Integrating the Internet into Your Measurement System

DataSocket Technical Overview

NATIONAL INSTRUMENTS™

Introduction

The Internet continues to become more integrated into our daily lives. This is particularly true for scientists and engineers, because designers of development systems view the Internet as a cost-effective worldwide standard for distributing data. Today, National Instruments customers can easily publish data from their programs to the Web using the LabVIEW and LabWindows/CVI Internet tools. With these Internet tools, programmers create applications that serve images of the front panels of their applications as Web pages with very little or no programming. Passing images over the Internet is easy; however, many users are looking for more interactive solutions in which users can actually control experiments remotely using a Web browser. Users also want to maximize performance, which often means passing raw data values rather than large images. Improving the performance of Web applications is possible, but only with specialized networking or Internet programming experience.

National Instruments now offers DataSocket, a new Internet programming technology that simplifies data exchange between computers and applications. With DataSocket, programmers can efficiently pass raw data over the Internet and respond to multiple users without the complexity of low-level TCP programming.

Why DataSocket? The Background

Getting all your hardware and software components linked together has always had a few challenges. For example, hooking up the hardware requires consideration of signal levels, impedance, etc. Software has its own set of challenges. A good example is porting data into and out of applications. First, you must measure the raw data, which is done using tools such as high-performance libraries for instrument control and data acquisition. Second, you must communicate between programs using another set of technologies. Some applications simply save results to a file, others may use custom TCP/IP networking solutions, DDE, or ActiveX. Each I/O mechanism has its own issues and requires certain expertise to implement.

DataSocket, a technology that is part of the National Instruments measurement suite, is a single, easy-to-use interface that provides easy access to several I/O mechanisms without entangling the end user in the low-level details. It pulls together established communication technologies for measurement and automation in much the same way that a web browser pulls together different Internet technologies into one easy-to-use tool.

Broadcasting Data – A Simple Example

National Instruments tools make it easy to configure a stand-alone measurement system. Now let’s take the next step. Suppose you want to share the measurements with several machines. A typical scenario is a college lab where one machine controls the experiment while several students do their own real-time analysis from individual workstations. Historically, to do this you would have turned to the TCP library to distribute the data. While these libraries can solve the problem, there are several steps to complete:

- Pick a TCP/IP port number (and hope it is not in use by any other apps on the system)
- Define the protocol (e.g. what gets sent when)
- Configure the Server to listens on selected port and create connection when client initiates request.

- Configure the Server to flatten the data and writes to all connections.
- Manage any errors
- Configure the Client applications to connect to the selected port, unflatten the data, and display it.

This is only an overview, but already you find several details that most developers would rather not have to deal with. Of course, when you make changes to the server, such as adding a new data item, you have to fix all the clients as well. With enough work, you can make a very robust implementation, but you have added a lot of code to your simple program.

To perform the same task using the DataSocket technology you perform two basics steps:

- Open a datasocket connection using a name you choose to identify the data.
- Write data to that connection as you compute new results

In this case, the low-level TCP/IP programming has been done for you.

Measurement Specific

DataSocket technology was designed from the ground up to meet the needs of measurement and automation engineers. For example, with TCP/IP you have to write code to convert your measurement data to an unstructured stream of bytes in the broadcasting application, as well as code to parse the stream of bytes back into its original form in subscribing applications. DataSocket, however, transfers data in a self-describing format that can represent data in an unlimited number of formats, including strings, scalars, Booleans, and waveforms. The DataSocket read and write operations transparently convert your measurements to and from the underlying byte streams for you, eliminating the need to write complicated parsing code. Furthermore, using the DataSocket data format, you can associate user-definable attributes with data. For example, you might associate a time stamp with a temperature measurement, or a sampling rate with an array. DataSocket greatly simplifies working with measurement data.

