LXI system setup is quick and easy: a complete example

by Tim Ludy, Data Translation Inc.

One thing about LXI is that the physical connections are so simple, creating an instrument setup and then programming it is quicker and simpler than ever. To illustrate that point, this article walks through the process of configuring a test application using two LXI instruments: a function generator creates a signal to simulate a device under test, and an oscilloscope then reads the signal. Meanwhile, a simple program controls the two instruments and displays the results in graphical and text format. Visual Basic.NET example applications demonstrate how to modify example code from the instrument suppliers to communicate with LAN devices, and to demonstrate communicating with the device drivers, this example uses an application-building program called Measure Foundry.

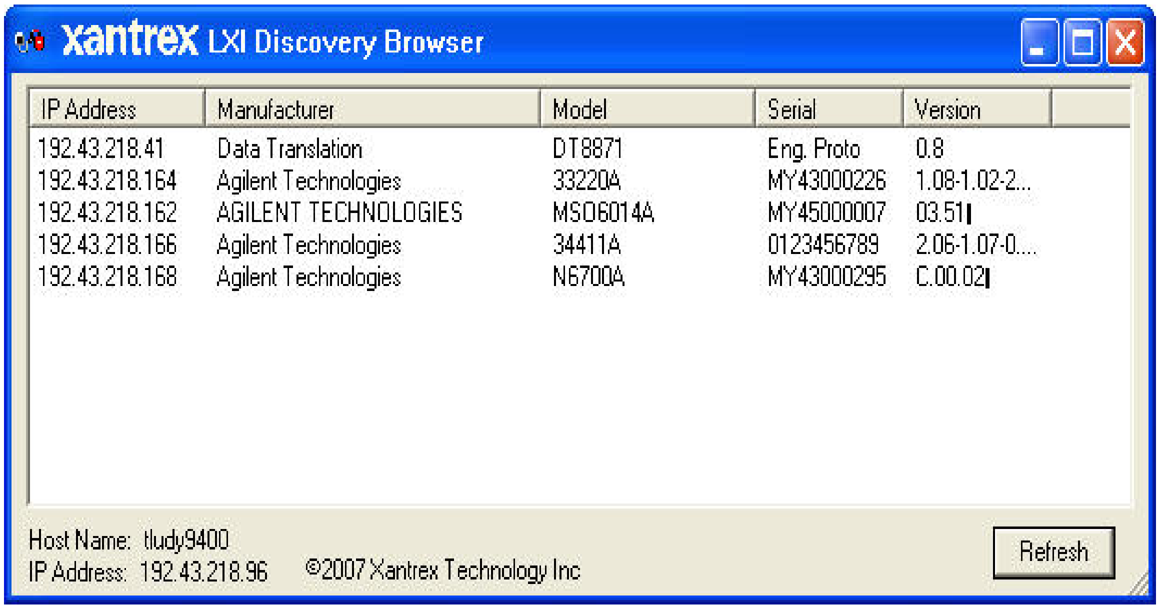
Hooking up the instruments

LXI-compliant instruments demonstrate predictable operation and behavior that makes them easy to set up, configure, and debug. All LXI-compliant instruments include a web interface that makes a user's manual virtually obsolete and makes it easy to configure and verify instrument settings. This step-by-step example describes how to configure multiple instruments on a private LAN using a commercial PC and a router. The recommended router utilizes DHCP (the Dynamic Host Configuration Protocol), which automatically assigns an IP address for an instrument. Most all new routers come with this DHCP server capability.

This example uses two instruments from Agilent: a Model 33220A function generator and a Model MSO6014A oscilloscope. Beginning with a running PC and router equipped with DHCP, follow these steps to configure each LXI instrument:

1. Connect the instrument to the router using a standard Ethernet cable.
2. Power on the instrument.
3. Reset the instrument to the default LAN configuration (use the instrument’s LAN reset).

4. Discover the instrument's IP address using a LXI discovery tool; this example uses a discovery utility written by Xantrex and that is available as a free download from the LXI Consortium website www.lxistandard.org . In this case, the utility's display looks like that in Fig. 1.



