a SUNSEA AIOT company

# SIM7028 Series_ TCPIP_Application Note

LPWA Module

**SIMCom Wireless Solutions Limited**
SIMCom Headquarters Building, Building 3, No. 289 Linhong
Road, Changning District, Shanghai P.R. China
Tel: 86-21-31575100
support@simcom.com
www.simcom.com

| | |
|---|---|
| **Document Title:** | SIM7028 Series_TCPIP_Application Note |
| **Version:** | 1.04 |
| **Date:** | 2023.03.30 |
| **Status:** | Released |

## GENERAL NOTES

SIMCOM OFFERS THIS INFORMATION AS A SERVICE TO ITS CUSTOMERS, TO SUPPORT APPLICATION AND ENGINEERING EFFORTS THAT USE THE PRODUCTS DESIGNED BY SIMCOM. THE INFORMATION PROVIDED IS BASED UPON REQUIREMENTS SPECIFICALLY PROVIDED TO SIMCOM BY THE CUSTOMERS. SIMCOM HAS NOT UNDERTAKEN ANY INDEPENDENT SEARCH FOR ADDITIONAL RELEVANT INFORMATION, INCLUDING ANY INFORMATION THAT MAY BE IN THE CUSTOMER'S POSSESSION. FURTHERMORE, SYSTEM VALIDATION OF THIS PRODUCT DESIGNED BY SIMCOM WITHIN A LARGER ELECTRONIC SYSTEM REMAINS THE RESPONSIBILITY OF THE CUSTOMER OR THE CUSTOMER'S SYSTEM INTEGRATOR. ALL SPECIFICATIONS SUPPLIED HEREIN ARE SUBJECT TO CHANGE.

## COPYRIGHT

**SIMCom Wireless Solutions Limited**
SIMCom Headquarters Building, Building 3, No. 289 Linhong Road, Changning District, Shanghai P.R. China
Tel: +86 21 31575100
Email: simcom@simcom.com

**For more information, please visit:**
https://www.simcom.com/download/list-863-en.html

**For technical support, or to report documentation errors, please visit:**
https://www.simcom.com/ask/ or email to: support@simcom.com

# About Document

## Version History

| Revision | Date | Chapter | Description |
|---|---|---|---|
| V1.00 | 2022.05.12 | All | New version |
| V1.01 | 2022.05.31 | All | Update file |
| V1.02 | 2022.07.06 | All | Update file |
| V1.03 | 2022.11.22 | All | Update file |
| V1.04 | 2023.03.07 | 3.1.1Network Environment | Word modification |

## Scope

This document could be applied to following modules.

| Name | Type | Size(mm) | Description |
|---|---|---|---|
| SIM7028 | NB2 | 17.6*15.7 | Band 1/2/3/4/5/8/12/13/14/17/18/19/20/25/26/28/66/70/85 |

# Contents

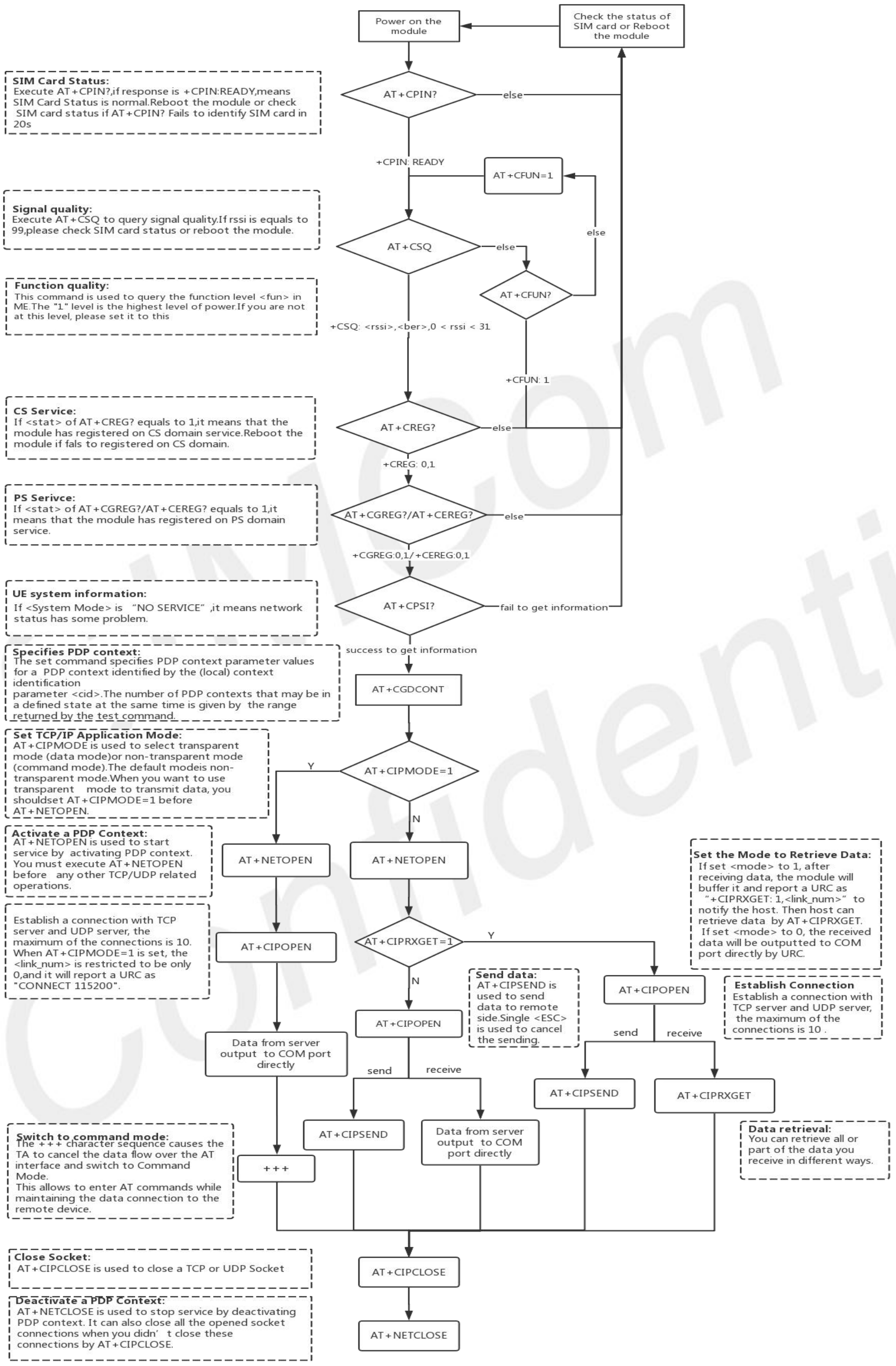
# 1 Introduction

## 1.1 Purpose of the document

Based on module AT command manual, this document will introduce TCPIP application process for SIM7028 series of module.Developers could understand and develop application quickly and efficiently based on this document.

## 1.2 The process of Using TCPIP AT Commands

Figure illustrates how to use TCP/IP AT commands:



**SIM Card Status:**
Execute AT+CPIN?,if response is +CPIN:READY,means SIM Card Status is normal.Reboot the module or check SIM card status if AT+CPIN? Fails to identify SIM card in 20s

**Signal quality:**
Execute AT+CSQ to query signal quality.If rssi is equals to 99,please check SIM card status or reboot the module.

**Function quality:**
This command is used to query the function level <fun> in ME.The "1" level is the highest level of power.If you are not at this level, please set it to this

**CS Service:**
If <stat> of AT+CREG? equals to 1,it means that the module has registered on CS domain service.Reboot the module if fals to registered on CS domain.

**PS Serivce:**
If <stat> of AT+CGREG?/AT+CEREG? equals to 1,it means that the module has registered on PS domain service.

**UE system information:**
If <System Mode> is "NO SERVICE" ,it means network status has some problem.

**Specifies PDP context:**
The set command specifies PDP context parameter values for a PDP context identified by the (local) context identification
parameter <cid>.The number of PDP contexts that may be in a defined state at the same time is given by the range returned by the test command.

**Set TCP/IP Application Mode:**
AT+CIPMODE is used to select transparent mode (data mode)or non-transparent mode (command mode).The default modeis non-transparent mode.When you want to use transparent mode to transmit data, you shouldset AT+CIPMODE=1 before AT+NETOPEN.

**Activate a PDP Context:**
AT+NETOPEN is used to start service by activating PDP context. You must execute AT+NETOPEN before any other TCP/UDP related operations.

Establish a connection with TCP server and UDP server, the maximum of the connections is 10. When AT+CIPMODE=1 is set, the <link_num> is restricted to be only 0,and it will report a URC as "CONNECT 115200".

**Set the Mode to Retrieve Data:**
If set <mode> to 1, after receiving data, the module will buffer it and report a URC as "+CIPRXGET: 1,<link_num>" to notify the host. Then host can retrieve data by AT+CIPRXGET.
If set <mode> to 0, the received data will be outputted to COM port directly by URC.

