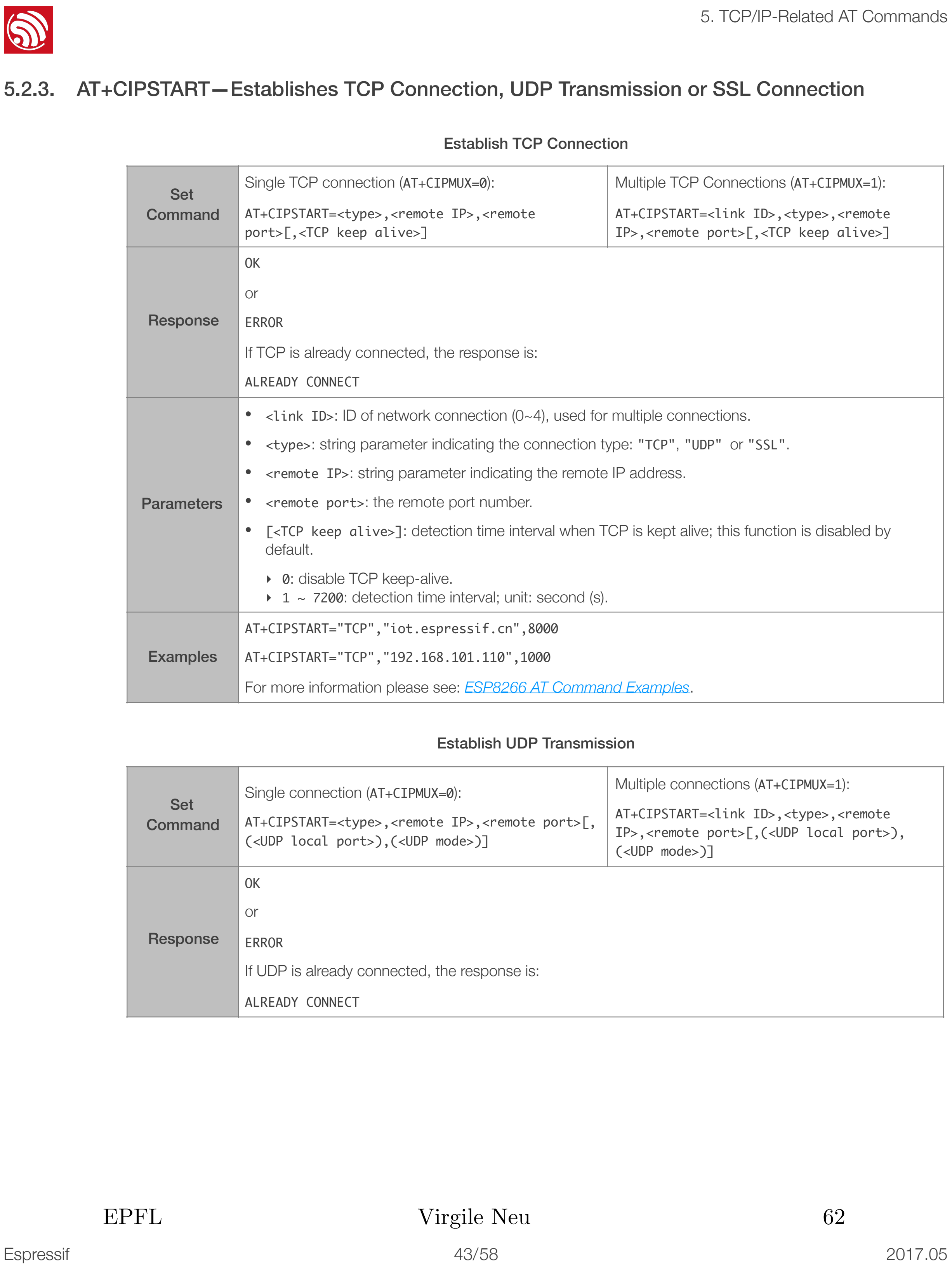
Response	`STATUS:<stat>`
	`+CIPSTATUS:<link ID>,<type>,<remote IP>,<remote port>,<local port>,<tetype>`

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>`
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. TCP/IP-Related AT Commands

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 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"`, `"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).	