IP Address	Manufacturer	Model	Serial	Version
192.43.218.41	Data Translation	DT8871	Eng. Proto	0.8
192.43.218.164	Agilent Technologies	33220A	MY43000226	1.08-1.02-2...
192.43.218.162	AGILENT TECHNOLOGIES	MSO6014A	MY45000007	03.51
192.43.218.166	Agilent Technologies	34411A	0123456789	2.06-1.07-0....
192.43.218.168	Agilent Technologies	N6700A	MY43000295	C.00.02

Host Name: tludy9400
IP Address: 192.43.218.96 ©2007 Xantrex Technology Inc

Fig 1. Results of running the Xantrex discovery utility while several instruments, which are turned on, are on the LXI bus. This utility finds all LXI instruments on the given network. As illustrated above, three additional instruments were found on this network.

5. Open a standard web browser and enter an instrument's IP address (taken from the discovery utility) to access the instrument's web page.
6. Perform a LAN identify within the web browser that causes the instrument to identify itself by either flashing an LED, or in some way indicates to the user which device is being accessed. This is helpful when using multiple devices in a rack and is also an LXI requirement.

Working with drivers

Now that the instruments are configured and communicating over the network, the next step is to create an application using the software of choice. IVI drivers ship with any LXI-compliant instrument, and they simplify instrument programming by wrapping up functions in a common programming interface. The IVI shared component library is required to communicate with the instrument drivers and can be found on the IVI foundation website. www.ivifoundation.org

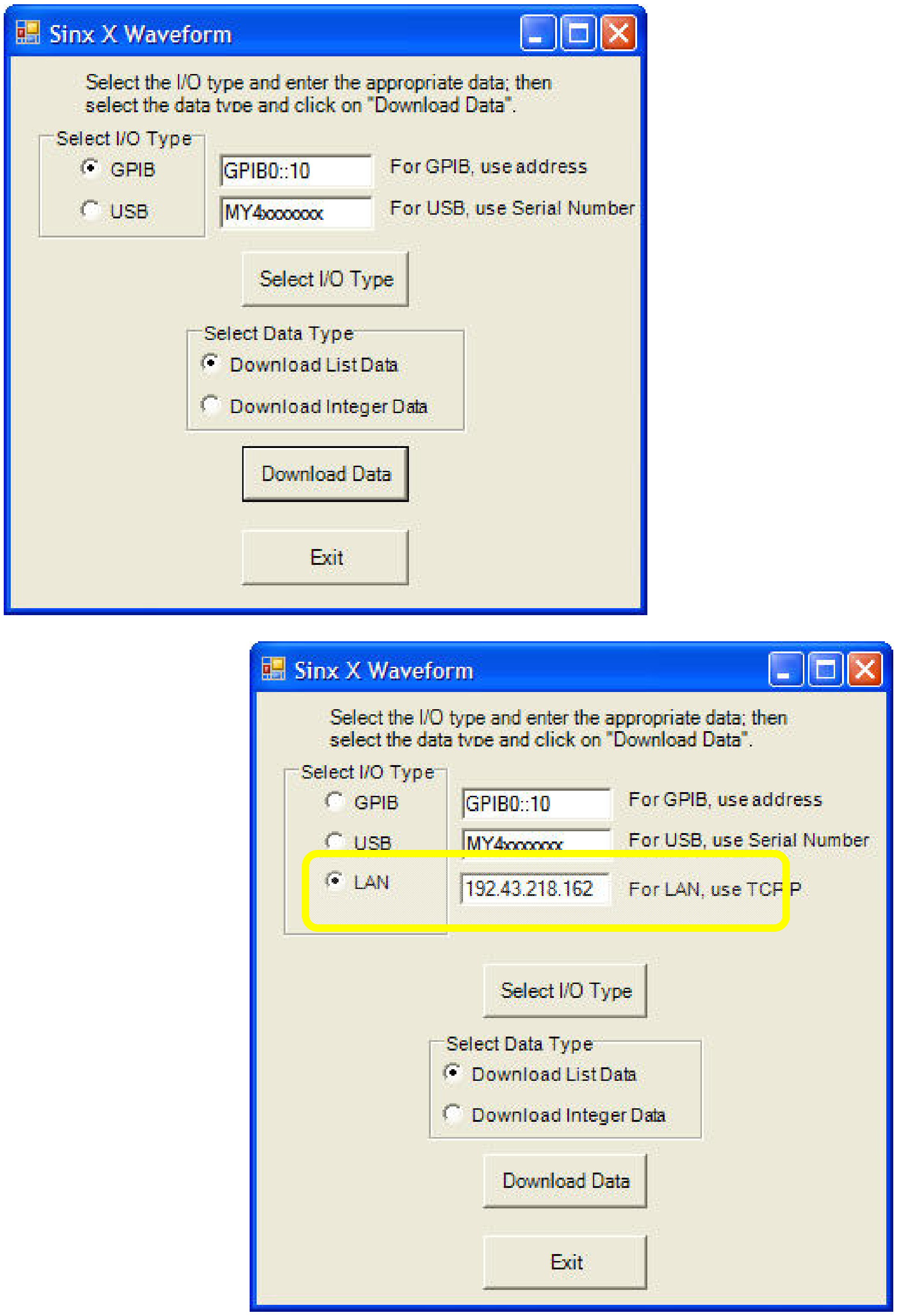
Install the IVI-COM instrument drivers for the function generator and oscilloscope. The drivers ship with the instruments, but if the disk isn’t readily at hand, you can also find these drivers on the IVI Foundation website or directly on the manufacturer's website. http://adn.tm.agilent.com/index.cgi?CONTENT_ID=1382