**Send data:**
AT+CIPSEND is used to send data to remote side.Single <ESC> is used to cancel the sending.

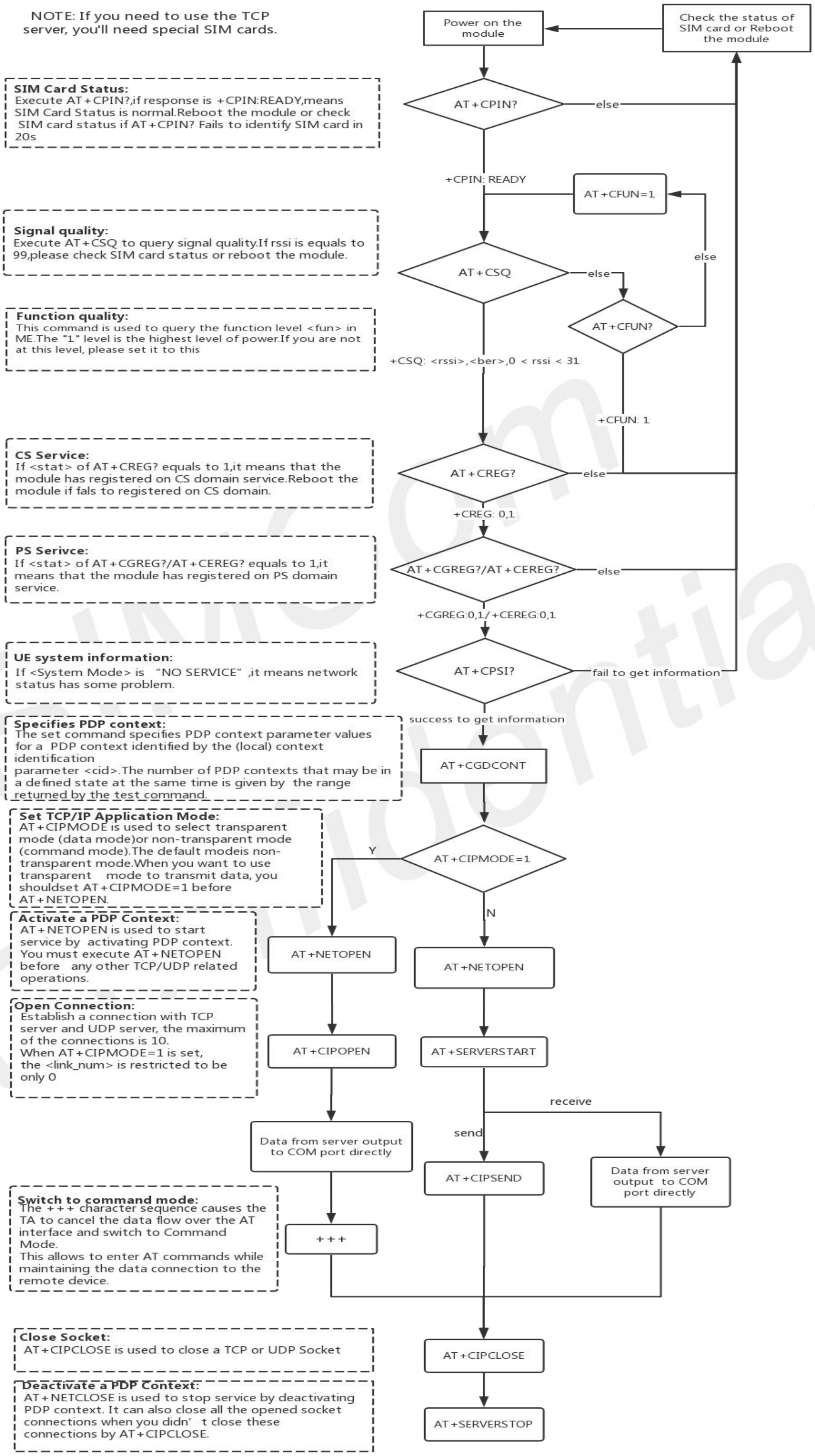
**Establish Connection**
Establish a connection with TCP server and UDP server, the maximum of the connections is 10 .

**Data retrieval:**
You can retrieve all or part of the data you receive in different ways.

**Switch to command mode:**
The +++ character sequence causes the TA to cancel the data flow over the AT interface and switch to Command Mode.
This allows to enter AT commands while maintaining the data connection to the remote device.

**Close Socket:**
AT+CIPCLOSE is used to close a TCP or UDP Socket

**Deactivate a PDP Context:**
AT+NETCLOSE is used to stop service by deactivating PDP context. It can also close all the opened socket connections when you didn’ t close these connections by AT+CIPCLOSE.



NOTE: If you need to use the TCP server, you'll need special SIM cards.

**SIM Card Status:**
Execute AT+CPIN?,if response is +CPIN:READY,means SIM Card Status is normal.Reboot the module or check SIM card status if AT+CPIN? Fails to identify SIM card in 20s

**Signal quality:**
Execute AT+CSQ to query signal quality.If rssi is equals to 99,please check SIM card status or reboot the module.

**Function quality:**
This command is used to query the function level <fun> in ME.The "1" level is the highest level of power.If you are not at this level, please set it to this

**CS Service:**
If <stat> of AT+CREG? equals to 1,it means that the module has registered on CS domain service.Reboot the module if fals to registered on CS domain.

**PS Serivce:**
If <stat> of AT+CGREG?/AT+CEREG? equals to 1,it means that the module has registered on PS domain service.

**UE system information:**
If <System Mode> is "NO SERVICE" ,it means network status has some problem.

**Specifies PDP context:**
The set command specifies PDP context parameter values for a PDP context identified by the (local) context identification parameter <cid>.The number of PDP contexts that may be in a defined state at the same time is given by the range returned by the test command.

**Set TCP/IP Application Mode:**
AT+CIPMODE is used to select transparent mode (data mode)or non-transparent mode (command mode).The default modeis non-transparent mode.When you want to use transparent mode to transmit data, you shouldset AT+CIPMODE=1 before AT+NETOPEN.

**Activate a PDP Context:**
AT+NETOPEN is used to start service by activating PDP context. You must execute AT+NETOPEN before any other TCP/UDP related operations.

**Open Connection:**
Establish a connection with TCP server and UDP server, the maximum of the connections is 10.
When AT+CIPMODE=1 is set, the <link_num> is restricted to be only 0

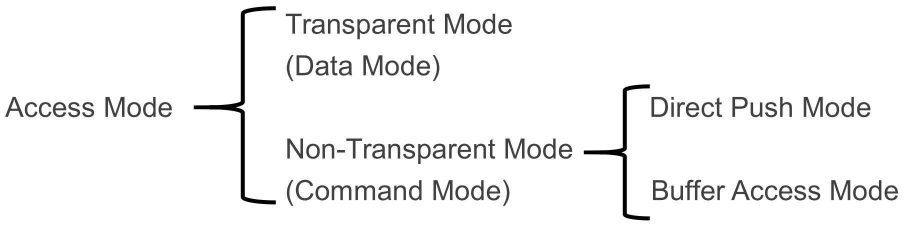
**Switch to command mode:**
The +++ character sequence causes the TA to cancel the data flow over the AT interface and switch to Command Mode.
This allows to enter AT commands while maintaining the data connection to the remote device.

**Close Socket:**
AT+CIPCLOSE is used to close a TCP or UDP Socket

**Deactivate a PDP Context:**
AT+NETCLOSE is used to stop service by deactivating PDP context. It can also close all the opened socket connections when you didn' t close these connections by AT+CIPCLOSE.

## 1.3 Description of Data Access Mode

SIM7028 series of module could support following data access mode for TCP\UDP data transmission.



The default mode is direct push mode.

### 1. Transparent Mode

AT+CIPMODE=1 is used to enter into transparent access mode. In transparent mode, all data received from COM port will be sent to remote side directly, and all received data from remote side will be output to COM port directly as well. “+++” could be used to exit from transparent access mode,when “+++” command returns OK, the module will be switched to command mode. In transparent access mode, host cannot execute any AT command. Currently, only one socket is available under transparent mode, either TCP\UDP client or TCP\UDP server.In transparent mode, the first server (<server_index> = 0) and the first client socket(<link_num> = 0) are used for transparent mode operation. Other servers index (<server_index> = 1-3) and other client sockets index (<link_num> = 1-9) are still running in command mode.

### 2. Non-Transparent Mode

**Direct Push Mode:**
AT+CIPRXGET=0 is used to enter direct push mode.In direct push mode, customer need to send data by AT+CIPSEND command. The received data will be outputted to COM port directly by URC as “+RECV FROM:<IP ADDRESS>:<PORT><CR><LF>+IPD(data length)<CR><LF><data>”.

**Buffer Access Mode:**
AT+CIPRXGET=1 is used to enter into buffer access mode. In buffer access mode, customer need to send data by AT+CIPSEND command. When receiving data, the module will buffer it internally and report a URC as “+CIPRXGET: 1,<link_num>” to notify the host. Then host can achieve data by AT+CIPRXGET.

### 3. Switch Between Data Mode and Command Mode

**(1) Data mode -> Command mode**
**Software switching:** By command sequence +++, this is a complete command, do not separate each character. And the time delay before and after this sequence should be more than 1000 milliseconds, the interval of each character should not be more than 900 milliseconds.
**Hardware switching:** DTR pin could be used to trigger data mode to command mode. AT&D1 should be configured before application.

