C# and Visual Basic .Net Instrument Control Tutorial

Publish Date: Jan 10, 2007

Table of Contents

1. Introduction

(http://zone.ni.com/devzone/cda/tut/p/id/4359)Controlling instruments with Microsoft .Net applications is challenging because of the the variety of buses instruments provide to connect to PCs. For each bus type, developers must learn the bus protocols and application programming interfaces (APIs) to write an application. VISA (Virtual Instrument Software Architecture) is a high-level driver that abstracts the lower-level drivers for each instrument hardware bus type and provides a single API for communicating with instruments regardless of the bus interface. This application note discusses how you can take advantage of VISA for .NET applicatior the VISA .Net libraries in C# and Visual Basic .Net to communicate with GPIB, serial, Ethernet / LAN, IEEE 1394, and US instruments.

2. What is VISA?

VISA is a driver software architecture developed to unify communication with GPIB, serial, Ethernet/LAN, IEEE 1394, anc instruments and simplify your instrument control applications. With the VISA API developers can use GPIB, serial, Etherr IEEE 1394, and USB instruments.

VISA offers the following benefits:

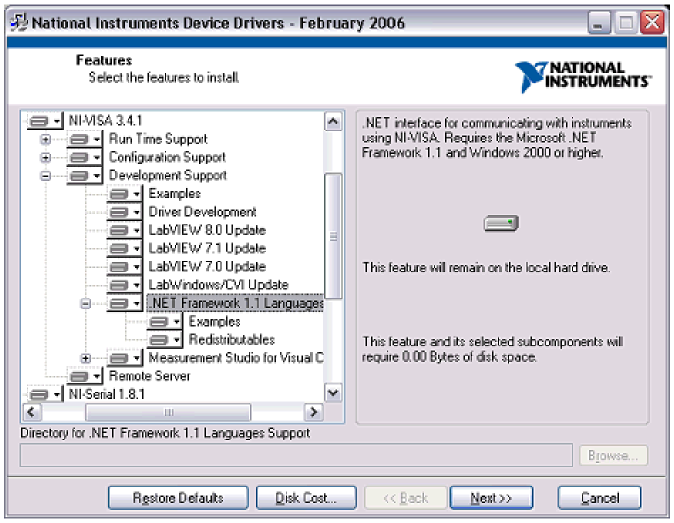
- Interface independence – VISA provides a single API with the same methods to communicate with instruments, regar the interface type. For example, the VISA command to send an ASCII string to a message-based instrument is the sa GPIB, serial, Ethernet/LAN, IEEE 1394, and USB interfaces.
- An object-oriented architecture that can easily adapt to new instrumentation interfaces developed in the future.
- Full-featured instrumentation programming functionality implemented in a very compact command set.

3. What is the VISA .Net API?

The VISA .Net API is an object-oriented interface made up of a set of .Net classes to communicate with instruments with You can use Net compliant languages such as C# and Visual Basic .Net.

4. How Can I Use the VISA .Net API?

The VISA .Net API is part of the free NI-VISA driver software. NI VISA can be downloaded free of charge from National I VISA (http://www.ni.com/visa/) for use with NI products. To install .Net support for NI VISA include .Net support by selecti Framework 1.1 Languages Support under the Development Support category as seen in Figure 1.



National Instruments Device Drivers - February 2006

Features
Select the features to install.

- NI-VISA 3.4.1
  - Run Time Support
  - Configuration Support
  - Development Support
    - Examples
    - Driver Development
    - LabVIEW 8.0 Update
    - LabVIEW 7.1 Update
    - LabVIEW 7.0 Update
    - LabWindows/CVI Update
    - .NET Framework 1.1 Languages
      - Examples
      - Redistributables
    - Measurement Studio for Visual C
  - Remote Server
- NI-Serial 1.8.1

.NET interface for communicating with instruments using NI-VISA. Requires the Microsoft .NET Framework 1.1 and Windows 2000 or higher.

This feature will remain on the local hard drive.

This feature and its selected subcomponents will require 0.00 Bytes of disk space.

Directory for .NET Framework 1.1 Languages Support

Figure 1: NI-VISA Installation Showing .Net Support

Adding References to the VisaNS Class Libraries

To create your C# or Visual Basic .Net instrument control application, begin with a Solution in Visual Studio. In the Solutio
Explorer you will find that each project has a category called References under which references are organized. In order
VISA .Net API class libraries add the NationalInstruments.Common and NationalInstruments.VisaNS references to your p
The NationalInstruments.Common namespace includes common methods and properties used by many of the National
Instruments drivers. The NationalInstruments.VisaNS namespace includes the methods and properties for the VISA .Net
add a new reference, right click on References category and select Add Reference.

