WISMO Quik Q2438 series

Application Note

TCP App

Reference : **WM_CCD_Q24x8_CTI_015**
Revision : **004**
Date : **03/10/2005**

Document Information

Revision	Date	History of the evolution	Author
004	03/10/2005	Updated module dormancy section 6.6	D. Buczynski
003	03/04/2005	Removed all AT command description and moved it to the AT command document. Updated for changes to the user interface and performance table.	D. Buczynski
002	02/07/2005	Added new section 3, Document Conventions and Section 11, ASCII character code chart. Minor corrections; sections 5.5.2 and 8. Removed all references to the +WHTX command.	D. Buczynski
001	12/03/2004	Initial document	D. Buczynski

Contents

Trademarks

Wavecom, WISMO are trademarks or registered trademarks of Wavecom S.A. All other company and/or product names mentioned may be trademarks or registered trademarks of their respective owners.

1 Overview

This application note provides information on the use of the TCP App feature. This feature provides Q24x8 module functionality for establishing TCP and UDP compliant non-blocking socket connections over an IP network. Before using this feature, verify that the module is in compliance with carrier network requirements. Also verify that the module is properly provisioned for the carrier network and that voice and data calls can be performed successfully.

When in a normal 1xRTT data call, the module is unresponsive to AT commands and suppresses all unsolicited AT responses to prevent corruption of the socket data stream. The TCP App feature provides a means of performing multiple socket data connections while retaining AT command level control of the Q24x8 module.

2 Scope

This document details the AT commands and responses associated with the TCP App feature. Usage examples are included to aide in the development of the host application that will interface with the module provided functionality. Where appropriate, operational considerations are presented which identify areas where multiple methodologies could be used to achieve an end result.

The reader is expected to have a good understanding of IP based data exchange and methodologies. A detailed description of the various IP protocols that are referred to in this document (e.g. TCP, UDP, PPP, etc.) is beyond the scope of this document. A search of the web will return useful information in this subject area. For example: http://www.developerfusion.com/show/28/

The use of the term 'Host Application' in this document refers to the program that is controlling the Q24x8 module using AT commands. The host application is responsible for the orderly startup, data exchange and shutdown of the TCP or UDP socket connections. The structure and design of the host application will effect the overall performance of the TCP App feature. A detailed consideration of the design tradeoffs is beyond the scope of this document.

3 Conventions

The following conventions are used in the AT command examples of this document to represent character data.

Character Data	Description
ABCD or "ABCD"	A string of ASCII characters as typed on a keyboard. Each character is a byte size value. See ASCII table in section 10 for all character code values.
0x41 0x42 0x43 0x44	Digits preceded by '0x' represent hexadecimal byte values; in this case, the ASCII codes for ABCD. The space separating each value

	is for readability purposes only.
<cr>	The ASCII character code corresponding to a carriage return. This byte size value is equal to 13 in decimal or 0x0d in hexadecimal.
<lf>	The ASCII character code corresponding to a new line or line feed. This byte size value is equal to 10 in decimal or 0x0a in hexadecimal.

Table 3-1 Document Conventions

4 TCP Application

The TCP App feature provides a means for a host application to make TCP or UDP socket connections with an internet server using the Wavecom WISMO Q24x8 CDMA module. The host application uses a set of AT commands for the setup and control of socket connections. Other AT commands and responses are provided for performing data transfers with a remote IP server. The use of an AT command interface minimizes the necessary functionality in the host application for internet packet communications.

The TCP App handles all of the lower level network protocol and data transmission tasks. It can connect to and exchange data with any server located on the Internet with a valid IP address and Port number. It supports multiple and simultaneous socket connections to different servers.

The TCP App supports both Mobile IP and Simple IP means of CDMA authentication and connectivity. It supports both TCP/IP and UDP/IP transport protocols including access to DNS services. It provides it's own dynamically assigned IP address to the host application.

In general, the host application is responsible for the following when using the functionality of the TCP App feature.