# 2 TCPIP AT Commands

## 2.1 Description of AT Commands

### 2.1.1 AT+NETOPEN Start Socket Service

AT+NETOPEN is used to start service by activating PDP context. You must execute AT+NETOPEN before any other TCP/UDP related operations.

| AT+NETOPEN Start Socket Service | |
|---|---|
| Read Command<br>**AT+NETOPEN?** | Response<br>**+NETOPEN: <net_state>**<br><br>**OK** |
| Execute Command<br>**AT+NETOPEN** | Response<br>1)If the PDP context has not been activated or the network closed abnormally, response:<br>**OK**<br><br>**+NETOPEN: <err>**<br>2)When the PDP context has been activated successfully, if you execute AT+NETOPEN again, response:<br>**+IP ERROR: Network is already opened**<br><br>**ERROR**<br>3)other:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | Range: 3000ms-120000ms<br>default: 120000ms<br>(it can be set by AT+CIPTIMEOUT) |
| Reference | 3GPP TS 27.005 |

### Defined Values

| | |
|---|---|
| **<net_state>** | Integer type, indicates the state of PDP context activation.<br>0 network close (deactivated)<br>1 network open(activated) |
| **<err>** | Integer type, the result of operation.<br>0 is success, other value is failure, please refer to Chapter 9.3.2 for details |

### Examples

```
AT+NETOPEN?
+NETOPEN: 1

OK
AT+NETOPEN
OK

+NETOPEN: 0
```

## 2.1.2 AT+NETCLOSE Stop Socket Service

AT+NETCLOSE is used to stop service by deactivating PDP context. It can also close all the opened socket connections when you didn't close these connections by AT+CIPCLOSE.

| AT+NETCLOSE Stop Socket Service | |
|---|---|
| Test Command<br>**AT+NETCLOSE=?** | Response<br>**OK** |
| Execute Command<br>**AT+NETCLOSE** | Response<br>1)If the PDP context has been activated, response:<br>**OK**<br><br>**+NETCLOSE: <err>**<br>2)If the PDP context has been activated and one connection is in non-transparent mode and transparent mode, response:<br><br>**OK**<br><br>**CLOSED**<br><br>**+CIPCLOSE: <link_num>,<err>** |

| | |
|---|---|
| | **+NETCLOSE: <err>**<br>3)If the PDP context has not been activated, response:<br>**+NETCLOSE: <err>**<br><br>**ERROR**<br>4)Others:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | Range: 3000ms-120000ms<br>default: 120000ms<br>(it can be set by AT+CIPTIMEOUT) |
| Reference | |

**Defined Values**

| | |
|---|---|
| **<err>** | Integer type, the result of operation.<br>0 is success, other value is failure, please refer to Chapter 9.3.2 for details |

**Examples**

```
AT+NETCLOSE
OK

+NETCLOSE: 0
```

### 2.1.3 AT+CIPOPEN Establish Connection in Multi-Socket Mode

You can use AT+CIPOPEN to establish a connection with TCP server and UDP server, the maximum of the connections is 2.

| AT+CIPOPEN Establish Connection in Multi-Socket Mode | |
|---|---|
| Test Command<br>AT+CIPOPEN=? | Response<br>**+CIPOPEN: (0-1),("TCP","UDP")**<br><br>**OK** |
| Read Command<br>AT+CIPOPEN? | Response<br>**+CIPOPEN:**<br>**<link_num>[,<type>,<serverIP>,<serverPort>,<index>]**<br>**+CIPOPEN:** |

| | |
|---|---|
| | **<link_num>[,<type>,<serverIP>,<serverPort>,<index>]**<br>**[…]**<br><br>**OK**<br>If a connection identified by <link_num>has not been established successfully, only +CIPOPEN: <link_num> will be returned. |
| Write Command<br>TCP connection<br>**AT+CIPOPEN=<link_num>,"TCP",<serverIP>,<serverPort>[,<localPort>]** | Response<br>1)if PDP context has been activated successfully, response:<br>**OK**<br><br>**+CIPOPEN: <link_num>,<err>**<br>2)when the <link_num> is greater than 9, response:<br>**+IP ERROR: Invalid parameter**<br><br>**ERROR**<br>3)If PDP context has not been activated, or the connection has been established, or parameter is incorrect, or when AT+CIPMODE=1 is set, the <link_num> is greater than 0, or other errors, response:<br>**+CIPOPEN: <link_num>,<err>**<br><br>**ERROR**<br>4)Transparent mode for TCP connection:<br>When you want to use transparent mode to transmit data, you should set AT+CIPMODE=1 before AT+NETOPEN. And if AT+CIPMODE=1 is set, the <link_num> is restricted to be only 0.<br>if success<br>**CONNECT [<text>]**<br>if failure<br>**CONNECT FAIL**<br>5)Others:<br>**ERROR** |
| Write Command<br>UDP Connection<br>**AT+CIPOPEN=<link_num>,"UDP",,,<localPort>** | 1)If PDP context has been activated successfully, response:<br>**+CIPOPEN: <link_num>,0**<br><br>**OK**<br>2)When the <link_num> is greater than 9, response:<br>**+IP ERROR: Invalid parameter**<br><br>**ERROR**<br>If PDP context has not been activated, or the connection has been established, or parameter is incorrect, or other errors, response:<br>**+CIPOPEN: <link_num>,<err>**<br><br>**ERROR** |

| | |
|---|---|
| | 3)Others:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | Range: 3000ms-120000ms<br>default: 120000ms<br>(it can be set by AT+CIPTIMEOUT) |
| Reference | |

## Defined Values

| | |
|---|---|
| **<link_num>** | Integer type, identifies a connection. Range is 0-1.<br>If AT+CIPMODE=1 is set, the <link_num> is restricted to be only 0. |
| **<type>** | String type, identifies the type of transmission protocol.<br>TCP Transmission Control Protocol<br>UDP User Datagram Protocol |
| **<serverIP>** | String type, identifies the IP address of server. The IP address format consists of 4 octets, separated by decimal point, like "AAA.BBB.CCC.DDD". Also the domain name is supported here. |
| **<serverPort>** | Integer type, identifies the port of TCP server, range is 0-65535.<br>NOTE:<br>When open port as TCP, the port must be the opened TCP port;<br>When open port as UDP, the port may be any port. |
| **<localPort>** | Integer type, identifies the port of local socket, range is 0-65535. |
| **<index>** | Integer type, indicates whether the module is used as a client or server. When used as server, the range is 0-3,<index> is the server index to which the client is linked.<br>-1 TCP client<br>0-3 TCP server index |
| **<text>** | String type, indicates CONNECT result code. |
| **<err>** | Integer type, the result of operation.<br>0 is success, other value is failure, please refer to Chapter 9.3.2 for details |

## Examples

```
AT+CIPOPEN=?
+CIPOPEN: (0-1),("TCP","UDP")

OK
AT+CIPOPEN?
+CIPOPEN: 0
+CIPOPEN: 1,"TCP","183.230.174.137",6031,-1
```

```
OK
AT+CIPOPEN=0,"TCP","183.230.174.137",6031
OK                                              //TCP connection

+CIPOPEN: 0,0
AT+CIPOPEN=1,"UDP",,,6031
+CIPOPEN: 1,0                                   // UDP Connection

OK
```

### 2.1.4 AT+CIPSEND Send data through TCP or UDP Connection

AT+CIPSEND is used to send data to remote side. If service type is TCP, the data is firstly sent to the module’s internal TCP/IP stack, and then sent to server by protocol stack. The <length> field may be empty. While it is empty, each <Ctrl+Z> character present in the data should be coded as <ETX><Ctrl+Z>. Each <ESC> character present in the data should be coded as <ETX><ESC>. Each <ETX> character will be coded as <ETX><ETX>. Single <Ctrl+Z> means end of the input data. Single <ESC> is used to cancel the sending.
<ETX> is 0x03, and <Ctrl+Z> is 0x1A,<ESC> is 0x1B.

