RG50xQ&RM5xxQ Series
TCP/IP Application Note

5G Module Series

Version: 1.0

Date: 2022-05-05

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2022-03-31	Ozzy ANG/ Demon YANG	Creation of the document
1.0	2022-05-05	Ozzy ANG/ Demon YANG	First official release

Contents

Table Index

1 Introduction

Quectel 5G RG50xQ series and RM5xxQ series modules feature embedded TCP/IP stack, which enables the host to access the Internet directly through AT commands. This greatly reduces the dependence on external PPP and TCP/IP protocol stacks and thus minimizes the cost.

RG50xQ series and RM5xxQ series modules provide the following socket services: TCP client, UDP client, TCP server and UDP server.

This document introduces how to use the TCP/IP function of the Quectel RG50xQ series and RM5xxQ series modules through AT commands.

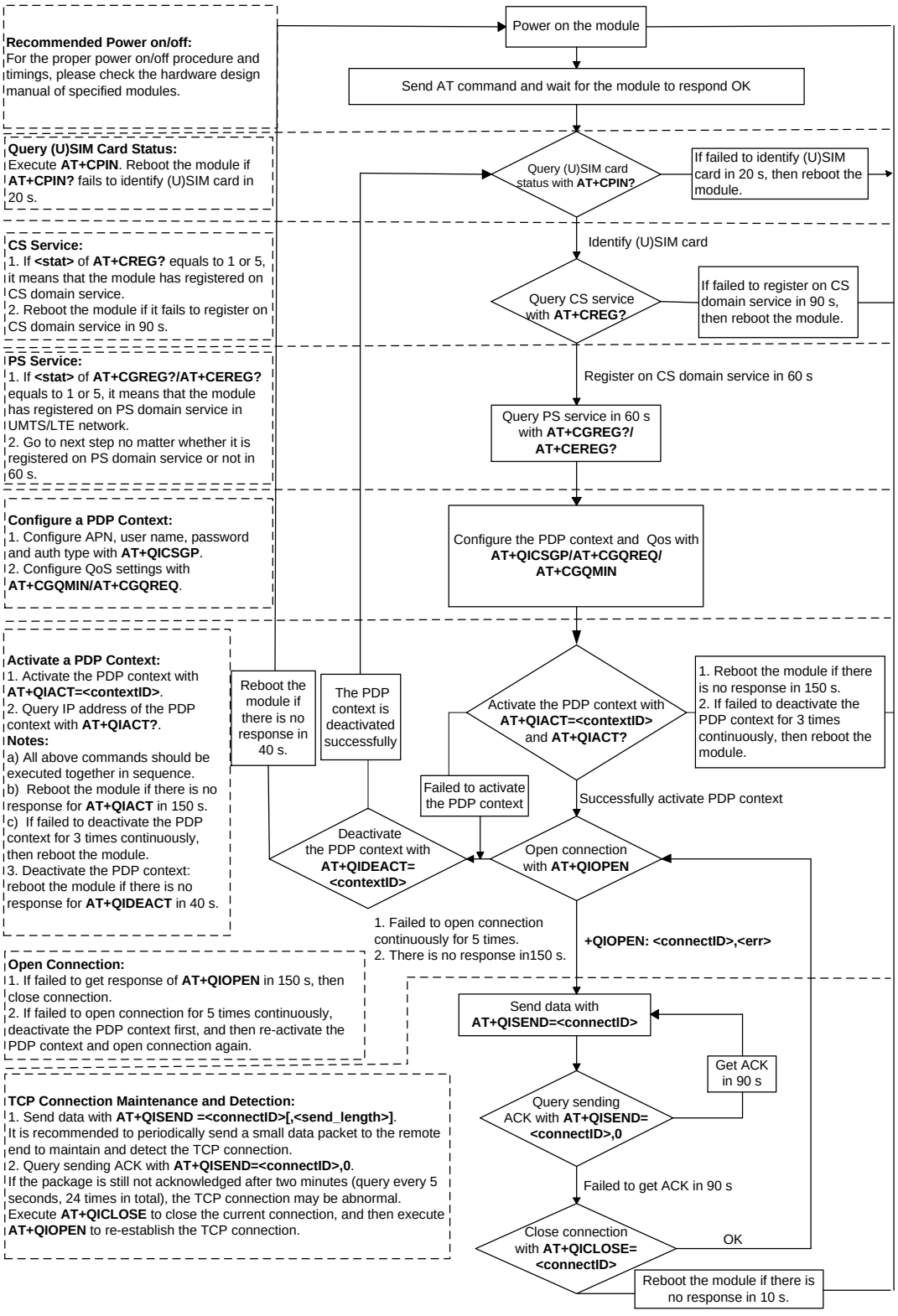
1.1. Applicable Modules

Table 1: Applicable Modules

Module Series	Model
RG50xQ	RG500Q Series
	RG501Q-EU
	RG502Q Series
RM5xxQ	RM500Q Series
	RM502Q-AE
	RM510Q-GL
	RM505Q-AE

1.2. The Process of Using TCP/IP AT Commands

Through TCP/IP AT commands, the host can configure a PDP context, activate/deactivate the PDP context, start/close socket service and send/receive data via socket service. The following figure illustrates how to use TCP/IP AT commands.



Notes:

1. Please note that users need to wait for the final response (for example "**OK**", "**CME ERROR**", "**CMS ERROR**") of the last AT command you entered before you enter the next AT command. You can reboot the module if the module fails to get response in 60 s.
2. Reboot the module if the module has not got response of **AT+QIACT** in 150 s or response of **AT+QICLOSE** in 10s and in 40 s.
3. It is NOT recommended to frequently reboot the module. When the module has been continuously rebooted for 3 times due to failed AT command execution, it can be rebooted immediately for the first time after that. If it still fails, reboot the module after 10 minutes for the second time, and reboot after 30 minutes for the third time, one hour for the fourth time, etc.

Figure 1: Flow Chart of Using TCP/IP AT Commands

1.3. Description of Data Access Modes

RG50xQ series and RM5xxQ series modules support the following three kinds of data access modes:

- Buffer access mode
- Direct push mode
- Transparent transmission mode

When opening a socket service via **AT+QIOPEN**, the data access mode can be specified with **<access_mode>**. After a socket service is opened, the access mode can be switched with **AT+QISWTMD**.

