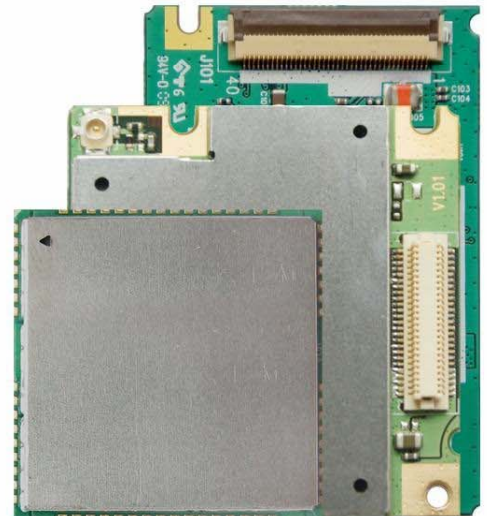




# Quectel Cellular Engine

## **GSM TCPIP Application Notes**

GSM\_TCPIP\_AN\_V1.01



<b>Document Title</b>	GSM TCPIP Application Notes
<b>Version</b>	1.01
<b>Date</b>	2009-11-20
<b>Status</b>	Release
<b>Document Control ID</b>	GSM_TCPIP_AN_V1.01

### **General Notes**

Quectel offers this information as a service to its customers, to support application and engineering efforts that use the products designed by Quectel. The information provided is based upon requirements specifically provided for Quectel by the customers. Quectel 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 Quectel 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**

This document contains proprietary technical information which is the property of Quectel Limited, copying of this document and giving it to others and the using or communication of the contents thereof, are forbidden without express authority. Offenders are liable to the payment of damages. All rights reserved in the event of grant of a patent or the registration of a utility model or design. All specification supplied herein are subject to change without notice at any time.

*Copyright © Quectel Wireless Solutions Co., Ltd. 2009*

## Contents

Contents .....	2
Table Index.....	4
Figure Index .....	5
0. Revision history .....	6
1. Introduction.....	7
1.1. Reference.....	7
1.2. Terms and abbreviations.....	7
2. The internal TCP/IP stack .....	8
3. One local IP address, single remote server.....	9
3.1. Initialize.....	9
3.2. Establish TCP/UDP session .....	10
3.3. Send/Receive data in transparent mode.....	11
3.4. Send/Receive data in non-transparent mode .....	12
3.5. Close session .....	14
4. One local IP address, multiple remote servers .....	16
4.1. Initialize.....	16
4.2. Establish TCP/UDP session .....	17
4.3. Send/Receive data .....	18
4.4. Close sessions.....	20
5. One local IP address, single remote client .....	22
5.1. Initialize.....	22
5.2. Listen and accept.....	22
5.3. Send/Receive data in transparent mode.....	24
5.4. Send/Receive data in non-transparent mode .....	24
5.5. Close session .....	24
6. One local IP address, multiple remote clients .....	26
6.1. Initialize.....	26
6.2. Listen and accept.....	26
6.3. Send/Receive data .....	27
6.4. Close sessions.....	28
7. Two local IP address.....	30
7.1. Introduction .....	30
7.2. Establish TCP/UDP sessions.....	30
7.3. Send/Receive data .....	32
7.4. Close sessions.....	33
8. Use TCP/IP stack in CMUX enabled.....	35
8.1. Introduction .....	35
8.2. Establish TCP/UDP sessions.....	35
8.3. Send/Receive data .....	36
8.4. Close session .....	37
9. DNS function .....	39
<b>GSM_TCPIP_AN_V1.01</b>	<b>- 2 -</b>

- 9.1. Visit a remote server with domain name.....39
- 9.2. Get IP according to the domain name .....40
- 10. TCP connection maintaining and detection.....42
  - 10.1. Maintain TCP connection.....42
  - 10.2. Detect TCP connection.....44

Quectel  
Confidential

## Table Index

TABLE 1: REFERENCE.....	7
TABLE 2: TERMS AND ABBREVIATIONS .....	7

## Figure Index

FIGURE 1: COMMUNICATE WITH A REMOTE SERVER .....	16
FIGURE 2: COMMUNICATE WITH MULTIPLE SERVERS .....	21
FIGURE 3: VISIT THE SERVER “QUECTEL.3322.ORG:7020” .....	40

## 0. Revision history

Revision	Date	Author	Description of change
1.00	2009-06-27	Colin.HU	Initial
1.01	2009-11-20	Colin.HU	Add the introduction for how to maintain and detect TCP connection

## 1. Introduction

This document introduces how to use the internal TCP/IP stack. Customer can find the powerful function and the apt operation of the internal TCP/IP stack.

This document is subject to change without notice at any time.

### 1.1. Reference

**Table 1: Reference**

SN	Document name	Remark
[1]	M10_ATC_V1.00	M10 AT commands set

### 1.2. Terms and abbreviations

**Table 2: Terms and abbreviations**

Abbreviation	Description
APN	Access Point Network
CMUX	Multiplexer of UART
CSD	Circuit Switched Data
FGCNT	Foreground Context. The internal TCP/IP stack supports to activate two GPRS PDP contexts at the same time and Foreground context is the context controlled by the UART at present.
GPRS	General Packet Radio Service
MUXIP	The function to visit several servers or listen to multiple clients based on the same GPRS/CSD context
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
VIRTUAL_UART_X	Virtual UART for AT command when CMUX is enabled and X is from 1 to 4.



## 2. The internal TCP/IP stack

The internal TCP/IP stack is designed to communicate with one or several TCP/UDP ends. The internal TCP/IP stack provides a series of AT commands to open TCP/UDP sessions and communicate with the other TCP/UDP ends.

The internal TCP/IP stack provides a lot of TCP/UDP applications. It includes the following applications.

- 1) Activate one GPRS/CSD context and communicate with single remote server. Both TCP server and UDP server are supported. This will be introduced in detail in Chapter 3.
- 2) Activate one GPRS/CSD context and communicate with several remote servers. The servers could be TCP servers or UDP servers. And it is supported to visit several TCP servers and several UDP servers at the same time. This will be introduced in detail in Chapter 4.
- 3) Activate one GPRS/CSD context and act as a server. Both TCP server and UDP server are supported. This will be introduced in detail in Chapter 5.
- 4) Activate one GPRS/CSD context and act as sever, listening to several clients. The clients could be TCP clients or UDP clients. This will be introduced in detail in Chapter 6.
- 5) The internal TCP/IP stack also supports to activate two GPRS contexts at the same time. And it is OK to establish TCP/UDP session based on the different GPRS contexts at the same time. This makes it easy to communicate with different TCP/UDP ends based on different GPRS contexts at the same time. This will be introduced in detail in Chapter 7.
- 6) The internal TCP/IP stack also supports to establish TCP/UDP session when CMUX is enabled. It is supported to establish different TCP/UDP sessions on different virtual UARTs at the same time. This will be introduced in detail in Chapter 8.
- 7) The internal TCP/IP stack also supports to establish TCP/UDP sessions with domain name directly. Besides, it provides AT command to get IP address according to a domain name. This will be introduced in detail in Chapter 9.

Some of these applications could be executed at the same time. It is OK to execute application 1, 3, 6, 7 at the same time. And it is OK to execute application 2, 4, 6, 7 at the same time. And it is OK to execute application 5, 6, 7 at the same time.

### 3. One local IP address, single remote server

#### 3.1. Initialize

It is recommended to do some initialization before establishing a TCP/UDP session, such as FGCNT ID, bearer type (CSD or GPRS), whether to enable the function of MUXIP, session mode (transparent or non-transparent), etc.

