Agilent

IVI-COM driver and VISA-COM I/O programming examples in Microsoft Visual C#

Application Note

Abstract

This paper details the installation instructions and Visual C# (C-sharp) programming examples for Agilent Technologies IVI-COM instrument drivers and VISA-COM I/O. This application note will demonstrate examples from the N4965A multi-channel BERT controller and the N4962A serial BERT 12.5 Gb/s.

William Sitch
Agilent Technologies, Inc.
April 2010
AN21

Anticipate __Accelerate __Achieve

Agilent Technologies

Table of Contents

Agilent Technologies IVI-COM driver and VISA-COM I/O programming examples in Microsoft Visual C#

Agilent Technologies offers a full suite of IVI-COM instrument drivers and Visual C# programming examples for remote instrument control. The drivers and programming examples are available as free downloads from the Agilent Technologies website. IVI Shared Components are also required; these are available as a free download from the IVI Foundation website.

Agilent Technologies IVI-COM drivers use a low-level VISA-COM interface to communicate with instruments over GPIB and/or USB interfaces. This means a program using the IVI-COM driver can be used with an instrument connected to the computer with either a GPIB cable or a USB cable.

Agilent Technologies also details how to communicate with instruments using the lower-level VISA-COM I/O libraries in Visual C#. Examples are included in this application note.

What is an instrument driver?

An instrument driver is a software program that can be used to control a programmable instrument, like a PRBS generator, BER tester, oscilloscope, switch matrix, or a simple DC power supply. The software program is comprised of individual routines that control a particular function of the instrument, like configuring options, writing data, reading data, and triggering the instrument.

Instrument drivers simplify instrument control and reduce test development time by providing an easier method to interface with the instrument. If you want to integrate an Agilent Technologies instrument into a pre-existing test system, or to build a new test system, we recommend you use the Agilent Technologies instrument driver when writing software to communicate with the instrument.

What is an IVI driver?

IVI drivers are a standardized instrument application programming interface (API). The driver encapsulates the instrument SCPI command stack and presents the instrument functionality in a consistent manner to many different Application Development Environments (ADEs).

What this means in practice is that an IVI driver converts an object-oriented class-based command like:

N4965A multi-channel BERT controller.Channels.get_Item("1").Generator. Pattern.Name = Agilent TechnologiesN4965A multi-channel BERT controllerGeneratorPatternEnum.PRBS31

into the following SCPI command that is sent to the appropriate instrument:

:GEN:DATA:PATT:NAME PRBS31 (@1)

What is an IVI-COM driver?

IVI drivers are available in two flavors: IVI-C and IVI-COM. IVI-C drivers are developed for use in ANSI C development environments. IVI-COM drivers are developed for use in development environments that support the Component Object Model (COM).

IVI-COM drivers are used in the typical Windows-based PC ADEs like Visual C# and Visual Basic .NET, but also including Agilent VEE, NI LabVIEW, NI TestStand, MATLAB, and other ADEs. Agilent Technologies recommends IVI-COM libraries based on the ease of integration into these advanced ADEs.

What is Visual C#?

Visual C# (abbreviated VC#, pronounced C-sharp) is Microsoft's® implementation of the C# programming language. Because Microsoft wrote the C# programming language specification and their implementation is the most popular, it is safe to assume that when someone is talking about C# they are also talking about VC#.

VC# includes a graphical development environment and supports very rapid development of Windows®-based applications.

C# is a simple, modern, general-purpose, object-oriented programming language. It has been developed within the .NET initiative and is compatible with the 3.5 framework. C# is ideally suited to controlling instruments when paired with IVI drivers.

What is VISA-COM?

Virtual Instrument Software Architecture (VISA) libraries are used for communicating with devices over GPIB, USB, and a variety of other buses. VISA libraries are managed by the IVI Foundation (**http://ivifoundation.org/**) and are available in two flavours: VISA-C and VISA-COM. VISA-C libraries were developed for use in ANSI C development environments, and VISA-COM libraries were developed for use in environments that support Microsoft's Component Object Model (COM).

VISA libraries allow us to write one version of code that can talk to our instruments over either GPIB or USB, which is nice. Agilent Technologies recommends VISA-COM (versus VISA-C) libraries based on the ease of integration into advanced ADEs.