1. In buffer access mode, data can be sent via **AT+QISEND**. If the module has received the data from the Internet, it will buffer them and report the URC: **+QIURC: "recv",<connectID>**. The host can retrieve the buffered data with **AT+QIRD**.

2. In direct push mode, data can be sent via **AT+QISEND**. If the module has received the data from the Internet, it will output them to COM port directly through the URC:
   +QIURC: "recv",<connectID>,<currectrecvlength><CR><LF><data> or **+QIURC: "recv",<connectID>,<currentrecvlength>,<remoteIP>,<remote_port><CR><LF><data>**.

3. In transparent transmission mode, the corresponding port (such as UART, USB modem port, etc.) is exclusively used for sending/receiving data directly to/from the Internet. It cannot be used for other purposes such as running AT commands, etc.

   - **Exit transparent transmission mode**

   To exit the transparent transmission mode, **+++** can be used. Follow the requirements below to prevent the **+++** from being misinterpreted as data:

   1) Do not input any character within 1 second before and after inputting**+++**.
   2) Input **+++** within 1 second, and wait until **OK** is returned. When **OK** is returned, the module is switched to buffer access mode.

   - **Return to transparent transmission mode**

   To return to transparent transmission mode either:

   1) Execute **AT+QISWTMD**. Before execution specify the **<access_mode>** as 2. Once transparent transmission mode is entered successfully, **CONNECT** is returned.

   OR

   2) Execute **ATO**. After a connection exits from transparent transmission mode, executing **ATO** switches the data access mode back to transparent transmission mode. Once transparent transmission mode is entered successfully, **CONNECT** is returned. If no connection has entered transparent transmission mode, **ATO** returns **NO CARRIER**.

NOTE

1. In buffer access mode, if the buffer is not empty, and the module receives data again, it does not report any new URC until all the received data have been retrieved with **AT+QIRD** from the buffer.
2. In transparent transmission mode, AT commands cannot be executed. If the socket connection is closed because of network error or other errors, the module reports **NO CARRIER** and exits the transparent transmission mode. In this case, execute **AT+QICLOSE** to close the socket service.

2 Description of TCP/IP AT Commands

2.1. AT Command Introduction

2.1.1. Definitions

- **<CR>** Carriage return character.
- **<LF>** Line feed character.
- **<...>** Parameter name. Angle brackets do not appear on the command line.
- **[...]** Optional parameter of a command or an optional part of TA information response. Square brackets do not appear on the command line. When an optional parameter is not given in a command, the new value equals its previous value or the default settings, unless otherwise specified.
- **Underline** Default setting of a parameter.

2.1.2. AT Command Syntax

All command lines must start with **AT** or **at** and end with **<CR>**. Information responses and result codes always start and end with a carriage return character and a line feed character: **<CR><LF><response><CR><LF>**. In tables presenting commands and responses throughout this document, only the commands and responses are presented, and **<CR>** and **<LF>** are deliberately omitted.

Table 2: Types of AT Commands

Command Type	Syntax	Description
Test Command	**AT+<cmd>=?**	Test the existence of the corresponding command and return information about the type, value, or range of its parameter.
Read Command	**AT+<cmd>?**	Check the current parameter value of the corresponding command.
Write Command	**AT+<cmd>=<p1>[,<p2>[,<p3>[...]]]**	Set user-definable parameter value.
Execution Command	**AT+<cmd>**	Return a specific information parameter or perform a specific action.

2.2. Declaration of AT Command Examples

The AT command examples in this document are provided to help you familiarize with AT commands and learn how to use them. The examples, however, should not be taken as Quectel's recommendation or suggestions about how you should design a program flow or what status you should set the module into. Sometimes multiple examples may be provided for one AT command. However, this does not mean that there exists a correlation among these examples and that they should be executed in a given sequence.

2.3. Description of AT Commands

2.3.1. AT+QICSGP Configure Parameters of a TCP/IP Context

This command configures the **<APN>**, **<username>**, **<password>** and other parameters of a TCP/IP context. The QoS settings can be configured with **AT+CGQMIN** and **AT+CGQREQ**. For more details about the AT commands, refer to ***document [1]***.

AT+QICSGP Configure Parameters of a TCP/IP Context	
Test Command **AT+QICSGP=?**	Response **+QICSGP: (**range of supported **<contextID>**s**),(**range of supported **<context_type>**s**),<APN>,<username>,<password>,(**range of supported **<authentication>**s**)** **OK**
Write Command Query a specific context configuration. **AT+QICSGP=<contextID>**	Response **+QICSGP: <context_type>,<APN>,<username>,<password>,<authentication>** **OK**
Write Command Configure the context. **AT+QICSGP=<contextID>[,<context_type>,<APN>[,<username>,<password>)[,<authentication>]]]**	Response If the optional parameters are omitted, query the current specific context configuration: **+QICSGP: <context_type>,<APN>,<username>,<password>,<authentication>** **OK** If the optional parameters are specified, configure the specified context: **OK** If there is any error:

	ERROR
Maximum Response Time	/
Characteristic	The command takes effect immediately. The configurations will not be saved.

Parameter

<contextID>	Integer type. The PDP context ID. Range: 1–16.
<context_type>	Integer type. The protocol type. 1 IPv4 2 IPv6 3 IPv4v6
<APN>	String type. Access point name.
<username>	String type. Username. The maximum length: 127 bytes.
<password>	String type. Password. The maximum length: 127 bytes.
<authentication>	Integer type. The APN authentication methods. 0 None 1 PAP 2 CHAP 3 PAP or CHAP

2.3.2. AT+QIACT Activate a PDP Context

Before activating a PDP context with **AT+QIACT**, the context should be configured with **AT+QICSGP**. After activation, the IP address can be queried with **AT+QIACT?**.

Although the range of **<contextID>** is 1–16, the module supports maximum 3 PDP contexts activated simultaneously. Depending on the network, it may take at most 150 seconds to return **OK** or **ERROR** after executing **AT+QIACT**. Before the response is returned, other AT commands cannot be executed.