A URL to Any Data Source

Before you can go much further, it is important to understand briefly how DataSocket connects to different I/O technologies. It all starts with how you name the device or resource you are transferring data to or from. Typically an I/O library will have a 'Open' function to which you pass a name or number to identify the source you want to read from or write to. For file I/O, the resource name is a file path, for TCP/IP there are two parts to the name – a machine name and port number. With DataSocket the resource name is in the form of a URL (uniform resource locator) much like the familiar web address used by a web browser. Consider how a web browser would interpret the URL **http://www.natinst.com/datasocket**. It tells the browser to use the TCP/IP-based protocol called HTTP (hyper text transfer protocol) to connect to the machine named www.natinst.com and to fetch the web page named datasocket. The URL is different from the names used by most I/O technologies in that it not only defines what you are interested in, it also indicates how to get it. The "how" encoded in the first part of the URL is called the access method or protocol. Web browsers typically use several access methods, such as HTTP, HTTPS (encrypted HTTP), FTP (file transfer protocol), and FILE (for reading files on your local machine). Of course, you don't usually care how the page is loaded, you just

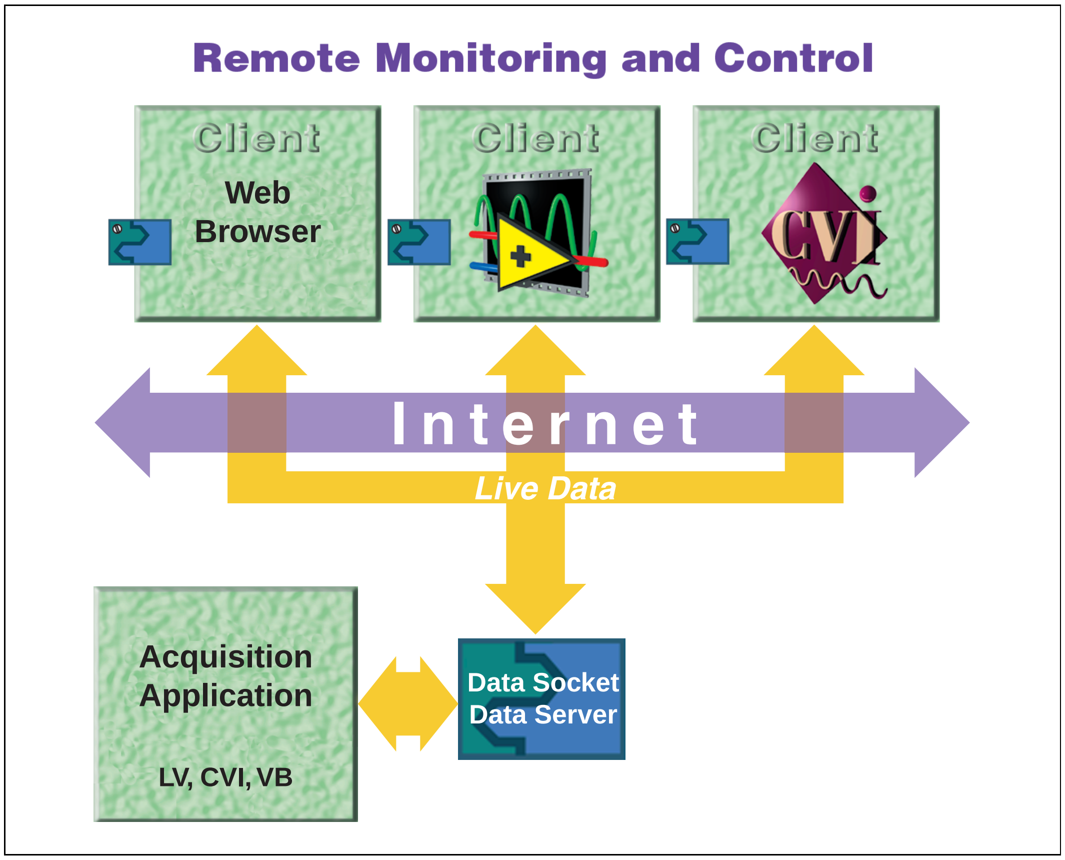
want to see it! DataSocket takes the same approach for measurement data. For example, in the data sharing example described above, DataSocket could have used the following URL to connect to a data item: **dstp://mytestmachine/wave1**
The "dstp" in the front tells DataSocket to open a data socket transfer protocol connection to my test machine and fetch a signal called wave1. Had the URL started with "file," the data would have been fetched from a file instead of the DataSocket server.

Several subsequent examples in this document show in more detail how DataSocket combined with the protocols it uses can solve common but challenging measurement and automation integration tasks.

What is DataSocket?

DataSocket, a new programming technology based on industry-standard TCP/IP, simplifies live data exchange between different applications on one computer or between computers connected via a network. Although a variety of different technologies exist today to share data between applications, such as TCP/IP and DDE, most of these tools are not targeted for live data transfer. DataSocket implements an easy-to-use, high-performance programming interface designed for sharing and publishing live data in measurement and automation applications.