From the National Instruments (NI) webpage (http://www.ni.com/visa/):

> "VISA provides the programming interface between the hardware and development environments such as LabVIEW, LabWindows/CVI, and Measurement Studio for Microsoft Visual Studio®. NI-VISA™ is the National Instruments implementation of the VISA I/O standard.
>
> NI-VISA™ includes software libraries, interactive utilities such as NI Spy and the VISA Interactive Control, and configuration programs through Measurement and Automation Explorer for all your development needs. NI-VISA is standard across the National Instruments product line. With NI-VISA, you can feel confident that your software development will not become obsolete as your instrumentation interface hardware needs evolve into the future."

National Instruments (NI) provides the VISA libraries free of charge. You are eligible for a 'free' deployment license when you download the VISA libraries for use with an application written using NI software.

Agilent Technologies also offers VISA-C and VISA-COM drivers with the Agilent IO Library Suite. Both the NI and Agilent VISA implementations cover the core VISA specifications and include extensions to interface with NI or Agilent hardware. NI and Agilent VISA libraries can be installed on the same computer, but only one vendor's libraries will be referenced by the VISA-COM API.

Installation instructions for IVI drivers

IVI drivers provided by a manufacturer are the easiest way to interface with the instrument hardware also provided by that manufacturer. Agilent Technologies provides IVI drivers for all instruments that have GPIB and/or USB interface capability. Please see the individual instrument webpage linked from the Agilent Technologies website.

There are three steps to installing an IVI driver. The first two steps, installing VISA I/O and installing IVI Shared Components, are only necessary if you haven't previously installed these software items.

Step 1: Installing a VISA library for low-level hardware I/O

Agilent Technologies IVI drivers all use the low-level Virtual Instrument Software Architecture (VISA) libraries for communicating with devices over GPIB, USB, and a variety of other buses. VISA libraries from any company will work, but we recommend the NI VISA library or the Agilent IO Library Suite.

Skip this step if you already have a VISA library installed on your computer. You probably have VISA installed if you have installed software from National Instruments or Agilent Technologies to connect to an instrument using a LAN (Ethernet), USB, GPIB, or VXI bus.

National Instruments, who provides the ubiquitous LabVIEW software, also sells hardware devices for connecting USB ports to GPIB ports (GPIB-USB-HS), PCI cards that connect to GPIB ports (PCI-GPIB) or ethernet-enabled GPIB host controllers (GPIB-ENET/100). If you are using a NI product to connect your computer to your instrument, use the NI VISA library.

Agilent Technologies, who offers a competing visual programming language called VEE, also offers a VISA library package wrapped around their free IO Library Suite. If you are using Agilent products to connect your computer to your instrument, you might want to use the Agilent VISA library.

For a simple USB cable connection from your computer to the instrument, you can use the NI 488.2 VISA libraries. The following instructions are specific to the NI-488.2 VISA libraries.

1.1   Download and un-zip the software package

You can download the NI VISA libraries directly from the NI webpage. Find and download the latest version of the NI VISA 488.2 software. Make sure you download the version designed for your operating system and in your native language.

Alternatively, Agilent Technologies hosts a mirror of version 2.5 (from September 2008). This version works just fine and is hosted on our website, the filename is ni488225.exe, and it is linked from each product page. The file is 259 MB and takes a while to download.

After downloading, run the self-extracting file. You may be presented with a security warning message. If you feel it is prudent to continue, unzip the contents into a directory. This is temporary and you can delete it later.

1.2 Install the NI-488.2 software and select appropriate features

Proceed with the software installation when presented with a menu. There is no need to read the documentation. You do not need an additional license.

The installation will next display a feature selection window, shown below in Figure 1.

Check to ensure you are installing the following features:

* GPIB Analyzer
* Application Support -> LabVIEW Support
* NI-VISA -> Run Time Support -> GPIB
* NI-VISA -> Run Time Support -> Serial
* NI-VISA -> Run Time Support -> USB
* NI-VISA -> Run Time Support -> COM Support
* NI-VISA -> Run Time Support -> (all others, can't hurt)
* NI-VISA -> Configuration Support -> VISA Configuration
* NI Measurement & Automation Explorer
* NI Spy