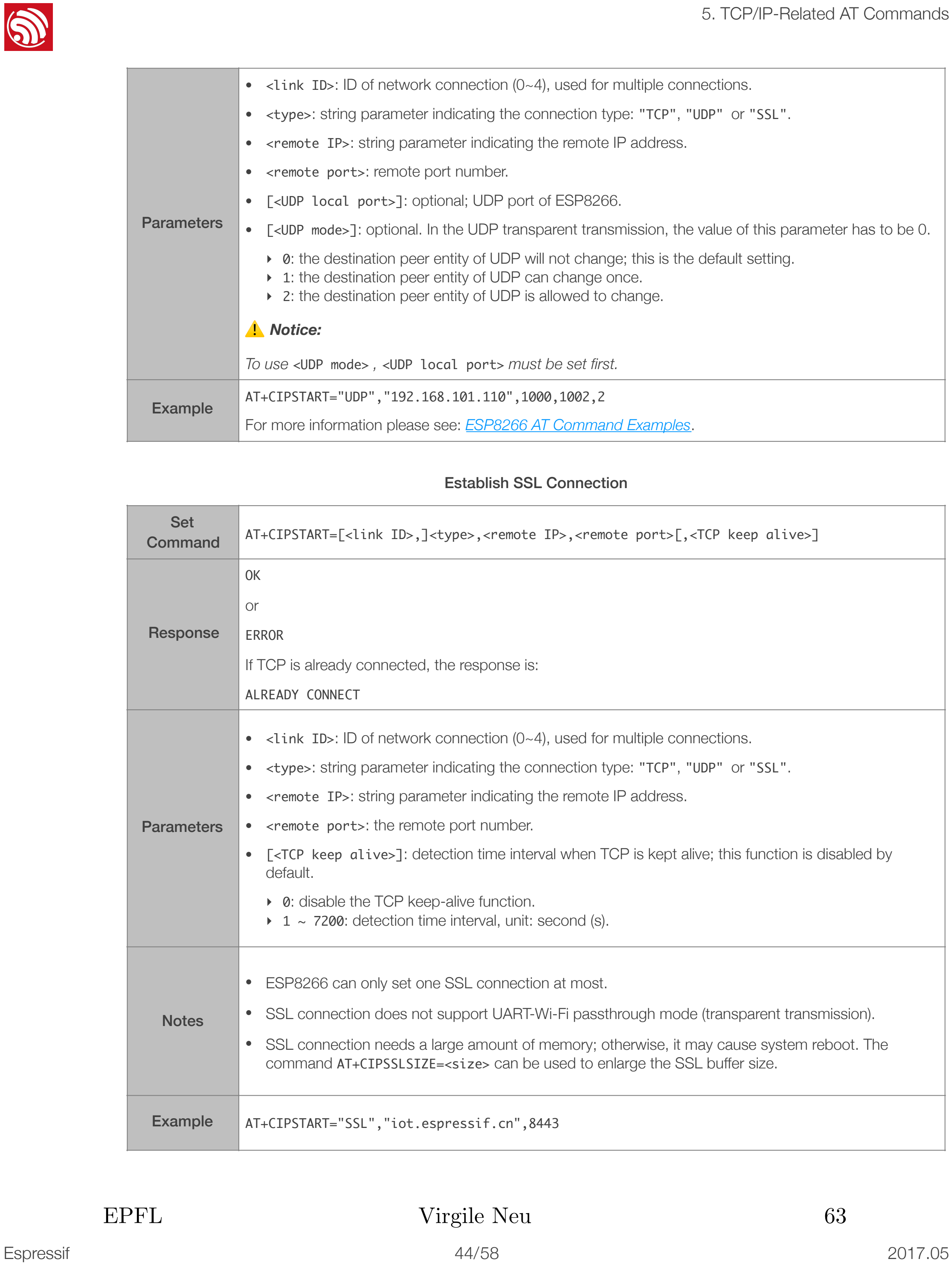
Examples	`AT+CIPSTART="TCP","iot.espressif.cn",8000`
	`AT+CIPSTART="TCP","192.168.101.110",1000`
	For more information please see: ESP8266 AT Command Examples.	