| **AT+CIPSEND Send data through TCP or UDP Connection** | |
|---|---|
| Test Command<br>**AT+CIPSEND=?** | Response<br>**+CIPSEND: (0-1),(1-1500)**<br><br>**OK** |
| Write Command<br>If service type is "TCP", send data with changeable length<br>**AT+CIPSEND=<link_num>**<br><br>Response ">", then type data to send, tap CTRL+Z to send data, tap ESC to cancel the operation | Response<br>1)If the connection identified by <link_num> has been established successfully, response:<br>**>**<br>**<input data>**<br>**CTRL+Z**<br>**OK**<br><br>**+CIPSEND: <link_num>,<reqSendLength>,<cnfSendLength>**<br>2)If <reqSendLength> is equal <cnfSendLength>, it means that the data has been sent to TCP/IP protocol stack successfully.<br>3)If the connection has not been established, abnormally closed, or parameter is incorrect, response:<br>**+CIPERROR: <err>**<br><br>**ERROR**<br>4)Others: |

| | |
|---|---|
| | **ERROR** |
| Write Command<br>If service type is "TCP", send data with fixed length<br>**AT+CIPSEND=<link_num>,<length>** | Response<br>1)If the connection identified by <link_num> has been established successfully, response:<br>**>**<br>**<input data with specified length>**<br>**OK**<br><br>**+CIPSEND: <link_num>,<reqSendLength>,<cnfSendLength>**<br>2)If <reqSendLength> is equal <cnfSendLength>, it means that the data has been sent to TCP/IP protocol stack successfully.<br>3)If the connection has not been established, abnormally closed, or parameter is incorrect, response:<br>**+CIPERROR: <err>**<br><br>**ERROR**<br><br>4)Others:<br>**ERROR** |
| Write Command<br>If service type is "UDP", send data with changeable length<br><br>**AT+CIPSEND=<link_num>,,<serverIP>,<serverPort>**<br>Response ">", then type data to send, tap CTRL+Z to send data, tap ESC to cancel the operation | Response<br>1)If the connection identified by <link_num> has been established successfully, response:<br>**>**<br>**<input data>**<br>**CTRL+Z**<br>**OK**<br><br>**+CIPSEND: <link_num>,<reqSendLength>,<cnfSendLength>**<br><br>2)If the connection has not been established, abnormally closed, or parameter is incorrect, response:<br>**+CIPERROR: <err>**<br><br>**ERROR**<br>3)Others:<br>**ERROR** |
| Write Command<br>If service type is "UDP", send data with fixed length<br>**AT+CIPSEND=<link_num>,<length>,<serverIP>,<serverPort>**<br>Response ">", type data until the data length is equal to <length> | Response<br>1)If the connection identified by <link_num> has been established successfully, response:<br>**>**<br>**<input data with specified length>**<br>**OK**<br><br>**+CIPSEND: <link_num>,<reqSendLength>,<cnfSendLength>**<br>2)If the connection has not been established, abnormally closed, |

| | |
|---|---|
| | or parameter is incorrect, response:<br>**+CIPERROR: <err>**<br><br>**ERROR**<br>3)Others:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | Range: 3000ms-120000ms<br>default: 120000ms<br>(it can be set by AT+CIPTIMEOUT) |
| Reference | |

## Defined Values

| | |
|---|---|
| **<link_num>** | Integer type, identifies a connection. Range is 0-1. |
| **<length>** | Integer type, indicates the length of sending data, range is 1-1500. |
| **<serverIP>** | String type, identifies the IP address of server. The IP address format consists of 4 octets, separated by decimal point, like "AAA.BBB.CCC.DDD". Also the domain name is supported here. |
| **<serverPort>** | Integer type, identifies the port of TCP server, range is 0-65535.<br>NOTE:<br>When open port as TCP, the port must be the opened TCP port;<br>When open port as UDP, the port may be any port.<br>But, for Qualcomm, connecting the port 0 is regarded as an invalid operation. |
| **<reqSendLength>** | Integer type, the length of the data requested to be sent |
| **<cnfSendLength>** | Integer type, the length of the data confirmed to have been sent<br>-1 the connection is disconnected.<br>0 own send buffer or other side’s congestion window are full.<br>Note: If the <cnfSendLength> is not equal to the <reqSendLength>, the socket then cannot be used further. |
| **<err>** | Integer type, the result of operation.<br>0 is success, other value is failure, please refer to Chapter 9.3.2 for details |

## Examples

```
AT+CIPSEND=?
+CIPSEND: (0-1),(1-1500)

OK
AT+CIPSEND=1,5                          // If service type is "TCP", send data with
>12345                                  fixed length
```

```
OK

+CIPSEND: 1,5,5
AT+CIPSEND=1,5,"183.230.174.137",6031
>12345                                           // If service type is "UDP", send data with
OK                                               fixed length

+CIPSEND: 1,5,5
```

**NOTE**

If you use UDP to send more than 1400 bytes of data when the server does not receive data, this may be the reason for the carrier, in this case please send no more than 1400 bytes of data.

If you use TCP to send data, the instruction can be followed by a comma just like "AT+CIPSEND=0," or "AT+CIPSEND=0,10," without an error, but it doesn't make any sense

### 2.1.5 AT+CIPRXGET Set the Mode to Retrieve Data

If set <mode> to 1, after receiving data, the module will buffer it and report a URC as "+CIPRXGET: 1,<link_num>" to notify the host. Then host can retrieve data by AT+CIPRXGET.
If set <mode> to 0, the received data will be outputted to COM port directly by URC as "RECV FROM:<IP ADDRESS>:<PORT><CR><LF>+IPD(data length)<CR><LF><data>".
The default value of <mode> is 0.

| **AT+CIPRXGET Set the Mode to Retrieve Data** | |
|---|---|
| Test Command<br>AT+CIPRXGET=? | Response<br>**+CIPRXGET: (0-4),(0-1),(1-1500)**<br><br>**OK** |
| Read Command<br>AT+CIPRXGET? | Response<br>**+CIPRXGET: <mode>**<br><br>**OK** |
| Write Command<br>AT+CIPRXGET=<mode><br>In this case,<mode> can only be 0 or 1 | Response<br>1)If the parameter is correct, response:<br>**OK**<br>2)If the parameter is incorrect, response:<br>**+IP ERROR: <err_info>** |

| | |
|---|---|
| | **ERROR**<br><br>3)Others:<br>**ERROR** |
| Write Command<br>**AT+CIPRXGET=2,<link_num>[,<len>]**<br>Retrieve data in ACSII form | 1)If <len> field is empty, the default value to read is 1500.<br>If the buffer is not empty, response:<br>**+CIPRXGET: <mode>,<link_num>,<read_len>,<rest_len>**<br>**<data>ACSII form**<br><br>**OK**<br>2)If the buffer is empty, response:<br>**+IP ERROR: No data**<br><br>**ERROR**<br>3)If the parameter is incorrect, response:<br>**+IP ERROR: <err_info>**<br><br>**ERROR**<br>4)Others:<br>**ERROR** |
| Write Command<br>**AT+CIPRXGET=3,<link_num>[,<len>]**<br>Retrieve data in hex form | Response<br>1)If <length> field is empty, the default value to read is 750.<br>If the buffer is not empty, response:<br>**+CIPRXGET: <mode>,<link_num>,<read_len>,<rest_len>**<br>**<data>**<br>hex form<br><br>**OK**<br>2)If the buffer is empty, response:<br>**+IP ERROR: No data**<br><br>**ERROR**<br>3)If the parameter is incorrect, response:<br>**+IP ERROR: <err_info>**<br><br>**ERROR**<br>4)Others:<br>**ERROR** |
| Write Command<br>**AT+CIPRXGET=4,<link_num>** | Response<br>1)If the parameter is correct, response:<br>**+CIPRXGET: 4,<link_num>,<rest_len>**<br><br>**OK**<br><br>2)If the parameter is incorrect, response: |

| | |
|---|---|
| | **+IP ERROR: <err_info>**<br><br>**ERROR**<br>3)Others<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | 8s |
| Reference | |

## Defined Values

| | |
|---|---|
| **<mode>** | Integer type, sets the mode to retrieve data<br>0 set the way to get the network data automatically<br>1 set the way to get the network data manually<br>2 read data, the max read length is 1500<br>3 read data in HEX form, the max read length is 750<br>4 get the rest data length |
| **<link_num>** | Integer type, identifies a connection. Range is 0-1. |
| **<len>** | Integer type, the data length to be read.<br>Not required, the default value is 1500 when <mode>=2, and 750 when <mode>=3. |
| **<read_len>** | Integer type, the length of data that has been read. |
| **<rest_len>** | Integer type, the length of data which has not been read in the buffer. |
| **<err_info>** | String type, displays the cause of occurring error, please refer to Chapter 15.3.1 for more details. |

## Examples

```
AT+CIPRXGET=?
+CIPRXGET: (0-4),(0-1),(1-1500)

OK
AT+CIPRXGET?
+CIPRXGET: 1

OK
AT+CIPRXGET=1
OK
AT+CIPRXGET=2,0
+CIPRXGET: 2,0,6,0
123456

OK
```

```
AT+CIPRXGET=3,0
+CIPRXGET: 3,0,6,0
313233343536

OK
AT+CIPRXGET=4,0
+CIPRXGET: 4,0,18

OK
```

NOTE

When data is received and reported, the maximum length of <data length> is 1500 each time.