1. AT command and unsolicited response processing using the modules 'Data' port.
2. Coordination of TCP App related activities with other concurrent module functionality.
3. Module configuration and setup for IP network operation.
4. Opening and closing of a PPP session with the wireless network IP server.
5. Opening and closing of socket connections (TCP or UDP) with the desired internet servers.
6. Insertion of socket data into the appropriate AT commands and transmission of those commands to the module.
7. Reception of module AT unsolicited responses and the extraction of socket data.
8. Maintain an accurate mapping of socket connections, data formatting and host application processes.
9. Related error detection and processing.

4.1 Performance

Many factors effect the overall data throughput of the TCP App feature. Some of these factors include: internet congestion, carrier CDMA network congestion and configuration, the quality and coverage of the associated data call, target server loading and host application design. The following provides an example of possible data throughputs utilizing a 1xRTT CDMA data call.

The throughput capacity is shared across all active TCP and UDP socket connections.

Transport	Max Packet Size (bytes)	Approximate Throughput (uplink/downlink)
TCP	536	22,000 / 45,000 bits per second
UDP	1330	60,000 / 75,000 bits per second

Table 4-1 Throughput Summary

5 AT Commands and Responses

The TCP App feature includes a set of AT commands and unsolicited responses for host application initialization and control of its functionality. Table 2 summarizes these commands and responses. Refer to the AT command document WI_SW_CDMA_PTS_001 version 1.35 or later for detailed usage information.

Command	Description
AT+WPPP	Start or end a PPP session
AT+WOSK	Open a socket
AT+WCSK	Close a socket
AT+WSTX	Transmit socket data
AT+WSRX	Receive polled socket data
AT+WGSS	Display connection status
AT+WTMO	Configure socket transmit timeout
AT+WCRX	Configure receive data mode
AT+WIPC	Show current module IP address
AT+WDNS	IP address lookup
AT+WFDM	Force dormant mode
Response	**Description**
+WPPP	PPP session status
+WSKS	Socket state change
+WSKE	Socket open/close error
+WSTX	Socket data transmission status
+WSTE	Socket data transmission error
+WSRX	Received socket data
+WSRE	Socket data error
+WDOR	Dormant mode status change
+WDNS	DNS lookup indication

Table 5-1 TCP App Command/Response Summary

6 Design Considerations

6.1 MIP and SIP

The use of Mobile IP or Simple IP is generally predicated by carrier requirements. There are a number of carrier specific module configurations that must be properly set to successfully perform data calls and establish TCP App socket connections. In most cases, these settings will be established when the Q24x8 module is provisioned for use on the carriers network. Refer to the carriers provisioning requirements for further information. Changes to the existing Q24x8 module provisioning settings must be done using the appropriate Wavecom software tool.

6.2 TCP and UDP Sockets

The use of a TCP or UDP socket connection is dependent on the host application requirements. The TCP protocol is well defined for data exchange between an internet server and client and includes error correction. When the TCP protocol is used there is a "guaranteed delivery" of data. This is due to the supported flow control that determines when data needs to be re-sent. Retransmission of data happens transparently to the host application layer. TCP frames are 536 bytes in length. TCP protocol socket connections are appropriate when connection to generally available internet services at typical speeds is required.

TCP Frame Structure				
16 bits				16 bits
Source Port				Destination Port
Sequence Number				
Acknowledgement Number				
Data Offset	Reserved	ECN	Control Bits	Window
Checksum				Urgent Pointer
Options				Padding
Payload Data				

Figure 6-1 TCP Frame Structure

The UDP protocol is used with applications where a high data transfer speed is required. An example is a streaming media such as audio or video. The UDP protocol offers no native flow control or data error detection. It is the responsibility of the host application layer to perform these functions. Since the data associated with this protocol is a user defined stream of bytes, it is easily customized for specific applications. The coding of appropriate server and client functionality is required. The TCP App feature limits the maximum size of UDP packets to 1330 bytes.