Now the desired application consists of this simple task: configure the LXI function generator to output a 1000-Hz sine wave. Make a physical connection between the function generator’s output and the input of the LXI oscilloscope. Then configure the function generator to create the signal and set up the scope to acquire it and send it to the PC for display and analysis.

Visual Basic example

One way to do this is to write a Visual Basic application that provides the desired functionality. An easy way to get the required code is to download example programs from the Agilent.com web site and modify them to communicate over the LAN. In particular, we get download these two programs: FGen and Scope. (http://adn.tm.agilent.com/index.cgi?CONTENT_ID=2257)

As written, the sine wave output program for the function generator and the scope input program accept GPIB and USB addresses but not a LAN address. Thus, the first thing to do is change the program so it also accepts an IP address. Do so by adding an I/O type and a corresponding text box (Fig 2).


Sinx X Waveform

Select the I/O type and enter the appropriate data; then select the data tvpe and click on "Download Data".

Select I/O Type
[✓ GPIB] GPIB0::10 For GPIB, use address
USB MY4xxxxxxx For USB, use Serial Number

Select Data Type
[✓ Download List Data]
Download Integer Data

Sinx X Waveform

Select the I/O type and enter the appropriate data; then select the data tvpe and click on "Download Data".

Select I/O Type
GPIB GPIB0::10 For GPIB, use address
USB MY4xxxxxxx For USB, use Serial Number
[✓ LAN] 192.43.218.162 For LAN, use TCP P

Select Data Type
[✓ Download List Data]
Download Integer Data

Fig 2. The modified VB code adds access for a LAN address.

The following code snippets show how to modify the program to access a LAN address.

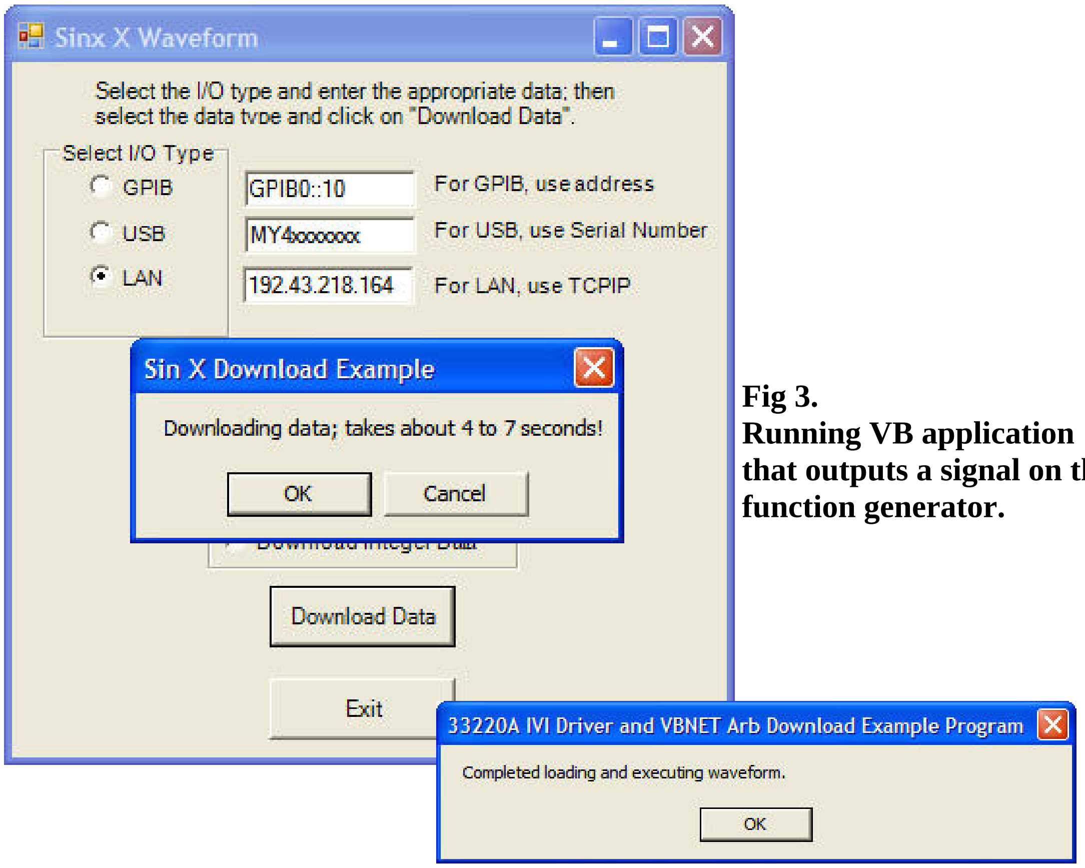
Original code:

```
' Get the instrument address form the text box
If GPIBType.Checked Then
addr = UCase$(gpibaddr.Text) + "::INSTR"
Else
addr = "usb0::2391::1031::" + usbaddr.Text +
"::INSTR"
End If

' Open the I/O session with the driver
Fg.Initialize(addr, True, False, "")
```