AT+QIACT Activate a PDP Context	
Test Command **AT+QIACT=?**	Response **+QIACT: (**range of supported **<contextID>**s**)** **OK**
Read command **AT+QIACT?**	Response Return the list of the currently activated contexts and their IP addresses: **+QIACT: 1,<context_state>,<context_type>[,<IP_address>]** **[...**

	+QIACT: 16,<context_state>,<context_type>[,<IP_address>]] **OK**
Write Command Activate a specified PDP context **AT+QIACT=<contextID>**	Response **OK** If there is any error: **ERROR**
Maximum Response Time	150 seconds, determined by the network.
Characteristics	/

Parameter

<contextID>	Integer type. The PDP context ID. Range: 1–16.
<context_state>	Integer type. The PDP context state. 0 Deactivated 1 Activated
<context_type>	Integer type. The protocol type. 1 IPv4 2 IPv6 3 IPv4v6
<IP_address>	String type. The local IP address after the context is activated.

2.3.3. AT+QIDEACT Deactivate a PDP Context

This command deactivates a specific context and closes all TCP/IP connections set up in this context. Depending on the network, it may take at most 40 seconds to return **OK** or **ERROR** after executing **AT+QIDEACT**. Before the response is returned, other AT commands cannot be executed.

AT+QIDEACT Deactivate a PDP Context	
Test Command **AT+QIDEACT=?**	Response **+QIDEACT: (**range of supported **<contextID>**s**)** **OK**
Write Command **AT+QIDEACT=<contextID>**	Response **OK** If there is any error: **ERROR**

Maximum Response Time	40 seconds, determined by the network.
Characteristics	/

Parameter

<contextID>	Integer type. The PDP context ID. Range: 1–16.

2.3.4. AT+QIOPEN Open a Socket Service

This command opens a socket service. The service type can be specified by **<service_type>**. The data access mode (buffer access mode, direct push mode and transparent transmission mode) can be specified by **<access_mode>**. The URC **+QIOPEN** indicates whether the socket service has been opened successfully.

1. If **<service_type>** is "TCP LISTENER", the module works as TCP server. After accepting a new TCP connection, the module automatically specifies a **<connectID>** and reports a URC **+QIURC: "incoming",<connectID>,<serverID>,<remoteIP>,<remote_port>**. The range of **<connectID>** is 0–11. The type of this new incoming connection is "TCP INCOMING" and the **<access_mode>** of "TCP INCOMING" is the same as that of "TCP LISTENER".

2. If **<service_type>** is "UDP SERVICE", UDP data can be sent to or received from the remote IP through **<local_port>**.
   - Send data: execute **AT+QISEND=<connectID>,<send_length>,<remoteIP>,<remote_port>**.
   - Receive data in direct push mode: the module reports the URC **+QIURC: "recv",<connectID>,<currentrecvlength>,<remoteIP>,<remote_port><CR><LF><data>**.
   - Receive data in buffer access mode: the module reports the URC **+QIURC: "recv",<connectID>**, and then the received data can be read through **AT+QIRD=<connectID>**.

3. It is suggested to wait for 150 seconds for **+QIOPEN: <connectID>,<err>** to be outputted after executing the Write Command. If the response cannot be received in 150 s, use **AT+QICLOSE** to close the socket.

AT+QIOPEN Open a Socket Service	
Test Command **AT+QIOPEN=?**	Response **+QIOPEN: (**range of supported **<contextID>**s**),(**range of supported **<connectID>**s**),"TCP/UDP/TCP LISTENER/UDP SERVICE","<IP_address>/<domain_name>",<remote_port>,<local_port>,(**range of supported **<access_mode>**s**)** **OK**

Write Command **AT+QIOPEN=<contextID>,<connectID>,<service_type>,<IP_address>/<domain_name>,<remote_port>[,<local_port>[,<access_mode>]]**	Response If the service is in transparent transmission mode (**<access_mode>**=2) and the service is opened successfully: **CONNECT** If there is any error: **ERROR** Error description can be got through **AT+QIGETERROR**. If the service is in buffer access mode (**<access_mode>**=0) or direct push mode (**<access_mode>**=1): **OK** **+QIOPEN: <connectID>,<err>** **<err>** is 0 when the service is opened successfully. In other cases, **<err>** is not 0.
Maximum Response Time	150 seconds, determined by the network.
Characteristic	/

Parameter

<contextID>	Integer type. The PDP context ID. Range: 1–16.
<connectID>	Integer type. Socket service index. Range: 0–11.
<service_type>	String type. Socket service type. "TCP" Start a TCP connection as a client "UDP" Start a UDP connection as a client "TCP LISTENER" Start a TCP server to listen to incoming TCP connections "UDP SERVICE" Start a UDP service
<IP_address>	String type. If **<service_type>** is "TCP" or "UDP", it indicates the IP address of remote server, such as 220.180.239.212. If **<service_type>** is "TCP LISTENER" or "UDP SERVICE", please input 127.0.0.1.
<domain_name>	String type. The domain name address of the remote server.
<remote_port>	Integer type. Port number of the remote server. Range: 0–65535. It is valid only when **<service_type>** is "TCP" or "UDP".
<local_port>	Integer type. The local port. Range: 0–65535. If **<service_type>** is "TCP LISTENER" or "UDP SERVICE", this parameter must be specified. If **<service_type>** is "TCP" or "UDP" and **<local_port>** is 0, the local port will be assigned automatically. Otherwise the local port is assigned as specified.

<access_mode>	Integer type. The data access mode of the socket service. 0 Buffer access mode 1 Direct push mode 2 Transparent transmission mode
<err>	Error codes. Please refer to ***Chapter 4***.

2.3.5. AT+QICLOSE Close a Socket Service

This command closes a specified socket service. Depending on the network, it will take at most 10 seconds (default value, can be modified by **<timeout>**) to return **OK** or **ERROR** after executing **AT+QICLOSE**. Before the response is returned, other AT commands cannot be executed.

AT+QICLOSE Close a Socket Service	
Test Command **AT+QICLOSE=?**	Response **+QICLOSE: (**range of supported **<connectID>**s**),(**range of supported **<timeout>**s**)** **OK**
Write Command **AT+QICLOSE=<connectID>[,<timeout>]**	Response If the socket service is closed successfully: **OK** If it is failed to close the socket service: **ERROR**
Maximum Response Time	10 s by default, determined by the setting of **<timeout>**.
Characteristic	/

Parameter

<connectID>	Integer type. The socket service index. Range: 0–11.
<timeout>	Integer type. The timeout value for the response to be outputted. If the FIN ACK of the other peer is not received within **<timeout>**, the module will be forced to close the socket. Range: 0–65535. Default value: 10. Unit: second.