DataSocket consists of two pieces – the DataSocket API and the DataSocket Server. The DataSocket API presents a single interface for communicating with multiple data types from multiple languages. DataSocket Server simplifies Internet communication by managing TCP/IP programming for you.



Commonality – DataSocket API

DataSocket is a single, unified, end-user API based on URLs for connecting to measurement and automation data located anywhere, be it on a local computer or anywhere on the Internet. It is a protocol-independent, language-independent, and OS-independent API designed to simplify binary data publishing. The DataSocket API is implemented as an ActiveX control, a LabWindows/CVI C library, and a set of LabVIEW VIs, so you can use it in any programming environment.

The DataSocket API automatically converts the user's measurement data into a stream of bytes that is sent across the network. The subscribing DataSocket application automatically converts the stream of bytes back into its original form. This automatic conversion eliminates network complexity, which accounts for a substantial amount of code that you must write when using TCP/IP libraries.

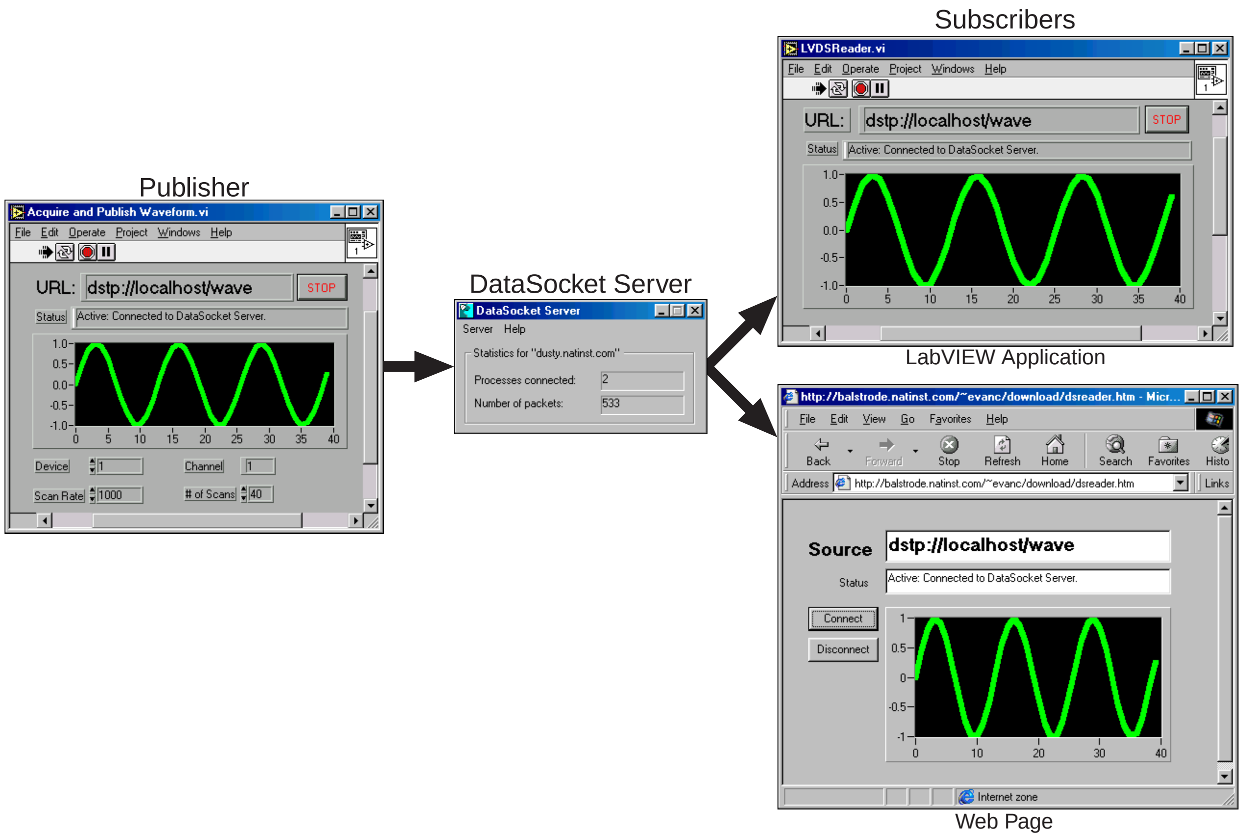
Learning the DataSocket API is simple. It consists of four basic actions (open, read, write, and close) that are similar to standard file I/O calls. You can use the same DataSocket API in your programs to read data from:

- Data items on HTTP servers
- Data items on FTP servers
- Local files
- Data items on OLE for Process Control (OPC) servers
- Data items on DSTP servers

Broadcasting Live Data – DataSocket Server

The DataSocket Server is a lightweight, stand-alone component with which programs using the DataSocket API can broadcast live measurement data at high rates across the Internet to several remote clients concurrently. DataSocket Server simplifies network TCP programming by automatically managing connections to clients.

Broadcasting data with the DataSocket Server requires three "actors" – a publisher, the DataSocket Server, and a subscriber. A publishing application uses the DataSocket API to write data to the server. A subscribing application uses the DataSocket API to read data from the server. Both the publishing and the subscribing applications are "clients" of the DataSocket Server. The three actors can reside on the same machine, but more often the three actors run on different machines. The ability to run the DataSocket server on another machine improves performance and provides security by isolating network connections from your measurement application.


Publisher
Acquire and Publish Waveform.vi
URL: dstp://localhost/wave
STOP
Status Active: Connected to DataSocket Server.
Device 1
Channel 1
Scan Rate 1000
# of Scans 40
DataSocket Server
Statistics for "dusty.natinst.com"
Processes connected: 2
Number of packets: 533
Subscribers
LVDSReader.vi
URL: dstp://localhost/wave
Status Active: Connected to DataSocket Server.
LabVIEW Application
http://balstrode.natinst.com/~evanc/download/dsreader.htm
Source dstp://localhost/wave
Status Active: Connected to DataSocket Server.
Connect
Disconnect
Web Page

The DataSocket Server restricts access to data by administering security and permissions. With DataSocket, you can share confidential measurement data over the Internet while preventing access by unauthorized viewers.

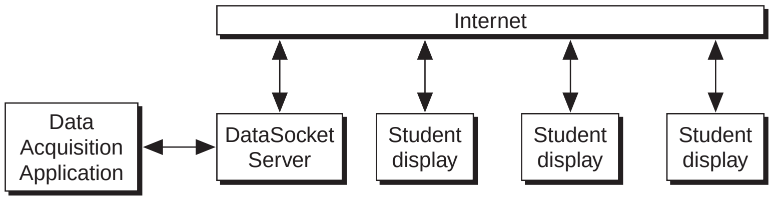
In essence, DataSocket Server is an easy-to-use, general solution to TCP/IP programming that replaces users' commonly written, home-grown networking code.

Applications Using DataSocket

Because DataSocket is a general-purpose programming tool for enhancing measurement applications, it can be used in a variety of different applications. Some example applications follow.

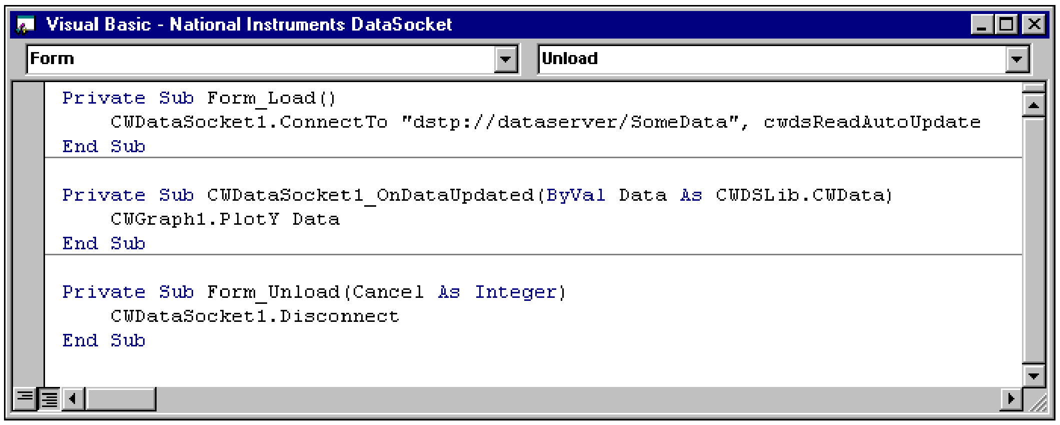
Building an Interactive Student Laboratory

Imagine a college signal-processing laboratory. Next week is the first day of class, and you, the professor, must prepare a laboratory experiment that introduces the students to signal analysis. The lab has 30 student workstations, all networked to the lab server. You want to demonstrate the fundamentals of analysis by acquiring and analyzing a signal on one of the student machines and then broadcasting it across the network to the rest of the student computers, so the students can see the effects of signal analysis without the need to acquire data on each machine.