The command “AT+QIFGCNT” is used to select FGCNT. It is recommended to implement this command before other TCP/UDP operations. The following is the example of this command.

```
AT+QIFGCNT=0 // set the context 0 as FGCNT.
OK // successfully set context 0 as FGCNT.
Or
AT+QIFGCNT=1 // set the context 1 as FGCNT.
OK // successfully set context 1 as FGCNT.
```

The command “AT+QICSGP” is used to select CSD or GPRS as the bearer to establish a TCP/UDP session. And at the same time, it is supported to set the configuration of CSD or GPRS by this command. For example:

- 1) Set bearer type as CSD and the call number is 17201 and the user name for the CSD call is 172 and the password for the CSD call is 172. This command should be implemented as following.

```
AT+QICSGP=0,"17201","172","172" // input through UART.
OK // successfully set the configuration.
```

- 2) Set bearer type as GPRS and the APN is “CMNET” and no user name and password for the APN. This command should be implemented as following.

```
AT+QICSGP=1,"CMNET" // input through UART.
OK // successfully set the configuration.
```

The command “AT+QIMUX” is used to decide whether to enable the function of MUXIP and default to disable the function of MUXIP. Here is to introduce how to visit single remote server, so this command should be implemented as following.

```
AT+QIMUX=0 // disable the function of MUXIP.
OK // successfully implement the command.
```

The command “AT+QIMODE” is used to set the session mode and the default mode is non-transparent mode. The following is the example of this command.

```
AT+QIMODE=0 // set the session mode as non-transparent.
OK // successfully implement the command.
Or
AT+QIMODE=1 // set the session mode as transparent.
```

```
OK // successfully implement the command.
```

The command “AT+QIDNSIP” is used to decide to establish a TCP/UDP session with domain name or IP address and default is IP address. Please refer to the following example.

```
AT+QIDNSIP=0 // use IP address as the address to establish TCP/UDP session.
```

```
OK // successfully implement the command.
```

Or

```
AT+QIDNSIP=1 // use domain name as the address to establish a TCP/UDP session.
```

```
OK // successfully implement the command.
```

If the session mode was set as transparent mode (refer to “AT+QIMODE”), it is recommended to configure some parameters for transparent session by the command “AT+QITCFG”. The following is the detail to use the command “AT+QITCFG”.

```
AT+QITCFG=3,2,512,1 // 3 means to set the retry times to resend as 3, this is a internal process, which doesn't have an obvious affection. 2 means to set wait interval as 200ms, which means system will wait for 200 ms before send the data in the buffer when the data in the buffer is less than 512. 512 means that once the length of the data in the buffer reaches 512, the data will be sent out. 1 means to enable escape data mode with “+++”.
```

```
OK
```

It is strongly recommended to implement the former commands after SIM PIN is unlocked. And all these commands should be implemented in the state IP INITIAL which can be queried by the command “AT+QISTAT”.

### 3.2. Establish TCP/UDP session

After the operation in the chapter 3.1, it is OK to establish a TCP/UDP session now. There are two methods to establish a TCP/UDP session.

1) Implement the following commands one by one to establish a TCP/UDP session.

```
AT+QIREGAPP // register the TCP/IP stack
```

```
OK
```

```
AT+QIACT // activate FGCNT.
```

```
OK
```

```
AT+QILOCIP // query the local IP address.
```

```
10.79.142.227
```

```
AT+QIOPEN="TCP", "124.79.167.121",7007 // visit the remote TCP server. And the
```

```

address of the remote server is an IP
address.

OK

CONNECT OK // successfully connect to the remote TCP
server. If the session mode is transparent
mode, it returns "CONNECT OK" here and
enter data mode.

```

2) Implement the command "AT+QIOPEN" to establish a TCP/UDP session

```

AT+QIOPEN="TCP", "124.79.167.121",7007 // visit the remote TCP server. And the
remote address is an IP address.

OK

CONNECT OK // successfully connect to the remote TCP
server. If the session mode is transparent
mode, it returns "CONNECT OK" here and
enter data mode.

```

### 3.3. Send/Receive data in transparent mode

In transparent mode, UART has two modes, AT command mode and data mode.

In AT command mode, all the input data through UART is considered as AT command and the received data from the remote server is saved in the buffer and can't be output through UART until the UART is switched to data mode.

In data mode, all the input data through UART is considered as the data to send to the remote server and all the received data from the remote server is output through UART. When the session mode is transparent mode, the UART will enter data mode after the TCP/UDP session is established.

The internal TCP/IP stack supports to switch the UART mode between AT command mode and data mode. The method to switch data mode to AT command mode is to input "+++". The interval time between the first "+" and the character before the first "+" **MUST NOT** be less than 500 ms and the interval time between the last "+" and the character next to the last "+" **MUST NOT** be less than 500 ms and the interval time between each "+" **MUST** be less than 20 ms. The method to switch AT command mode to data mode is to implement the command "ATO".

Besides, the internal TCP/IP stack provides hardware method to switch the UART mode from data mode to AT mode. In order to adopt the hardware method, it is necessary to issue the command AT&D1 at first. Then it is supported to change DTR from ON to OFF in data mode to switch the UART mode to AT mode.

The internal TCP/IP stack provides the command “AT+QISACK” to query the total length of the sent data and the total length of the acknowledged data. Of course, it is necessary to switch the UART to AT command mode before use the command to query. The following is an example to use the command.

```

AT+QISACK // query the total size of the data sent and acknowledged.
+QISACK: 1024, 536, 488 // the total size of the data sent is 1024 and the total size of the
// data acknowledged is 536, and the length of the data
// unacknowledged is 488.

OK

```

### 3.4. Send/Receive data in non-transparent mode

In non-transparent mode, it has to use the command “AT+QISEND” to send data to the remote server. The detail to send data by this command is described as following.

Firstly, it is recommended to make sure the command “AT+QISEND” is used to send data to a remote server. This is realized by the command “AT+QISRVC”. The following is the detail operation.

```

AT+QISRVC?
+QISRVC: 1 // “AT+QISEND” is used to send data to a remote server.

OK

```

If the return of the command “AT+QISRVC?” is “+QISRVC: 2”, it is necessary to implement the following command.

```

AT+QISRVC=1 // set “AT+QISEND” for the data to a remote server.

OK

```

Secondly, implement the command “AT+QISEND” to send data. The internal TCP/IP stack supports three methods to send data.

- 1) Use <Ctrl+Z> as the end character to send the input data to the remote server. Please refer to the following operation.

```

AT+QISEND
> // echo from the UART to indicate the following input data is
// considered as data to send.
<data> // input data to send through UART and the maximum length of
// the data is 1460. The data beyond 1460 will be omitted.
<Ctrl+Z> // input <Ctrl+Z> to send data.
SEND OK // the data is sent.

```

In this method, the sending operation can be escaped through inputting <ESC> before sending out.

- 2) Send data with fixed length. In this manner, <Ctrl+Z> is considered as a common character to send. Please refer to the following for the details.

```

AT+QISEND=<length>      // ready to send the data whose length is <length> and the
                          // maximum value of <length> is 1460.
>                          // echo from the UART to indicate the following input data is
                          // considered as data to send.
<data>                   // input data to send through UART and when the data length
                          // reaches to <length>, the input data will be sent to the remote
                          // server.
SEND OK                   // the data is sent.

```

- 3) Send data after a preset timer expired. The following are the detail steps.

```

AT+QIAUTOS=1,<period>   // set the time to wait before send data to the remote server.
                          // <period> is time to wait and the unit is second.
OK
AT+QISEND                // after this command, the preset timer starts.
>                          // echo from the UART to indicate the following input data is
                          // considered as data to send.
<data>                   // input data through UART and the maximum length of the
                          // data is 1460. The data beyond 1460 will be omitted. After the
                          // timer expires, the data will be sent.
SEND OK                   // the data is sent, if the length of the input data is 0, it returns
                          // "SEND FAIL" here.

```

In this method, if input <Ctrl+Z> before the timer expires, the data before <Ctrl+Z> will be sent and the timer stops. If input <Ctrl+Z> before the timer expires, the sending operation will be cancelled.

Thirdly, it is optional to query the total size of the data sent and the data acknowledged by the command "AT+QISACK". The following is an example to use the command "AT+QISACK".

```

AT+QISACK                // query the total size of the data sent and acknowledged.
+QISACK: 1024, 536, 488  // the total size of the data sent is 1024 and the total size of the
                          // data acknowledged is 536, and the length of the data
                          // unacknowledged is 488.
OK

```

Same as the command "AT+QISEND", it is necessary to make sure that the command "AT+QISACK" is used to query the total size about the connection with a remote server by the command "AT+QISRVC".

In default setting, the received data from the server will be output through UART without any information. Besides, it is allowed to add some information at the beginning of the received data controlled by some commands. The detail is described as following.