2.3.6. AT+QISTATE Query Socket Service Status

This command queries the socket service status. If the **<query_type>** is 0, it will return the status of all existing socket services in the specified context. If the **<query_type>** is 1, it will return the status of a specified socket service.

AT+QISTATE Query Socket Service Status	
Test Command **AT+QISTATE=?**	Response **OK**
Read/Execution Command **AT+QISTATE?** or **AT+QISTATE**	Response Return the status of all existing connections: **+QISTATE: <connectID>,<service_type>,<IP_address>,<remote_port>,<local_port>,<socket_state>,<contextID>,<serverID>,<access_mode>,<AT_port>** **[…]** **OK**
Write Command If **<query_type>** is 0, query the connection status of a specified context **AT+QISTATE=<query_type>,<contextID>**	Response Return the status of all existing connections in a specified context: **+QISTATE: <connectID>,<service_type>,<IP_address>,<remote_port>,<local_port>,<socket_state>,<contextID>,<serverID>,<access_mode>,<AT_port>** **[…]** **OK**
Write Command If **<query_type>** is 1, query the connection status of a specified socket service **AT+QISTATE=<query_type>,<connectID>**	Response **+QISTATE: <connectID>,<service_type>,<IP_address>,<remote_port>,<local_port>,<socket_state>,<contextID>,<serverID>,<access_mode>,<AT_port>** **OK**
Maximum Response Time	/
Characteristic	/

Parameter

<query_type>	Integer type. The query type. 0 Query connection status of all socket services in a specified context 1 Query connection status of a specified socket service
<contextID>	Integer type. The PDP context ID. Range: 1–16.
<connectID>	Integer type. The socket service index. Range: 0–11.
<service_type>	String type. The socket service type. "TCP" Start a TCP connection as a client "UDP" Start a UDP connection as a client "TCP LISTENER" Start a TCP server to listen to incoming TCP connection

	"TCP INCOMING"	Start a TCP connection accepted by a TCP server
	"UDP SERVICE"	Start a UDP service
<IP_address>	String type. IP address.	
	If **<service_type>**="TCP" or "UDP", it is the IP address of remote server.	
	If **<service_type>**="TCP LISTENER" or "UDP SERVICE", it is the local IP address.	
	If **<service_type>**="TCP INCOMING", it is the IP address of remote client.	
<remote_port>	Integer type. Remote port number. Range: 0–65535.	
	If **<service_type>**="TCP" or "UDP", it is the port of remote server.	
	If **<service_type>**="TCP LISTENER" or "UDP SERVICE", **<remote_port>** equals to 0 and the port is invalid.	
	If **<service_type>**="TCP INCOMING", it is the port of remote client.	
<local_port>	Integer type. Local port number. Range: 0–65535.	
	If **<local_port>** is 0, then the local port is assigned automatically.	
<socket_state>	Integer type. The socket service status.	
	0	"Initial": connection has not been established
	1	"Opening": client is connecting or server is trying to listen
	2	"Connected": client/incoming connection has been established
	3	"Listening": server is listening
	4	"Closing": connection is closing
<serverID>	Integer type. It is valid only when **<service_type>** is "TCP INCOMING". **<serverID>** represents which server accepts this TCP incoming connection, and the value is the same as **<connectID>** of this server's "TCP LISTENER".	
<access_mode>	Integer type. Data access mode.	
	0	Buffer access mode
	1	Direct push mode
	2	Transparent transmission mode
<AT_port>	String type. COM port of socket service.	
	"usbmodem"	USB modem port
	"usbat"	USB AT port
	"uart1"	UART port1
	"cmux1"	MUX port 1
	"cmux2"	MUX port 2
	"cmux3"	MUX port 3
	"cmux4"	MUX port 4

2.3.7. AT+QISEND Send Data

In buffer access mode (**<access_mode>**=0) or direct push mode (**<access_mode>**=1), the data can be sent with **AT+QISEND**. If the data have been sent to the module successfully, **SEND OK** is returned, otherwise **SEND FAIL** or **ERROR** is returned.

- **SEND FAIL** indicates the sending buffer is full. In this case, resending of the data can be tried.
- **ERROR** indicates an error in the sending data process. In this case, wait for some time before resending the data. Maximum length of data to be sent: 1460 bytes.
- **SEND OK** means that the data have been sent to the peer, but it does not mean they have reached

the server successfully. You can query whether the data have reached the server with **AT+QISEND=<connectID>,0**.

AT+QISEND Send Data	
Test Command **AT+QISEND=?**	Response **+QISEND: (**range of supported **<connectID>**s**),(**range of supported **<send_length>**s**)** **OK**
Write Command Send variable-length data when **<service_type>** is "TCP", "UDP" or "TCP INCOMING" **AT+QISEND=<connectID>**	Response **>** After the response **>**, input the data to be sent. Tap **Ctrl + Z** to send, and tap **Esc** to cancel the operation If the connection has been established and the data are sent successfully: **SEND OK** If the connection has been established but the sending buffer is full: **SEND FAIL** If the connection has not been established, abnormally closed, or any parameter is incorrect: **ERROR**
Write Command Send fixed-length data when **<service_type>** is "TCP", "UDP" or "TCP INCOMING" **AT+QISEND=<connectID>,<send_length>**	Response **>** After the response **>**, input the data until the data length equals to **<send_length>**. If the connection has been established and the data are sent successfully: **SEND OK** If the connection has been established but the sending buffer is full: **SEND FAIL** If the connection has not been established, abnormally closed, or any parameter is incorrect: **ERROR**
Write Command If **<service_type>** is "UDP SERVICE"	Response This command sends fixed length data to a specified remote

AT+QISEND=<connectID>,<send_length>,<remoteIP>,<remote_port>	IP address and remote port. The **<service_type>** must be "UDP SERVICE". **>** After the response **>**, input the data until the data length equals to **<send_length>** If the connection has been established and the data are sent successfully: **SEND OK** If the connection has been established but the sending buffer is full: **SEND FAIL** If the connection has not been established, abnormally closed, or any parameter is incorrect: **ERROR**
Write Command When **<send_length>** is 0, query the sent data **AT+QISEND=<connectID>,0**	Response If the specified connection exists: **+QISEND: <total_send_length>,<ackedbytes>,<unackedbytes>** **OK** If there is any error: **ERROR**
Maximum Response Time	/
Characteristic	/

Parameter

<connectID>	Integer type. Socket service index. Range: 0–11.
<send_length>	Integer type. The length of data to be sent. Range: 1–1460. Unit: byte.
<remoteIP>	String type. The remote IP address (must be dot format). It is valid only when **<service_type>** is "UDP SERVICE".
<remote_port>	Integer type. Remote port. Range: 0–65535. It is only valid when **<service_type>** is "UDP SERVICE".
<total_send_length>	Integer type. The total length of sent data. Unit: byte.
<ackedbytes>	Integer type. The total length of acknowledged data. Unit: byte.
<unackedbytes>	Integer type. The total length of unacknowledged data. Unit: byte.