You may wish to install other features as well, just make sure you get the important ones listed above. Continue through the license agreement and summary screens and install the software.
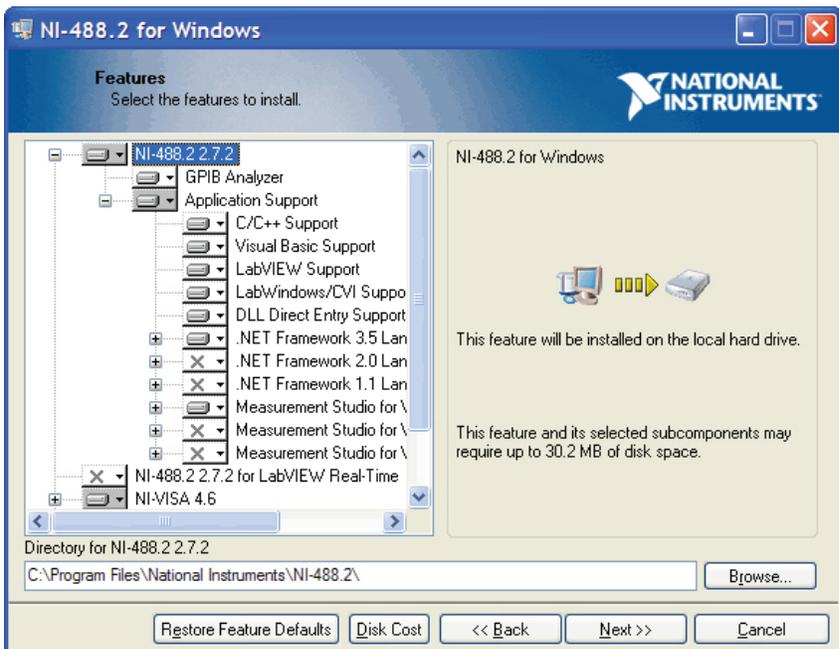

Figure 1. NI-488.2 software installation, feature selection screen

After the installation is complete, which may take as long as 30 minutes, you will probably be prompted to reboot.

Step 2: Installing the IVI shared components

IVI Shared Components allow all IVI drivers to use a common code base. This simplifies IVI driver development, and as detailed on the IVI Foundation website, at: **http://www.ivifoundation.org/shared_components/Default.aspx**

> "To improve users' experience when they combine drivers and other software from various vendors, it is important to have some key software components common to all implementations. In order to accomplish this, the IVI Foundation provides a standard set of shared components that must be used by all compliant drivers and ancillary software. These components provide services to drivers and driver clients that need to be common to all drivers, for instance, the administration of system-wide configuration."

Skip this step if you already have the IVI Shared Components installed on your computer.

2.1   Download the latest MSI or EXE package

Download the latest Microsoft Windows Installer (MSI) or executable installer (EXE) shared component package. These files are directly available from the IVI Foundation website at:
http://www.ivifoundation.org/shared_components/Default.aspx

As of 3 March 2010, the current version is 2.1.0.

After downloading, install the MSI or execute the EXE file. Accept the license agreement, install the files in whatever directory you feel appropriate (default is best), and install the shared components.
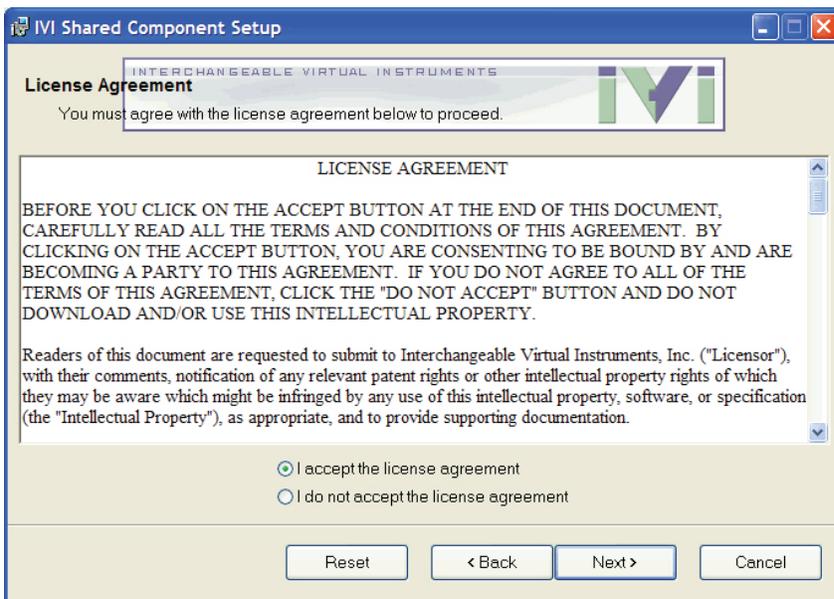

Figure 2. IVI Shared Component installation, license agreement screen

Step 3: Installing the Agilent Technologies IVI drivers

IVI drivers simplify communications with the instrument they are written for. Each type of instrument has its own dedicated IVI driver, and you will need to install each driver separately if you want to control multiple types of instruments.