### 2.1.6 AT+CIPCLOSE Close TCP or UDP Socket

AT+CIPCLOSE is used to close a TCP or UDP Socket

| AT+CIPCLOSE Close TCP or UDP Socket | |
|---|---|
| Test Command<br>**AT+CIPCLOSE=?** | Response<br>**+CIPCLOSE: (0-1)**<br><br>**OK** |
| Read Command<br>**AT+CIPCLOSE?** | Response<br>**+CIPCLOSE: <link0_state>,<link1_state>**<br><br>**OK** |
| Write Command<br>**AT+CIPCLOSE=<link_num>** | Response<br>1)If service type is TCP and the connection identified by <link_num> has been established, response<br>**OK**<br><br>**+CIPCLOSE: <link_num>,<err>**<br>2)If service type is TCP and the access mode is transparent mode, response:<br>**OK**<br><br>**CLOSED**<br><br>**+CIPCLOSE: <link_num>,<err>** |

| | |
|---|---|
| | 3)If service type is UDP and the connection identified by <link_num> has been established and closed successfully, response:<br>**+CIPCLOSE: <link_num>,0**<br><br>**OK**<br>4)If service type is UDP and access mode is transparent mode, response:<br>**CLOSED**<br><br>**+CIPCLOSE: <link_num>,<err>**<br><br>**OK**<br>5)If the connection has not been established, abnormally closed, or parameter is incorrect, response:<br>**+CIPCLOSE: <link_num>,<err>**<br><br>**ERROR**<br>6)Others:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | Range: 3000ms-120000ms<br>default: 120000ms<br>(it can be set by AT+CIPTIMEOUT) |
| Reference | |

## Defined Values

| | |
|---|---|
| **<link_num>** | Integer type, identifies a connection. Range is 0-1. |
| **<linkX_state>** | Integer type, indicates state of connection identified by <link_num>. Range is 0-1.<br>0 disconnected<br>1 connected |
| **<err>** | Integer type, the result of operation.<br>0 is success, other value is failure, please refer to Chapter 5.3.2 for details |

## Examples

```
AT+CIPCLOSE=?
+CIPCLOSE: (0-1)

OK
```

```
AT+CIPCLOSE?
+CIPCLOSE: 0,0

OK
AT+CIPCLOSE=0
OK

+CIPCLOSE: 0,0
```

### 2.1.7 AT+IPADDR Inquire Socket PDP address

AT+IPADDR is used to get active PDP address.

| AT+IPADDR Inquire Socket PDP Address | |
|---|---|
| Test Command<br>AT+IPADDR=? | Response<br>**OK** |
| Execute Command<br>AT+IPADDR | Response<br>1)If PDP context has been activated successfully, response<br>**+IPADDR: <ip_address>**<br><br>**OK**<br>2)<br>**+IP ERROR: Network not opened**<br><br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<ip_address>** | String type, identifies the IP address of current active socket PDP. |

### Examples

```
AT+IPADDR
+IPADDR: 10.84.17.161

OK
```

## 2.1.8 AT+CIPHEAD Add an IP Header When Receiving Data

AT+CIPHEAD is used to add an IP header when receiving data.

| AT+CIPHEAD Add an IP Header When Receiving Data | |
|---|---|
| Test Command<br>**AT+CIPHEAD=?** | Response<br>**+CIPHEAD: (0-1)**<br><br>**OK** |
| Read Command<br>**AT+CIPHEAD?** | Response<br>**+CIPHEAD: <mode>**<br><br>**OK** |
| Write Command<br>**AT+CIPHEAD=<mode>** | Response<br>1)If the parameter is correct, response:<br>**OK**<br>2)<br>**ERROR** |
| Execute Command<br>**AT+CIPHEAD** | Response<br>Set default value:(<mode>=1)<br>**OK** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<mode>** | Integer type, indicates whether adding an IP header or not when receiving data<br>0 not add IP header<br>1 add IP header, the format is "+IPD(data length)" |

### Examples

```
AT+CIPHEAD=?
+CIPHEAD: (0-1)

OK
AT+CIPHEAD?
```

```
+CIPHEAD: 1

OK
AT+CIPHEAD=1
OK
AT+CIPHEAD
OK
```

### 2.1.9 AT+CIPSRIP Show Remote IP Address and Port

AT+CIPSRIP is used to set whether to display IP address and port of server when receiving data.

| AT+CIPSRIP Show Remote IP Address and Port | |
|---|---|
| Test Command<br>**AT+CIPSRIP=?** | Response<br>**+CIPSRIP: (0-1)**<br><br>**OK** |
| Read Command<br>**AT+CIPSRIP?** | Response<br>**+CIPSRIP: <mode>**<br><br>**OK** |
| Write Command<br>**AT+CIPSRIP=<mode>** | Response<br>1)If the parameter is correct, response:<br>**OK**<br>2)<br>**ERROR** |
| Execute Command<br>**AT+CIPSRIP** | Response<br>Set default value:(<mode>=1)<br>**OK** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<mode>** | Integer type, indicates whether to show IP address and port of server or not when receiving data.<br>0 not show<br>1 show, the format is as follows:<br>"RECV FROM:<IP ADDRESS>:<PORT>" |

### Examples

```
AT+CIPSRIP=?
+CIPSRIP: (0-1)

OK
AT+CIPSRIP?
+CIPSRIP: 1

OK
AT+CIPSRIP=0
OK
AT+CIPSRIP
OK
```

### 2.1.10 AT+CIPMODE Set TCP/IP Application Mode

AT+CIPMODE is used to select transparent mode(data mode) or non-transparent mode(command mode).The default mode is non-transparent mode.

| AT+CIPMODE Set TCP/IP Application Mode | |
|---|---|
| Test Command<br>AT+CIPMODE=? | Response<br>**+CIPMODE: (0-1)**<br>**OK** |
| Read Command<br>AT+CIPMODE? | Response<br>**+CIPMODE: <mode>**<br>**OK** |
| Write Command<br>AT+CIPMODE=<mode> | Response<br>1)If the parameter is correct, response:<br>**OK**<br>2)<br>**ERROR** |
| Execute Command<br>AT+CIPMODE | Response<br>Set default value:(<mode>=0)<br>**OK** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |

| | |
|---|---|
| Reference | - |

### Defined Values

| | |
|---|---|
| **<mode>** | Integer type, sets TCP/IP application mode<br>0 Non transparent mode<br>1 Transparent mode |

### Examples

```
AT+CIPMODE=?
+CIPMODE: (0-1)

OK
AT+CIPMODE?
+CIPMODE: 0

OK
AT+CIPMODE=1
OK
AT+CIPMODE
OK
```

**NOTE**

When you want to use transparent mode to transmit data, you should set AT+CIPMODE=1 before AT+NETOPEN.

### 2.1.11 AT+CIPSENDMODE Set Sending Mode

AT+CIPSENDMODE is used to select sending mode when service type is "TCP".
If set <mode> to 1, when sending data by AT+CIPSEND, the URC "+CIPSEND: <link_num>,<reqSendLength>,<cnfSendLength>" will not be returned until module receives the server's ACK message to the sent data last time.
If set <mode> to 0, the URC "+CIPSEND: <link_num>,<reqSendLength>,<cnfSendLength>" will be returned If the data has been sent to module's internal TCP/IP protocol stack. In this case, the module doesn't need to wait for the server's ACK message.
The default mode is sending without waiting peer TCP ACK mode.

| AT+CIPSENDMODE Set Sending Mode | |
|---|---|
| Test Command<br>**AT+CIPSENDMODE=?** | Response<br>**+CIPSENDMODE: (0-1)**<br><br>**OK** |
| Read Command<br>**AT+CIPSENDMODE?** | Response<br>**+CIPSENDMODE: <mode>**<br><br>**OK** |
| Write Command<br>**AT+CIPSENDMODE=<mode>** | Response<br>1)If the parameter is correct, response:<br>**OK**<br>2)<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<mode>** | Integer type, sets sending mode<br>0 sending without waiting peer TCP ACK mode<br>1 sending wait peer TCP ACK mode |

### Examples

```
AT+CIPSENDMODE=?
+CIPSENDMODE: (0-1)

OK
AT+CIPSENDMODE=1
OK
AT+CIPSENDMODE?
+CIPSENDMODE: 1

OK
```

### 2.1.12 AT+CIPTIMEOUT Set TCP/IP Timeout Value

AT+CIPTIMEOUT is used to set timeout value for AT+NETOPEN/AT+CIPOPEN/AT+CIPSEND.

| AT+CIPTIMEOUT Set TCP/IP Timeout Value | |
|---|---|
| Read Command<br>**AT+CIPTIMEOUT?** | Response<br>**+CIPTIMEOUT: <netopen_timeout>,<cipopen_timeout>,<cipsend_timeout>**<br><br>**OK** |
| Write Command<br>**AT+CIPTIMEOUT=[<netopen_timeout>][,[<cipopen_timeout>][,[<cipsend_timeout>]]]** | Response<br>1)If the parameter is correct, response:<br>**OK**<br>2)<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<netopen_timeout>** | Integer type, timeout value for AT+NETOPEN.<br>default is120000ms. Range is 3000ms-120000ms. |
| **<cipopen_timeout>** | Integer type, timeout value for AT+CIPOPEN.<br>default is120000ms. Range is 3000ms-120000ms. |
| **<cipsend_timeout>** | Integer type, timeout value for AT+CIPSEND.<br>default is120000ms. Range is 3000ms-120000ms. |

### Examples

```
AT+CIPTIMEOUT?
+CIPTIMEOUT: 120000,120000,120000

OK
AT+CIPTIMEOUT=3000,3000,3000
OK
```

### 2.1.13 AT+CIPCCFG Configure Parameters of Socket

AT+CIPCCFG is used to configure parameters of socket.

