Basic TCP/IP Communication in LabVIEW

Publish Date: Sep 06, 2006

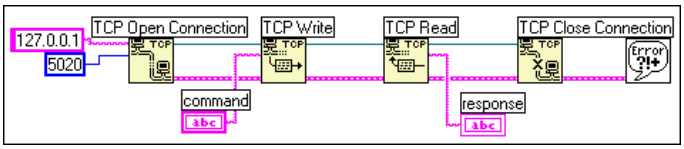
1. Description

Internet Protocol (IP), User Datagram Protocol (UDP), and Transmission Control Protocol (TCP) are the basic tools for network communication. The name TCP/IP comes from two of the best-known protocols of the internet protocol suite, the Transmission Control Protocol and the Internet Protocol. With TCP/IP you can communicate over single networks or interconnected networks (Internet).

TCP/IP communication provides a simple user interface that conceals the complexities of ensuring reliable network communications. Refer to the Using LabVIEW with TCP/IP and UDP Application Note (linked below) for more information about how TCP/IP communication works. Refer to the TCP specification (linked below) for technical details about TCP.

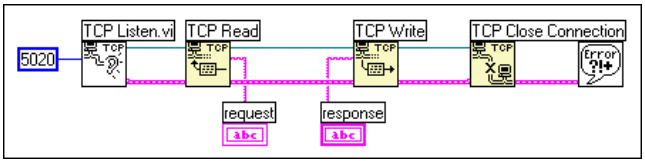
Use the TCP/IP functions located on the **Functions»Communication»TCP** palette for TCP communication in LabVIEW. As with DAQ, instrument, and File I/O communication, the process involves opening the connection, reading and writing the information, and closing the connection.

With most I/O communication, the processor is always the client that initiates a connection to the disk drive server, the external instrument server, or the DAQ board server. With TCP/IP connections, a computer can function either as the client or the server. The following block diagram represents a client application that initiates a connection to a remote server with TCP Open Connection. The server, or daemon, listens for remote connections and responds appropriately.



LabVIEW users can develop custom applications for TCP/IP communication. The programmer is responsible for developing both the client and the server. Refer to the Using LabVIEW with TCP/IP and UDP Application Note (linked below) for more information about creating a TCP client with LabVIEW.

Because anyone can initiate a connection to a server, you might want server access control. The following block diagram shows how the server uses the remote address output value of the TCP Listen VI to determine whether a remote client has permission to access the server.

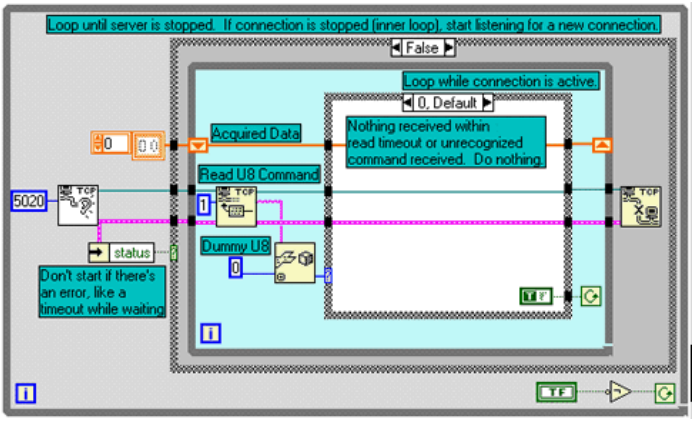


Developing Communication Applications

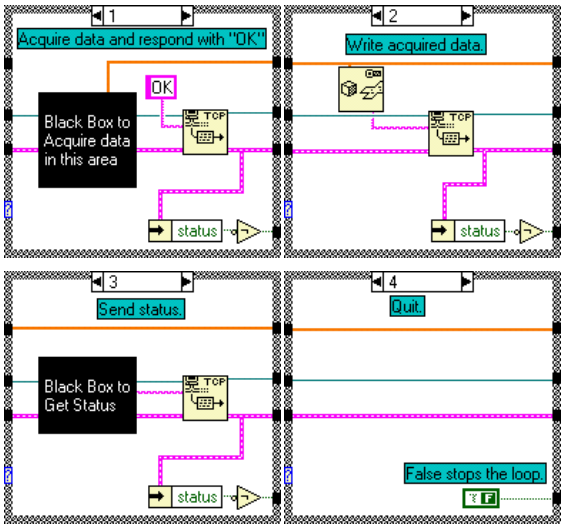
Most applications do more than write and read one value. Communication is an ongoing process that involves protocol. For example, suppose a client sends the following four commands by 8-bit integer to the server:

1 = acquire data and confirm
2 = send data
3 = get status
4 = close connection

In the following block diagram, a While Loop surrounds the rest of the VI. This allows the VI to handle multiple sequential connections without having to restart after each connection closes. The VI cannot handle multiple simultaneous connections. The outer Case structure determines whether a valid connection occurred. If not, nothing happens. If a valid connection occurs, the VI enters a While Loop that reads one byte from the TCP/IP port. This byte holds commands 1 through 4 from the client. If no command is received within the read timeout period, the default case of the inner Case structure sends a TRUE value to the continuation terminal of the inner While Loop to keep the connection active.



The following block diagram shows the other four cases of the inner case statement. Each case handles a specific command that the server can send. Each case sends information to the continuation terminal, which determines whether or not to continue the loop. In particular, the Quit case always returns a value of FALSE. After leaving the loop, the server closes the connection with the client.



This type of server architecture allows you to develop flexible servers for the more complex network communication procedures. The protocols you develop might be more complex than the preceding example.

2. Common Applications

Sending acquired data to remote locations
Controlling data acquisition from remote locations

Related Links:

Transmission Control Protocol Specification
Using LabVIEW with TCP/IP and UDP