UDP Frame Structure

16 bits	16 bits
Source Port	Destination Port
Length	Checksum
Payload Data	

Figure 6-2 UDP Frame Structure

6.3 Write Data Flow Control

The TCP App feature provides the +WSTX command for use by the host application to send data to a socket connection. The data throughput capacity of the Q24x8 module is dynamic and many factors effected it including carrier CDMA network congestion and signal quality. The available radio link capacity is used by all active socket connections. If the host application sends data to open socket connections faster than it can be transmitted over-the-air by the module, a blocking condition will occur and be reported to the host application.

A blocking condition is indicated to the host application by the unsolicited +WSTE: <type>,<socket>,102 response. The term <type> refers to the socket type; 0 for TCP and 1 for UDP. The term <socket> refers to the socket number. This unsolicited response will occur in addition to an AT command ERROR that is returned for the associated +WSTX command.

In response to this condition, the host application must wait for the currently buffered socket data to be transmitted. Once socket buffer space is again available, the +WSKS: <type>,<socket>,1 unsolicited response will be sent to the host application. The host application should then resend the +WSTX blocked data to the socket connection.

Figure 6-3 shows the flow for a blocked TCP socket transmission. Section 7.4 provides an example of a blocked UDP socket transmission.

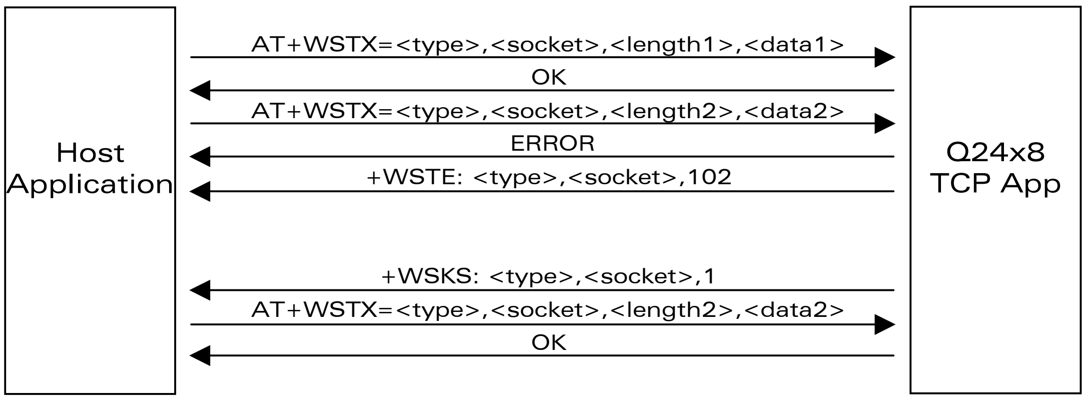
Host Application
Q24x8 TCP App
AT+WSTX=<type>,<socket>,<length1>,<data1>
OK
AT+WSTX=<type>,<socket>,<length2>,<data2>
ERROR
+WSTE: <type>,<socket>,102
+WSKS: <type>,<socket>,1
AT+WSTX=<type>,<socket>,<length2>,<data2>
OK

Figure 6-3 Transmit Data Flow Control

In other words, the +WSTE and +WSKS unsolicited responses are transmit data flow control indicators for socket connections. These indications are socket specific and multiple indications can occur if multiple socket connections are active.

In a blocked condition, the host application should wait a reasonable amount of time for a +WSKS unsolicited response that indicates the blocking condition has cleared. If not received, the host application can use the +WGSS command to evaluate the <PPPstate> and <SocketState> values for the open connections. The host application can then initiate appropriate error recovery processes.

6.4 Polled and Unsolicited Received Data

The TCP App feature provides two methods for processing received socket data. The method used will depend on the design and desired operational characteristics of the host application. Socket data received from the IP network can be delivered to the host application in either 'Polled' or 'Unsolicited' mode. The +WCRX command is used to establish the desired settings.

In 'Polled' mode, the module sends a +WSKS unsolicited response to the host application when receive data is available in a socket buffer. The host application must then use the +WSRX command to retrieve the socket data from the module. This methodology is suited to single thread host applications with one open socket and low to medium data throughput rates.

