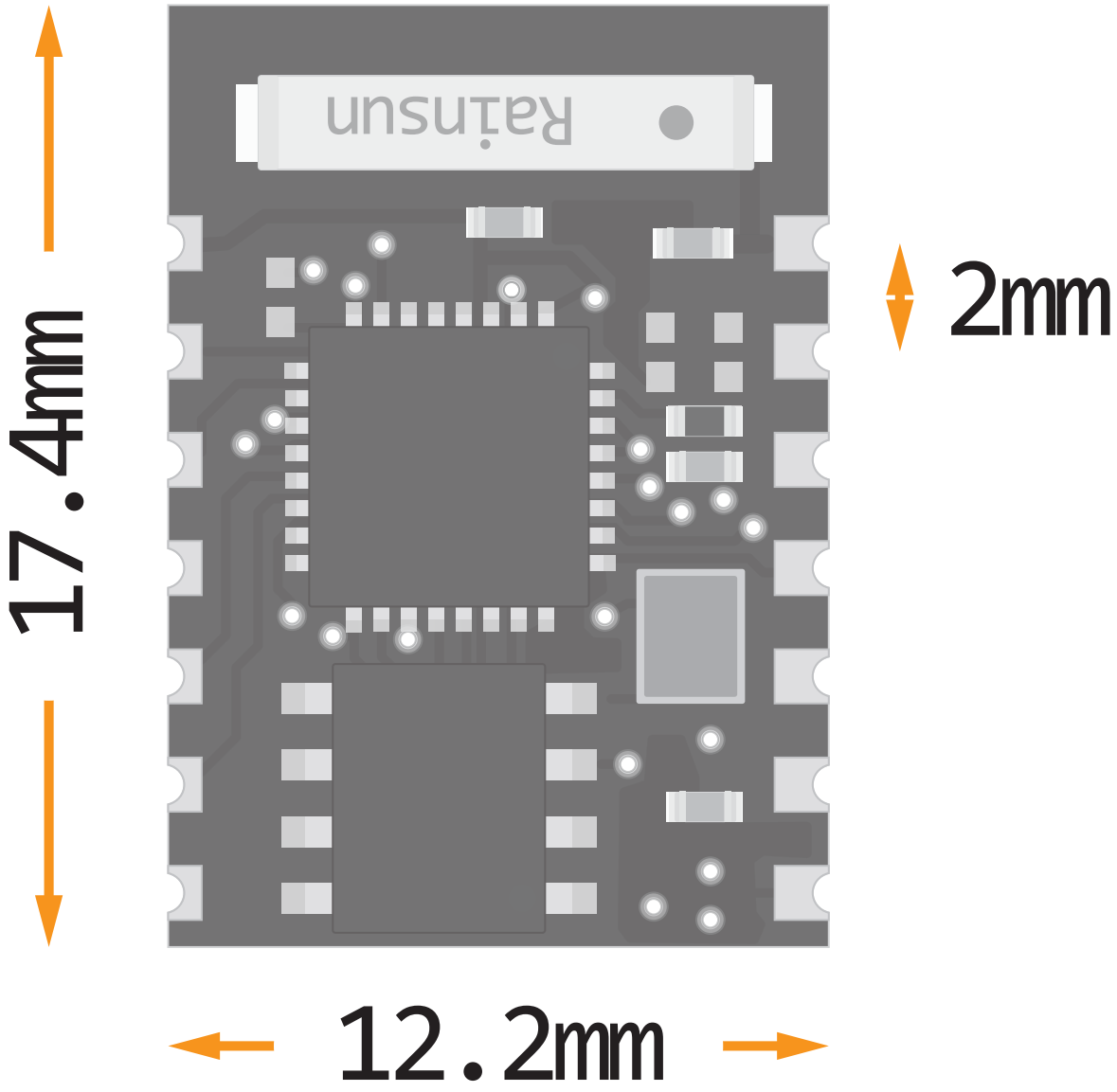