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 UDP is already connected, the response is:
	`ALREADY CONNECT`	

Espressif 43/58 2017.05



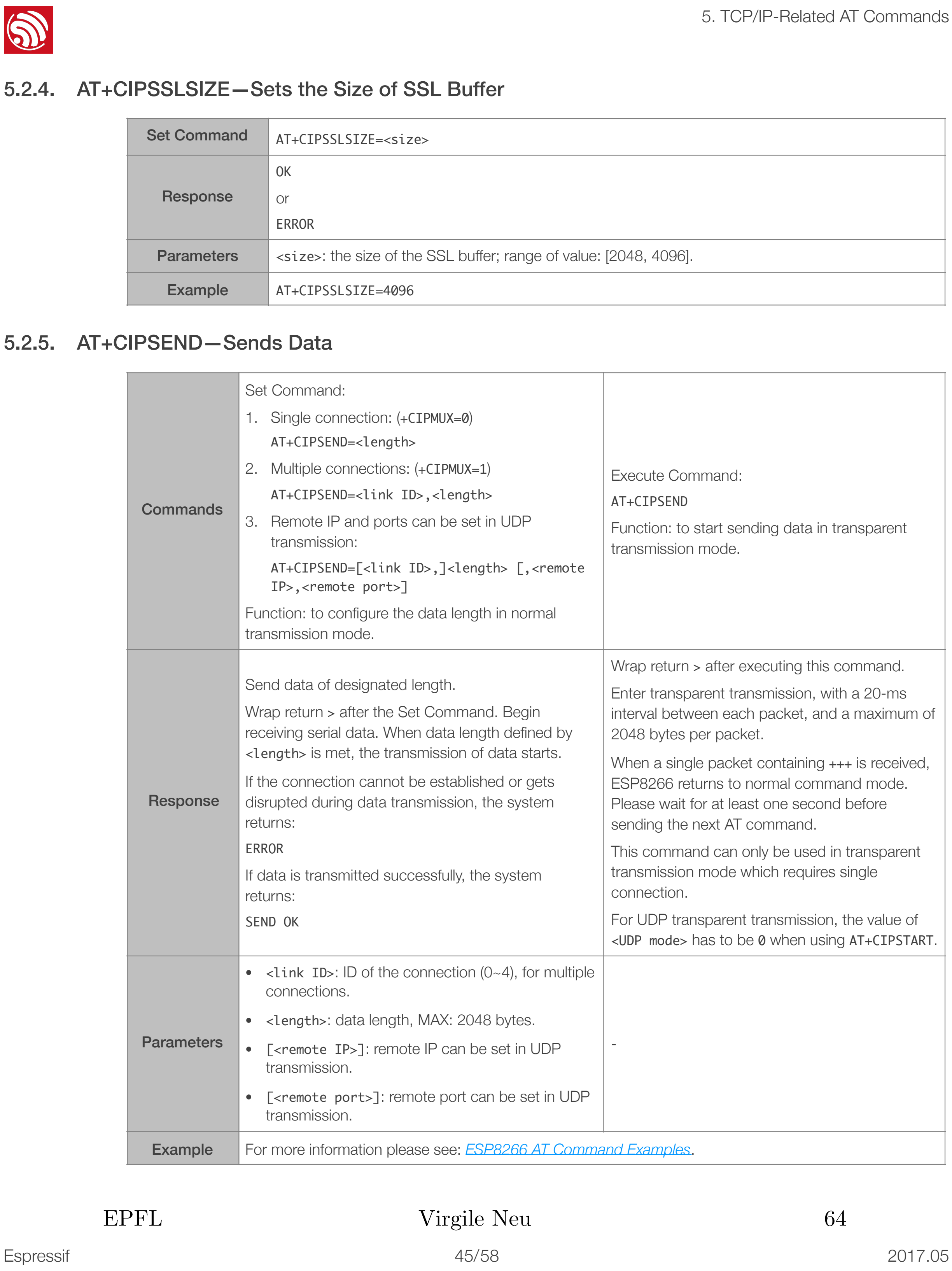
Parameters	• `<link ID>`: ID of network connection (0~4), used for multiple connections. • `<type>`: string parameter indicating the connection type: `"TCP"`, `"UDP"` or `"SSL"`. • `<remote IP>`: string parameter indicating the remote IP address. • `<remote port>`: remote port number. • `[<UDP local 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 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"`, `"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`
	or
	`ERROR`