In 'Unsolicited' mode, the module sends a +WSRX unsolicited response to the host application which includes the associated socket data. Fields within these responses identify the associated socket connection and data length. This methodology is well suited for multiple thread host applications with many open sockets and high data throughput rates.

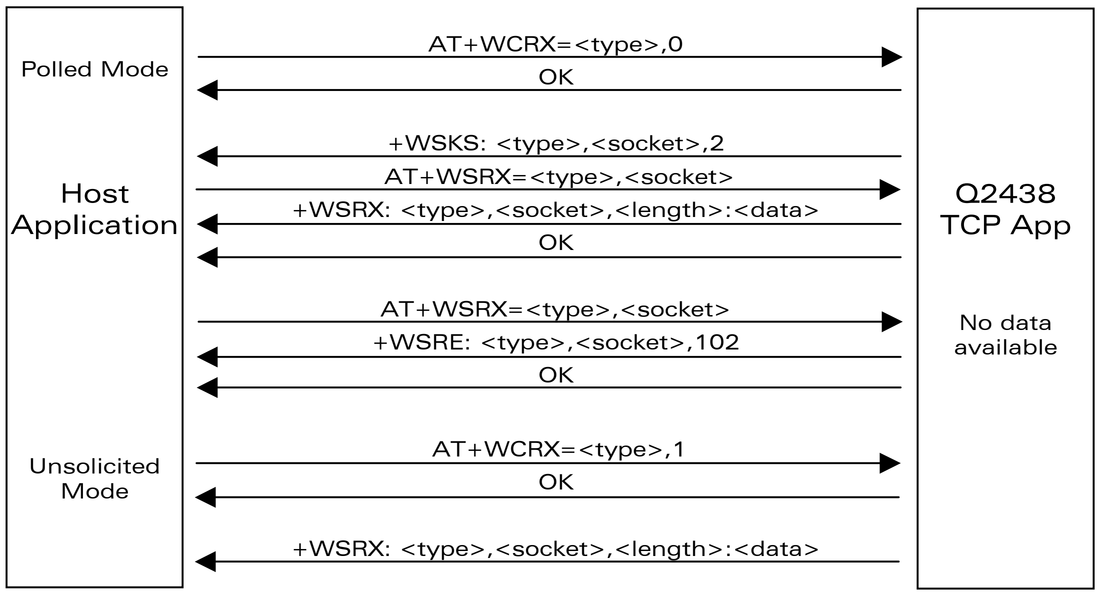

Figure 6-4 Receive Data Flow Control

The Q24x8 software returns payload data up to approximately 600 bytes using a single +WSRX unsolicited response. This limit does not effect TCP sockets since the maximum TCP packet size is 536 bytes. For UDP sockets however, where the payload packet size can be larger, multiple +WSRX unsolicited responses will be used as necessary to return the packet data to the host application. Each response will contain the next sequential portion of the received UDP packet data.

For example, using a UDP packet size of 1330 bytes, three +WSRX unsolicited responses will be used to return the socket data packet. If using polled mode, only one +WSRX command is required to return all three portions of the socket data packet; three +WSRX unsolicited responses will occur. The host application must be designed to handle multiple +WSRX responses when using large server/client UDP data packets.

6.5 Unsolicited Responses

When a socket connection is open, the TCP App feature utilizes unsolicited response messages to inform the host application of status changes. For example, when polling mode is used, a +WSKS response indicates that receive data is available. Since IP socket data communication is generally asynchronous, unsolicited responses can occur at any time; including while waiting for an unrelated AT command result (e.g. AT+CSQ ... OK). All socket related unsolicited responses include the socket type and number that is associated with the message to permit proper handling by the host application. The host application must be designed in a way that ensures that unsolicited responses are not missed while other processing is being performed.