You have already written an application that acquires and publishes data using DataSocket. You are now challenged with disseminating the data to the rest of the students. You decide to use the Web browser as a vehicle for displaying the live data. To build the Web page, you use Visual Basic to build a user interface and convert it to an HTML document. You can view an example Web interface on the previous page.

You use ComponentWorks DataSocket to read the data published to the server. Reading data from a DataSocket server consists of three easy steps: 1) open a session to the DataSocket server using the DataSocket ActiveX control, 2) read item from the DataSocket Server, and 3) close your DataSocket connection when your application is terminated. The figure below illustrates how to read data using ComponentWorks and Visual Basic.



```
Private Sub Form_Load()
    CWDataSocket1.ConnectTo "dstp://dataserver/SomeData", cwdsReadAutoUpdate
End Sub

Private Sub CWDataSocket1_OnDataUpdated(ByVal Data As CWDSLib.CWData)
    CWGraph1.PlotY Data
End Sub

Private Sub Form_Unload(Cancel As Integer)
    CWDataSocket1.Disconnect
End Sub
```

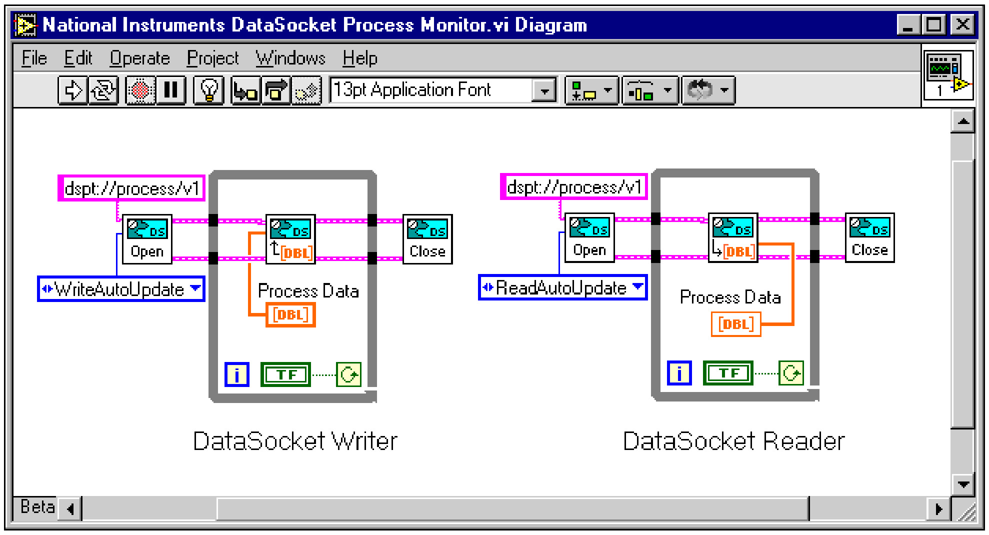
With DataSocket, you can view data from anywhere on a network or on the Internet. You can write applications that read information from a network or Internet with little or no coding. In this example you saw how to create a Visual Basic application for reading from a DataSocket server. For more information on how to build an interactive Web application, similar to the example described, see Application Note 127, *Building an Interactive Web Page with DataSocket*.

Using DataSocket to Monitor Process Variables

Imagine a cookie factory that makes several different types of cookies. Each type of cookie has its own production line, and each line has a computer that monitors the process variables. You are a systems engineer chosen to write a LabVIEW application that continuously monitors each of the process variables. Your application writes the live data to the central office over the local network. A computer in the central office gathers data and displays a live summary by production line and by process variable so that factory managers have an up-to-date picture of how the factory is performing.