3.1 Download the latest MSI or EXE package

Download the latest Agilent Technologies IVI drivers for the instruments you want to control. The drivers are packed in Microsoft Windows Installer (MSI) files located on the Agilent Technologies website. You can search the website and find the files from the individual product webpage, for example, the IVI driver for the N4965A-CTR MULTI-CHANNEL BERT CONTROLLER is available from: **http://www.Agilent Technologies.com/products/testmeas/N4965A-CTR MULTI-CHANNEL BERT CONTROLLER**

After downloading, install the MSI file. Agree to the license agreement, select 'Typical' installation (there are no customizations available), and install the IVI driver. Note that the help files can actually take several minutes to fully register on your computer. Don't cancel the process early.
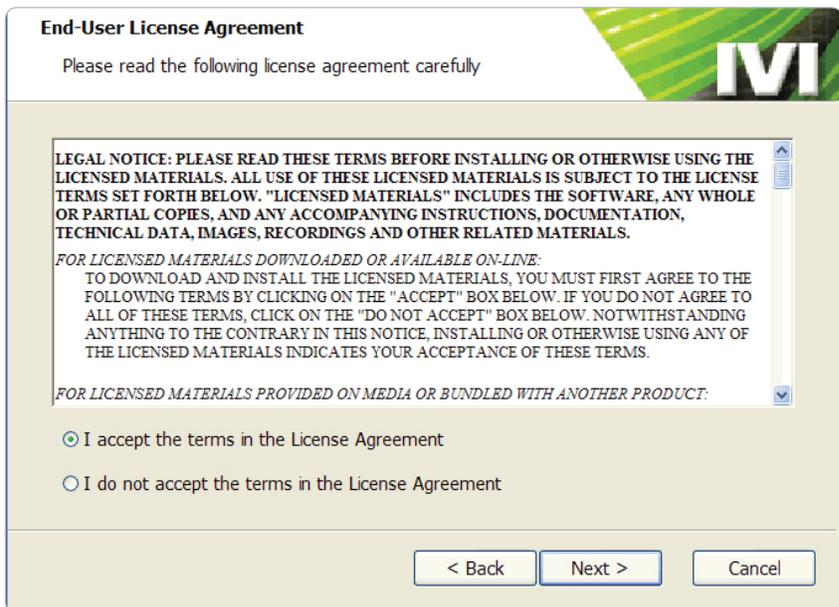

Figure 3. Agilent Technologies N4965A-CTR multi-channel BERT controller IVI driver installation, license agreement screen

After installation, the Agilent Technologies IVI drivers are stored in a subdirectory of the IVI Foundation installation folder. For default installations, these files will be under the directory: C:\Program Files\IVI Foundation\IVI\Drivers\.

VISA-COM programming examples in Visual C#

VISA-COM libraries are used to communicate with instruments on relatively low-level message-based communication sessions. For most instruments this means sending SCPI commands and reading/processing data read from the instrument.

For most customers and most applications, we do not recommend using VISA-COM libraries for instrument communication. Instead, we recommend using the Agilent Technologies IVI-COM drivers detailed below. The IVI-COM drivers have been optimized for each instrument, implement robust error-checking routines, and offer significant conveniences that most software developers will appreciate. Additionally, using IVI-COM drivers rather than VISA-COM libraries has a minimal impact on performance.

To use the VISA-COM libraries in your software development efforts, you first must reference the VISA-COM library.

4.1 Referencing VISA-COM libraries in Microsoft Visual Studio (VC#)

The Microsoft Visual Studio IDE requires you to reference the VISA-COM library in each Visual C# project you are intending to use the driver. First, in the Solution Explorer window, expand the project you want to add the VISA-COM reference to.
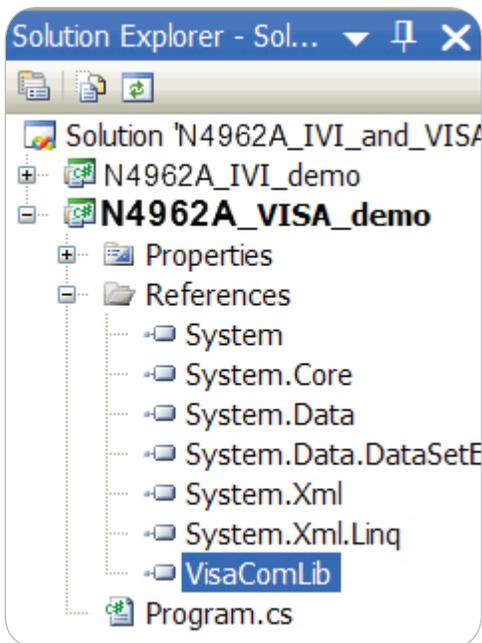
Figure 4. Microsoft Visual Studio, solution explorer window, project reference list

From the Solution Explorer window, right click on "References" and select "Add Reference". Alternatively, you can select the Project menu, and select "Add Reference" from the menu list.