When polling mode is used, a missed +WSKS response will result in the stopping of the received socket data flow. This is because only one socket specific +WSKS response is sent by the TCP App feature when received data is available. Subsequent +WSKS responses can not be sent by the TCP App until the current buffered data is read by the host application using the +WSRX command.

When unsolicited mode is used, a missed +WSRX response will result in the loss of its associated payload data. A recovery process whereby the lost data is retransmitted from the server to the host application must be performed.

When polling mode and multiple open socket connections are used, the host application must include provisions for handling multiple +WSKS responses. These responses can occur for any other open socket connection while waiting for a +WSRX response to a +WSRX command. Recursive capable response handler code can address this condition. The use of unsolicited mode is recommended when multiple open socket connections are required.

In high data rate applications using UDP sockets, the bandwidth of the Q24x8 'Data' port may become a host application design factor for received socket data. In such cases, the highest possible baud rate between the host application and Q24x8 module should be used to minimize the need for socket data retransmission. It might also be appropriate to implement some form of application layer throttling of received socket data from the server.

6.6 Module Dormancy

After approximately ten seconds of inactivity (carrier specific) during a data call, the Q24x8 module automatically enters "dormant" mode. In this mode, the module releases CDMA traffic channel resources to conserve power and minimize loading on the carrier network. The module exits dormant mode when an event occurs that requires CDMA traffic channel resources. The entry and exit of dormant mode is automatically performed based on the need for carrier network resources and does not require any specific action on the part of the host application.

The TCP App feature will send the +WDOR unsolicited response to inform the host application of a change in dormant mode status. The host application can use this indication, if required, to make timing allowances for an AT command that causes the module to exit dormant mode. When in dormant mode, the first host application initiated client to server data transmission (e.g. AT+WSTX) will experience a completion delay. That is, the +WSTX unsolicited response indicating the number of bytes sent. This is a result of the Q24x8 module reacquiring the CDMA traffic during the dormant mode exit process. This time delay varies and is dependent on carrier network congestion. It is typically in the 3 to 5 second range.

The carrier network can also force the Q24x8 module into dormant mode during active socket data transmissions due its management of CDMA traffic channel congestion/bandwidth. The +WDOR unsolicited response will be reported. This will result in a blocked unsolicited response (+WSTE: x,x,102) being reported to the host application for subsequent AT+WSTX commands. The host application should perform a delay and resend loop for the blocked socket data packet until dormant mode is exit and the blocked packet is successfully transmitted.

While the host application is waiting for receive data (a +WSRX or +WSKS response), dormant mode may likewise occur due to no client/server data transmission or some other carrier network event. The +WDOR unsolicited response is reported to the host application for this condition. If the dormant mode condition lasts long enough, the server TCP layer will timeout and the open socket connections with the client will be closed. In this state, the +WGSS command may continue to report normal TCP socket connectivity to the host application because the server initiated socket close event may not have been sent to the Q24x8 module TCP stack by the carrier network.

To handle this condition, the host application should implement some form of periodic data exchange or "ping" with the server when normal data transmissions are not performed for a long time period. This will cause the module to exit dormant mode for the data exchange and verify ongoing socket connectivity. If socket or PPP session errors are returned, the host application can initiate appropriate recovery steps.

6.7 Power Save Mode

The AT+W32K command is a standard Q24x8 module command that can be used to enable or disable sleep mode operation. When enabled, the module enters sleep mode when inactive for 10 to 15 seconds. This mode conserves power by turning off a significant portion of the module hardware. A host application issued AT command will cause the module to exit sleep mode. However, the first few characters of this "wake up" command are lost due to the modules 'Data' port being powered off. Since the module software does not see the initial AT character sequence, the remainder of the command string is ignored.

When utilizing the Q24x8 sleep mode feature, the host application should ensure that sleep mode is exit prior to sending an AT command to be processed. This is easily accomplished by first sending a separate command consisting only of "AT". This "AT" only command can be repeated at intervals until the module responds "OK".