Without DataSocket, you would have to write a TCP/IP server and client application to transfer the data from the factory floor to the central office. The server application would acquire the process data, flatten the data into a bit stream, and transfer the data to the server. The server reads the data, unflattens the data, and displays the data. In addition to the code required to read information from the server, the client application must also contain the code required to manage multiple connections, one connection for each process line. Writing all the low-level TCP/IP code to handle such data transfers would add a significant amount of overhead to the development process.

With DataSocket, you can easily handle the network communication required to move data from the factory floor to the central office. The sample LabVIEW block diagram below illustrates how to write and read process variable data using the DataSocket API. Because data is being written to a DataSocket server, the central office application does not need to implement extra code to handle the extra connections of multiple production lines. It simply reads the data item for each line.



This example illustrates how to broadcast data over a local network. You could expand upon the cookie scenario by passing process control information from the central office back to the production lines. Because DataSocket communication can span either a network or the Internet, the central office can be located either in the same building or halfway around the world.

Live Excel Reports

Imagine a company that produces 500,000 cellular telephones per month. You are the lead test technician and, your boss has just called from the UK and notified you that he needs updated production statistics for a meeting on Monday.

You are familiar with Excel and have developed small macros for generating reports in the past. Usually, generating such reports is not a problem, but time is short. You decide to create an Excel spreadsheet that can connect over the Internet to live production data. With the new spreadsheet, your boss can open the spreadsheet with his laptop, connect to the Internet, and press the UPDATE button to download the most recent production data. Now, whenever your boss needs an updated view of the system, he simply uses the Excel spreadsheet you created for him.

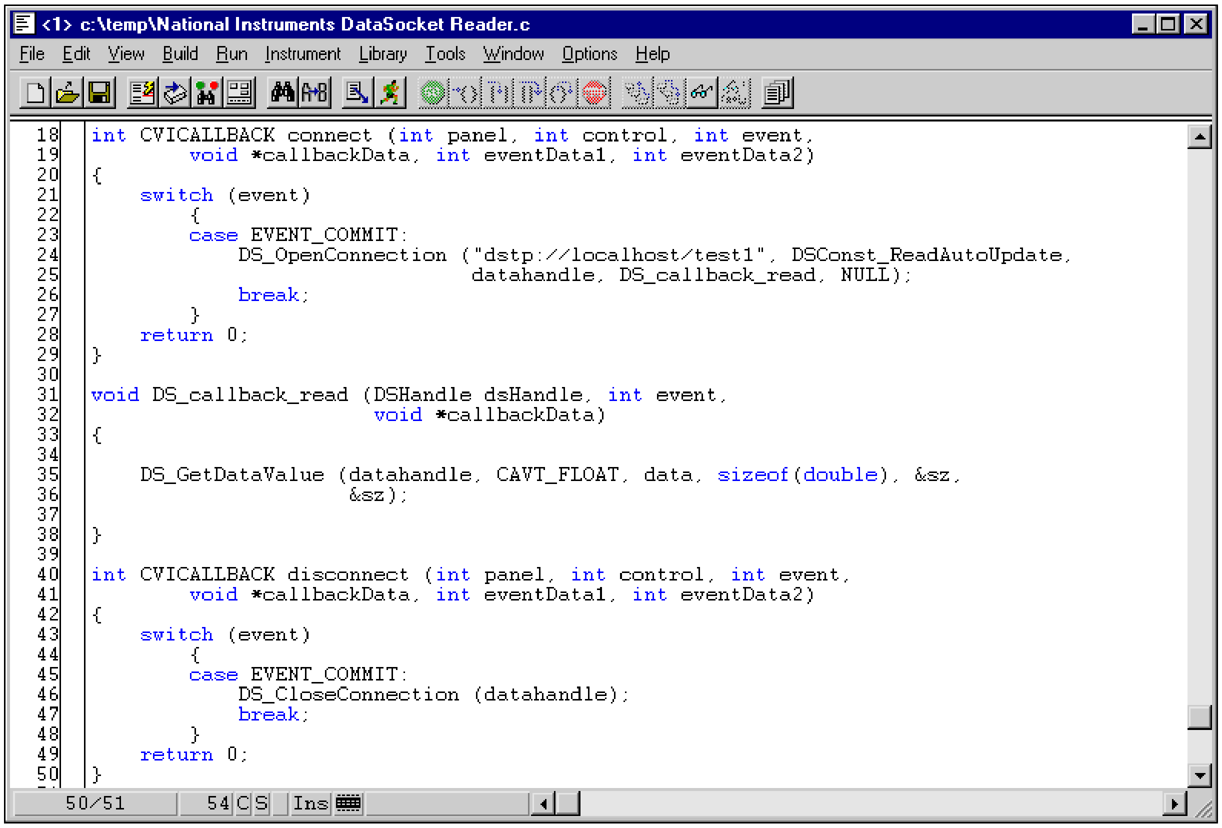
The spreadsheet is implemented using ComponentWorks DataSocket ActiveX controls. Because Excel macros are programmed using Visual Basic for Applications, you can copy many of the ComponentWorks examples directly into macros for use in Excel. By entering the URL to connect to the DataSocket server of the test machine, you are able to share your confidential test data over the Internet preventing access by unauthorized viewers.