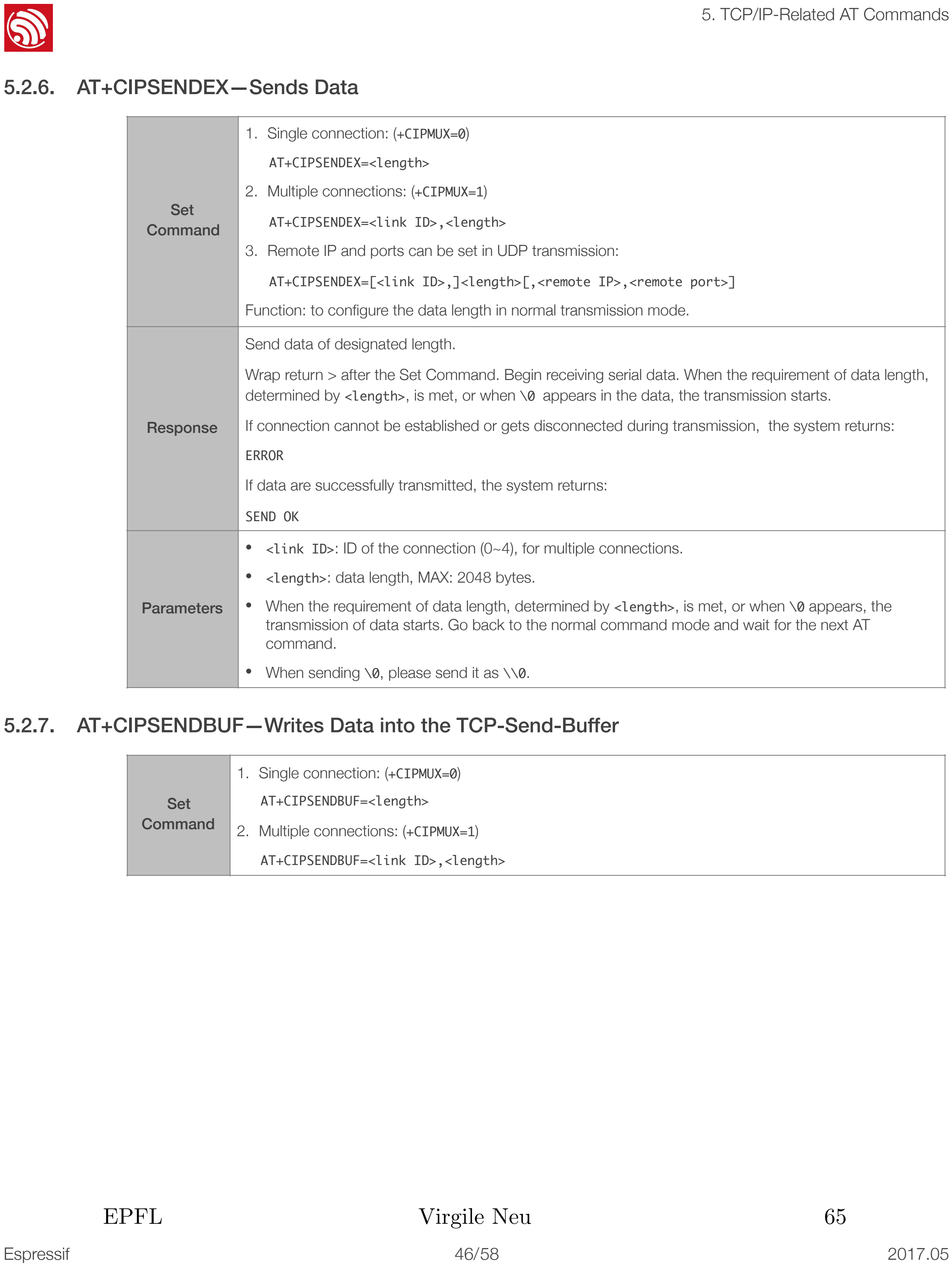
Parameters	`<size>`: the size of the SSL buffer; range of value: [2048, 4096].
Example	`AT+CIPSSLSIZE=4096`

5.2.5. 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`
		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*.	



5.2.6. 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`

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`.