6.8 Radio Link

The TCP App feature provides a robust methodology for performing socket data transfers utilizing a IP network PPP session. However, the differences between wired and wireless connections must be kept in mind when designing a server/client application. For example, socket connections or the PPP session itself can terminate unexpectedly due to a dropped call. Data transmission rates may vary depending on carrier network loading and radio coverage conditions. Appropriate recovery measures must be included in the application design to handle these and other related circumstances.

7 Usage Examples

This section provides usage examples for the TCP App feature. The examples show typical AT command exchanges between the Q24x8 module and host application when using the TCP App feature.

7.1 TCP Connection, Polled Bi-Directional Data Transfer

```
AT+CMEE=1                   Enable detailed reporting of mobile equipment errors.
  OK
AT+WGSS=0                   Display TCP connection status.
  +WGSS: 0,0,1,0,0,0,0            TCP status, PPPstate "closed",  RxMode "Unsolicited", All
sockets "closed".
  OK
AT+WCRX=0,0                       Set TCP receive mode; polled received data.
  OK
AT+WGSS=0                   Display TCP connection status.
  +WGSS: 0,0,0,0,0,0,0            TCP status, PPPstate "closed",  RxMode "Polled", All sockets
"closed".
  OK

AT+WPPP=0                   Start a MIP data call and open a PPP session.
  OK
  +WPPP: 201                Unsolicited response; PPP session startup in progress.
  +WPPP: 200                Unsolicited response; PPP session established and available.
AT+WGSS=0                   Display TCP connection status.
  +WGSS: 0,2,0,0,0,0,0            TCP status, PPPstate "open", RxMode "Polled", All sockets
"closed".
  OK

AT+WIPC                           Display module IP address.
  +WIPC: 68.25.209.28
  OK

AT+WOSK=0,12,57,125,2,24          Open a TCP socket to IP address 12.57.125.2 port 24.
  +WOSK: 0,0                TCP socket zero allocated.
  OK
  +WSKS: 0,0,1                    Unsolicited response; TCP socket zero is open.
AT+WGSS=0                   Display TCP connection status.
  +WGSS: 0,2,0,2,0,0,0            TCP status, PPPstate "open", RxMode "Polled", Socket zero
"open".
  OK

AT+WSTX=0,0,5,<cr>HELLO           Send "HELLO" to TCP socket zero.
  OK
  +WSTX: 0,0,5                    Unsolicited response; five bytes transmitted on TCP socket
zero.
  ...
  +WSKS: 0,0,2                    Unsolicited response; Received data available on socket zero.
```

```
AT+WSRX=0,0                         Read and clear TCP socket zero data buffer.
  +WSRX: 0,0,5:WORLD                Socket zero data "WORLD".
  OK
  ...

AT+WCSK=0,0                         Close TCP socket zero.
  OK
  +WSKS: 0,0,4                      Unsolicited response; TCP socket zero is closed.
AT+WPPP=2                     Close the PPP session and end the data call.
  OK
  +WPPP: 203                  Unsolicited response; PPP session shutdown in progress.
  +WPPP: 202                  Unsolicited response; PPP session closed.
```

7.2 TCP Connection, Unsolicited Received Data

```
  ...                        For this example, we'll assume that TCP socket zero has been
previously
                             opened for some other non-related purpose and TCP "Unsolicited"
                           receive
                             mode has been set.
AT+WOSK=0,168,0,0,2,42             Open a TCP socket to IP address 168.0.0.2 port 42.
  +WOSK: 0,1                 TCP socket one allocated.
  OK
  +WSKS: 0,1,1                     Unsolicited response; TCP socket one is open.

AT+WGSS=0                    Display TCP connection status.
  +WGSS: 0,2,1,2,2,0,0             TCP status, PPPstate "open", RxMode "Unsolicited", Socket
zero "open",
  OK                         Socket one "open".

AT+WSTX=0,1,5,<cr>HELLO            Send "HELLO" to TCP socket one.
  OK
  +WSTX: 0,1,5                     Unsolicited response; five bytes transmitted on TCP socket
one.
  ...
 +WSRX: 0,1,5:WORLD                Unsolicited response; 5 bytes of data received on TCP socket
one.
                             "WORLD"
AT+WCSK=0,1                        Close TCP socket one.
  OK
  +WSKS: 0,1,4                     Unsolicited response; TCP socket one is closed.
AT+WGSS=0                    Display TCP connection status.
  +WGSS: 0,2,1,2,0,0,0             TCP status, PPPstate "open", RxMode "Unsolicited", Socket
zero "open",
  OK                         Socket one "closed".
```