Field Testing over the Internet

Imagine a New York City office equipment supplier who leases and maintains photocopiers. The company carries many different models of copiers and is continually adding more models to meet growing customer demands. Each technician in the company's field service group is armed with a laptop and a number of procedures for diagnosing and repairing copiers. The problem is that as the supplier continually adds more models of copiers, the field service group continually needs to be supplied with more diagnostic procedures. The office supplier is looking for a solution to optimize the field service group's ability to diagnose and repair copiers.

You are a measurement system solutions provider tasked with the responsibility of designing a mobile diagnostics application for the supplier's field service group. Your solution consists of moving all of the diagnostic procedures back to a single server in the central office where they easily can be kept up to date as well as readily accessed by field service agents using wireless modems attached to their laptop computers. You have designed a LabWindows/CVI application that interfaces with the copiers through a serial port and a data acquisition card. This application requests copier model information from the field service agent and sends the information back to the central office's DataSocket Server using the LabWindows/CVI DataSocket API. During test execution, your LabWindows/CVI application acquires data from the copier and sends it back to the DataSocket Server. The central office program determines which diagnostic test to run based upon the copier model supplied. TestStand, a test sequence manager, runs the diagnostic procedures from the central office. TestStand reads the DataSocket Server and uses the data passed from your application to carry out the diagnostic test. Once the test is complete, TestStand writes diagnostic information to the DataSocket Server. Your LabWindows/CVI application then reads the data from the server. The field service agent uses this information to repair the copier.

In the screen below, the LabWindows/CVI code illustrates three basic operations of DataSocket – Open, Read, and Close. Notice that the Open Connection function references the callback `DS_callback_read` function. When the data is updated at the DataSocket Server, LabWindows/CVI is notified and the `DS_callback_read` function is invoked. This event-driven architecture simplifies communicating with other applications.


<1> c:\temp\National Instruments DataSocket Reader.c
File Edit View Build Run Instrument Library Tools Window Options Help
50/51
54 C S
Ins

```
int CVICALLBACK connect (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
        {
        case EVENT_COMMIT:
            DS_OpenConnection ("dstp://localhost/test1", DSConst_ReadAutoUpdate,
                               datahandle, DS_callback_read, NULL);
            break;
        }
    return 0;
}

void DS_callback_read (DSHandle dsHandle, int event,
                       void *callbackData)
{

    DS_GetDataValue (datahandle, CAVT_FLOAT, data, sizeof(double), &sz,
                     &sz);

}

int CVICALLBACK disconnect (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
        {
        case EVENT_COMMIT:
            DS_CloseConnection (datahandle);
            break;
        }
    return 0;
}
```

Because the company moved all photocopier diagnostic procedures back to the central office, the field service technician no longer has to maintain any diagnostic routines. The company can now add new models without needing to update field computers. By using DataSocket, you did not need to implement TCP/IP programming to implement multipoint communication via a network. You were free to spend your time organizing and implementing your test application.

Measurement Communication for the Masses

DataSocket simplifies live data broadcast across the Internet. DataSocket makes it easier to exchange measurement data over the Internet, so you can concentrate your efforts on the design of your distributed measurement system. For more information on DataSocket, visit the DataSocket Website at **www.natinst.com/datasocket**.

How Do I Get DataSocket?

DataSocket is available for LabVIEW, LabWindows/CVI, and ComponentWorks. DataSocket is included with LabVIEW 5.1 and ComponentWorks 2.0. DataSocket for LabWindows/CVI is available via download from the DataSocket Web site, **www.natinst.com/datasocket**.