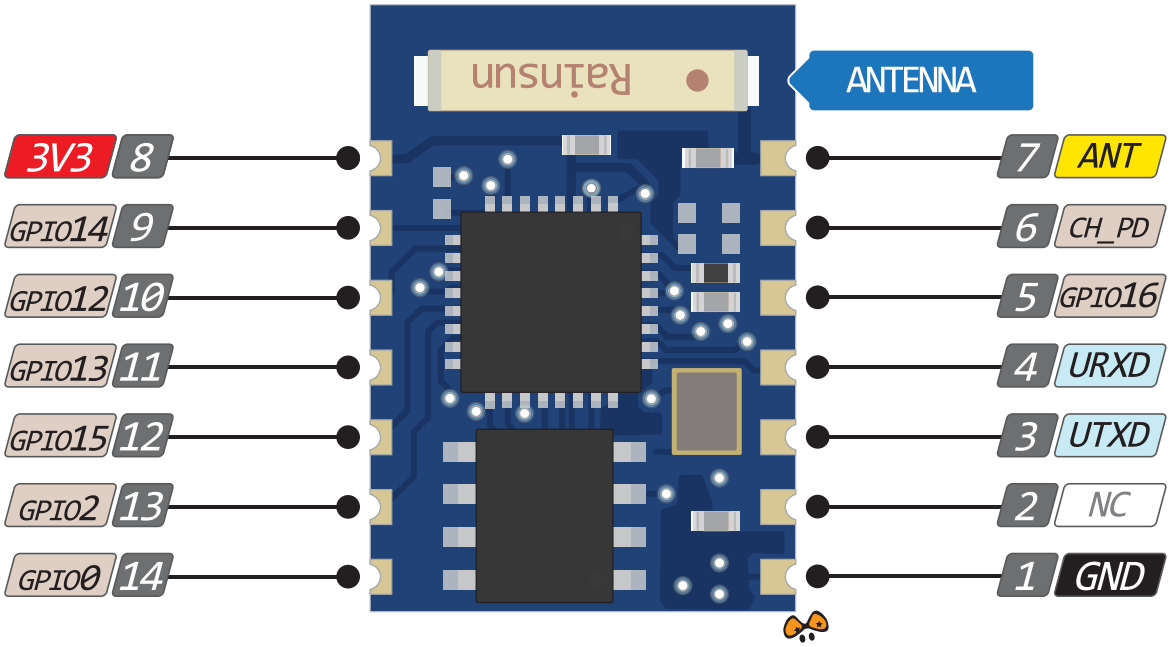
ESP8266