7.3 UDP Connection, DNS Lookup, Unsolicited Received Data

```
AT+WCRX=1,1                        Set UDP receive mode; unsolicited received data.
  OK
AT+WPPP=1,user,password            Start a SIP data call and open a PPP session.
  OK
  +WPPP: 201                 Unsolicited response; PPP session startup in progress.
  +WPPP: 200                 Unsolicited response; PPP session established and available.
AT+WGSS=1                    Display UDP connection status.
  +WGSS: 1,2,1,0,0,0,0             UDP status, PPPstate "open", RxMode "Unsolicited", All
sockets "closed".
  OK

AT+WDNS="www.myurl.org"            Look up IP address on DNS server.
  OK
  ...
  +WDNS: 0                   Unsolicited response; Communication with DNS server completed.
AT+WDNS="www.myurl.org"            Display DNS server lookup result.
  +WDNS : 216.37.68.117            IP address of "www.myurl.org".
  OK

AT+WOSK=1,216,37,68,117,250 Open a UDP socket to IP address 216.37.68.117 port 250.
  +WOSK: 1,0                 UDP socket zero allocated.
  OK
  +WSKS: 1,0,1                     Unsolicited response; UDP socket zero is open.
AT+WGSS=1                    Display UDP connection status.
  +WGSS: 1,2,1,2,0,0,0             UDP status, PPPstate "open", RxMode "Unsolicited", Socket
zero "open".
```

```
  OK

AT+WSTX=1,0,1024,<cr>HELLO ...       Send HELLO to UDP socket zero; 1024 byte packet.
  OK
  +WSTX: 1,0,1024           Unsolicited response; 1024 bytes transmitted on UDP socket zero.
  ...
  +WSRX: 1,0,600:WORLD ...     Unsolicited response; first 600 bytes of a 1024 byte size
packet.
  +WSRX: 1,0,424: ...          Unsolicited response; last 424 bytes of a 1024 byte size
packet.
```

```
AT+WCSK=1,0                     Close UDP socket zero.
  OK
  +WSKS: 1,0,4                  Unsolicited response; UDP socket zero is closed.
AT+WGSS=1                  Display UDP connection status.
  +WGSS: 1,2,1,0,0,0,0          UDP status, PPPstate "open", RxMode "Unsolicited", Socket
zero "closed".
  OK
AT+WPPP=2                  Close the PPP session and end the data call.
  OK
  +WPPP: 203               Unsolicited response; PPP session shutdown in progress.
  +WPPP: 202               Unsolicited response; PPP session closed.
```