2.3.8. AT+QIRD Read the Received TCP/IP Data

In buffer access mode, after receiving data, the module buffers it and reports **+QIURC: "recv",<connectID>**, and then the data can be read via **AT+QIRD**.

Please note that if the buffer is not empty, and the module receives data again, it will not report a new URC until all the received data has been read via **AT+QIRD** from buffer.

AT+QIRD Read the Received TCP/IP Data	
Test Command **AT+QIRD=?**	Response **+QIRD: (**range of supported **<connectID>**s**),(**range of supported **<read_length>**s**)** **OK**
Write Command When **<service_type>** is "TCP", "UDP", "TCP INCOMING" **AT+QIRD=<connectID>[,<read_length>]**	Response If the specified connection has received the data: **+QIRD: <read_actual_length>** **<data>** **OK** If there is no data: **+QIRD: 0** **OK** If the connection does not exist: **ERROR**
Write Command When **<service_type>** is "UDP SERVICE" **AT+QIRD=<connectID>**	Response If data exists: **+QIRD: <read_actual_length>,<remoteIP>,<remote_port>** **<data>** **OK** If there is no data: **+QIRD: 0** **OK** If the connection does not exist: **ERROR**
Write Command	Response

When **<read_length>** is 0, query the retrieved data length **AT+QIRD=<connectID>,0**	If the specified connection exists: **+QIRD: <total_recvlength>,<have_read_length>,<unread_length>** **OK** If there is any error: **ERROR**
Maximum Response Time	/
Characteristic	/

Parameter

<connectID>	Integer type. The socket service index. Range: 0–11.
<read_length>	Integer type. Maximum length of data to be read. Range: 0–1500. Unit: byte.
<read_actual_length>	Integer type. Length of actually retrieved data. Unit: byte.
<remoteIP>	String type. The remote IP address. It is valid only when **<service_type>** is "UDP SERVICE".
<remote_port>	Integer type. Remote port. Range: 0–65535. It is valid only when **<service_type>** is "UDP SERVICE".
<data>	String type.The data that has been read.
<total_recvlength>	Integer type. The total length of the received data. Unit: byte.
<have_read_length>	Integer type. The length of data that has been read. Unit: byte.
<unread_length>	Integer type. The length of data that has not been read. Unit: byte.

2.3.9. AT+QISENDEX Send Hex String Data

This command sends hex string data and cannot be applied for "UDP SERVICE" and "TCP LISTENER" Socket service type.

AT+QISENDEX Send Hex String Data	
Test Command **AT+QISENDEX=?**	Response **+QISENDEX: (**range of supported **<connectID>**s**),<hex_string>** **OK**
Write Command **AT+QISENDEX=<connectID>,<hex_string>**	Response If the hex string is sent successfully: **SEND OK** If the sending buffer is full: **SEND FAIL**

	If the connection does not exist: **ERROR**
Maximum Response Time	/
Characteristic	/

Parameter

<connectID>	Integer type. The Socket service index. Range: 0–11.
<hex_string>	String type. Hex string data. The max. length is 512 bytes.

2.3.10. AT+QISWTMD Switch Data Access Mode

This command switches the data access mode which includes buffer access mode, direct push mode and transparent transmission mode. When a socket service is established, the data access mode can be specified via the **<access_mode>** of **AT+QIOPEN**. After a socket has been opened, the data access mode can be changed with **AT+QISWTMD**.

AT+QISWTMD Switch Data Access Mode	
Test Command **AT+QISWTMD=?**	Response **+QISWTMD: (**range of supported **<connectID>**s**),(**range of supported **<access_mode>**s**)** **OK**
Write Command **AT+QISWTMD=<connectID>,<access_mode>**	Response If data access mode is switched successfully and **<access_mode>** is 0 or 1: **OK** If data access mode is switched successfully and **<access_mode>** is 2, the module will enter data mode: **CONNECT** If there is any error: **ERROR**
Maximum Response Time	/
Characteristic	This command takes effect immediately. The configuration will not be saved.

Parameter

<connectID>	Integer type. The socket service index. Range: 0–11.
<access_mode>	Integer type. The data access modes of the connection. 0 Buffer access mode 1 Direct push mode 2 Transparent transmission mode

2.3.11. AT+QPING Ping a Remote Server

This command tests the reachability of a host on an Internet protocol network. Before using the ping utility, the host should activate the context of the corresponding **<contextID>** with **AT+QIACT**. The command returns the result within **<timeout>** and the default value of **<timeout>** is 4 seconds.

AT+QPING Ping a Remote Server	
Test Command **AT+QPING=?**	Response **+QPING: (**range of supported **<contextID>**s**),<host>,(**range of supported **<timeout>**s**),(**range of supported **<pingnum>**s**)** **OK**
Write Command **AT+QPING=<contextID>,<host>[,<timeout>[,<pingnum>]]**	Response If a remote server is pinged successfully: **OK** **+QPING: <result>[,<IP_address>,<bytes>,<time>,<ttl>]** **[…]** **+QPING: <finresult>[,<sent>,<rcvd>,<lost>,<min>,<max>,<avg>]** If there is any error: **ERROR**
Maximum Response Time	/
Characteristic	This command takes effect immediately. The configuration will not be saved.

Parameter

<contextID>	Integer type. The PDP context ID. Range: 1–16.
<host>	String type. The host address. It is a domain name or a dotted decimal IP address.