REFERENCE

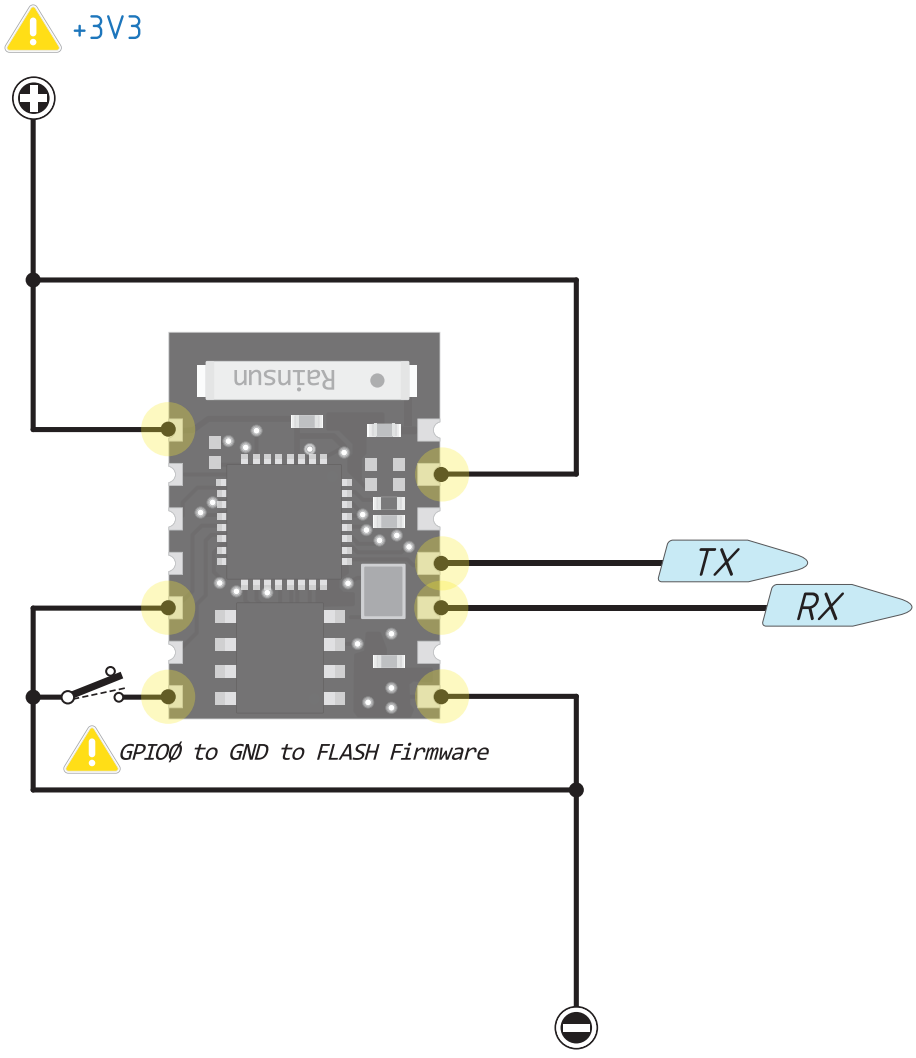
ESP8266 Dimensions



ESP8266 Pinout



ESP8266 Basic connect



AT Instruction Set
Overview

This is the documentation for ESP8266 AT command instruction set and usage. Instruction set is divided into: **Basic AT commands**, **WiFi function**, **AT commands**, **TCP/IP** Toolbox AT commands.

Version Info

Date	Version	Author	Changes
09 Dec 2014	1.0	Pighixxx	Draft

Disclaimer and Copyright Notice

Instruction Description

Each instruction set contains four types of AT commands.

Type	Format	Description
Test	**AT+<x>=?**	Query the Set command or internal parameters and its range values.
Query	**AT+<x>?**	Returns the current value of the parameter.
Set	**AT+<x>=<...>**	Set the value of user-defined parameters in commands and run.
Execute	**AT+<x>**	Runs commands with no user-defined parameters.

NOTE:

1. Not all AT instruction has four commands.
2. [] = default value, not required or may not appear
3. String values require double quotation marks, for example: AT+CWSAP=“ESP756190”,”21030826”,1,4
4. Baud rate = 115200
5. AT instruction ends with “\r\n”

AT Instruction Listing

Instruction	Description
Basic	
AT	Test AT startup
AT+RST	Restart
AT+GMR	View version info
AT+GSLP	Enter deep-sleep mode
ATE	AT commands echo
WiFi	
AT+CWMODE	WIFI mode(station/softAP/station+softAP)
AT+CWJAP	Connect to AP
AT+CWLAP	Lists available APs
AT+CWQAP	Disconnect from AP
AT+CWSAP	Set parameters under AP mode
AT+CWLIF	Get stations' ip which are connected to ESP8266 softAP
AT+CWDHCP	Enable/Disable DHCP
AT+CIPSTAMAC	Set mac address of ESP8266 station
AT+CIPAPMAC	Set mac address of ESP8266 softAP
AT+CIPSTA	Set ip address of ESP8266 station
AT+CIPAP	Set ip address of ESP8266 softAP
TCP/IP	
AT+CIPSTATUS	Get connection status
AT+CIPSTART	Establish TCP connection or register UDP port
AT+CIPSEND	Send data
AT+CIPCLOSE	Close TCP/UDP connection
AT+CIFSR	Get local IP address
AT+CIPMUX	Set multiple connections mode
AT+CIPSERVER	Configure as server
AT+CIPMODE	Set transmission mode
AT+CIPSTO	Set timeout when ESP8266 runs as TCP server
AT+CIUPDATE	Force OTA(upgrade through network)
Data RX	
+IPD	Data received from network

Basic AT Instruction Set
Overview

Instruction	Description
AT	Test AT startup
AT+RST	Restart module
AT+GMR	View version info
AT+GSLP	Enter deep-sleep mode
ATE	AT commands echo on/off

Instructions

AT – Test AT startup	
Instruction: **AT**	Response: **OK** _____ Param description: null

AT+RST – Restart Module	
Instruction: **AT+RST**	Response: **OK** _____ Param description: null

AT+GMR – View version Info	
Instruction: **AT+GMR**	Response: **<number>** **OK** _____ Param description: <number> Version Info, length: 8bytes
Note: response is 0017xxxxxx, then 0017 means the AT version.	