Modified code:

```
' Get the instrument address form the text box
If GPIBType.Checked Then
addr = UCase$(gpibaddr.Text) + "::INSTR"
End If
If usbtype.checked Then
addr = "usb0::2391::1031::" + usbaddr.Text +
"::INSTR"
End If
If LANtype.Checked Then
addr = "TCPIP::" + LANaddr.Text + "::INSTR"
End If

[[need Initialize line?]]
```

This example now instructs the function generator to generate an arbitrary waveform.



Select the I/O type and enter the appropriate data; then select the data type and click on "Download Data".

GPIB0::10 For GPIB, use address

MY4xxxxxxx For USB, use Serial Number

[✓ LAN] 192.43.218.164 For LAN, use TCPIP

Downloading data; takes about 4 to 7 seconds!

Completed loading and executing waveform.

Fig 3.
Running VB application that outputs a signal on the function generator.

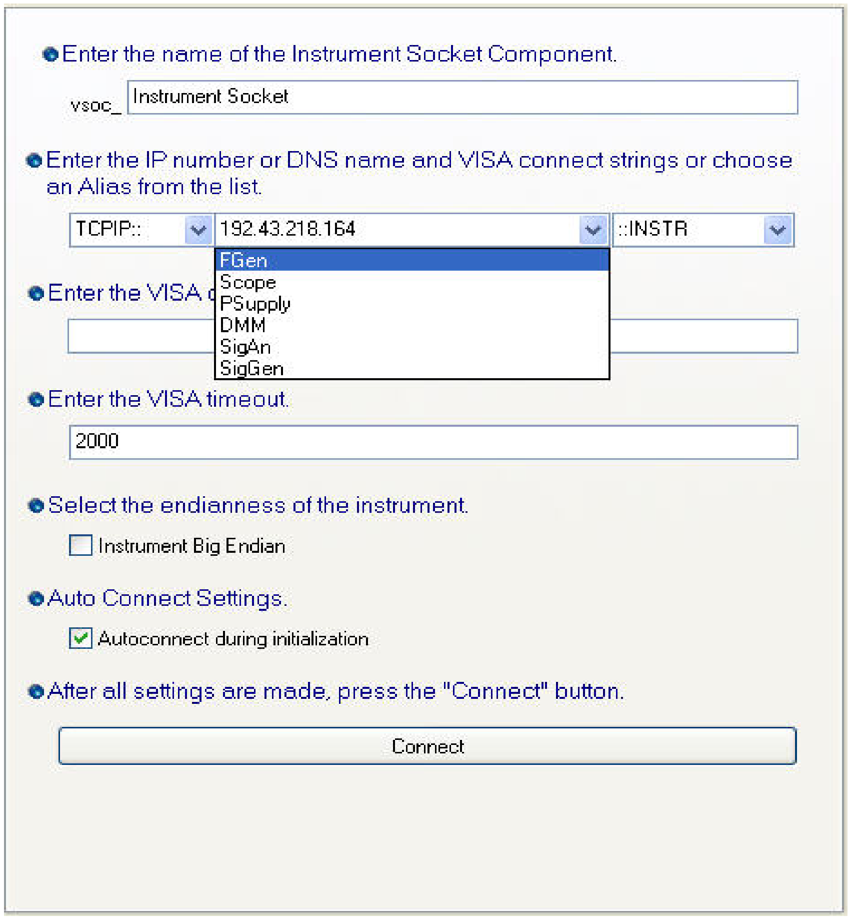
When the program finishes running, a new dialog box appears (Fig. 3). At this time the signal is being displayed on the oscilloscope.