The command “AT+QISHOWRA” is used to set whether to add the address and port of the remote server. Please refer to [1] for the detail style and position of the address and port of the remote server. The following is the example to use the command.

```
AT+QISHOWRA=1      // add the address and port of the remote server before the received
                    data.
OK
Or
AT+QISHOWRA=0      // do not add the address or port of the remote server.
OK
```

The command “AT+QISHOWLA” is used to set whether to add the local address. Please refer to [1] for the detail style and position of the local address. The following is the example to use this command.

```
AT+QISHOWLA=1      // add the local address before the received data.
OK
Or
AT+QISHOWLA=0      // do not add the local address.
OK
```

The command “AT+QIHEAD” is used to set whether to add the header information (data length and transmission layer type if necessary). Please refer to [1] for the detail style and position to add the header information. The following is the detail to use the command.

```
AT+QIHEAD=1        // add the header information before the received data.
OK
Or
AT+QIHEAD=0        // do not add the header information.
OK
```

The command “AT+QISHOWPT” is used to set whether to add the transmission layer protocol type (TCP or UDP) at the end of header information. Please refer to Document [1] for the detail style and position to add the transmission layer protocol type (TCP or UDP). The following is the example to use the command.

```
AT+QISHOWPT=1      // add the transmission layer protocol type.
OK
Or
AT+QISHOWPT=0      // do not add the transmission layer protocol type.
OK
```

### 3.5. Close session

It is supported to close only the TCP/UDP session by the command “AT+QICLOSE” and it is also supported to deactivate GPRS/CSD context at the same time by the command “AT+QIDEACT”.

The following are the detail steps to close TCP/UDP session.

Firstly, same as the check-up before send data, it is recommended to make sure the command “AT+QICLOSE” is used for the session with a remote server.

```
AT+QISRVC?  
+QISRVC: 1 // “AT+QICLOSE” is used to close the session with a remote server.  
  
OK
```

If the return of the command “AT+QISRVC?” is “+QISRVC: 2”, it is necessary to implement the following command.

```
AT+QISRVC=1 // set “AT+QICLOSE” for the session with a remote server.  
  
OK
```

It is unnecessary to implement the former operation if use the command “AT+QIDEACT” to close the session and deactivate GPRS/CSD context.

Secondly, close the current session by the command “AT+QICLOSE” or “AT+QIDEACT”.

```
AT+QICLOSE // close the session with the remote server.  
CLOSE OK  
Or  
AT+QIDEACT // close the session with the remote server and then deactivate  
GPRS/CSD context.  
DEACT OK
```

It is recommended to close all sessions by the command “AT+QICLOSE” before deactivate GPRS/CSD context by the command “AT+QIDEACT”. It is also recommended to close all sessions before deactivate GPRS/CSD context in other applications introduced later.

The following figure gives the process to communicate with the remote TCP server “116.226.41.43:7020”.



```

ATI
Quectel_Ltd

Quectel_M10
Revision:M10R04A02M32_SST

OK
AT+QIFGCNT=0
OK
AT+QICSGP=1,"CMNET"
OK
AT+QIOPEN="TCP","116.226.41.43",7020
OK

CONNECT OK
AT+QISEND
> Welcome to use Quectel's module!
SEND OK
AT+QISHOWRA=1
OK

RECV FROM:116.226.41.43:7020
Quectel's module is great!AT+QICLOSE
CLOSE OK
AT+QIDEACT
DEACT OK
-

```

Figure 1: communicate with a remote server

## 4. One local IP address, multiple remote servers

### 4.1. Initialize

The commands to initialize the internal TCP/IP stack to work in this mode are almost same as the commands introduced in the chapter 3.1. The detail steps are described as following.

```

AT+QIFGCNT=0           // set context 0 as the FGCNT.
OK

AT+QICSGP=1,"CMNET"   // set the APN as "CMNET".
OK

AT+QIMUX=1            // enable MUXIP. This is different from chapter 3.1.
OK                    // successfully implement the command.

AT+QIMODE=0           // set the session mode as non-transparent. It doesn't support
                        // transparent mode if MUXIP is enabled.
OK                    // successfully implement the command.

AT+QIDNSIP=0          // use IP address as the address to establish TCP/UDP session.
OK                    // successfully implement the command.

```

Same as the description in the chapter 3.1, it is recommended to implement the former commands when SIM PIN is unlocked.

## 4.2. Establish TCP/UDP session

After the operation in the chapter 4.1, it is OK to establish a TCP/UDP session now. There are two methods to establish a TCP/UDP session.

1) Implement the following commands one by one to establish a TCP/UDP session.

```

AT+QIREGAPP // register the TCP/IP stack
OK

AT+QIACT // activate FGCNT.
OK

AT+QILOCIP // query the local IP address.
10.79.142.227

AT+QIOPEN=0,"TCP", "124.79.167.121",7007 // use the profile 0 to visit the remote TCP
server. And the remote address is an IP
address. Notice: there is an index "0" at the
beginning of all parameters.

OK

0, CONNECT OK // use profile 0 successfully to connect to
the remote TCP server.

```

2) Implement the command "AT+QIOPEN" to establish a TCP/UDP session

```

AT+QIOPEN=0,"TCP", "124.79.167.121",7007 // use the profile 0 to visit the remote TCP
server. And the remote address is an IP
address.

OK

0, CONNECT OK // use the profile 0 successfully to connect to
the remote TCP server.

```

After profile 0 was successfully to used to establish a TCP session, it is OK to establish other TCP/UDP session on other profiles, such as 1, 2, 3, 4 and 5. And the method to use other profiles is same as the method to use the profile 0.

### 4.3. Send/Receive data

When “AT+QIMUX” is set as 1, the internal TCP/IP stack doesn’t support transparent mode, so it is necessary to use the command “AT+QISEND” to send data to a remote server. The detail to send data by the command is described as following.

Firstly, it is recommended to make sure the “AT+QISEND” is used to send data to a remote server. This is realized by the command “AT+QISRVC”. The following is the detail operation.

```
AT+QISRVC?
+QISRVC: 1 // AT+QISEND is used to send data to a remote server.
OK
```

If the return of the command “AT+QISRVC?” is “+QISRVC: 2”, it is necessary to implement the following command.

```
AT+QISRVC=1 // set “AT+QISEND” for the data to a remote server.
OK
```

Secondly, implement the command “AT+QISEND” to send data. The internal TCP/IP stack supports three manners to send data.

- 1) Use <Ctrl+Z> as the end character to send the input data to the remote server. The following are the detail steps.

```
AT+QISEND=0 // 0 means to use profile 0 to send data.
> // echo from the UART to indicate the following input
// data is considered as data to send.
<data> // input data to send through UART and the maximum
// length of the data is 1460. The data beyond 1460 will be
// omitted.
<Ctrl+Z> // input <Ctrl+Z> to send data.
SEND OK // the data is sent.
```

- 2) Send fixed length data. In this manner, <Ctrl+Z> is considered as a common character to send. The following are the detail steps.

```
AT+QISEND=0,<length> // ready to use profile 0 to send the data whose length is
// <length> and the maximum value of <length> is 1460
> // echo from the UART to indicate the following input
// data is considered as data to send.
<data> // input data to send through UART and when the data
// length reaches to <length>, the input data will be sent to
// the remote server through the profile 0.
SEND OK // the data is sent.
```

- 3) Send data after a preset timer expired. The following are the detail steps.

```

AT+QIAUTOS=1,<period> // set the time to wait before send data to the remote
                           server.
OK

AT+QISEND=0 // after this command, the the preset timer starts.
> // echo from the UART to indicate the following input
  // data is considered as data to send.
<data> // input data to send through UART and the maximum
        // length of the data is 1460. The data beyond 1460 will be
        // omitted. After the timer expires, the data will be sent.
SEND OK // the data is sent, if the length of the input data is 0, it
          // returns "SEND FAIL" here.

```

In this manner, if input <Ctrl+Z> before the timer expires, the data before <Ctrl+Z> will be sent and the timer stops.

Same as the description in the chapter 3.4, all the former manners can be escaped through inputting <ESC>.

Thirdly, it is optional to query the total size of the data sent and the data acknowledged by the command "AT+QISACK". The usage of the command "AT+QISACK" is a little different from the chapter 3.4. It is necessary to specify the profile index here and the following is an example to query the total size of acknowledged data.

```

AT+QISACK=0 // query the total size of the data sent and acknowledged
              // through profile 0.
+QISACK: 1024, 536, 488 // the total size of the data sent is 1024 and the total size of
                          // the data acknowledged is 536, and the length of the data
                          // unacknowledged is 488.
OK

```

Same as the command "AT+QISEND", it is necessary to make sure that the command "AT+QISACK" is used to query the total size about the connection with a remote server by the command "AT+QISRVC".