| AT+CIPCCFG Configure Parameters of Socket | |
|---|---|
| Test Command<br>AT+CIPCCFG=? | Response<br>**+CIPCCFG: (0-10),(0-1000),(0),(0-1),(0-1),(0-1),(500-120000)**<br><br>**OK** |
| Read Command<br>AT+CIPCCFG? | Response<br>**+CIPCCFG: <NmRetry>,<DelayTm>,<Ack>,<errMode>,<Header-Type>,<AsyncMode>,<TimeoutVal>**<br><br>**OK** |
| Write Command<br>AT+CIPCCFG=[<NmRetry>][,[<DelayTm>][,[<Ack>][,[<errMode>][,]<HeaderType>][,[[<AsyncMode>][,[<TimeoutVal>]]]]]]] | Response<br>1)If the parameter is correct, response:<br>**OK**<br>2)<br>**ERROR** |
| Execute Command<br>AT+CIPCCFG | Response<br>Set default value:<br>**OK** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<NmRetry>** | Integer type, number of retransmission to be made for an IP packet. Range is 0-10. The default value is 10. |
| **<DelayTm>** | Integer type, number of milliseconds to delay to output data of Receiving. Range is 0-1000. The default value is 0. |
| **<Ack>** | Integer type, it can only be set to 0. It's used to be compatible with old TCP/IP command set. |
| **<errMode>** | Integer type, sets mode of reporting <err_info>, default value is 1.<br>0 error result code with numeric values<br>1 error result code with string values |
| **<HeaderType>** | Integer type, select which data header is used when receiving data, it only takes effect in multi-client mode. Default value is 0.<br>0 add data header, the format is "+IPD<data length>"<br>1 add data header, the format is "+RECEIVE,<link num>,<data length>" |

| | |
|---|---|
| **<AsyncMode>** | Integer type, range is 0-1. Default value is 0.<br>It's used to be compatible with old TCP/IP command set. |
| **<TimeoutVal>** | Integer type, set the minimum retransmission timeout value for TCP connection. Range is 500ms-120000ms. Default is 500ms. |

### Examples

```
AT+CIPCCFG=?
+CIPCCFG: (0-10),(0-1000),(0),(0-1),(0-1),(0-1),(500-120000)

OK
AT+CIPCCFG?
+CIPCCFG: 10,0,0,1,0,0,500

OK
AT+CIPCCFG=2
OK
AT+CIPCCFG
OK
```

### 2.1.14 AT+SERVERSTART Startup TCP Sever

AT+SERVERSTART is used to startup a TCP server, and the server can receive the request of TCP client. After the command executes successfully, an unsolicited result code is returned when a client tries to connect with module and module accepts request. The unsolicited result code is+CLIENT: <link_num>,<server_index>,<client_IP>:<port>.

| **AT+SERVERSTART Startup TCP Sever** | |
|---|---|
| Test Command<br>**AT+SERVERSTART=?** | Response<br>**+SERVERSTART: (0-65535),(0-1)**<br><br>**OK** |
| Read Command<br>**AT+SERVERSTART?** | Response<br>1)If the PDP context has not been activated successfully, response:<br>**+CIPERROR: <err>**<br><br>**ERROR**<br><br>2)If there exists opened server, response:<br>**[+SERVERSTART: <server_index>,<port>** |

| | |
|---|---|
| | **…]**<br><br>**OK**<br>3)Others:<br>**ERROR** |
| Write Command<br>**AT+SERVERSTART=<port>,<server_index>[,<backlog>]** | Response<br>1)If there is no error, response:<br>**OK**<br>2)If the PDP context has not been activated, or the server identified by <server_index> has been opened, or the parameter is not correct, or other errors, response:<br>**+CIPERROR: <err>**<br><br>**ERROR**<br>3)Others:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<port>** | Integer type, identifies the listening port of module when used as a TCP server. Range is 0-65535. |
| **<server_index>** | Integer type, the TCP server index, range is 0-1. |
| **<backlog>** | Integer type, the maximum connections can be queued in listening queue. Range is 1-3. Default is 3. |

### Examples

```
AT+SERVERSTART=?
+SERVERSTART: (0-65535),(0-1)

OK
AT+SERVERSTART?
OK
AT+SERVERSTART=8080,0
OK
```

### 2.1.15 AT+SERVERSTOP Stop TCP Sever

AT+SERVERSTOP is used to stop TCP server. Before stopping a TCP server, all sockets <server_index> of which equals to the closing TCP server index must be closed first.

| **AT+SERVERSTOP Stop TCP Sever** | |
|---|---|
| Write Command<br>**AT+SERVERSTOP=<server_index>** | Response<br>1)If there exists open connection with the server identified by <server_index>, or the server identified by <server_index> has not been opened, or the parameter is incorrect, response:<br>**+SERVERSTOP: <server_index>,<err>**<br><br>**ERROR**<br>2)If the server socket is closed immediately, response:<br>**+SERVERSTOP: <server_index>,0**<br><br>**OK**<br>(In general, the result is shown as below.)<br>3)If the server socket starts to close, response:<br>**OK**<br><br>**+SERVERSTOP: <server_index>,<err>**<br>4)Others:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<server_index>** | Integer type, the TCP server index, range is 0-1. |
| **<err>** | Integer type, the result of operation.<br>0 is success, other value is failure, please refer to Chapter 9.3.2 for details |

### Examples

```
AT+SERVERSTOP=0
OK

+SERVERSTOP: 0,0
```

### 2.1.16 AT+CIPACK Query TCP Connection Data Transmitting Status

AT+CIPACK is used to query TCP connection data transmitting status.

| AT+CIPACK | Query Connection Data Transmitting State |
|---|---|
| Test Command<br>AT+CIPACK=? | Response<br>**+CIPACK: (range of supported <link_num>s)**<br><br>**OK** |
| Write Command<br>AT+CIPACK=<link_num> | Response<br>1)If the PDP context has not been activated, or the connection identified by <link_num> has not been established, abnormally closed, or the parameter is incorrect, or other errors, response:<br>**+IP ERROR: <err_info>**<br><br>**ERROR**<br>2)If the connection has been established, and the service type is "TCP", response:<br>**+CIPACK: <sent_data_size>,<ack_data_size>,<recv_data_size>**<br><br>**OK** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<link_num>** | Integer type, identifies a connection. Range is 0-1. |
| **<sent_data_size>** | Integer type, the total length of sent data |
| **<ack_data_size>** | reserve |
| **<recv_data_size>** | Integer type, the total length of received data |
| **<err>** | Integer type, the result of operation.<br>0 is success, other value is failure, please refer to Chapter 9.3.2 for details |
| **<err_info>** | String type, displays the cause of occurring error, please refer to Chapter 3 for details. |

### Examples

```
AT+CIPACK=?
+CIPACK: (0-1)
```

```
OK
AT+CIPACK=0
+CIPACK: 10,0,5

OK
```

### 2.1.17 AT+CDNSGIP Query the IP Address of Given Domain Name

AT+CDNSGIP is used to query the IP address of given domain name.

| AT+CDNSGIP Query the IP Address of Given Domain Name | |
|---|---|
| Test Command<br>**AT+CDNSGIP=?** | Response<br>**OK** |
| Write Command<br>**AT+CDNSGIP=<domain name>** | Response<br>1)If the given domain name has related IP, response:<br>**+CDNSGIP: 1,<domain name>,<IP address>**<br><br>**OK**<br>2)If the given name has no related IP, response:<br>**+CDNSGIP: 0,<dns error code>**<br><br>**ERROR**<br>3)Others:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Max Response Time | default: 6s |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<domain name>** | String type (string should be included in quotation marks), indicates the domain name. The maximum length of domain name is 254.<br>Valid characters allowed in the domain name area include a-z, A-Z, 0-9, "-" (hyphen)and ".". A domain name is made up of one label name or more label names separated by "." (eg: AT+CDNSGIP="aa.bb.cc").<br>For label names separated by ".", length of each label must be no more than 63 characters. The beginning character of the domain name and of labels should be an alphanumeric character. |
| **<IP address>** | String type, indicates the IP address corresponding to the domain name. |
| **<dns error code>** | Integer type, indicates the error code. |

| | |
|---|---|
| | 10 DNS GENERAL ERROR |

### Examples

```
AT+CDNSGIP=?
OK
AT+CDNSGIP="www.baidu.com"
+CDNSGIP: 1,"www.baidu.com","61.135.169.121"

OK
```

## 2.1.18 AT+CSOCKSETPN Set active PDP context's profile

This command sets default active PDP context's profile number and type. When we activate PDP by using AT+NETOPEN command, we need use the default profile number and type.,and the context of this profile is set by AT+CGDCONT command.

| **AT+CSOCKSETPN Set acitve PDP context's profile** | |
|---|---|
| Test Command<br>**AT+CSOCKSETPN=?** | Response<br>**+CSOCKSETPN: 1,(1,6)**<br><br>**OK** |
| Read Command<br>**AT+CSOCKSETPN?** | Response<br>**+CSOCKSETPN: <profile_num>,<ip_family>**<br><br>**OK** |
| Write Command<br>**AT+CSOCKSETPN=<profile_num>[,<ip_family>]** | Response<br>1)If the parameter is correct, response:<br>**OK**<br>2)If the parameter is wrong,or NETOPEN is already active, response:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Maximum Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<profile_num>** | Packet Data Protocol context's profile number.<br>Now only 1 is supported for this parameter value. |
| **<ip_family>** | Packet Data Protocol type<br>1 IPV4<br>6 IPV6 |