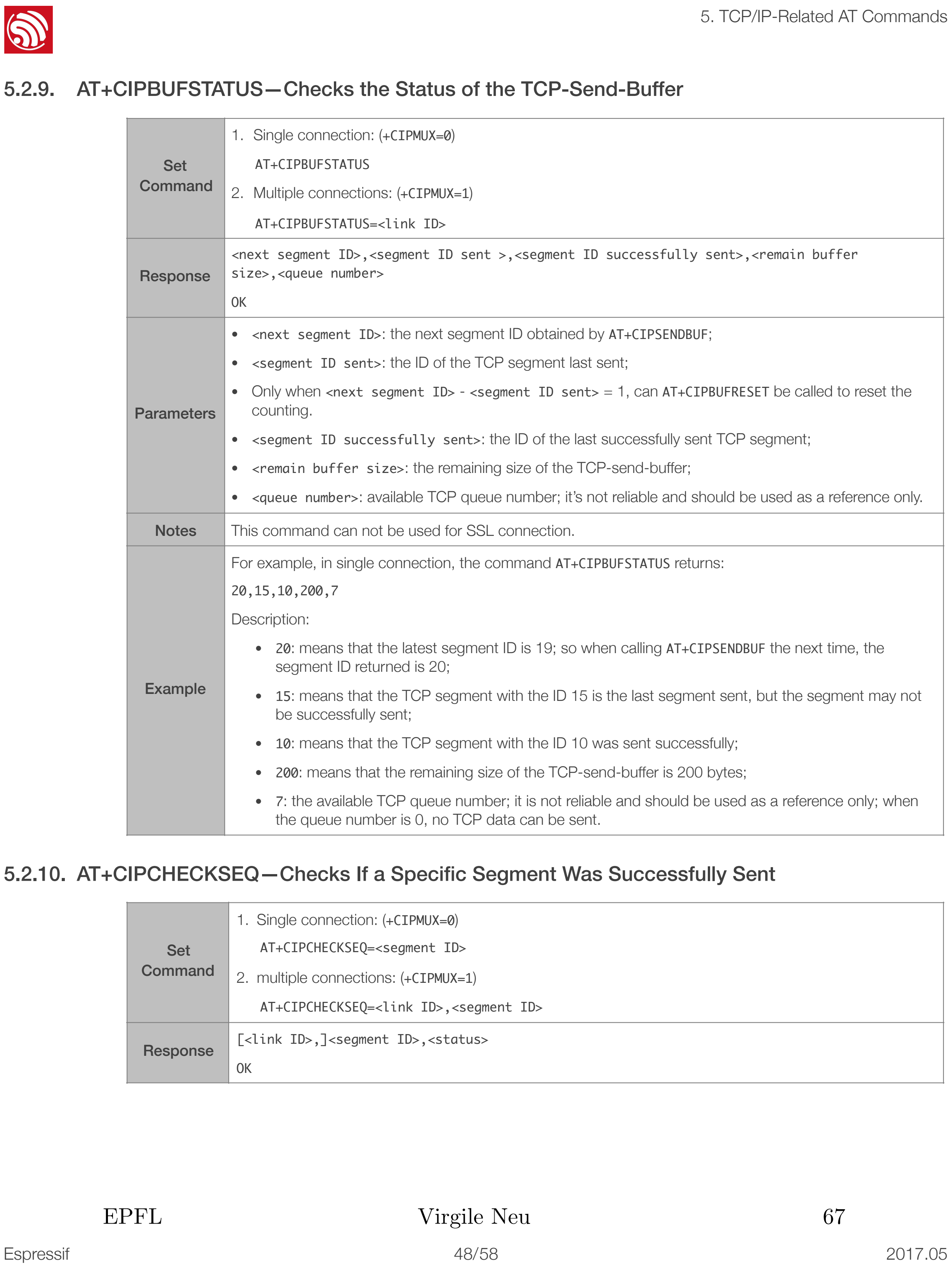
5.2.7. AT+CIPSENDBUF—Writes Data into the TCP-Send-Buffer

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
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.

5.2.8. 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.