In default setting, the received data from the server will be output through UART with only "+RECEIVE: <index>, <length>" (<index> is the index of the profile to receive data and <length> is the length of the received data). Besides, it is allowed to add some other information at the beginning of the received data controlled by the commands "AT+QISHOWRA", "AT+QISHOWLA", "AT+QIHEAD" and "AT+QISHOWPT". About the usage of these commands, please refer to the chapter 3.4

#### 4.4. Close sessions

When a profile is not necessary to communicate with the remote server any more, it is OK to close the session based on the profile. It is supported to use the command “AT+QICLOSE=<n>” to close the session based on the given profile and it is also supported to close all sessions and then deactivate GPRS/CSD context by the command “AT+QIDEACT” here. The following are the detail steps to close the given session.

Firstly, same as the check-up before send data, it is recommended to make sure the command “AT+QICLOSE” is used for the session with a remote server.

```
AT+QISRVC?  
+QISRVC: 1 // “AT+QICLOSE” is used to close the session with a remote server.  
  
OK
```

If the return of the command “AT+QISRVC?” is “+QISRVC: 2”, it is necessary to implement the following command.

```
AT+QISRVC=1 // set “AT+QICLOSE” for the session with a remote server.  
  
OK
```

It is unnecessary to implement this step when use the command “AT+QIDEACT” to close all sessions and deactivate GPRS/CSD context.

Secondly, close the session based on the profile 0 by the command “AT+QICLOSE” or close all sessions and then deactivate GPRS/CSD context by “AT+QIDEACT”.

```
AT+QICLOSE=0 // close the session based on the profile 0.
```

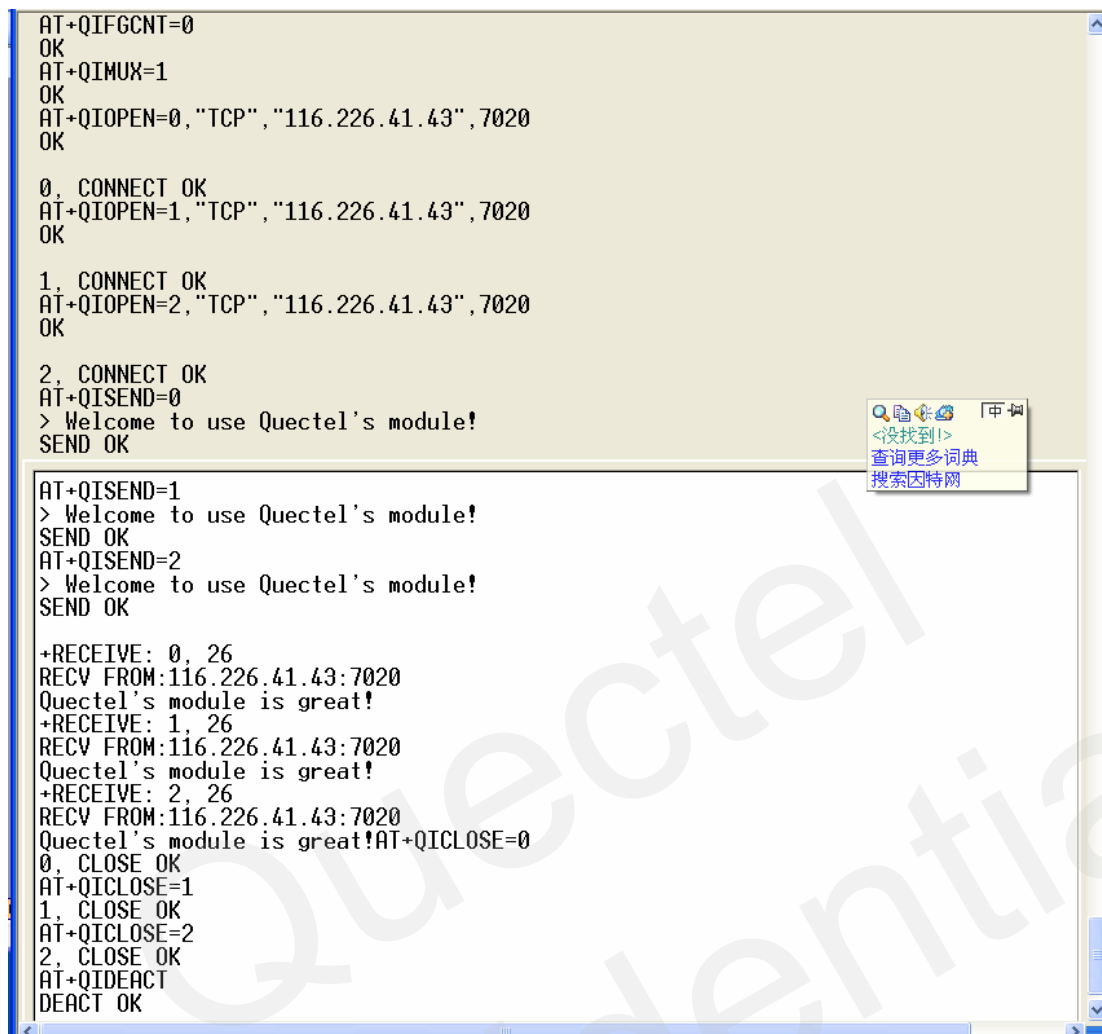
**0, CLOSE OK**

Or

```
AT+QIDEACT // close all sessions and deactivate GPRS/CSD context.
```

**DEACT OK**

The following figure is an example to visit several remote servers on several profiles.



```
AT+QIFGCNT=0
OK
AT+QIMUX=1
OK
AT+QIOPEN=0,"TCP","116.226.41.43",7020
OK

0, CONNECT OK
AT+QIOPEN=1,"TCP","116.226.41.43",7020
OK

1, CONNECT OK
AT+QIOPEN=2,"TCP","116.226.41.43",7020
OK

2, CONNECT OK
AT+QISEND=0
> Welcome to use Quectel's module!
SEND OK

AT+QISEND=1
> Welcome to use Quectel's module!
SEND OK
AT+QISEND=2
> Welcome to use Quectel's module!
SEND OK

+RECEIVE: 0, 26
RCV FROM:116.226.41.43:7020
Quectel's module is great!
+RECEIVE: 1, 26
RCV FROM:116.226.41.43:7020
Quectel's module is great!
+RECEIVE: 2, 26
RCV FROM:116.226.41.43:7020
Quectel's module is great!AT+QICLOSE=0
0, CLOSE OK
AT+QICLOSE=1
1, CLOSE OK
AT+QICLOSE=2
2, CLOSE OK
AT+QIDEACT
DEACT OK
```

Figure 2: communicate with multiple servers

## 5. One local IP address, single remote client

### 5.1. Initialize

Besides to visit a remote server, the internal TCP/IP stack also supports to listen to other client. The commands to initialize the internal TCP/IP stack here are almost same as the commands introduced in the chapter 3.1. The detail steps are described as following.

```

AT+QIFGCNT=0           // set context 0 as the FGCNT.
OK

AT+QICSGP=1,"CMNET"   // set the APN as "CMNET".
OK

AT+QIMUX=0            // disable MUXIP. So the module can listen to only one client
OK                    // successfully implement the command.

AT+QIMODE=0           // set the session mode as non-transparent. It also supports
                        // transparent mode when listen to only one client.
OK                    // successfully implement the command.

```

Same as the description in the chapter 3.1, it is recommended to implement the former commands when SIM PIN is unlocked.

### 5.2. Listen and accept

In order to listen to a client, it is necessary to enter listening state at first. The internal TCP/IP stack provides two methods to enter listening state.

- 1) Implement the following commands one by one to enter listening state.

```

AT+QIREGAPP
OK

AT+QIACT             // activate FGCNT.
OK

AT+QILOCIP          // query the local IP address.
10.79.142.227

AT+QISERVER        // enter listening state to listen a TCP client to visit and it is also
                        // supported to implement "AT+QISERVER=1" to listen a UDP

```

```

client.
OK

SERVER OK // successfully enter listening state.

```

2) Implement the command “AT+QISERVER” to enter listening state directly

```

AT+QISERVER // enter listening state to listen TCP client to visit and it is
supported to implement “AT+QISERVER=1” to listen UDP
client.

OK

SERVER OK // successfully enter listening state.