### Examples

```
AT+CSOCKSETPN=?
+CSOCKSETPN: 1,(1,6)

OK
AT+CSOCKSETPN?
+CSOCKSETPN: 1,1

OK
AT+CSOCKSETPN=1,6
OK
```

### 2.1.19 AT+CTCPKA Conigure TCP heartbeat

This command is used to set TCP heartbeat parameters. Set this up after we activate PDP by using AT+NETOPEN command.

| **AT+CTCPKA Conigure TCP heartbeat** | |
|---|---|
| Test Command<br>**AT+CTCPKA=?** | Response<br>**OK** |
| Read Command<br>**AT+CTCPKA?** | Response<br>**+CTCPKA: <keepalive>,<keepidle>,<keepcount>,<keepinterval>**<br><br>**OK** |
| Write Command<br>**AT+CTCPKA=<keepalive>,<keepidle>,<keepcount>,<keepinterval>** | Response<br>1)If successfully:<br>**OK**<br>2)If failed:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |
| Maximum Response Time | default: 9000ms |
| Reference | - |

## Defined Values

| | |
|---|---|
| **<keepalive >** | Set TCP keepalive option.<br>0 Disable TCP keep alive mechanism<br>1 Enable TCP keep alive mechanism |
| **<keepidle>** | The unit is minute. If there is no data interaction within this period, the probe is performed. (1-120) |
| **<keepcount>** | Number of probe retries. If all times out, the connection is considered Invalid.(1-10) |
| **<keepinterval>** | The unit is minute. Interval for sending probe packets during probe. |

## Examples

```
AT+CTCPKA=1,2,5,1
OK
AT+CTCPKA?
+CTCPKA: 1,2,5,1

OK
```

### 2.1.20 AT+CDNSCFG Configure Domain Name Server

This command is used to configure Domain Name Server.

| **AT+CDNSCFG Configure Domain Name Server** | |
|---|---|
| Test Command<br>**AT+CDNSCFG =?** | Response<br>**+CDNSCFG: ("Primary DNS"),("Secondary DNS"),type**<br><br>**OK** |
| Read Command<br>**AT+CDNSCFG?** | Response<br>**Primary IPv4 DNS: <pri_dns>,Secondary IPv4 DNS: <pri_dns>**<br>**Primary IPv6 DNS: <pri_dns>,Secondary IPv6 DNS: <pri_dns>**<br><br>**OK** |
| Write Command<br>**AT+CDNSCFG=<pri_dns>[,<sec_dns>][,<type>]** | Response<br>1)If successfully:<br>**OK**<br>2)If failed:<br>**ERROR** |
| Parameter Saving Mode | NO_SAVE |

| | |
|---|---|
| Maximum Response Time | default: 9000ms |
| Reference | - |

### Defined Values

| | |
|---|---|
| **<pri_dns>** | A string parameter which indicates the IP address of the primary domain name server. |
| **<sec_dns>** | A string parameter which indicates the IP address of the secondary domain name server. |
| **<type>** | 0 Set the server for the ipv4 network<br>1 Set the server for the ipv6 network |

### Examples

```
AT+CDNSCFG?
Primary IPv4 DNS: 183.230.126.224,Secondary IPv4
DNS: 183.230.126.225
Primary IPv6 DNS: 2409:8060:20EA:101::1,Secondary
IPv6 DNS: 2409:8060:20EA:201::1

OK
AT+CDNSCFG=183.230.126.224,183.230.126.225,0
OK
```

## 2.2 Description of URC

**Table 1:Description of URC**

| URC | Description |
|---|---|
| **+CIPEVENT: NETWORK CLOSED UNEXPECTEDLY** | Network is closed for network error(Out of service, etc). When this event happens, user's application needs to check and close all opened sockets, and then uses AT+NETCLOSE to release the network library if AT+NETOPEN? shows the network library is still opened. |
| **+IPCLOSE: <client_index>,<close_reason>** | Socket is closed passively.<br><client_index> is the link number.<br><close_reason>:<br>0 Closed by local, active<br>1 Closed by remote, passive |

| | |
|---|---|
| | 2 Closed for sending timeout or DTR off |
| **+CLIENT: <link_num>,<server_index>,<client_IP>:<port>** | TCP server accepted a new socket client, the index is<link_num>, the TCP server index is <server_index>. The peer IP address is <client_IP>, the peer port is <port>. |

# 3 Examples

## 3.1 Configure and Activate context

### 3.1.1 Network Environment

TCP/IP application is based on NB network. Please make sure that NB network is available before TCP/IP setup.

```
AT+CSQ
+CSQ: 23,0

OK
AT+CEREG?
+CEREG: 0,1

OK
```

### 3.1.2 Configure Context

**If based on ipv4**

```
AT+CGDCONT=1,"IP","CMNET"
OK
```

**If based on ipv6**

```
AT+CGDCONT=1,"IPV6","CMNET"
OK                                  //The CGDCONT IP_TYPE is set to IPV6 instead
                                    of IP
```

### 3.1.3 Activate context

**If based on ipv4**

```
AT+NETOPEN
OK

+NETOPEN: 0
AT+IPADDR
+IPADDR: 10.148.0.17

OK
```

**If based on ipv6**

```
AT+CSOCKSETPN=1,6
OK
AT+NETOPEN
OK

+NETOPEN: 0
AT+IPADDR
+IPADDR: 2409:8960:1e64:94d8:1:0:3b3b:7118    //The queried IP address is an ipv6 address
OK
```

Other commands are used in the same way based on IPV4 or IPV6.

### 3.1.4 Deactivate Context

```
AT+NETCLOSE
OK

+NETCLOSE: 0
AT+IPADDR
+IP ERROR: Network not opened

ERROR
```

## 3.2 TCP Client

### 3.2.1 TCP Client Works in Direct Push Mode

```
//Set up TCP Client Connection
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPOPEN=1,"TCP","117.131.85.139",5253     // set up a TCP connection, <link_num> is 1.
OK                                           Before using AT+CIPOPEN, host should activate
                                             PDP Context with AT+NETOPEN first.
+CIPOPEN: 1,0
```

```
//Send Data To Server
AT+CIPSEND=1,5                               // send data with fixed length
>HELLO
OK

+CIPSEND: 1,5,5
```

```
//Receive Data From Server
RECV FROM:117.131.85.139:5253                // data from server directly output to COM
+IPD16

data from server
```

```
//Close TCP Connection
AT+CIPCLOSE=1
OK

+CIPCLOSE: 1,0
```

### 3.2.2 TCP Client Works in Buffer Access Mode

```
//Set up TCP Client Connection
```

```
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPRXGET=1                         // buffer access mode, get data by AT+CIPRXGET
OK
AT+CIPOPEN=1,"TCP","117.131.85.139",5253
OK

+CIPOPEN: 1,0
```

```
//Send Data to Server
AT+CIPSEND=1,5                        // send data with fixed length
>hello
OK

+CIPSEND: 1,5,5
```

```
//Receive Data from Server
+CIPRXGET: 1,1                        // URC to notify host of data from server
AT+CIPRXGET=4,1                       // query the length of data in the buffer of socket
+CIPRXGET: 4,1,16                     with
                                      // <link_num>=1
OK
AT+CIPRXGET=2,1,5                     // get data in ASCII form
+CIPRXGET: 2,1,5,11                   // read 5 bytes data and left 11 bytes
Data1

OK
AT+CIPRXGET=3,1,5                     // get data in hex form
+CIPRXGET: 3,1,5,6
66726F6D20

OK
AT+CIPRXGET=4,1                       // read the length of unread data in buffer
+CIPRXGET: 4,1,6

OK
AT+CIPRXGET=2,2                       // the connection identified by link_num=2 has not
+IP ERROR: No data                    been established

ERROR
AT+CIPRXGET=2,1
```

```
+CIPRXGET: 2,1,6,0
server

OK
AT+CIPRXGET=4,1                          // all the data in buffer has been read, the rest_len
+CIPRXGET: 4,1,0                         is 0.

OK
```

```
//Close TCP Connection
AT+CIPCLOSE=1
OK

+CIPCLOSE: 1,0
```

### 3.2.3 TCP Client Works in Transparent Access Mode

```
//Set up TCP Client Connection
AT+CIPMODE=1                             // Enter into transparent mode by at+cipmode=1
OK
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPOEPN=0,"TCP","117.131.85.139",5253  // only <link_num>=0 is allowed to operate with
CONNECT 115200                           transparent mode.
```

```
//Send Data to Server
All data got from com port will be sent to
remote directly
```

```
//Receive Data From Server
DATA FROM SERVERDATA FROM SERVER
OK                                       //all the received data from server will be output to
                                         com port directly
                                         //sequence of +++ to quit transparent mode
AT+CIPOPEN?
+CIPOPEN: 0,"TCP","117.131.85.139",5253,-1
+CIPOPEN: 1

OK
```

```
ATO                                         //ATO to enter transparent mode again
CONNECT 115200
HELLO CLIENT
OK
```

```
//Close TCP Connection
AT+CIPCLOSE=0                                                                     /
OK

CLOSED

+CIPCLOSE: 0,0
```