WiFi Functions
Overview

Instruction	Description
AT+CWMODE	WIFI mode(station/softAP/station+softAP)
AT+CWJAP	Connect to AP
AT+CWLAP	Lists available APs
AT+CWQAP	Disconnect from AP
AT+CWSAP	Set parameters under AP mode
AT+CWLIF	Get stations’ ip which are connected to ESP8266 softAP
AT+CWDHCP	Enable/Disable DHCP
AT+CIPSTAMAC	Set mac address of ESP8266 station
AT+CIPAPMAC	Set mac address of ESP8266 softAP
AT+CIPSTA	Set ip address of ESP8266 station
AT+CIPAP	Set ip address of ESP8266 softAP

Instructions

AT+CWMODE – WiFi mode(station/softAP/station+softAP)	
Type: **test** Function: **Get value scope of WiFi mode.** Instruction: **AT+CWMODE=?**	Response: **+CWMODE:(value scope of <mode>)** **OK** Param description: <mode> 1 – Station Mode 2 – AP Mode 3 – AP + Station Mode
Type: **query** Function: **Query ESP8266's current wifi mode.** Instruction: **AT+CWMODE?**	Response: **+CWMODE:<mode>** **OK** Param description: <mode> 1 – Station Mode 2 – AP Mode 3 – AP + Station Mode
Type: **set** Function: **Set ESP8266 wifi mode.** Instruction: **AT+CWMODE=<mode>**	Response: **OK** Param description: <mode> 1 – Station Mode 2 – AP Mode 3 – AP + Station Mode

AT+CWJAP – Connect to AP

Type: **test** Function: **Query AP's info which is connect by ESP8266.** Instruction: **AT+CWJAP?**	Response: **+CWJAP:<ssid>** **OK** Param description: <ssid> string, AP's SSID
Type: **set** Function: **Set AP's info which will be connect by ESP8266.** Instruction: **AT+CWJAP=<ssid>,<pwd>**	Response: **OK** **ERROR** Param description: <ssid> string, AP's SSID <pwd> string, MAX: 64bytes

AT+CWLAP – List available APs

Type: **set** Function: **Search available APs with specific conditions.** Instruction: **AT+ CWLAP = <ssid>,<mac>,<ch>**	Response: **+CWLAP:<ecn>,<ssid>,<rssi>,<mac>** **OK** **ERROR** Param description: <ssid> string, AP's SSID
Type: **execute** Function: **Lists all available APs.** Instruction: **AT+CWLAP**	Response: **+CWLAP:<ecn>,<ssid>,<rssi>,<mac>** **OK** **ERROR** Param description: <ecn> 0 – OPEN 1 – WEP 2 – WPA_PSK 3 – WPA2_PSK 4 – WPA_WPA2_PSK <ssid> string, AP's SSID <rssi> signal strength <mac> string, MAC address

AT+CWQAP – Disconnect from AP

Type: **test** Function: **Only for test.** Instruction: **AT+CWQAP=?**	Response: **OK** Param description: null
Type: **execute** Function: **Disconnect from AP.** Instruction: **AT+CWQAP**	Response: **OK** Param description: *null*

AT+CWSAP – Configuration of softAP mode

Type: **query** Function: **Query configuration of softAP mode.** Instruction: **AT+ CWSAP?**	Response: **+CWSAP:<ssid>,<pwd>,<chl>,<ecn>** **OK** **ERROR** Param description: *The same as below*
Type: **set** Function: **Set configuration of softAP mode.** Instruction: **AT+CWSAP=<ssid>,<pwd>,<chl>,<ecn>**	Response: **OK** **ERROR** Param description: <ecn> 0 – OPEN 1 – WEP 2 – WPA_PSK 3 – WPA2_PSK 4 – WPA_WPA2_PSK <ssid> string, AP's SSID <pwd> string, MAX: 64bytes <chl> channel ID

Note: This CMD is only available when softAP mode enable, and need to follow by AT+RST to make it works.

AT+CWLIF — ip of stations which are connected to ESP8266 softAP

Type: **execute**

Function:

Get ip of stations which are connected to ESP8266 softAP.

Instruction:

AT+CWLIF

Response:

<ip addr>
OK

Param description:

<ip addr>
ip address of stations which are connected to ESP8266 softAP

AT+CWDHCP — Enable/Disable DHCP

Type: **set**

Function:

Enable/Disable DHCP.

Instruction:

AT+CWDHCP=<mode>,<en>

Response:

OK

Param description:

<mode>
1 - Station Mode
2 - AP Mode
3 - AP + Station Mode
<en>
0 - enable DHCP
1 - disable DHCP

AT+CIPSTAMAC — Set mac address of station

Type: **query**

Function:

Get mac address of ESP8266 station.

Instruction:

AT+CIPSTAMAC?

Response:

+CIPSTAMAC:<mac>
OK

Param description:

<mac>
string, mac address of station

Type: **set**

Function:

Set mac address of ESP8266 station.