```

After enter listening state, the module can accept a client to visit. The following are two examples separately for TCP server and UDP server.

1) The module is a TCP server to listen to a TCP client.

The TCP client whose address is “10.79.141.26” tries to visit the module. After the module listens to the visit and accepts the visit successfully, it will output the following string through UART.

```

REMOTE IP: 10.79.141.26
[CONNECT] // it will output “CONNECT” here in transparent mode.

```

Now, it is OK to send data to and receive data from the client “10.79.141.26”.

2) The module is a UDP server to listen to a UDP client

The UDP client whose address is “10.79.141.26” tries to visit the module. There is a little difference from the last example for TCP server. It doesn’t output the following string until receive the first packet from the UDP client “10.79.141.26”.

```

REMOTE IP: 10.79.141.26
[CONNECT] // it will output “CONNECT” here in transparent mode.
<data> // data received from the UDP client “10.79.141.26”.

```

Now it is ready to send data to the UDP client “10.79.141.26”. Of course, it is also ready to receive data from the UDP client.

The internal TCP/IP stack supports to query the listening state of the module by the command “AT+QISERVER?”. The following is an example to query the listening state.

```

AT+QISERVER? // query the server state.
+QISERVER: 1, 1 // the first 1 means the module is in the listening state and if
here is 0, it means the module is not in the listening state. The
second 1 means the number of the clients connected is 1.

OK

```



### 5.3. Send/Receive data in transparent mode

After outputs “CONNECT”, the UART enters data mode immediately. Then it is same as the introduction in the chapter 3.3 to communicate with the remote client.

### 5.4. Send/Receive data in non-transparent mode

It is almost same as the introduction in the chapter 3.4 to send data to the remote client. The only difference is to make sure that the command “AT+QISEND” is used to send data to a remote client but not a remote server. The following are the detail steps to send data.

Firstly, it is recommended to make sure the “AT+QISEND” is used to send data to a remote client. This is realized by the command “AT+QISRVC”. The following is the detail operation.

```
AT+QISRVC?
+QISRVC: 2           // “AT+QISEND” is used to send data to a remote client.
OK
```

If the return of the command “AT+QISRVC?” is “+QISRVC: 1”, it is necessary to implement the following command.

```
AT+QISRVC=2         // set AT+QISEND for the data to a remote client.
OK
```

Secondly, implement the command “AT+QISEND” to send data. Please refer to 2.4 for the details of the usage of “AT+QISEND”.

Thirdly, it is optional to use the command “AT+QISACK” to query the total size of the data sent and the data acknowledged. And the usage of the command here is same as the chapter 3.4. Same as the command “AT+QISEND”, it is necessary to make sure that the command “AT+QISACK” is used to query the total size about the connection with a remote client by the command “AT+QISRVC”.

Same as the introduction in the chapter 3.4, in default settings, the received data will be output without any other information. And the operations in the chapter 3.4 to add some information at the beginning of the received data are also useful here.

### 5.5. Close session

It is supported to close only the TCP/UDP session by the command “AT+QICLOSE” and it is also supported to deactivate GPRS/CSD context at the same time by the command “AT+QIDEACT”. The following are the detail steps to close TCP/UDP session.

Firstly, same as the check-up before send data, it is recommended to make sure the command

“AT+QICLOSE” is used for the session with a remote client.

```
AT+QISRVC?
```

```
+QISRVC: 2 // “AT+QICLOSE” is used to close the session with a remote client.
```

```
OK
```

If the return of the command “AT+QISRVC?” is “+QISRVC: 1”, it is necessary to implement the following command.

```
AT+QISRVC=2 // set “AT+QICLOSE” for the session with a remote client.
```

```
OK
```

It is unnecessary to implement this step if use the command “AT+QIDEACT” to close the session and deactivate GPRS/CSD context.

Secondly, close the current session by the command “AT+QICLOSE” or “AT+QIDEACT”.

```
AT+QICLOSE // close the session with the remote client.
```

```
CLOSE OK
```

Or

```
AT+QIDEACT // close the session with the remote client and then deactivate  
GPRS/CSD context.
```

```
DEACT OK
```

If the session is closed by the command “AT+QICLOSE”, the module is still in the listening state, i.e. it is OK to accept other client to visit again. And it will escape listening state after implement “AT+QICLOSE” when no client connects to the module.

## 6. One local IP address, multiple remote clients

### 6.1. Initialize

Chapter 5 introduces the function to listen to single client to visit. The internal TCP/IP stack of Quectel also supports to listen to multiple clients, up to 5 clients. Just like listen to single client, it is also recommended to do some initialization here. The following are the detail steps to initialize.

```

AT+QIFGCNT=0           // set context 0 as the FGCNT.
OK

AT+QICSGP=1,"CMNET"   // set the APN as "CMNET".
OK

AT+QIMUX=1            // enable MUXIP. Then the module can listen to multiple
                      // clients
OK                    // successfully implement the command.

AT+QIMODE=0           // set the session mode as non-transparent. It doesn't support
                      // transparent mode if MUXIP is enabled.
OK                    // successfully implement the command.

```

Same as the description in the chapter 3.1, it is recommended to implement the former commands when SIM PIN is unlocked.

### 6.2. Listen and accept

In order to listen to clients, it is necessary to enter listening state at first. The internal TCP/IP stack provides two methods to enter listening state.

- 1) Implement the following commands one by one to enter listening state.

```

AT+QIREGAPP
OK

AT+QIACT              // activate FGCNT.
OK

AT+QILOCIP           // query the local IP address.
10.79.142.227

AT+QISERVER=0,3      // 0 means to set the module as a TCP server and if here is 1, it

```

means to set the module as a UDP server. **3** is the maximum clients to listen, i.e. after the number of clients reaches 3, no more other client can visit the module.

**OK**

**SERVER OK** // successfully enter listening state.

2) Implement the command “AT+QISERVER” to enter listening state directly

**AT+QISERVER=0,3** // the meaning of each parameters is same as former.

**OK**

**SERVER OK** // successfully enter listening state.

After enter listening state, the module can accept clients. The following gives two examples separately for TCP server and UDP server.

1) The module is used as a TCP server to listen to TCP clients.

The TCP client whose address is “10.79.141.26” tries to visit the module. After the module listen to the visit and accept the visit successfully, it will output the following string through UART.

**0, REMOTE IP: 10.79.141.26** // profile 0 is took up to establish the session with the client “10.79.141.26”.

Now, it is OK to send data to or receive data from the client “10.79.141.26”.

2) The module is used as a UDP server to listen to UDP clients

The UDP client whose address is “10.79.141.26” tries to visit the module. There is a little difference between TCP and UDP. Only after the module receive the first packet from the UDP client “10.79.141.26”, it will output the following string through UART.

**0, REMOTE IP: 10.79.141.26** // profile 0 is took up to establish the session with the client “10.79.141.26”.

**<data>** // data received from the UDP client “10.79.141.26”.

Now it is OK to send data to the UDP client “10.79.141.26”. Of course, it is still OK to receive data.

Same as the introduction in the chapter 5.2, it is supported to query the listening state of the module by the command “AT+QISERVER?”.

### 6.3. Send/Receive data

When “AT+QIMUX” is set as 1, the system doesn’t support transparent mode, so it is necessary to use the command “AT+QISEND” to send data to a remote client. The detail to send data by the command is described as following.

Firstly, it is recommended to make sure the “AT+QISEND” is used to send data to a remote client.

This is realized by the command “AT+QISRVC”. The following is the detail operation.

```
AT+QISRVC?
```

```
+QISRVC: 2 // “AT+QISEND” is used to send data to s remote client.
```

```
OK
```

If the return of the command “AT+QISRVC?” is “+QISRVC: 1”, it is necessary to implement the following command.

```
AT+QISRVC=2
```

```
// set “AT+QISEND” for the data to a remote client.
```

```
OK
```

Secondly, implement the command “AT+QISEND” to send data. Please refer to 3.3 for the details about how to use “AT+QISEND”.

Thirdly, it is optional to use the command “AT+QISACK” to query the total size of the data sent and the data acknowledged. And the usage of the command here is same as the chapter 4.3.

Same as the command “AT+QISEND”, it is necessary to make sure that the command “AT+QISACK” is used to query the total size about the connection with a remote client by the command “AT+QISRVC”.