5.2.9. 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>`

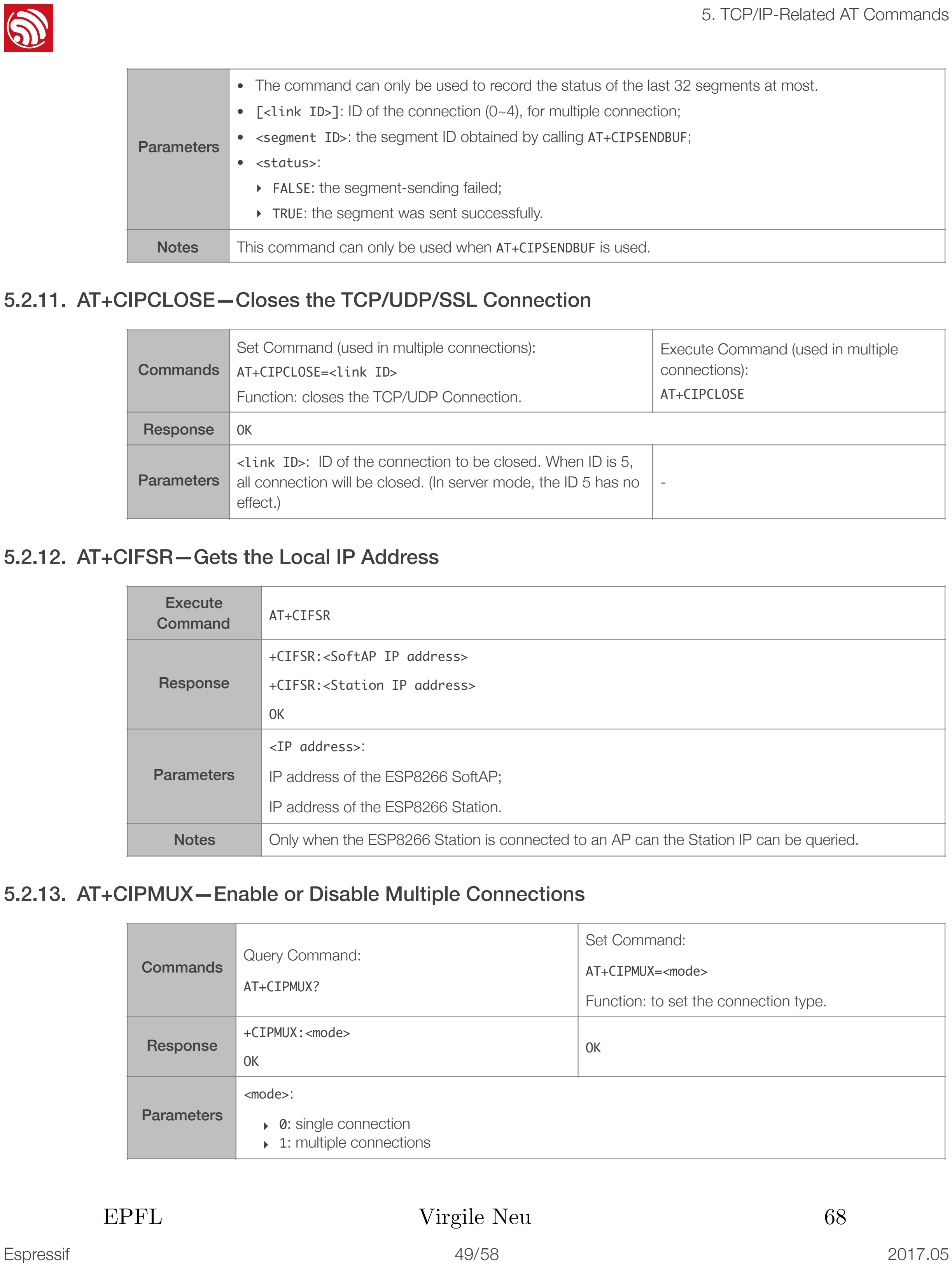
Response	`<next segment ID>,<segment ID sent >,<segment ID successfully sent>,<remain buffer size>,<queue number>`
	`OK`

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.

5.2.10. 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.