Instruction:

AT+CIPSTAMAC=<mac>

Response:

OK

Param description:

<mac>
string, mac address of station

AT+CIPAPMAC – Set mac address of softAP

Type: **query** Function: **Get mac address of ESP8266 softAP.** Instruction: **AT+CIPAPMAC?**	Response: **+CIPAPMAC:<mac>** **OK** Param description: <mac> string, mac address of softAP
Type: **set** Function: **Set mac address of ESP8266 softAP.** Instruction: **AT+CIPAPMAC=<mac>**	Response: **OK** Param description: <mac> string, mac address of softAP

AT+CIPSTA – Set ip address of ESP8266 station

Type: **query** Function: **Get ip address of ESP8266 station.** Instruction: **AT+CIPSTA?**	Response: **+CIPSTA:<ip>** **OK** Param description: <ip> string, ip address of station
Type: **set** Function: **Set ip address of ESP8266 station.** Instruction: **AT+CIPSTA=<ip>**	Response: **OK** Param description: <ip> string, ip address of station

AT+CIPAP – Set ip address of ESP8266 softAP

Type: **query** Function: **Get ip address of ESP8266 softAP.** Instruction: **AT+CIPAP?**	Response: **+CIPAP:<ip>** **OK** Param description: <ip> string, ip address of softAP
Type: **set** Function: **Set ip address of ESP8266 softAP.** Instruction: **AT+CIPAP=<ip>**	Response: **OK** Param description: <ip> string, ip address of softAP

TCP/IP Related
Overview

Instruction	Description
AT+CIPSTATUS	Get connection status
AT+CIPSTART	Establish TCP connection or register UDP port
AT+CIPSEND	Send data
AT+CIPCLOSE	Close TCP/UDP connection
AT+CIFSR	Get local IP address
AT+CIPMUX	Set multiple connections mode
AT+CIPSERVER	Configure as server
AT+CIPMODE	Set transmission mode
AT+CIPSTO	Set timeout when ESP8266 runs as TCP server
AT+CIUPDATE	Force OTA(upgrade through network)

Instructions

AT+CIPSTATUS – Information about connection

Type: **execute**

Function:

Get information about connection

Instruction:

AT+CIPSTATUS

Response:

STATUS:<stat>
+CIPSTATUS:<id>,<type>,<addr>,<port>,<tetype>

OK

Param description:

<stat>
    2 – Got IP
    3 – Connected
    4 – Disconnected
<id>
    id of the connection (0~4), for multi-connect
<type>
    “TCP” or “UDP”
<addr>
    string, IP address
<port>
    port number
<tetype>
    0 – ESP8266 run as a client
    1 – ESP8266 run as a server

AT+CIPSTART
Establish TCP connection or register UDP port, start connection

Type: **test** Function: **Get the information of param.** Instruction: **AT+CIPSTART=?**	Response: **If AT+CIPMUX=0** **+CIPSTART:(<type>),(<IPaddress>),(<port>)[,(<localport>),(<mode>)]** **+CIPSTART:(<type>),(<domainname>),(<port>)[,(<localport>),(<mode>)]** **OK** **If AT+CIPMUX=1** **+CIPSTART:(id),(<type>),(<IPaddress>),(<port>)[,(<localport>),(<mode>)] +CIPSTART:(id),(<type>),(<domain name>),(<port>)[,(<localport>),(<mode>)]** **OK** Param description: null
Type: **set** Function: **Start a connection as client.** Instruction: SINGLE CONNECTION **(+CIPMUX=0)** **AT+CIPSTART=<type>,<addr>,<port> [,(<localport>),(<mode>)]** MULTIPLE CONNECTIONS **(+CIPMUX=1)** **AT+CIPSTART=<id><type>,<addr>,<port> [,(<localport>),(<mode>)]**	Response: **OK** **ERROR** **ALREADY CONNECT** Param description: <id> ID of connection (0–4) <type> “TCP” or “UDP” <addr> string, remote IP <port> string, remote port [<localport>] for UDP only [<mode>] for UDP only 0 – destination peer entity of UDP will not change. 1 – destination peer entity of UDP can change once 2 - destination peer entity of UDP is allowed to change.

Note: [<mode>] can only be used when [<local port>] is set.