7.4 UDP Connection, Blocked Data Transmission

```
AT+WPPP=0                  Start a MIP data call and open a PPP session.
  OK
  +WPPP: 201               Unsolicited response; PPP session startup in progress.
  +WPPP: 200               Unsolicited response; PPP session established and available.
AT+WOSK=1,216,37,68,117,250 Open a UDP socket to IP address 216.37.68.117 port 250.
  +WOSK: 1,0               UDP socket zero allocated.
  OK
  +WSKS: 1,0,1                  Unsolicited response; UDP socket zero is open.
AT+WGSS=1                  Display UDP connection status.
  +WGSS: 1,2,1,2,0,0,0          UDP status, PPPstate "open", RxMode "Unsolicited", Socket
zero "open".
  OK

AT+WSTX=1,0,500,<data>          Send 500 bytes of data on UDP socket zero.
  OK
  +WSTX: 1,0,500                Unsolicited response; 500 bytes transmitted on UDP socket
zero.

  ...                      Other +WSTX commands and +WSTX responses for data transfers.

AT+WSTX=1,0,500,<data>          Send 500 bytes of data on UDP socket zero.
  +CME ERROR : 3
  +WSTE: 1,0,102                Blocked data transmission on UDP socket zero.
AT+WGSS=1                  Display UDP connection status.
  +WGSS: 1,2,1,2,0,0,0          UDP status: PPPstate "open" and Socket zero "open".
Blocking condition due
  OK                       to full socket buffer.
  ...
  +WSKS: 1,0,1                  UDP socket zero available for data transmission.

AT+WSTX=1,0,500,<data>          Resend the failed 500 bytes of data on UDP socket zero.
  OK
  +WSTX: 0,500                  Unsolicited response; resend good. 500 bytes transmitted on
UDP socket zero.

  ...                      Other +WSTX commands and +WSTX responses for data transfers.

AT+WSTX=1,0,500,<data>          Send 500 bytes of data on UDP socket zero.
  +CME ERROR : 3
  +WSTE: 0,114                  Lost UDP socket zero connection.
AT+WGSS=1                  Display UDP connection status.
  +WGSS: 1,2,1,0,0,0,0          UDP status: PPPstate "open" and Socket zero "closed".
  OK
AT+WOSK=1,216,37,68,117,250 Re-open a UDP socket to IP address 216.37.68.117 port 250.
  +WOSK: 1,0               UDP socket zero allocated.
  OK
  +WSKS: 1,0,1                  Unsolicited response; UDP socket zero is open.

  ...                      Host application specific recovery process.
```

8 Status and Error Codes

This section summarizes the status and error codes that can be returned by the TCP App feature.

Socket Status Events	
Code	**Description**
1	Socket is open
2	Receive data is available
4	Socket is closed
Socket Error Codes	
Code	**Description**
100	Invalid socket descriptor
101	Invalid buffer or argument
102	Operation would block
103	Address family not supported
104	Wrong protocol for socket type
105	Socket parameter not supported
106	Protocol not supported
107	No more sockets available for opening
108	Operation not supported
109	Address already in use
110	Destination address required
111	Connection establishment in progress
112	Connection already established
113	IP address changed, causing TCP reset
114	Socket not connected
115	Connection attempt refused
116	Connection attempt timed out
117	Connection reset
118	Connection aborted
119	Broken pipe
120	Network subsystem unavailable
121	No more applications available
122	Invalid application ID
123	There are existing sockets
124	Invalid operation

Table 8-1 TCP App Error Codes

DNS Server Codes	
Code	**Description**
125	Domain Name Error or not found
126	Domain Name not found
127	Network is not opened
128	Out of memory
129	DNS Server busy
130	Reserved
131	Reserved
132	Unrecoverable error
133	No address for the domain name
PPP Network Codes	
Code	**Description**
200	PPP established and available
201	PPP initialization in progress
202	PPP closed or unavailable
203	PPP is closing

Table 8-1 TCP App Error Codes (continued)

9 Terms and Acronyms

The following defines the acronyms and terms used in this document.

Acronym	Definition
<tbd>	To be determined
0x	Hexadecimal prefix. Digits and letters following represent a hexadecimal value.
1xRTT	1x Radio Transmission Technology
ASCII	American Standard Code for Information Interchange
AT	Attention – Identifies a string as a Q24x8 module command
CDMA	Code division multiple access
DNS	Domain Name Server
DTE	Data Terminal Equipment
IP	Internet Protocol
LSB	Least Significant Byte or Bit
MIP	Mobile Internet Protocol
MSB	Most Significant Byte or Bit
NVRAM	Non-Volatile Random Access Memory
PPP	Point-to-Point Protocol
SIP	Simple Internet Protocol
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol over Internet Protocol
UDP	User Datagram Protocol
UDP/IP	User Datagram Protocol over Internet Protocol

Table 9-1 Terms and Acronyms

10 ASCII Character Code Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Table 10-1 ASCII Character Codes