A dialog box will open with a tabbed grouping of reference lists. Several options are available, including .NET, COM, and local Projects on your computer. The VISA libraries available from NI or Agilent are COM objects, and as such are available from the COM references.

Figure 5 . Microsoft Visual Studio, add reference window, VISA-COM library

Find and select the VISA-COM 3.0 Type Library from the list.

When you add this reference to your project, a VisaComLib reference will be listed under the project "References" treenode, as shown in Figure 4. This reference is now available in your project.

4.2   Using VISA-COM library references in your program code
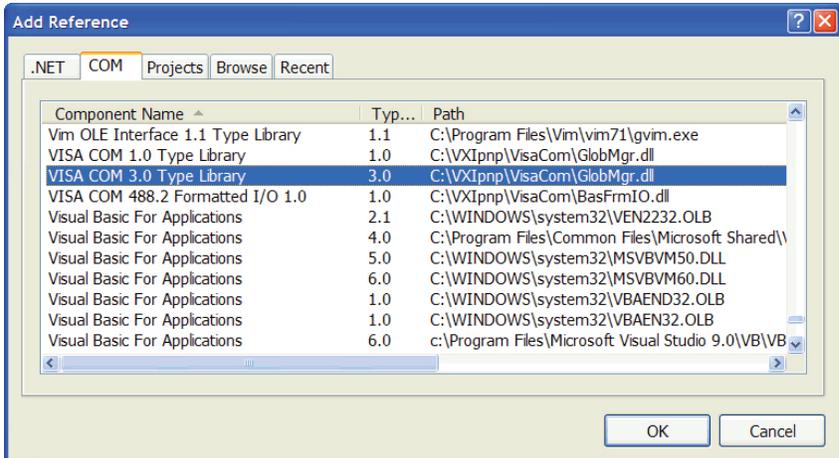
Now you've added the VISA-COM library reference to your project, you can interact with the VISA library from any code block in your project. Select the code file you want to interact with the instrument and add a "using" statement to simplify the namespace associated with the library reference.

```
Using Ivi.Visa.Interop;
```

Next you will want to create an instance of the VISA Resource Manager (to interface with the resources, like GPIB or USB or VXI or com ports) and create an instance of the VISA message-based session appropriate for your instrument. For Agilent Technologies instruments, which are all designed for IEEE 488.2 GPIB message-based communication, we recommend the FormattedIO488 class for instrument communication.

```
ResourceManager rMgr = new ResourceManagerClass();
FormattedIO488 src = new FormattedIO488Class();
```

Now we have a reference to the VISA resource manager and a reference to a message-based session we're going to use to communicate with an instrument. To open a message-based session with an instrument and set a reasonable timeout value, we use the following code.

```
src.IO = (IMessage)rMgr.Open(srcAddress, AccessMode.NO_
LOCK, 2000, null);
src.IO.Timeout = 2000;
```

11

4.3 Communicating with an instrument using the VISA-COM library

The next step is to assign the message-based session to a specific instrument using the resource manager. There are easy and difficult methods to do this, the hardest of which would be an automated scan of the resources available.

The code example shown below details how you connect to an instrument where the GPIB address is known.

4.4 10-second BER measurement example using the N4962A serial BERT 12.5 Gb/s and N4963A clock synthesizer 13.5 GHz with a VISA-COM library in Visual C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Ivi.Visa.Interop;

namespace TG1B1A_VISA_demo
{
    class Program
    {
        static void Main(string[] args)
        {
            // resource manager and message-based session manager
            ResourceManager rMgr = new ResourceManagerClass();
            FormattedIO488 src = new FormattedIO488Class();
            FormattedIO488 BERT = new FormattedIO488Class();

            // measurement setup
            string srcAddress = "GPIB::16";  // GPIB address of N4963A CLOCK
SYNTHESIZER13.5 GHZ
            double srcFreq = 10e9;           // frequency in Hz
            string BERTAddress = "GPIB::5";  // GPIB address of N4962A SERIAL BERT 12.5
GB/S
            string BERTPattern = "PRBS7";    // PRBS pattern
            double BERTAmplitude = 0.5;      // output amplitude
            double BERTGateTime = 10;        // measurement gate time
            double BERTErrThreshold = 1e-5;  // pre-measurement error threshold
            bool   BERTErrInj = false;       // error injection
            string BERTErrInjRate = "1E2";   // error injection rate