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

Commands	Set Command (used in multiple connections): `AT+CIPCLOSE=<link ID>` Function: closes 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 connection will be closed. (In server mode, the ID 5 has no effect.)	-

5.2.12. AT+CIFSR—Gets the Local IP Address

Execute Command	`AT+CIFSR`

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

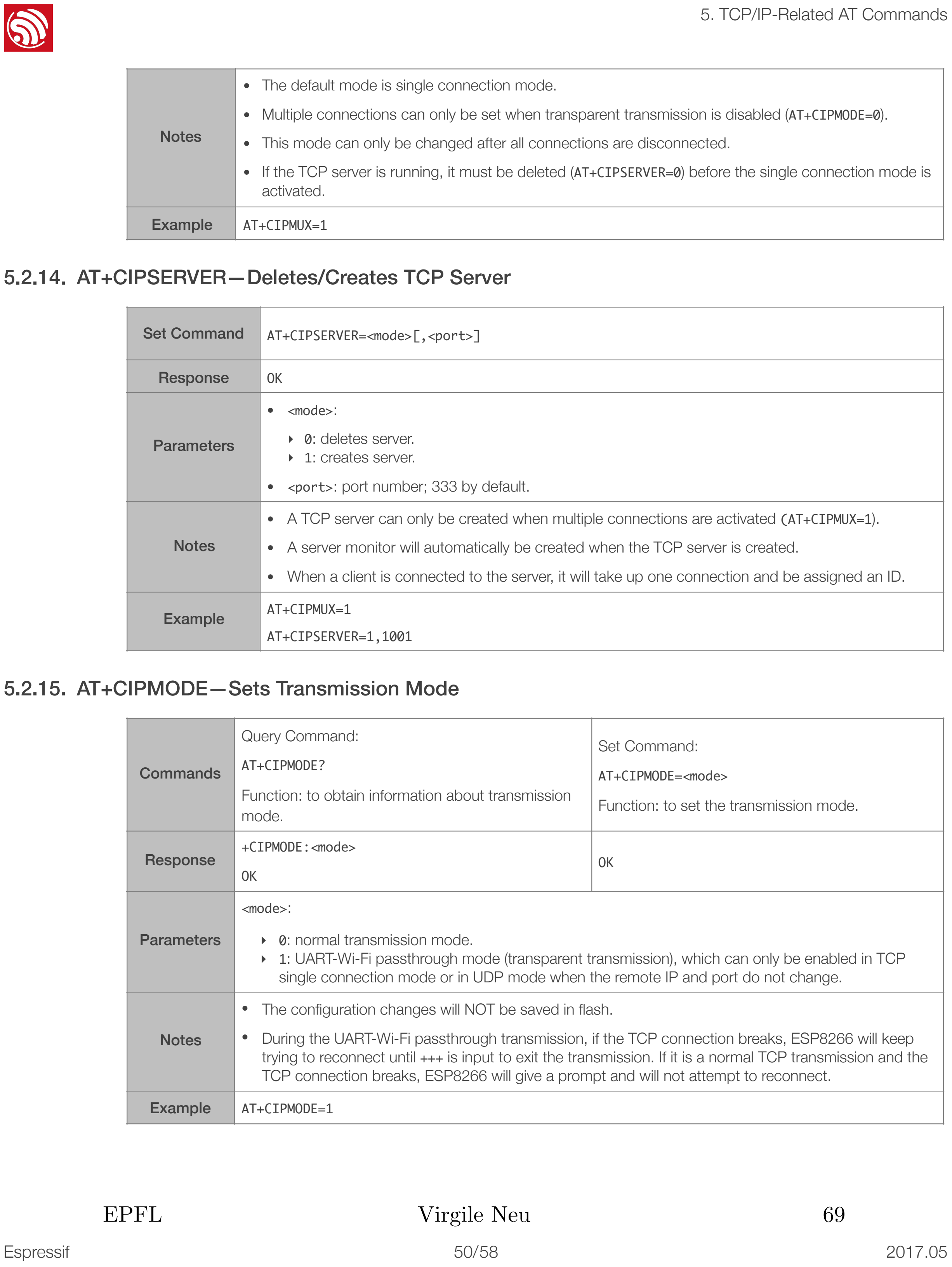
Parameters	`<IP address>`: IP address of the ESP8266 SoftAP; IP address of the ESP8266 Station.
Notes	Only when the ESP8266 Station is connected to an AP can the Station IP can be queried.

5.2.13. 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`

5.2.14. 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`

5.2.15. 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`	