AT+CIPSEND – Send data

Type: **test** Function: **Only for test.** Instruction: **AT+CIPSEND=?**	Response: **OK** Param description: null
Type: **set** Function: **Set length of the data that will be sent. For normal send.** Instruction: SINGLE CONNECTION **(+CIPMUX=0)** **AT+CIPSEND=<length>** MULTIPLE CONNECTIONS **(+CIPMUX=1)** **AT+CIPSEND=<id>,<length>**	Response: Wrap return “>” after set command. Begins receive of serial data, when data length is met, starts transmission of data. If connection cannot be established or gets disconnected during send, returns **ERROR** If data is transmitted successfully, returns **SEND OK** Param description: <id> ID of transmit connection <length> data length, MAX 2048 bytes
Type: **execute** Function: **Send data. For unvarnished transmission mode.** Instruction: **AT+CIPSEND**	Response: Wrap return “>” after execute command. Enters unvarnished transmission, 20ms interval between each packet, maximum 2048 bytes per packet. When single packet containing “+++” is received, it returns to command mode. This command can only be used in unvarnished transmission mode which require to be single connection mode.

AT+CIPCLOSE – Close TCP or UDP connection

Type: **test** Function: **Only for test.** Instruction: **AT+CIPCLOSE=?**	Response: **OK** Param description: null
Type: **set** Function: **Close TCP or UDP connection.** Instruction: MULTIPLE CONNECTIONS **AT+CIPCLOSE=<id>**	Response: Wrap return “>” after set command. Begins receive of serial data, when data length is met, starts transmission of data. If connection cannot be established or gets disconnected during send, returns **ERROR** If data is transmitted successfully, returns **SEND OK** Param description: <id> ID no. of connection to close, when id=5, all connections will be closed.
Type: **execute** Function: **For single connection mode** Instruction: **AT+CIPCLOSE**	Response: **OK** If no such connection, returns **ERROR** **UNLINK** when there is no connection.
Note: id=5 has no effect in server mode	

AT+CIFSR – Get local IP address

Type: **test** Function: **Only for test.** Instruction: **AT+CIFSR=?**	Response: **OK** Param description: null
Type: **execute** Function: **Get local IP address.** Instruction: **AT+CIFSR**	Response: **+CIFSR:<IP address>** **OK** **ERROR** Param description: <IP address> IP address of ESP8266

Note: <IP address> for softAP or station

AT+CIPMUX – Enable multiple connections or not

Type: **query** Function: **Get param config.** Instruction: **AT+CIPMUX?**	Response: **+CIPMUX:<mode>** **OK** Param description: The same as below
Type: **set** Function: **Set connection mode.** Instruction: **AT+CIPMUX=<mode>**	Response: **OK** If already connected, returns **LINK IS BUILDED** Param description: <mode> 0 – Single connection 1 – Multiple connections

Note: This mode can only be changed after all connections are disconnected. If server is started, reboot is required.

AT+CIPSERVER – Configure as TCP server

Type: **set** Function: **Set TCP server.** Instruction: **AT+CIPSERVER= <mode>[,<port>]**	Response: **OK** Param description: <mode> 0 – Delete server (need to follow by restart) 1 – Create server <port> port number, default is 333

Note:
1. Server can only be created when **AT+CIPMUX=1**
2. Server monitor will automatically be created when Server is created.
3. When a client is connected to the server, it will take up one connection,be gave an id.

AT+CIPMODE – Set transfer mode

Type: **query** Function: **Query transfer mode.** Instruction: **AT+CIPMODE?**	Response: **+CIPMODE:<mode>** **OK** Param description: The same as below
Type: **set** Function: **Set transfer mode.** Instruction: **AT+CIPMODE=<mode>**	Response: **OK** If already connected, returns **LINK IS BUILDED** Param description: <mode> 0 – Normal mode 1 – unvarnished transmission mode

AT+CIPSTO – Set server timeout

Type: **query** Function: **Query server timeout.** Instruction: **AT+CIPSTO?**	Response: **+CIPSTO:<time>** **OK** Param description: The same as below
Type: **set** Function: **Set server timeout.** Instruction: **AT+CIPSTO=<time>**	Response: **OK** Param description: <time> server timeout, range 0–7200 seconds

AT+CIUPDATE – update through network

Type: **execute** Function: **Start upgrade.** Instruction: **AT+CIPMODE?**	Response: **+CIPUPDATE:<n>** **OK** Param description: <mode> 0 – found server 1 – connect to server 2 – download firmware 4 – flash firmware

+IPD – Receive network data

Instruction:	Param description:
SINGLE CONNECTION **(+CIPMUX=0)** **+IPD,<len>:<data>** MULTIPLE CONNECTIONS **(+CIPMUX=1)** **+IPD,<id>,<len>:<data>**	<id> number ID of connection <len> data length <data> data received

Note: When the module receives network data, it will send the data through the serial port using +IPD command