            // connect to instruments
            src.IO = (IMessage)rMgr.Open(srcAddress, AccessMode.NO_LOCK, 2000, null);
            src.IO.Timeout = 2000;
            BERT.IO = (IMessage)rMgr.Open(BERTAddress, AccessMode.NO_LOCK,
                                          2000, null);
            BERT.IO.Timeout = 2000;

            // setup clock
            Console.Write("N4963A CLOCK SYNTHESIZER13.5 GHZ " + srcAddress + " setup..");
            src.IO.Clear();
            src.WriteString("*RST;*OPC?", true);
            string temp = src.ReadString();
            src.WriteString(":FREQ " + srcFreq, true);
            src.WriteString(":AMPL 1 1.5", true);
            src.WriteString(":AMPL 2 1.5", true);
            src.WriteString(":AMPL 3 0.8", true);
            src.WriteString(":AMPL 4 0.8", true);
            src.WriteString(":OUTP ON", true);
            Console.WriteLine("done");
```

```csharp
            // setup BERT
            Console.Write("N4962A SERIAL BERT 11.3 GB/S " + BERTAddress + " setup..");
            BERT.IO.Clear();
            BERT.WriteString("*RST;*OPC?", true);
            temp = BERT.ReadString();
            BERT.WriteString(":SOUR:ROSC:SOUR EXT", true);
            System.Threading.Thread.Sleep(200);
            BERT.WriteString(":SOUR:ROSC:FREQ " + (srcFreq/1e6), true);
            BERT.WriteString(":SOUR:PATT " + BERTPattern, true);
            BERT.WriteString(":SOUR:VOLT " + (BERTAmplitude*1000), true);
            if (BERTErrInj)
                BERT.WriteString(":SOUR:PRBS:IERR:RAT ERR" + BERTErrInjRate, true);
            BERT.WriteString(":MOD NORM", true);
            Console.WriteLine("done");

            // autoalign and check for error-free operation
            Console.Write("Autoalign and error check..");
            BERT.WriteString(":SENS:ROSC:PHAS AUTO", true);
            BERT.WriteString("*OPC?", true);
            temp = BERT.ReadString();
            BERT.WriteString(":SENS:ROSC:PHAS?", true);
            temp = BERT.ReadString().TrimEnd('\r', '\n');
            Console.Write(temp + " degrees..");
            BERT.WriteString(":SENS:SWE:TIME 0.1", true);
            BERT.WriteString(":TRIG:INIT", true);
            System.Threading.Thread.Sleep(100);
            BERT.WriteString(":TRIG:SAMP;:SENS:DATA? ALL", true);
            double[] results = (double[])BERT.ReadList(
                            IEEEASCIIType.ASCIIType_R8, ",");
            if (results[3] > BERTErrThreshold)
            {
                Console.WriteLine("failed");
                Console.WriteLine(results[2] + ", BER: " + results[3] +
                        "; bits: " + results[0] + "; errs: " + results[1]);
                Environment.Exit(1);
            }
            Console.WriteLine("passed");

            // start measurement
            Console.WriteLine(BERTGateTime.ToString() + "s BER measurement");
            BERT.WriteString(":SENS:SWE:TIME " + BERTGateTime.ToString(), true);
            BERT.WriteString(":TRIG:INIT", true);
            bool running = true;
            while (running)
            {
                BERT.WriteString(":STAT:OPER:MEAS:COND?", true);
                running = ((double)BERT.ReadNumber(
                            IEEEASCIIType.ASCIIType_R4, true) == 1);
                System.Threading.Thread.Sleep(1000);

                BERT.WriteString(":TRIG:SAMP;:SENS:DATA? ALL", true);
                results = (double[])BERT.ReadList(IEEEASCIIType.ASCIIType_R8, ",");
                Console.WriteLine(results[2] + ", BER: " + results[3] +
                        "; bits: " + results[0] + "; errs: " + results[1]);
                if (results[2] == BERTGateTime) running = false;
            }
        }
    }
}
```

4.5 Example code output

If the autophase procedure is unable to find a clock phase location that yields a BER result below the threshold, the output looks like:

```
N4963A CLOCK SYNTHESIZER13.5 GHZ GPIB::16 setup..done
N4962A SERIAL BERT 12.5 GB/S GPIB::5 setup..done
Autoalign and error check..14 degrees..failed
0.1, BER: 0.000100685; bits: 985661000; errs: 99241
```

When the autophase is able to successfully align the clock and data, the BER measurement runs for 10 seconds and the output looks like:

```
N4963A CLOCK SYNTHESIZER13.5 GHZ GPIB::16 setup..done
N4962A SERIAL BERT 11.3 GB/S GPIB::5 setup..done
Autoalign and error check..12 degrees..passed
10s BER measurement
1.021, BER: 0; bits: 9839830000; errs: 0
2.054, BER: 0; bits: 20049800000; errs: 0
3.087, BER: 0; bits: 30251400000; errs: 0
4.121, BER: 0; bits: 40460300000; errs: 0
5.157, BER: 0; bits: 50729000000; errs: 0
6.19, BER: 0; bits: 60974700000; errs: 0
7.223, BER: 0; bits: 70917300000; errs: 0
8.257, BER: 0; bits: 81128300000; errs: 0
9.29, BER: 0; bits: 91336200000; errs: 0
10, BER: 0; bits: 98367900000; errs: 0
```

4.6 Useful code snippets

Once you've referenced the VISA-COM library in your Visual Studio project, assigned a using command to simplify the object namespace, initialized a resource manager object and a message-based session object, and connected to an instrument, now what?

To reset the N4963A clock synthesizer13.5 GHz, set the frequency to a pre-specified value, turn all output amplitudes to the maximum level, and enable all outputs:

```
src.IO.Clear();
src.WriteString("*RST;*OPC?", true);
string temp = src.ReadString();
src.WriteString(":FREQ " + srcFreq, true);
src.WriteString(":AMPL 1 1.5", true);
src.WriteString(":AMPL 2 1.5", true);
src.WriteString(":AMPL 3 0.8", true);
src.WriteString(":AMPL 4 0.8", true);
src.WriteString(":OUTP ON", true);
```

The same type of code is used to reset and initialize the N4962A serial BERT 12.5 Gb/s. Once initialized, the following example performs an autophase and checks to see if the resulting BER is better than a pre-determined threshold value (if it is, it exits):