The next step is to modify the second example program to address the scope, capture the signal over the hardwired connection between the two instruments, and read it in on the PC. Then you tie the two examples together to create the finished application. We leave this as an exercise for the reader and move on to the second application-building option to demonstrate the same concept with another type of program-development environment, specifically Measure Foundry.

The drag-and-drop method

When you open up Measure Foundry, a Form appears on the screen. To start, drag-and-drop an Instrument Socket component and also a Function Generator component onto the Form. Double-click the Instrument Socket component to access its property pages and configure its initial properties (Fig 4), which you can later change during run time using other controlling objects just as in programming languages such as Visual Basic and C. A video of this exact program is available at http://www.measurefoundry.com/support/videos/default.asp .

On the property page, enter the instrument's IP address, or if you set up an alias name for it, choose the alias from the list to make connection to the device. Click Connect and then click OK to open a connection to the instrument.



● Enter the name of the Instrument Socket Component.

vsoc_ Instrument Socket

● Enter the IP number or DNS name and VISA connect strings or choose an Alias from the list.

TCPIP:: | 192.43.218.164 | ::INSTR

FGen
Scope
PSupply
DMM
SigAn
SigGen

● Enter the VISA

● Enter the VISA timeout.

2000

● Select the endianness of the instrument.

☐ Instrument Big Endian

● Auto Connect Settings.