## 3.3 UDP Client

### 3.3.1 UDP Client Works in Direct Push Mode

```
//Set up UDP Client Connection
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPOPEN=1,"UDP",,,5000                   // when set a UDP connection, the remote IP
+CIPOPEN: 1,0                               address and port is not necessary, but the local
                                            port
OK                                          must be specified.
```

```
//Send data to Server
AT+CIPSEND=1,,"117.131.85.139",5254         // for UDP connection, when sending data, user
>HELLOSERVER                                must specify the remote IP address and port
OK    <CTRL+Z>                              //send data with changeable length, <CTRL+Z> to
                                            end
+CIPSEND: 1,11,11
AT+CIPSEND=1,5,"117.131.85.139",5254        //send data with fixed length
>HELLO
OK
```

```
+CIPSEND: 1,5,5
```

```
//Receive Data From Server
RECV FROM:117.131.85.139:5254          //data from server output to COM port directly
+IPD14
HELLO CLIENT
```

```
//Close UDP Connection
AT+CIPCLOSE=1
+CIPCLOSE: 1,0

OK
```

### 3.3.2 UDP Client Works in Buffer Access Mode

```
//Set up UDP Client Connection
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPRXGET=1                          // buffer access mode, get data by AT+CIPRXGET
OK
AT+CIPOPEN=1,"UDP",,,5000              // when set a UDP connection, the remote IP
+CIPOPEN: 1,0                          address and port is not necessary, but the local
                                       port
OK                                     must be specified.
```

```
//Send Data to Server
AT+CIPSEND=1,,"117.131.85.139",5254    // for UDP connection, when sending data, user
>HELLOSERVER                           must specify the remote IP address and port
OK     <CTRL+Z>                        //send data with changeable length, <CTRL+Z> to
                                       end

+CIPSEND: 1,11,11
AT+CIPSEND=1,5,"117.131.85.139",5254   //send data with fixed length
>HELLO
OK

+CIPSEND: 1,5,5
```

```
//Receive Data From Server
+CIPRXGET: 1,1                  // URC to notify host of data from server
AT+CIPRXGET=4,1                 // query the length of data in the buffer of socket
+CIPRXGET: 4,1,16               with <link_num>=1

OK
AT+CIPRXGET=2,1,5               // get data in ASCII form
+CIPRXGET: 2,1,5,11
data

OK
AT+CIPRXGET=3,1,5               // get data in hex form
+CIPRXGET: 3,1,5,6
66726F6D20

OK
AT+CIPRXGET=4,1                 // read the length of unread data in buffer
+CIPRXGET: 4,1,6

OK
AT+CIPRXGET=2,2                 // the connection identified by link_num=2 has not
+IP ERROR: No data              been established

ERROR
AT+CIPRXGET=2,1
+CIPRXGET: 2,1,6,0
server

OK
AT+CIPRXGET=4,1                 // all the data in buffer has been read, the rest_len
+CIPRXGET: 4,1,0                is 0.

OK
```

```
//Close UDP Connection
AT+CIPCLOSE=1
OK

+CIPCLOSE: 1,0
```

### 3.3.3 UDP Client Works in Transparent Access Mode

```
//Set up UDP Client Connection
```

```
AT+CIPMODE=1
OK
AT+NETOPEN
OK

+NETOPEN: 0
AT+CIPOPEN=0,"UDP","117.131.85.139",5254,    //only <link_num>=0 is allowed to operate with
5000                                          transparent mode.
CONNECT 115200
```

```
//Send Data to Server
All data got from com port will be sent to
internet directly
```

```
//Receive Data From Server
HELLO CLIENT                                  //data from server output to COM port directly
HELLO CLIENT
OK                                            // sequence of +++ to quit transparent mode
AT+CIPOPEN?
+CIPOPEN: 0,"UDP","117.131.85.139",5254,-1
+CIPOPEN: 1

OK
AT+CIPOPEN=0,"UDP","117.131.85.139",5254,    //only <link_num>=0 is allowed to operate with
5000                                          transparent mode.

CONNECT 115200
```

## 3.4 TCP Server

### 3.4.1 Transparent Mode

```
AT+CIPMODE=1
OK
AT+NETOPEN
OK

+NETOPEN: 0
AT+SERVERSTART=8080, 0                        //only <server_index>=0 is allowed to operate with
OK                                            transparent mode.
+CLIENT: 0,0,192.168.108.5:57202              //only <link_num> 0 can be used for transparent
```

```
CONNECT 115200                              mode operation.

OK                                          // sequence of +++ to quit data mode
AT+CIPCLOSE=0                               // close client connection
OK

CLOSED
+CIPCLOSE: 0,0
AT+SERVERSTOP=0                             // close server socket
+SERVERSTOP: 0,0

OK
```

### 3.4.2 Non-Transparent Mode

```
AT+NETOPEN
OK

+NETOPEN: 0
AT+SERVERSTART=8080,0                       //only <server_index>=0 is allowed to operate with
OK                                          transparent mode.
AT+SERVERSTART=9090,1
OK
+CLIENT: 0,0,192.168.108.5:57202            //If a socket is accepted, the following URC will be
                                            reported:
AT+CIPOPEN?                                 //User can use AT+CIPOPEN? to check the
+CIPOPEN: 0,"TCP","192.168.108.5",57202,1   accepted socket
+CIPOPEN: 1                                 //last parameter of 1 indicates this is an accepted
                                            socket, this server index is 1
OK
AT+CIPSEND=0,5                              // only supports fixed-length to send
>HELLO
OK

+CIPSEND: 0,5,5
AT+SERVERSTOP=0                             // if unspecified, it will close 0 channel
+SERVERSTOP: 0,0
OK
AT+SERVERSTOP=1
+SERVERSTOP: 1,0
OK
```

```
AT+NETCLOSE
OK

+NETCLOSE: 0
```

### 3.4.3 Query Connection Status

```
AT+CIPOPEN=1,"TCP","117.131.85.139",5253
OK

+CIPOPEN: 1,0
AT+CIPOPEN?                                 // query the current state of all sockets
+CIPOPEN: 0
+CIPOPEN: 1,"TCP","117.131.85.139",5253,-1

OK
AT+CIPCLOSE?
+CIPCLOSE: 0,1

OK
AT+CIPCLOSE=1
OK

+CIPCLOSE: 1,0
AT+CIPCLOSE?
+CIPCLOSE: 0,0

OK
```

# 4 Error Handling

## 4.1 Executing TCPIP AT Commands Fails

When executing TCPIP AT commands, if ERROR response is received from the module, please check whether the U(SIM) card is inserted and whether it is +CPIN: READY returned when executing AT+CPIN?,also please check by AT+CGDCONT? If the 1st PDP context has correct APN.

## 4.2 PDP Activation Fails

If it fails to activate a PDP context with AT+NETOPEN command, please make sure the module has registered to network successfully, AT+CEREG? and AT+CPSI? could be used to verify registration state,also please check by AT+CGDCONT? If the 1st PDP context has correct APN.

# 5 Summary of Error Codes

## 5.1 Description of <err_info>

The fourth parameter <errMode> of AT+CIPCCFG (TODO)is used to determine how <err_info> is displayed.
If <errMode> is set to 0, the <err_info> is displayed with numeric value.
If <errMode>is set to 1, the <err_info> is displayed with string value.
The default is displayed with string value.

**Table 2:description of <err_info>**

| Numeric Value | String Value |
|---|---|
| 0 | Connection time out |
| 1 | Bind port failed |
| 2 | Port overflow |
| 3 | Create socket failed |
| 4 | Network is already opened |
| 5 | Network is already closed |
| 6 | No clients connected |
| 7 | No active client |
| 8 | Network not opened |
| 9 | Client index overflow |
| 10 | Connection is already created |
| 11 | Connection is not created |
| 12 | Invalid parameter |
| 13 | Operation not supported |
| 14 | DNS query failed |
| 15 | TCP busy |
| 16 | Net close failed for socket opened |
| 17 | Sending time out |
| 18 | Sending failure for network error |
| 19 | Open failure for network error |
| 20 | Server is already listening |
| 21 | Operation failed |
| 22 | No data |

## 5.2 Description of <err>

Table 3:description of <err>

| <err> | Description of <err> |
|---|---|
| 0 | operation succeeded |
| 1 | Network failure |
| 2 | Network not opened |
| 3 | Wrong parameter |
| 4 | Operation not supported |
| 5 | Failed to create socket |
| 6 | Failed to bind socket |
| 7 | TCP server is already listening |
| 8 | Busy |
| 9 | Sockets opened |
| 10 | Timeout |
| 11 | DNS parse failed for AT+CIPOPEN |
| 12 | Unknown error |

# 6 Appendix A Reference

## 6.1 Related documents

Table 4:Related documents

| SN | Document Name | Remark |
|---|---|---|
| [1] | SIM7028 Series_AT Command Manua | AT Command of SIM7028 module |

## 6.2 Conventions and abbreviations

Table 5:Conventions and abbreviations

| Abbreviation | Description |
|---|---|
| ME | Mobile Equipment |
| MS | Mobile Station |
| TA | Terminal Adapter |
| DCE | Data Communication Equipment |
| TE | Terminal Equipment |
| DTE | Data Terminal Equipment |
| PDP | Packet Data Protocol |
| TCP | Terminal Control Protocol |
| UDP | User Datagram Protocol |