In default setting, the received data from the server will be output through UART with only “+RECEIVE: <index>, <length>” (<index> is the index of the profile to receive the data and <length> is the length of the received data). Besides, it is allowed to add some other information at the beginning of the received data controlled by the commands “AT+QISHOWRA”, “AT+QISHOWLA”, “AT+QIHEAD” and “AT+QISHOWPT”. About the usage of these commands, please refer to the chapter 3.4.

## 6.4. Close sessions

When a profile is not necessary to communicate with the remote server any more, it is OK to close the session based on the profile. It is supported to use the command “AT+QICLOSE=<n>” to close the session based on the given profile and it is also supported to close all sessions and then deactivate GPRS/CSD context by the command “AT+QIDEACT” here. The following are the detail steps to close the given session.

Firstly, same as the check-up before send data, it is recommended to make sure the command “AT+QICLOSE” is used for the session with a remote client.

```
AT+QISRVC?
```

```
+QISRVC: 2 // “AT+QICLOSE” is used to close the session with a remote client.
```

```
OK
```

If the return is “+QISRVC: 1”, it is necessary to implement the following command.

```
AT+QISRVC=2          // set "AT+QICLOSE" for the session with a remote client.  
OK
```

It is unnecessary to implement this step when use the command "AT+QIDEACT" to close all sessions and deactivate GPRS/CSD context.

Secondly, close the session based on the profile 0 by the command "AT+QICLOSE" or close all sessions and deactivate GPRS/CSD context by "AT+QIDEACT".

```
AT+QICLOSE=0        // close the session based on the profile 0.  
0, CLOSE OK  
Or  
AT+QIDEACT          // close all sessions and deactivate GPRS/CSD context.  
DEACT OK
```

The internal TCP/IP stack also supports to escape listening state and keep GPRS/CSD context activated by the command "AT+QICLOSE". The following is the detail.

```
AT+QICLOSE          // try to close all sessions and escape listening state no matter how  
                    // many clients has visited to the module.  
CLOSE OK           // successfully close all the sessions and escape listening state.
```

## 7. Two local IP address

### 7.1. Introduction

The internal TCP/IP stack of Quectel supports to activate two GPRS contexts at the same time. In another word, the internal TCP/IP stack supports to establish TCP/UDP sessions based on two different activated GPRS contexts. For example, establish a TCP/UDP session based on the GPRS context whose APN is “CMNET” and it is OK to establish another TCP/UDP session based on another GPRS context whose APN is “CMWAP”. As the introduction in Chapter 3.1, the command “AT+QIFGCNT” is used to select a context as FGCNT. The purpose of the command is to select a GPRS context to control.

Notice: The two contexts activated at the same time should be both GPRS.

### 7.2. Establish TCP/UDP sessions

The following is an example to activate two GPRS contexts one by one.

1) Establish a TCP session to visit a remote server based on the context 0

Firstly, use the command “AT+QIFGCNT” to select context 0 as FGCNT as following.

```
AT+QIFGCNT=0           // set context 0 as FGCNT, then all the operations in the UART
                        are for the context 0.
OK
```

Secondly, initialize some configuration as the former chapters.

```
AT+QICSGP=1,"CMNET"   // set the APN as "CMNET".
OK

AT+QIMUX=0            // disable MUXIP, that means it visits only one remote server
                        based on the context 0. It also supports to enable MUXIP here.
OK

AT+QIMODE=0           // set the session mode as non-transparent. It doesn't support
                        transparent mode if try to activate two GPRS contexts on the
                        same UART.
OK

AT+QIDNSIP=0          // use IP address as the address to establish TCP/UDP session
OK                    // successfully implement the command.

AT+QIDNSIP=0          // use IP address as the address to establish TCP/UDP session
OK                    // successfully implement the command.
```

Thirdly, activated the context 0 and establish a session based on the context 0.

Here takes “AT+QIOPEN” as an example to establish a session with a remote server. Of course, it supports to establish a session by the steps introduced in Chapter 4.2 or enter listening state as the

introduction in Chapter 5.2 and Chapter 6.2.

```

AT+QIOPEN="TCP", "124.79.167.121",7007 // visit the remote TCP server
                                         "124.79.167.121:7007".
OK

CONNECT OK // successfully connect to the remote TCP
            server.

AT+QILOCIP // query the local IP address.
10.79.141.224 // the local IP for the context 0 is
              "10.79.141.224".

```

Now, it is successful to establish a TCP session based on the context 0.

2) Establish a TCP server based on context 1

Firstly, use the command "AT+QIFGCNT" to select context 1 as FGCNT as following.

```

AT+QIFGCNT=1 // set context 1 as FGCNT, and then all the operations
              in the UART are for the context 1.
OK

```

Secondly, initialize some configuration as the former chapters.

```

AT+QICSGP=1,"CMNET" // set the APN as "CMNET".
OK

AT+QIMUX=1 // enable MUXIP, that means it listens to multiple
            remote clients based on the context 1.
OK

AT+QIMODE=0 // set the session mode as non-transparent. It doesn't
             support transparent here.
OK // successfully implement the command.

```

Thirdly, activated the context 1 and enter listening state.

```

AT+QISERVER=0,3 // 0 means to enter the listening state to listen to TCP clients
                and if here is 1, it means to listen to UDP clients. 3 is the
                maximum clients to listen, i.e. after the number of clients
                reaches 3, no more other client can visit the module.
OK

SERVER OK // successfully enter listening state.

AT+QILOCIP // query the local IP address.
10.79.140.214 // the local IP for the context 1 is "10.79.140.214".

```

The TCP client whose address is "10.79.141.26" tries to visit the module. After the module listens to the visit and accepts the visit successfully, it will output the following string through UART.

```

0, REMOTE IP: 10.79.141.26 // profile 0 is took up to establish the session with the client

```



“10.79.141.26”.

### 7.3. Send/Receive data

Because there are two GPRS contexts are activated, so it is necessary to make sure which context to send data based on. The command “AT+QIFGCNT” is designed for the purpose. The following is an example to send data by the session based on context 0.

Firstly, query FGCNT. If it is the context 0, then it is OK to send data.

```

AT+QIFGCNT?           // query FGCNT.
+QIFGCNT: 1, 0       // FGCNT is the context 1, so it is necessary to switch
                       FGCNT to the context 0. And the second parameter 0
                       means the context 1 is controlled by VIRTUAL_UART_1
                       and it is very useful when CMUX enabled.

OK

AT+QIFGCNT=0         // set the context 0 as FGCNT.
OK

```

Secondly, send data by the command “AT+QISEND”. The detail steps to send data are same as the introduction in the chapter 3.4.

Then it is necessary to switch FGCNT to the context 1 to send data by the session based on the context 1.

```

AT+QIFGCNT=1         // set the context 1 as FGCNT.
OK

```

Secondly, send data by the command “AT+QISEND”. The detail steps to send data is same as the introduction in the chapter 6.3.

Thirdly, it is optional to use the command “AT+QISACK” to query the total size of the data sent and the data acknowledged. And the usage of the command here is same as Chapter 4.3.

Same as the command “AT+QISEND”, it is necessary to make sure that the command “AT+QISACK” is used to query the total size about the connection with a remote client by the command “AT+QISRVC”.

In order to distinguish which context the received data belongs to, the internal TCP/IP stack provides the command “AT+QISHOWLA” to control whether to add the local IP address before the received data. It is recommended to implement the following command when activate two GPRS context at the same time.

```

AT+QISHOWLA=1       // add local IP address before the received data.
OK

```

About other information, please refer to Chapter 3.4.