<timeout>	Integer type. The maximum time to wait for the response of each ping request. Range: 1–255. Default value: 4. Unit: second.
<pingnum>	Integer type. The maximum number of times for sending ping request. Range: 1–10. Default value: 4.
<result>	Integer type. The result of each ping request. 0 Received the ping response from the server. In this case, it is followed by **,<IP_address>,<bytes>,<time>,<ttl>**. Others Error codes. Please refer to ***Chapter 4***.
<IP_address>	String type. The IP address of the remote server formatted as a dotted decimal IP.
<bytes>	Integer type. The length of each sent ping request. Unit: byte.
<time>	Integer type. The time wait for the response of the ping request. Unit: ms.
<ttl>	Integer type. Time to live value of the response packet for the ping request.
<finresult>	Integer type. The final result of the command. 0 It is finished normally. It is successful to activate the context and find the host. In this case, it is followed by **,<sent>,<rcvd>,<lost>,<min>,<max>,<avg>**. Others Error codes. Please refer to ***Chapter 4***.
<sent>	Integer type. Total number of sent ping requests.
<rcvd>	Integer type. Total number of the ping requests that have received responses.
<lost>	Integer type. Total number of timed out ping requests.
<min>	Integer type. The minimum response time. Unit: ms.
<max>	Integer type. The maximum response time. Unit: ms.
<avg>	Integer type. The average response time. Unit: ms.

2.3.12. AT+QISDE Control Whether to Echo the Data for AT+QISEND

This command controls whether to echo the data for **AT+QISEND**.

AT+QISDE Control Whether to Echo the Data for AT+QISEND	
Test command **AT+QISDE=?**	Response **+QISDE: (**list of supported **<echo>**s**)** **OK**
Read command **AT+QISDE?**	Response **+QISDE: <echo>** **OK**
Write Command **AT+QISDE=<echo>**	Response **OK** If there is any error: **ERROR**

Maximum Response Time	/
Characteristic	This command takes effect immediately. The configuration will not be saved.

Parameter

<echo>	Integer type. Whether to echo the data for **AT+QISEND**.	
	0	Do not echo the data
	1	Echo the data

2.3.13. AT+QIGETERROR Query the Error Code of the Last AT Command

If **ERROR** is returned after TCP/IP commands are executed, the detailed information about a result code can be queried with **AT+QIGETERROR**. Please note that **AT+QIGETERROR** only returns the result code of the last TCP/IP AT command.

AT+QIGETERROR Query the Error Code of the Last AT Command	
Test command **AT+QIGETERROR=?**	Response **OK**
Execution Command **AT+QIGETERROR**	Response **+QIGETERROR: <err>,<errcode_description>** **OK**
Maximum Response Time	/
Characteristic	/

Parameter

<err>	Error codes. Please refer to ***Chapter 4***.
<errcode_description>	A string parameter indicates the details of error information. Please refer to ***Chapter 4*** for details.

2.4. Description of URCs

+QIURC: is used at the beginning of the URC of TCP/IP AT commands to be reported to the host. The URC contains the reports about incoming data, closed connection and incoming connection and etc. Actually, there is **<CR><LF>** both before and after URC, but **<CR><LF>** is intentionally not presented.

2.4.1. +QIURC: "closed" URC Indicating Connection Closed

When TCP socket service is closed by remote client or due to network error, the URC will be outputted, and the status of socket service will be "Closing" (**<socket_state>=4**). **AT+QICLOSE=<connectID>** can be used to change the **<socket_state>** to "Initial".

+QIURC: "closed" URC Indicating Connection Closed	
+QIURC: "closed",<connectID>	Socket service connection is closed.

Parameter

<connectID>	Integer type. The socket service index. Range: 0–11.

2.4.2. +QIURC: "recv" URC Indicating Incoming Data

In buffer access mode or direct push mode, after receiving data, the module will report a URC to the host.

- In buffer access mode, after receiving data, the module will report URC **+QIURC: "recv",<connectID>** to notify the host. Then host can retrieve data through **AT+QIRD**.
- In direct push mode, the received data will be outputted to COM port directly.

+QIURC: "recv" URC Indicating Incoming Data	
+QIURC: "recv",<connectID>	Indicates incoming data in buffer access mode. The host can retrieve data via **AT+QIRD**.
+QIURC: "recv",<connectID>,<current_recvlength><CR><LF><data>	Indicates incoming data in direct push mode when the **<service_type>** is "TCP", "UDP" or "TCP INCOMING".
+QIURC: "recv",<connectID>,<current_recvlength>,<remoteIP>,<remote_port><CR><LF><data>	Indicates incoming data in direct push mode when **<service_type>** is "UDP SERVICE".

Parameter

<connectID>	Integer type. The socket service index. Range: 0–11.
<currentrecvlength>	Integer type. The length of actually received data.
<remoteIP>	String type. Remote IP address.
<remote_port>	Integer type. Remote port. Range: 0–65535.
<data>	String type. The received data.

NOTE

If the buffer is not empty, and the module receives data again, it does not report a new URC until all the received data have been retrieved with **AT+QIRD** from the buffer.

2.4.3. +QIURC: "incoming full" Indicating Incoming Connection Reaches the Limit

If the incoming connection reaches the limit, or no socket system resources can be allocated, then the module will report the URC as **+QIURC: "incoming full",<serverID>** for the new incoming connection request.

+QIURC: "incoming full" Indicating Incoming Connection Reaches the Limit	
+QIURC: "incoming full",<serverID>	Indicates that the number of incoming connections has reached the limit.

Parameter

<serverID>	Integer type. The incoming **<connectID>** accepted by the server whose **<service_type>** is "TCP LISTENER" and listening socket ID is **<serverID>**.

2.4.4. +QIURC: "incoming" Indicating Incoming Connection

If the **<service_type>** is "TCP LISTENER", when a remote client connects to this server, the host will automatically assign a free **<connectID>** for the new connection. The range of **<connectID>** is 0–11. In such a case, the module will report the URC. The **<service_type>** of the new connection will be "TCP INCOMING", and the **<access_mode>** of "TCP INCOMING" is the same with that of "TCP LISTENER".

+QIURC: "incoming" Indicating Incoming Connection	
+QIURC: "incoming",<connectID>,<serverID>,<remoteIP>,<remote_port>	When the new incoming connection is accepted by **<serverID>**, the allocated **<connectID>**, **<remoteIP>** and **<remote_port>** will be informed by this URC.

Parameter

<connectID>	Integer type. Assign this socket service for the incoming connection, which is automatically specified by the module. Range: 0–11.
<serverID>	Integer type. The incoming **<connectID>** accepted by the server whose **<service_type>** is "TCP LISTENER" and listening socket ID is **<serverID>**.
<remoteIP>	String type. Remote IP address of the incoming **<connectID>**.
<remote_port>	Integer type. Remote port of the incoming **<connectID>**. Range: 0–65535.