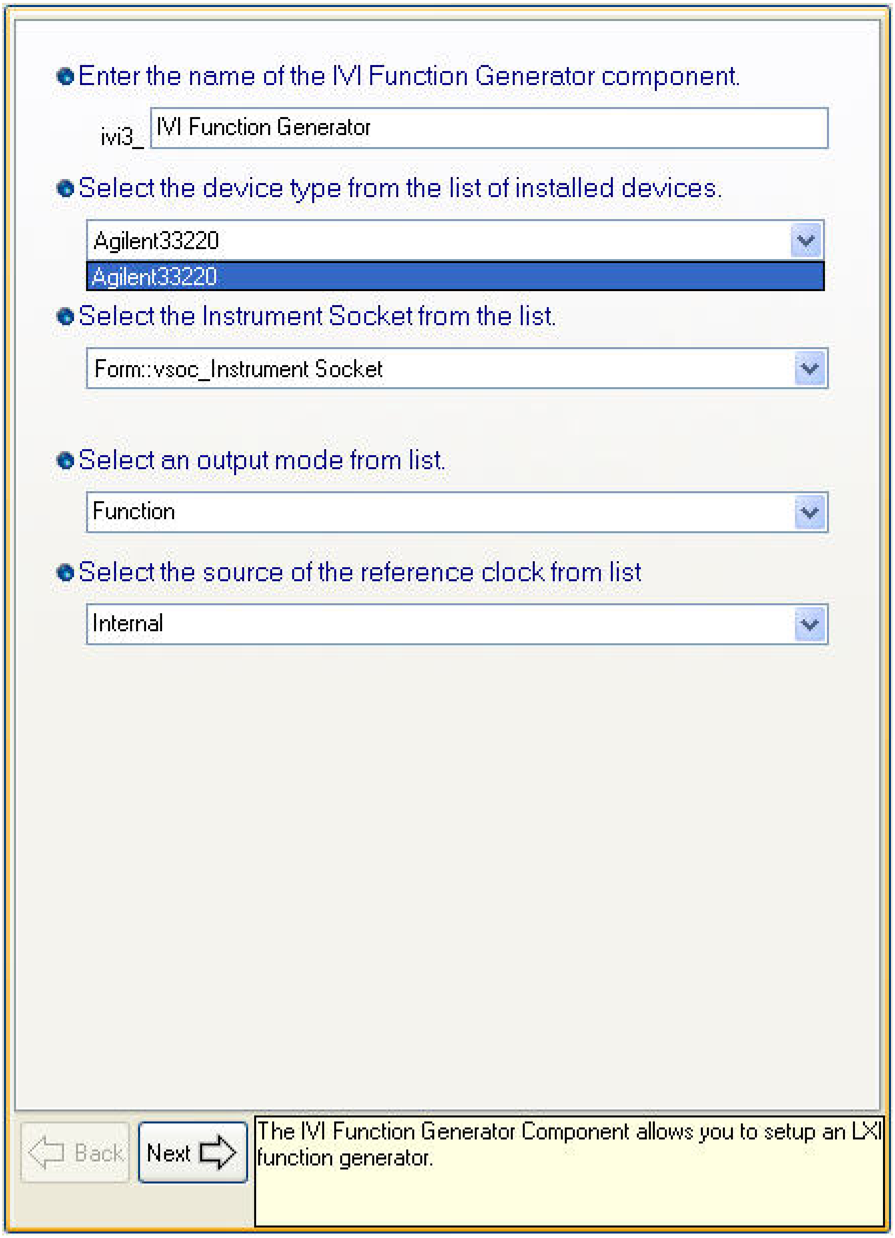
☑ Autoconnect during initialization

● After all settings are made, press the "Connect" button.

Connect

Fig 4: On the property page for the Instrument Socket that will access the function generator, either enter the instrument's IP address or select its alias from the drop-down menu.

A socket connection is required for each device being accessed. Next double-click the Function Generator component you dragged to the form to open its property page. If the device driver has been installed, select it from the available drivers in the pull-down menu (Fig 5). This associates the Function Generator component with the socket connection. The socket and device objects combine to provide the easiest access to standard LXI instruments.



Enter the name of the IVI Function Generator component.

ivi3_ IVI Function Generator

Select the device type from the list of installed devices.

Agilent33220

Agilent33220

Select the Instrument Socket from the list.

Form::vsoc_Instrument Socket

Select an output mode from list.

Function

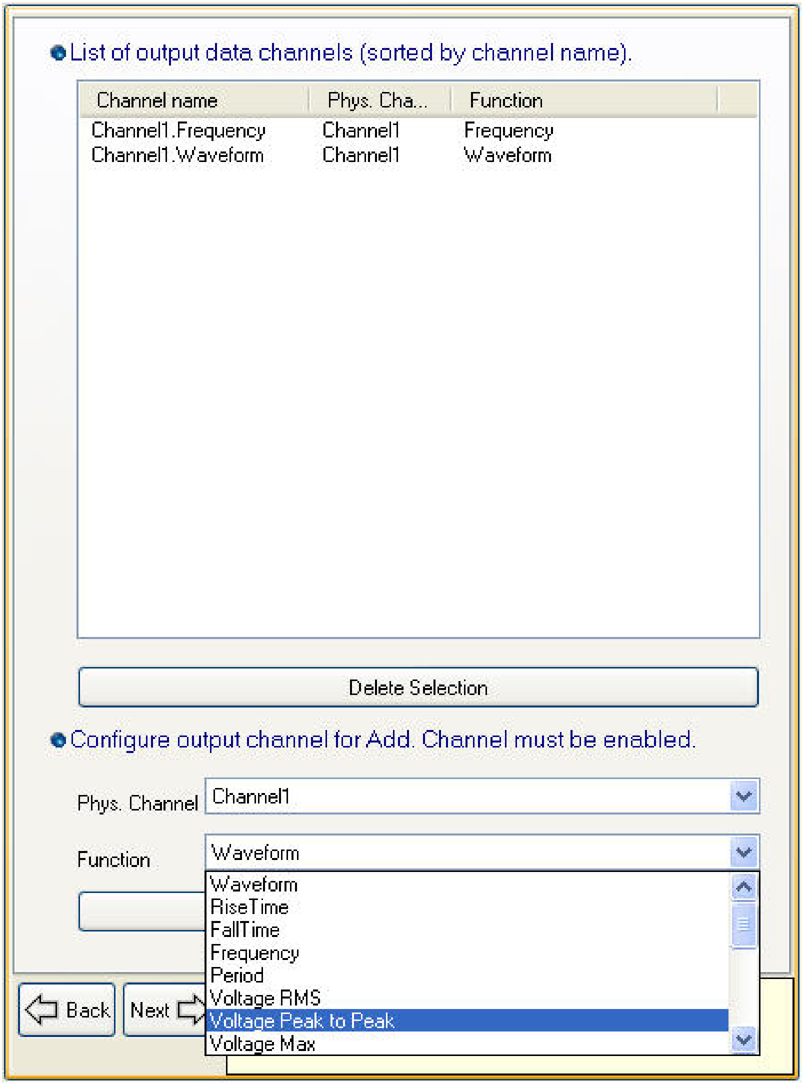
Select the source of the reference clock from list

Internal

The IVI Function Generator Component allows you to setup an LXI function generator.