Importing the VisaNS Namespace into Your Application
When you reference the VisaNS classes, you can use them in your project. By importing the VisaNS namespace, you ca
directly access objects by reducing typing. Import the VisaNS namespace by adding the following line of code to the begi
your application:

```
[C#]
using NationalInstruments.VisaNS;

[VB .Net]
Imports NationalInstruments.VisaNS
```

Determining the Instrument Resource Name
A resource is the GPIB, serial, Ethernet/LAN, IEEE 1394, and USB instrument or controller with which you want to comm
The resource name, also know as instrument descriptor, specifies the exact name and location of the VISA resource. Fo
the resource name ASRL1::INSTR describes the instrument on COM port 1 of your computer, and GPIB0::13::INSTR de
GPIB instrument at address 13. To communicate with an instrument using VISA. we must determine its address and instr
descriptor. The following link will take you to the Instrument Control Fundamentals hardware page where you can find art
how to find a visa resource for any hardware bus.

Instrument Control Fundamentals: Hardware and Bus Technologies (http://zone.ni.com/devzone/cda/tut/p/id/15)

We will use this resource name later on in the tutorial to specify which instrument we want to communicate with.

Opening a VISA Session
A session is a connection or link from the VISA .Net API to a resource. The VISA .Net API contains different Session clas
designed for particular applications. For example, the MessageBasedSession class is used for instruments that commun
sending and receiving messages in the form of text strings. In contrast, the RegisterBasedSession class is used to comm
with instruments that communicate by writing and reading from registers. In this tutorial we will discuss the MessageBase
class. If you want more information on the other types of sessions, please refer to the NI-VISA .Net Framework help docu
is installed with NI VISA.
The first step in creating a new MessageBasedSession is to declare it as a variable in our application. The following code
demonstrates how to declare a new MessageBasedSession:

```
[C#]
private MessageBasedSession mbSession;

[VB .Net]
Private mbSession As MessageBasedSession
```

It is important to declare this MessageBasedSession variable as a global variable in the form or class. Any function that
communicates with the instrument will have to access this object.

After declaring the MessageBasedSession variable, we need to instantiate a MessageBasedSession object. To do so, w
the static (or shared as it is called in Visual Basic) method, GetLocalManager of the ResourceManager class. This metho
instantiates a new ResourceManager object. This new ResourceManager object contains a function called Open, which t
instantiates a new Session object. Finally, we will cast the newly created Session object to a MessageBasedSession. Th

ResourceManager and Session classes contain much more functionality than we discuss in this tutorial. If you want more information on these classes, please refer to the NI-VISA .Net Framework help document that is installed with NI VISA. T following code shows how to perform these three operations in one line of code:

```
[C#]
mbSession = (MessageBasedSession)ResourceManager.GetLocalManager().
Open(resourceString.Text);

[VB .Net]
mbSession = CType(ResourceManager.GetLocalManager().
Open(resourceString.Text), MessageBasedSession)
```

The Open method of the ResourceManager objects accepts as a parameter the resource name of the instrument as a str discussed how to find the resource name for our instrument earlier in this tutorial.
In order to make our application more robust, we will add try and catch statements. Try and catch statements let us respo errors that occur during the execution of certain statements. In particular we will catch exceptions of the type InvalidCastE and all other exceptions by using the Exception type. The following code demonstrates how to implement try and catch statements:

```
[C#]

try
{
mbSession = (MessageBasedSession)ResourceManager.GetLocalManager().
Open(resourceString.Text);
}
catch(InvalidCastException)
{
MessageBox.Show("Resource selected must be a message-based session");
}
catch(Exception exp)
{
MessageBox.Show(exp.Message);
}


[VB .Net]

Try

mbSession = CType(ResourceManager.GetLocalManager().
Open(resourceString.Text), MessageBasedSession)

Catch exp As InvalidCastException

MessageBox.Show("Resource selected must be a message-based session")

Catch exp As Exception

MessageBox.Show(exp.Message)

End Try
```

Transferring Data

After creating a Session to communicate with our instrument, we can start transferring data to the instrument and readinc response. The three most common operations for communicating with a message-based instrument are query, write, anc The query operation writes a command to an instrument and reads back the response. On the other hand, the write comi sends a command to the instrument, and the read command reads information from the instrument. The following piece c demonstrates how to use the query operation:

[C#]

```
try
{
string responseString = mbSession.Query(stringToWrite.Text);
}
catch(Exception exp)
{
MessageBox.Show(exp.Message);
}
```

[VB .Net]

```
Try

Dim responseString As String = mbSession.Query(stringToWrite.Text)

Catch exp As Exception

MessageBox.Show(exp.Message

End Try
```

The read and write operations, because they are methods of the MessageBasedSession class, are implemented very sir the query operation. If you want more information on these operations, please refer to the NI-VISA .Net Framework help that is installed with NI VISA.

Closing the VISA Session

To close the VISA session we created to communicate with our instrument, we must use the Dispose method of the MessageBasedSession class. This method releases any resources allocated to the session. The following code demons how to dispose of a session:

[C#]

```
mbSession.Dispose();
```

[VB .Net]

```
mbSession.Dispose()
```

5. Conclusion

National Instruments VISA is a fast-and-easy solution for diverse instrument control. The VISA .Net API offer an object-o interface for easily communicating with your instrument.