2.4.5. +QIURC: "pdpdeact" Indicating PDP Deactivation

PDP context may be deactivated by the network. The module will report the URC to the host about PDP deactivation. In such a case, the host must execute **AT+QIDEACT** to deactivate the context and reset all connections.

+QIURC: "pdpdeact" Indicating PDP Deactivation	
+QIURC: "pdpdeact",<contextID>	**<contextID>** context is deactivated.

Parameter

<contextID>	Integer type. The PDP context ID. Range: 1–16.

3 Examples

3.1. Configure and Activate a Context

3.1.1. Configure a Context

```
AT+QICSGP=1                          //Query the configuration of context 1.
+QICSGP: 1,"","","",0

OK
AT+QICSGP=1,1,"UNINET","","",1       //Configure context 1. China Unicom APN: “UNINET”.
OK
```

3.1.2. Activate a Context

```
AT+QIACT=1                           //Activate context 1. Depending on the network, the maximum
                                       response time is 150 s.
OK                                   //Activated the context successfully.
AT+QIACT?                            //Query the context state.
+QIACT: 1,1,1,"10.7.157.1"

OK
```

3.1.3. Deactivate a Context

```
AT+QIDEACT=1                         //Deactivate context 1.
OK                                   //Deactivated the context successfully. Depending on the
                                       network, the maximum response time is 40 s.
```

3.2. TCP Client Works in Buffer Access Mode

3.2.1. Set up a TCP Client Connection and Enter Buffer Access Mode

```
AT+QIOPEN=1,0,"TCP","220.180.239.212",8009,0,0    //Context is 1 and <connectID> is 0. Before using AT+QIOPEN, the host should activate the context with AT+QIACT first.

OK

+QIOPEN: 0,0                                      //TCP client is connected successfully. It is suggested to wait for 150 s for the URC +QIOPEN: <connectID>,<err>. If the URC cannot be received in 150 s, the host could use AT+QICLOSE to close the socket.
AT+QISTATE=1,0                                    //Query the connection status of socket service when <connectID> is 0.
+QISTATE: 0,"TCP","220.180.239.201",8009,65514,2,1,0,0,"usbmodem"

OK
```

3.2.2. Send Data in Buffer Access Mode

```
AT+QISEND=0                                       //Send variable-length data.
> test1<ctrl+Z>
SEND OK                                           //SEND OK does not mean the data has been sent to the server successfully. The host can query whether the data has reached the server through AT+QISEND=0,0.

AT+QISEND=0,4                                     //Send fixed-length data and the data length is 4 bytes.
> test
SEND OK
AT+QISEND=0,0                                     //Query the length of sent data.
+QISEND: 9,9,0

OK
AT+QISENDEX=0,"3132333435"                        //Send hex string data.
SEND OK
AT+QISEND=0,0                                     //Query the length of sent data, acknowledged data and unacknowledged data.
+QISEND: 14,14,0

OK
```

3.2.3. Receive Data from Remote Server in Buffer Access Mode

```
+QIURC: "recv",0                    //The received data when <connectID>=0.
AT+QIRD=0,1500                      //Read data, the maximum length of data to be retrieved is 1500 bytes.
+QIRD: 5                            //The length of actually received data is 5 bytes.
test1

OK
AT+QICFG="recvind",1
OK
+QIURC: "recv",0,5                  //The <connectID>= 0, and the length of received data is 5 bytes.
AT+QIRD=0,1500                      //Read data, the maximum length of data to be retrieved is 1500 bytes.
+QIRD: 5                            //The length of actual received data is 5 bytes.
test1

OK

AT+QIRD=0,1500
+QIRD: 0                            //No data in buffer.

OK
AT+QIRD=0,0                         //Query the total length of received data, including read and unread data.
+QIRD: 10,10,0

OK
```

3.2.4. Close a Connection

```
AT+QICLOSE=0                        //Close a connection whose <connectID> is 0. Depending on the network, the maximum response time is 10 s.

OK
```

3.3. TCP Client Works in Transparent Transmission Mode

3.3.1. Set up a TCP Client Connection and Enter Transparent Transmission Mode

```
AT+QIOPEN=1,0,"TCP","220.180.239.212",8009,0,2  //Context is 1 and <connectID> is 0. Before using AT+QIOPEN, the host should activate the context with AT+QIACT first.
CONNECT                                         //TCP client is connected successfully. It is
```

	suggested to wait for 150 seconds for the URC **CONNECT**. If the URC response cannot be received in 150 seconds, the host could use **AT+QICLOSE** to close the socket.

3.3.2. Send Data in Transparent Transmission Mode

<All data got from COM port will be sent to internet directly>

3.3.3. Receive Data from Remote Server in Transparent Transmission Mode

Test 1	//All data received from internet will be outputted through COM port directly.

3.3.4. Close a TCP Client

AT+QICLOSE=0	//After using **+++** to exit transparent transmission mode, the host could use **AT+QICLOSE** to close the TCP link. Depending on the network, the maximum response time is 10 s.
OK	

3.4. TCP Client Works in Direct Push Mode

3.4.1. Set up a TCP Client Connection and Enter Direct Push Mode

AT+QIOPEN=1,0,"TCP","220.180.239.212",8009,0,1	//Context is 1 and **<connectID>** is 0. Before using **AT+QIOPEN**, the host should activate the context with **AT+QIACT** first.
OK	
+QIOPEN: 0,0	//TCP client is connected successfully. It is suggested to wait for 150 seconds for the URC **+QIOPEN: <connectID>,<err>**. If the URC cannot be received in 150 seconds, the host could use **AT+QICLOSE** to close the socket.
AT+QISTATE=1,0	//Query the connection status of socket service when **<connectID>** is 0.

```
+QISTATE: 0,"TCP","220.180.239.201",8009,65344,2,1,0,1,"usbmodem"

OK
```

3.4.2. Send Data in Direct Push Mode

```
AT+QISEND=0                    //Send variable-length data.
> test1<ctrl+Z>
SEND OK                        //SEND OK does not mean the data has been sent to
                                 the server successfully. Host can query whether the
                                 data has reached the server via AT+QISEND=0,0.
AT+QISEND=0,5                  //Send fixed length data and the data length is 5 bytes.
> test2
SEND OK
AT+QISEND=0,0                  //Query the length of sent data, acknowledged data and
                                 unacknowledged data.
+QISEND: 10,10,0

OK
```