Fig 5: In this page of the function generator's component, you select the appropriate driver

You also want to be able to access the scope, so drag and drop another Instrument Socket and an IVI Oscilloscope component onto the Form. Configure the second Instrument Socket to connect to the scope. Double-click the IVI Oscilloscope component to open its property pages, and then click Next to access additional properties. The oscilloscope driver publishes many measurements that can be very useful to include in an application. In this case, select the Waveform and Frequency channels for the component to publish as data channels to the rest of the application (Fig 6).


List of output data channels (sorted by channel name).

Channel name	Phys. Cha...	Function
Channel1.Frequency	Channel1	Frequency
Channel1.Waveform	Channel1	Waveform

Configure output channel for Add. Channel must be enabled.

Phys. Channel: Channel1

Function: Waveform

Fig 6: Publishing signal outputs from the scope so that these values are available throughout the software application.

In the application, you also want to be able to start/stop the function generator's waveform output and display the waveform from the hardware oscilloscope, respectively. For these purposes, add a Control Button and an Oscilloscope display components from the Foundry on the right side of the Form as shown in Fig. 7.

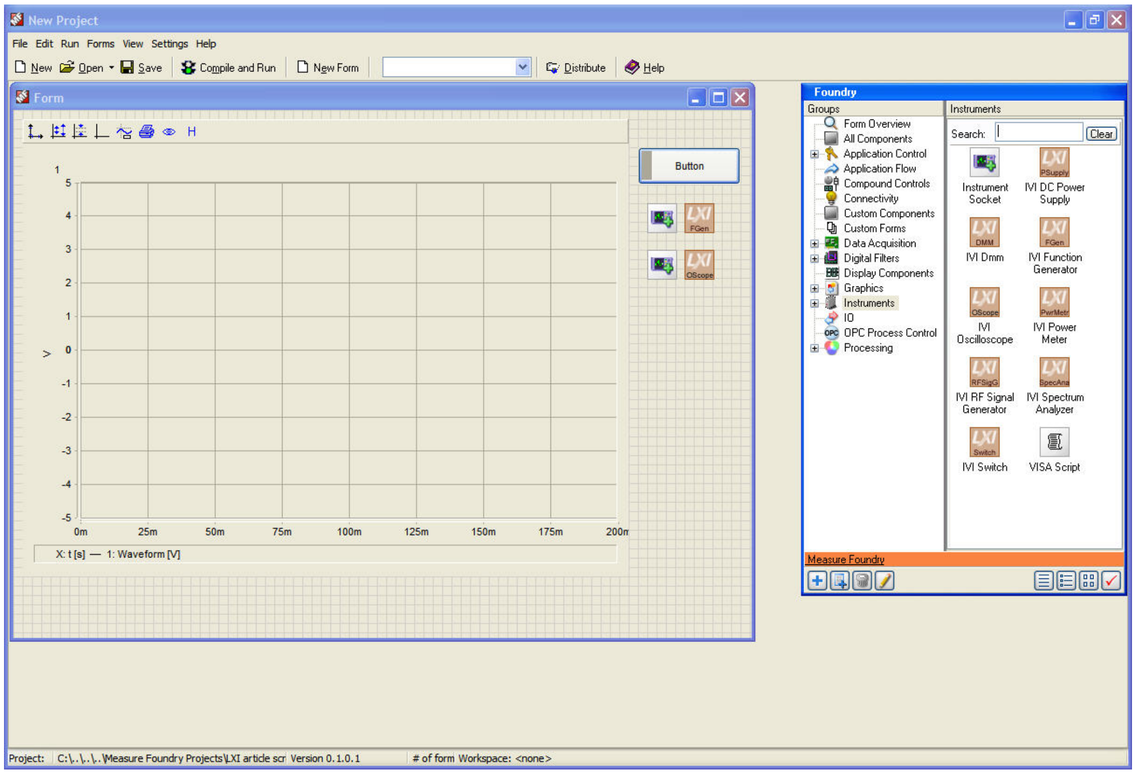


Fig 7: Adding a scope display and control button for the function generator.

To make the application even more useful, next add control components for the waveform type and frequency setting as well as a display component that reads the frequency measurement from the oscilloscope (Fig 8).

