Visual C++ TCP/IP Communications

Overview

This application demonstrates Ethernet communications using Windows sockets. This simple program, initializes the socket and sends a POL command to the attached QuickSilver Device. The device's response is received and displayed.

It is assumed the reader is familiar with Windows, Visual C++, QuickControl®, programming QuickSilver E-485 Bridge products, TCP/IP and QuickSilver's serial communication. For more information see:

- QCI-TD053 Serial Communications
- QCI-TD056 E-485 Bridge-Ethernet
- SilverLode User Manual

This document is meant to explain the setup and general design of the example. The details concerning communicating with a QuickSilver device are left up to the documents mentioned above and the source code comments.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Setup

The following assumptions are made about the QuickSilver device:

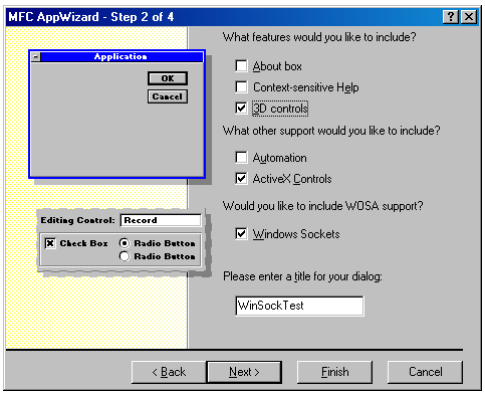
1) The device has an E-485 Bridge (i.e. QCI-D2-IG8-E)
2) The device has already been initialized (using QuickControl) over the E-485 Bridge.
3) Unit ID of 16.
4) IP Address 10.10.13.160 (see below to a use different address)

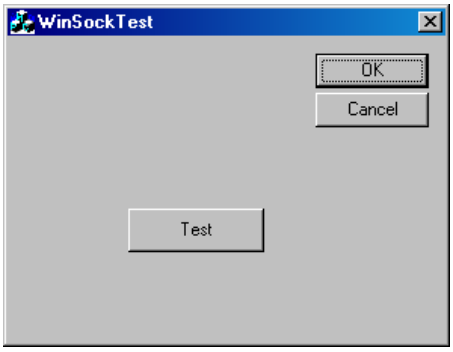
Supplied Files

File/Folder	Description
WinSockTest	Source Code folder

Application Setup

When the project is first created, be sure to include "Windows Sockets" (see example to the right).





Dialog Box

Pressing the 'Test' button will send a POL command.

Notes On Documentation

Since the Microsoft framework generates some of the code for you, only those classes with QCI specific code will be addressed.

Note: QCI specific code is tagged with the comment character "///QCI". This allows the programmer to quickly search the application for the "important" stuff.

CWinSockTestDlg

This is the dialog box class. It contains the code that enables us to communicate with the device.

It first establishes a TCP connection with the device and then sends the POL command. This is where the default IP Address of "10.10.13.160" can be changed. The port must remain at 10001.

This is a simple example that uses CAsyncSocket::Receive to receive data from the device. A more sophisticated approach would be to override CAsyncSocket::OnReceive message handler.