## 7.4. Close sessions

Firstly, close the session based on the context 0, it is necessary to make sure that the context 0 is FGCNT before “AT+QICLOSE” as the introduction in Chapter 7.3.

```
AT+QIFGCNT?           // query FGCNT.
+QIFGCNT: 1, 0

OK

AT+QIFGCNT=0         // set the context 0 as FGCNT.
OK
```

The following operation to close the session is same as the introduction in Chapter 3.5.

```
AT+QISRVC?
+QISRVC: 1           // “AT+QICLOSE” is used to close the session with a remote server.

OK
```

If the return of the command “AT+QISRVC?” is “+QISRVC: 2”, it is necessary to implement the following command.

```
AT+QISRVC=1         // set “AT+QICLOSE” for the session with a remote server.
OK
```

It is unnecessary to implement this step if use the command “AT+QIDEACT” to close the session and deactivate GPRS/CSD context.

```
AT+QICLOSE          // close the session with the remote server.
CLOSE OK
Or
AT+QIDEACT          // close the session the remote server and then deactivate
                    // GPRS/CSD context.
DEACT OK
```

Secondly, close the session based on the context 1, it is necessary to set FGCNT as the context 1 at first.

```
AT+QIFGCNT=1       // set the context 1 as FGCNT.
OK
```

The following operation to close the session is the same as the introduction in Chapter 6.4.

```
AT+QISRVC?
+QISRVC: 2           // “AT+QICLOSE” is used to close the session with a remote client.
```

**OK**

If the return is “+QISRVC: 1”, it is necessary to implement the following command.

```
AT+QISRVC=2 // set AT+QICLOSE for the session with a remote client.
```

**OK**

It is unnecessary to implement this step when use the command “AT+QIDEACT” to close the session and deactivate GPRS/CSD context.

```
AT+QICLOSE=0 // close the session based on the profile 0.
```

**0, CLOSE OK**

Or

```
AT+QIDEACT // close all sessions and then deactivate GPRS/CSD context.
```

**DEACT OK**

## 8. Use TCP/IP stack in CMUX enabled

### 8.1. Introduction

The internal TCP/IP stack of Quectel supports to establish TCP/UDP session when CMUX is enabled. But if a context has been controlled by some virtual UART and the state of the context is not IP INITIAL, then no other virtual UART can control the context unless the state of the context changes back to IP INITIAL. The internal TCP/IP stack supports to activate two GPRS contexts on a same virtual UART or two different virtual UARTs. The operation to establish TCP/UDP sessions on a same virtual UART is same as the introduction in the former chapters. So, this document doesn't introduce this here and it introduces how to establish TCP/UDP sessions on two different virtual UARTs following.

### 8.2. Establish TCP/UDP sessions

The following gives an example to activate two GPRS contexts in different virtual UART.

- 1) Establish a TCP server based on the context 0 on VIRTUAL\_UART\_1. Firstly, implement the command "AT+QIFGCNT=0" to select context 0 as FGCNT through VIRTUAL\_UART\_1 as following.

```
AT+QIFGCNT=0           // set context 0 as FGCNT, then all the operations for TCP/IP in
                        // VIRTUAL_UART_1 are for the context 0.
OK
```

Secondly, initialize some configuration as the former chapters.

```
AT+QICSGP=1,"CMNET"   // set the APN as "CMNET".
OK

AT+QIMUX=0            // disable MUXIP, that means visiting only one remote server
                        // based on the context 0. It also supports to enable MUXIP here.
OK

AT+QIMODE=0           // set the session mode as non-transparent. It also supports
                        // transparent mode here when QIMUX was set as 0.
OK

AT+QIDNSIP=0          // use IP address as the address to establish TCP/UDP session.
OK                    // successfully implement the command.
```

Thirdly, activate the context 0 and establish a TCP server.

```
AT+QISERVER           // enter listening state to listen TCP client to visit.
OK

SERVER OK            // successfully enter listening state.
```

```

AT+QILOCIP // query the local IP address.
10.79.140.214 // the local IP for the context 1 is "10.79.140.214".

```

Now, other TCP client can visit the TCP server, and all the commands implemented formerly are through VIRTUAL\_UART\_1.

- 2) Establish a TCP session to visit the server on VIRTUAL\_UART\_1 based on context 1 on VIRTUAL\_UART\_2

Firstly, implement the command "AT+QIFGCNT" to select context 1 as FGCNT through VIRTUAL\_UART\_2 as following.

```

AT+QIFGCNT=1 // set context 1 as FGCNT, then all the operations in the UART
                // are for the context 1.
OK // if it returns ERROR here, it means the context 1 has been
      // controlled by some other virtual UART and the state is not IP
      // INITIAL. Then it is not allowed to control the context 1
      // through VIRTUAL_UART_2 until the state of the context 1
      // changes to IP INITIAL.

```

Secondly, initialize some configuration as the former chapters.

```

AT+QICSGP=1,"CMNET" // set the APN as "CMNET".
OK
AT+QIMUX=0 // disable MUXIP.
OK
AT+QIMODE=0 // set the session mode as non-transparent.
OK // successfully implement the command.

```

Thirdly, activated the context 1 and visit the server on VIRTUAL\_UART\_1.

```

AT+QIOPEN="TCP", "10.79.140.214",2020 // visit the TCP server "10.79.140.214:2020".
OK
CONNECT OK // successfully connect to the remote TCP
              // server.

```

At this time, VIRTUAL\_UART\_1 will output the following string.

```

REMOTE IP: 10.79.141.126 // "10.79.141.126" is the IP address of the
                          // context 1.

```

All the commands implemented formerly are through VIRTUAL\_UART\_2.

### 8.3. Send/Receive data

Firstly, send data from VIRTUAL\_UART\_1 to VIRTUAL\_UART\_2 by the session between

VIRTUAL\_UART\_1 and VIRTUAL\_UART\_2 established formerly.

Implement the following commands through VIRTUAL\_UART\_1.

```
AT+QISRVC=2           // make sure "AT+QISEND" is used to send data to a remote client.
OK
```

```
AT+QISEND             // send data without fixed length.
>Hello, welcome to use Quectel's module<Ctrl+Z>
SEND OK
```

Then VIRTUAL\_UART\_2 will output the received data as following.

```
Hello, welcome to use Quectel's module
```

Secondly, send data from VIRTUAL\_UART\_2 to VIRTUAL\_UART\_1 through the session.

Implement the following commands through VIRTUAL\_UART\_2

```
AT+QISRVC=1           // make sure "AT+QISEND" is used to send data to a remote server.
OK
```

```
AT+QISEND=26          // send data with fixed length 26.
>Quectel's module is great!
SEND OK
```

Then VIRTUAL\_UART\_2 will output the received data as following.

```
Quectel's module is great!
```

## 8.4. Close session

1) Close the session based on the context 0 through VIRTUAL\_UART\_1.

```
AT+QISRVC?
+QISRVC: 2           // "AT+QICLOSE" is used to close the session with a remote client.
OK
```

If the return is "+QISRVC: 1", it is necessary to implement the following command.

```
AT+QISRVC=2           // set "AT+QICLOSE" for the session with a remote client.
OK
```

It is unnecessary to implement this step when use the command "AT+QIDEACT" to close the session and deactivate GPRS/CSD context.

```
AT+QICLOSE            // close the session with the client "10.79.141.126".
CLOSE OK
```

Or

```
AT+QIDEACT            // close the session and then deactivate GPRS context.
```

**DEACT OK**

After these operations, VIRTUAL\_UART\_2 will output the following string.

**CLOSED** // The session has been closed by the remote server.

All the commands implemented formerly should be through VIRTUAL\_UART\_1.

2) Deactivate the GPRS context 1 on VIRTUAL\_UART\_2

**AT+QIDEACT** // close all sessions and then deactivate GPRS context.

**DEACT OK**

## 9. DNS function

### 9.1. Visit a remote server with domain name

Besides to visit a remote server with its IP address, the internal TCP/IP stack also supports to visit a remote server with its domain name. The following is an example to visit “www.quectel.com” directly.

Firstly, initialize some configuration as the former chapters.

```

AT+QIFGCNT=0           // set context 0 as the FGCNT.
OK

AT+QICSGP=1,“CMNET”   // set the APN as “CMNET”.
OK

AT+QIMUX=0            // disable MUXIP.
OK                    // successfully implement the command.

AT+QIMODE=0           // set the session mode as non-transparent.
OK                    // successfully implement the command.

AT+QIDNSIP=1          // use domain name as the address to establish TCP/UDP
                       session.
OK                    // successfully implement the command.

```

The former configuration is just an example. It is unnecessary to be completely same as the former configuration. For example, it is OK to enable and set session mode as transparent mode.

Secondly, visit the remote server “www.quectel.com” by the command “AT+QIOPEN”

```

AT+QIOPEN=“TCP”, “www.quectel.com”,80
OK

CONNECT OK

```

Then, it is allowed to send data to or receive data from the server “www.quectel.com:80” in the same steps introduced in Chapter 3.4. And it is supported to close the session by the command “AT+QICLOSE” or “AT+QIDEACT”. The following figure is an example to visit the server “quectel.3322.org:7020”.



```

ATI
Quectel_Ltd
Quectel_M10
Revision:M10R04A02M32_SST

OK
AT+QIFGCNT=0
OK
AT+QICSGP=1,"CMNET"
OK
AT+QIDNSIP=1
OK
AT+QIOPEN="TCP","quectel.3322.org",7020
OK

CONNECT OK
AT+QISEND
> Welcome to use Quectel's module!
SEND OK

RCV FROM:116.226.41.43:7020
Quectel's module is great!