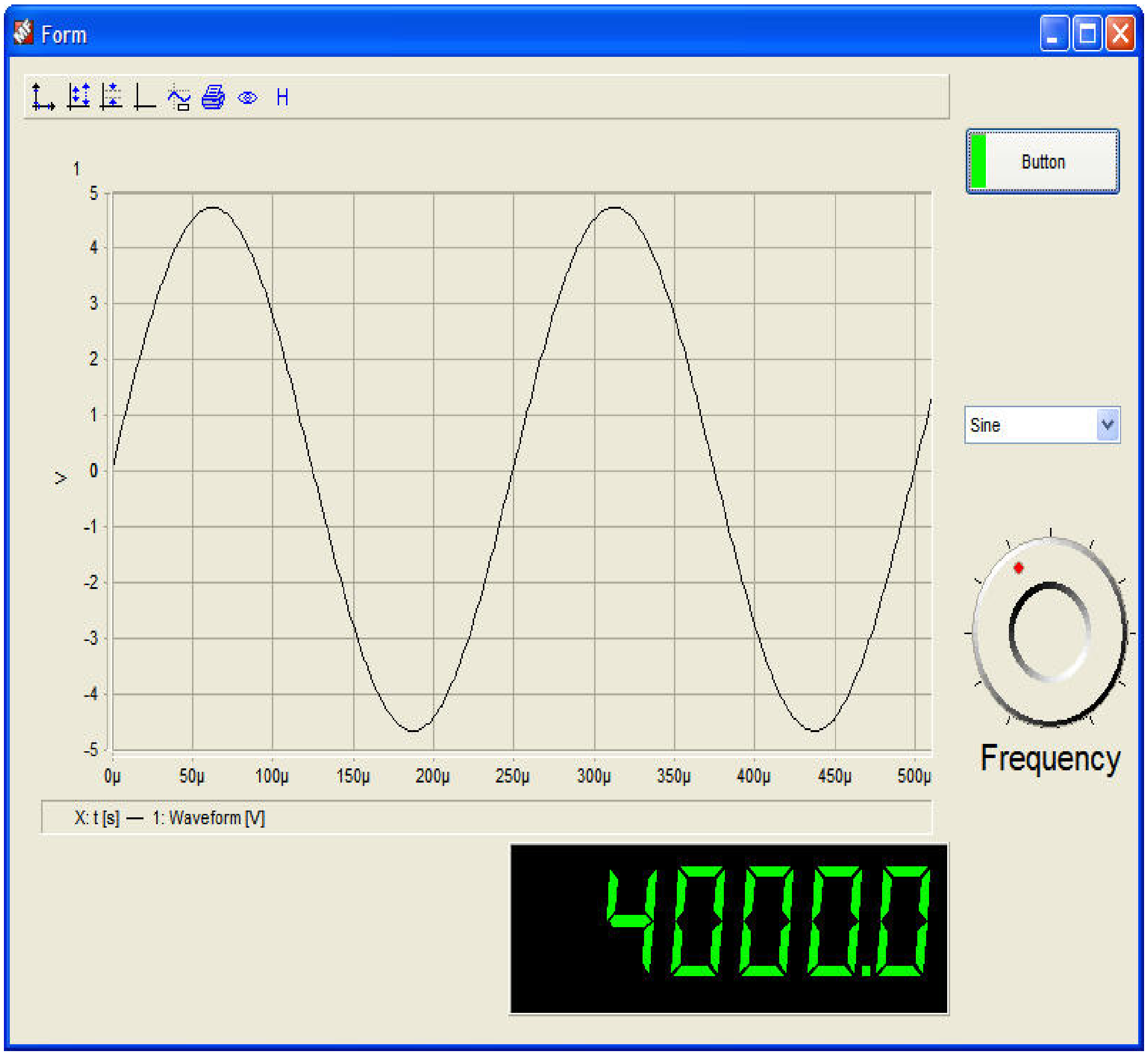


Fig. 8: Additional components on the Form allow selection of the waveform type and frequency.

The Windows application just described controls the amplitude and signal type that is output from the function generator, and it also captures the signal back from the oscilloscope. It displays both the waveform and the frequency-measurement data.

Better tools are here

Software is simply a tool, but it tends to be complicated for non-programmers to use efficiently when developing test applications. In order to speed creation and maintenance of automated test equipment, engineers need better tools. Utilizing the standards that exist today and eliminating the complex nature of traditional programming, this new breed of test-and-measurement development tools allows the engineer to build complete applications faster and make them easier to maintain. Test engineers can now focus on what needs to be tested and what results they require rather than how to do it.