3.4.3. Receive Data from Remote Server in Direct Push Mode

```
+QIURC: "recv",0,4             //Receive data from remote server.
test
```

3.4.4. Close a TCP Client

```
AT+QICLOSE=0                   //Close the connection whose <connectID> is 0.
                                 Depending on the network, the maximum response time
                                 is 10 seconds.
OK
```

3.5. TCP Server Works in Buffer Access Mode

3.5.1. Start a TCP Server

```
AT+QIOPEN=1,1,"TCP LISTENER","127.0.0.1",0,2020,0 //Context is 1 and <connectID> is 1. Before
                                                    using AT+QIOPEN, the host should
```

```
                                                    activate the context with AT+QIACT first.
OK

+QIOPEN: 1,0                                        //TCP server is opened successfully.
AT+QISTATE=0,1                                      //Query whether the connection state of
                                                    <contextID> is 1.
+QISTATE: 1,"TCP LISTENER","10.7.157.1",0,2020,3,1,1,0,"usbmodem"

OK
```

3.5.2. Accept TCP Incoming Connection

```
+QIURC: "incoming",11,1,"172.31.242.222",54091    //Accept a TCP connection. The <service_type>
                                                    is "TCP incoming", and <connectID> is 11.
```

3.5.3. Receive Data from Incoming Connection

```
+QIURC: "recv",11                                 //Receive data from remote incoming connection.
AT+QIRD=11,1500                                   //Read data received from incoming connection.
+QIRD: 4                                          //Actual data length is 4 bytes.
test

OK
AT+QIRD=11,1500
+QIRD: 0                                          //No data in buffer.

OK
AT+QIRD=11,0                                      //Query the total length of received data, including
                                                    read and unread data.
+QIRD: 4,4,0

OK
```

3.5.4. Close a TCP Server

```
AT+QICLOSE=11                                     //Close the incoming connection. Depending on
                                                    the network, the maximum response time is 10 s.
OK
AT+QICLOSE=1                                      //Close TCP server listening.
OK
```

3.6. UDP Service

3.6.1. Start a UDP Service

```
AT+QIOPEN=1,2,"UDP SERVICE","127.0.0.1",0,3030,0 //Start a UDP service. The <connectID> is 2 and <contextID> is 1. Before using AT+QIOPEN, the host should activate the context with AT+QIACT first.

OK

+QIOPEN: 2,0                                     //UDP service is opened successfully.
AT+QISTATE=0,1                                   // Query the connection status of socket service when <connectID> is 1.
+QISTATE: 2,"UDP SERVICE","10.7.157.1",0,3030,2,1,2,0,"usbmodem"

OK
```

3.6.2. Send UDP Data to Remote Client

```
AT+QISEND=2,10,"10.7.89.10",6969                 //Send 10-byte data to remote client whose IP is 10.7.89.10 and the remote port is 6969.

>1234567890
SEND OK
```

3.6.3. Receive Data from Remote Client

```
+QIURC: "recv",2                                 //Receive data from remote client.
AT+QIRD=2                                        //Read UDP data. One whole UDP packet will be outputted. There is no need to specify the read length.
+QIRD:  4,"10.7.76.34",7687                      //Data length is 4. The remote IP address is 10.7.76.34 and remote port is 7687.

AAAA

OK
AT+QIRD=2                                        //Read data.
+QIRD: 0                                         //No data in buffer.

OK
AT+QISEND=2,10,"10.7.76.34",7687                 //Send data to the remote whose IP is 10.7.76.34 and remote port is 7687.
```

```
>1234567890
SEND OK
```

3.6.4. Close a UDP Service

```
AT+QICLOSE=2                                    //Close the service.
OK
```

3.7. PING

```
AT+QPING=1,"www.baidu.com"                      //Ping www.baidu.com in context 1. Before pinging the destination IP address, the host should activate the context with AT+QIACT first.
OK

+QPING: 0,"61.135.169.125",32,192,255

+QPING: 0,"61.135.169.125",32,240,255

+QPING: 0,"61.135.169.125",32,241,255

+QPING: 0,"61.135.169.125",32,479,255

+QPING: 0,4,4,0,192,479,288
```

3.8. Get Last Error Code

```
AT+QIOPEN=1,"TCP","220.180.239.212",8009,0,1    //Send AT+QIOPEN with missing <connectID>.
ERROR
AT+QIGETERROR
+QIGETERROR: 552, invalid parameters

OK
```

4 Summary of Error Codes

If **ERROR** is returned after executing TCP/IP AT commands are executed, the details of error type can be queried via **AT+QIGETERROR**. Please note that **AT+QIGETERROR** just returns error code of the last TCP/IP AT command.

Table 3: Summary of Error Codes

<err>	<errcode_description>
0	Operation success
550	Unknown error
551	Operation blocked
552	Invalid parameters
553	Memory allocation failed
554	Socket creation failed
555	Operation not supported
556	Socket bind failed
557	Socket listen failed
558	Socket write failed
559	Socket read failed
560	Socket accept failed
561	PDP context activate failed
562	PDP context deactivate failed
563	Socket identity has been used
564	DNS busy

565	DNS parse failed
566	Socket connect failed
567	Socket has connection reset
568	Operation busy
569	Operation timeout
570	PDP context deactivated
571	Cancel sending
572	Operation not allowed
573	Port busy
574	PDP has been activated

5 Appendix References

Table 4: Related Document

Document Name
[1] Quectel_RG50xQ&RM5xxQ_Series_AT_Commands_Manual

Table 5: Terms and Abbreviations

Abbreviation	Description
3GPP	3rd Generation Partnership Project
ACK	Acknowledgement
APN	Access Point Name
ASCII	American Standard Code for Information Interchange
CHAP	Challenge Handshake Authentication Protocol
CS	Circuit Switching
DNS	Domain Name System
FIN	Finish
ID	Identifier
IP	Internet Protocol
NV	Non-Volatile
PAP	Password Authentication Protocol l
PDP	Packet Data Protocol
PPP	Point-to-Point Protocol
PS	Packet Switching

QoS	Quality of Service
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver& Transmitter
UDP	User Datagram Protocol
URC	Unsolicited Result Code
USB	Universal Serial Bus
(U)SIM	(Universal) Subscriber Identity Module