```

Figure 3: visit the server "quectel.3322.org:7020"

## 9.2. Get IP according to the domain name

The internal TCP/IP stack supports to get IP address according to the given domain name. The following is an example to parse the domain name "www.google.com".

- 1) Initialize GPRS/CSD context

```

AT+QIFGCNT=0 // set context 0 as the FGCNT.
OK

AT+QICSGP=1,"CMNET" // set the APN as "CMNET".
OK

```

- 2) Activate GPRS/CSD context and parse the domain name.

```

AT+QIREGAPP
OK

AT+QIACT // activate FGCNT.
OK

AT+QIDNSCFG="211.136.18.171","211.136.112.50" // "211.136.18.171" is the primary
// "211.136.112.50" is the secondary DNS server.
OK

AT+QIDNSGIP="www.google.com" // get the IP address of the domain
// name "www.google.com".
OK

```

**64.233.189.99****64.233.189.104****64.233.189.147**

Actually, the internal TCP/IP stack negotiates DNS servers automatically after activate GPRS/CSD. Hereby, it is unnecessary to implement the command “AT+QIDNSCFG” to set DNS servers. And the “AT+QIDNSGIP” can activate FGCNT automatically, so it is unnecessary to implement “AT+QIREGAPP” and “AT+QIACT” to activate FGCNT, either. So, the former commands can be simplified as following.

```
AT+QIDNSGIP="www.google.com"           // get the IP address of the domain
                                         name "www.google.com".
```

**OK****64.233.189.99****64.233.189.104****64.233.189.147**

## 10. TCP connection maintaining and detection

TCP is a reliable stream delivery service that guarantees delivery of a data stream sent from one host to another without duplication or losing data. But GSM network is very complicated so that some unknown errors occur to the internal TCP/IP stack based on GSM. This seriously affects the capability of TCP connection. So it is necessary to check whether the TCP connection is OK and whether the sent data has been received by the server. The chapter intends to recommend several methods to maintain TCP connection and detect the status of TCP connection.

### 10.1. Maintain TCP connection

The internal TCP/IP stack is based on GSM network and the resource of GSM network is limited. So the TCP connection may be disconnected by GSM network without any notification if there is no data transmission on the TCP connection for a period of time. Here is a method recommended by Quectel.

In order to maintain a TCP connection, it is recommended to send a small data packet to the remote end through the TCP connection and then wait for a moment to check whether the packet has been received by the remote end. Please refer to the following example for the details.

```

AT+QIFGCNT=0
OK

AT+QICSGP=1,"CMNET"           // select GPRS as the bearer for the
                                TCP connection and the APN is
                                "CMNET".

OK

AT+QIOPEN="TCP","116.226.33.129",7020 // try to visit the remote TCP server
                                        116.226.33.129:7020.

CONNECT OK                       // successfully to establish the TCP
                                    connection with the remote server
                                    116.226.33.129:7020.

.....                             // When there is no data transmission
                                    in an established TCP connection for a
                                    certain period, the GPRS network
                                    would force module to release an
                                    active PDP context which breaks the
                                    TCP connection. Therefore, it is
                                    suggested to send a heart beating small
                                    packet to server periodically to avoid

```

```
AT+QISTAT
OK
STATE: CONNECT OK
.....
AT+QISACK
+QISACK: 0, 0, 0
OK
AT+QISEND
> TCP detect
```

kicked off by network and maintain the TCP connection. For example, the periodic interval should be shorter than 8 minutes in China Mobile network. The PDP context active timer is decided by GPRS network, thus customer has to find a right number for its specific network.

// check the status of the TCP connection by the command “AT+QISTAT” at first.

// the TCP connection is still OK. This just means the internal TCP/IP stack didn’t receive any abnormal report for the TCP connection.

// no data was transmitted through the TCP connection for a period of time, such as 8 minutes.

// check the information for the sending data. The purpose is to check whether some sent data still wait for the acknowledgement from the remote end. If some sent data still wait for the acknowledgement from the remote end, the TCP connection MUST be abnormal. Then it is strongly recommended to re-establish the TCP connection.

// the third parameter 0 means no data wait for the acknowledgement from the remote end. If the third parameter is not 0, please re-establish the TCP connection.

// no data was transmitted through the TCP connection for so long time, it is recommended to send a small packet to the remote end to maintain the TCP connection.

// this is the small packet to send to the remote end. Of course, any other data

```

SEND OK
.....
AT+QISACK
+QISACK: 10, 10, 0
OK

```

is OK.  
// the internal TCP/IP stack has sent the data through the TCP connection.  
// wait for a moment (for example, 30 seconds) to check whether the packet has been received by the remote end.  
// check the information for the sending data.  
// the third parameter is 0, the packet has been received by the remote end successfully. This means the TCP connection is still normal.

In the above example, it waits for a period of time after sending the small packet “TCP detect” to issue the command “AT+QISACK” to check whether the packet has been received by the remote end. Here is another method. After send the packet, wait for 5 seconds, and then issue the command “AT+QISACK”. If the packet has been received by the remote end successfully, the TCP connection is still OK, so stop the check operation. If the packet has not been received by the remote end (according to the value of the third parameter of the response of “AT+QISACK”), please wait for 10 seconds again, and the repeat the former operation. Repeat the former operation for several times until the packet has been received by the remote end successfully or the time of repeat reaches to a limitation (for example, 12 times). If the time of repeat reaches to the limitation, the TCP connection is abnormal, so it is suggested to re-establish the TCP connection.

## 10.2. Detect TCP connection

The principle to detect TCP connection is similar to the method to maintain TCP connection. After a TCP connection has been established. Here is a sample process to detect the status of the TCP connection.

```

AT+QISTAT
OK
STATE: CONNECT OK

```

// check the status of the TCP connection by the command “AT+QISTAT” at first.  
// the TCP connection is still OK. This just means the internal TCP/IP stack doesn’t receive any abnormal report for the TCP connection. If it is not “CONNECT OK”, the TCP connection is abnormal. Please refer to

```
.....  
  
AT+QISACK  
  
+QISACK: 10, 10, 0  
  
OK  
AT+QISEND  
  
> TCP detect  
  
SEND OK  
  
.....  
  
AT+QISACK  
+QISACK: 20, 20, 0
```

Document [1] for the details of status.  
// no data is transmitted through the TCP connection for a period of time, such as 8 minutes.  
// check the information for the sending data. The purpose is to check whether some sent data still wait for the acknowledgement from the remote end. If some sent data still wait for the acknowledgement from the remote end, the TCP connection MUST be abnormal.  
// the third parameter 0 means no data wait for the acknowledgement from the remote end. If the third parameter is not 0, the TCP connection is abnormal.

// no data was transmitted through the TCP connection for so long time, it is recommended to send a small packet to the remote end to check whether the connection is still OK.  
// this is the small packet to send to the remote end. Of course, any other data is OK.  
// the internal TCP/IP stack has sent the data through the TCP connection.  
// wait for 5 second to check whether the packet has been received by the remote end.  
// check the information for the sending data.  
// the third parameter is 0. The packet has been received by the remote end successfully. This means the TCP connection is still normal. If it isn't 0, the packet hasn't been received by the remote end and it is strongly recommended to repeat check the information for sending data by the command "AT+QISACK" for at most 12 times. If the third parameter of the

response of “AT+QISACK” becomes 0 in some time of repeat, the packet has been sent to the remote end successfully which indicates the TCP connection is still OK, and then please stop the repeat operation. If the third parameter is always not 0, it means the TCP connection is abnormal.

**OK**

**Note:**

If the module was set to enable MUXIP, please specify the TCP connection to detect, i.e. use “AT+QISACK=<index>” to replace “AT+QISACK” in the above examples.

# QUECTEL



**Quectel Wireless Solutions Co, Ltd.**  
Room 801, Building E, No.1618, Yishan Road, Shanghai, China 201103  
Tel: +86 21 5108 2965  
Mail: [info@quectel.com](mailto:info@quectel.com)