```
BERT.WriteString(":SENS:ROSC:PHAS AUTO", true);
BERT.WriteString("*OPC?", true);
temp = BERT.ReadString();
BERT.WriteString(":SENS:ROSC:PHAS?", true);
temp = BERT.ReadString().TrimEnd('\r', '\n');
BERT.WriteString(":SENS:SWE:TIME 0.1", true);
BERT.WriteString(":TRIG:INIT", true);
System.Threading.Thread.Sleep(100);
BERT.WriteString(":TRIG:SAMP;:SENS:DATA? ALL", true);
double[] results = (double[])BERT.ReadList(IEEEASCIIType.
ASCIIType_R8, ",");
if (results[3] > BERTErrThreshold)
{
    Environment.Exit(1);
}
```

Other examples are available. Please contact Agilent Technologies for more information.

IVI-COM Programming Examples in Visual C#

IVI drivers use the VISA-COM library to communicate with instruments, but the driver encapsulates error-checking and optimized instrument communication methods into a format that's much easier to use in an object-oriented programming language like Visual C#.

To use the Agilent Technologies IVI drivers in your software development efforts, you first must reference the IVI driver in the VC# project before writing code that uses the driver to interface to a particular instrument.

5.1   Referencing IVI-COM drivers in Microsoft Visual Studio (VC#)

The Visual Studio IDE requires you to reference drivers in each Visual C# project you are intending to use the driver. First, open the project you want to add the IVI driver reference to.
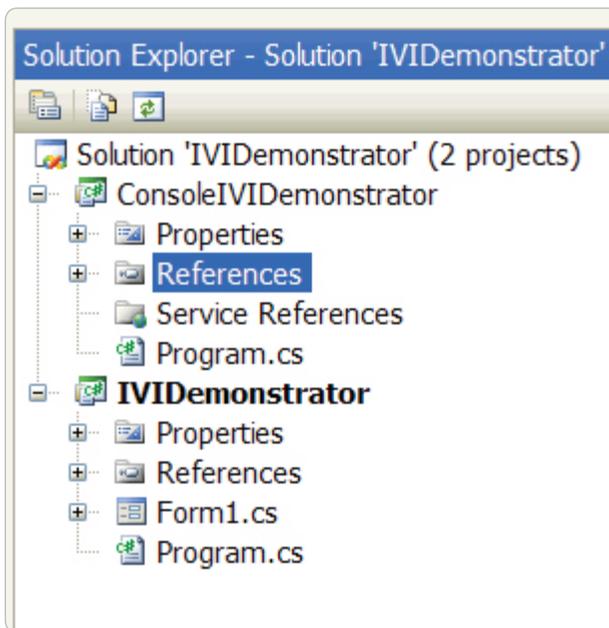

Figure 6. Microsoft Visual Studio, solution explorer window, project reference list

From the Solution Explorer window, right click on "References" and select "Add Reference". Alternatively, you can select the Project menu, and select "Add Reference" from the menu list.

A dialog box will open with a tabbed grouping of reference lists. Several options are available, including .NET, COM, and local Projects on your computer. The Agilent Technologies IVI drivers are IVI-COM drivers, and as such are available from the COM references.
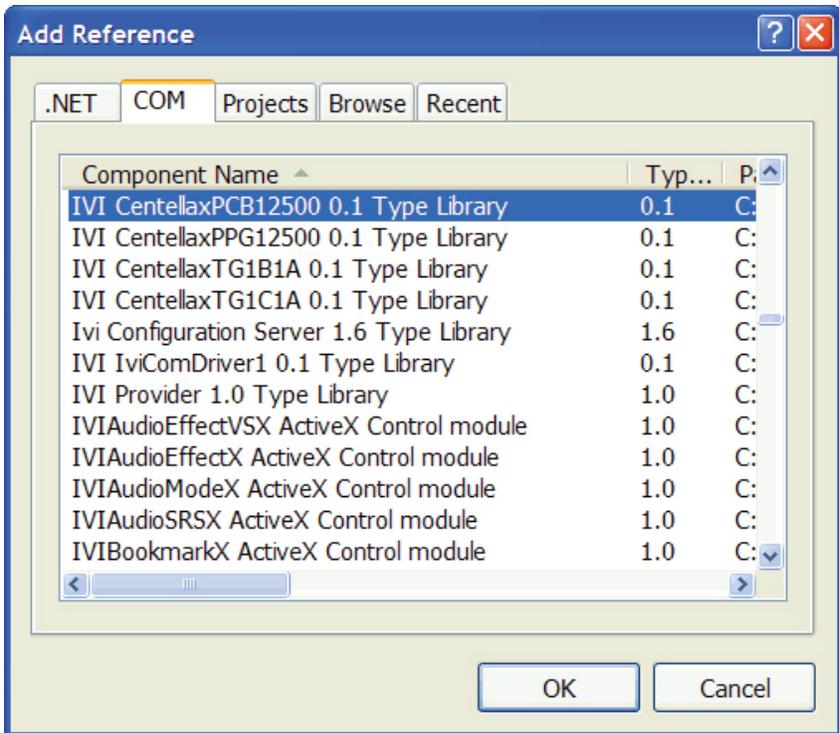
Figure 7. Microsoft Visual Studio, add reference window, Agilent Technologies IVI drivers list

Find the IVI drivers in the list, and select the specific IVI driver you want to add a reference to. In this example, we select IVI Agilent Technologies N4962A and IVI Agilent Technologies N4963A libraries.

When you add these references, another reference to IviDriverLib will be added to your project reference list. These references are now available in your project, but are not specifically associated with any particular block of code in your project.

5.2 Using IVI-COM driver references in your program code

Now you've added the IVI driver reference to your project, you can interact with the IVI driver from any code block in your project. Select the code file you want to interact with the instrument and add a "using" statement to simplify the namespace associated with the IVI driver reference.

```
Using Agilent Technologies.Agilent TechnologiesN4962A.Interop;
Using Agilent Technologies.Agilent TechnologiesN4963A.Interop;
```

Next you will want to create an instance of the Agilent Technologies IVI driver interface class. Because we want to share this reference with sub-classes, we use the protected variable definition. You could use private or public if you prefer.

```
protected Agilent TechnologiesN4963A syn;
protected Agilent TechnologiesN4962A BERT;
```

Now we have a reference to the IVI driver and variables that we've initialized to communicate with the driver. We're not done yet!

5.3 Communicating with an instrument using the IVI-COM driver

The next step is to connect an instance of the IVI driver class to a specific instrument. This code requires that you know the resource address of the instrument, which is a VISA string that identifies how the instrument is connected to the computer.

This code is shown in the next section.

5.4 10-second BER measurement example using the N4962A serial BERT 12.5 Gb/s and N4963A clock synthesizer 13.5 GHz with an IVI-COM driver in Visual C#

```csharp
using System;
using System.Text;
using Agilent Technologies.Agilent TechnologiesN4963A.Interop;
using Agilent Technologies.Agilent TechnologiesN4962A.Interop;
using Ivi.Driver.Interop;

namespace TG1B1A_IVI_demo
{
    class Program
    {
        static void Main(string[] args)
        {
            // vars
            double bitCount=0, errCount=0, elapTime=0, BER=0;

            // measurement setup
            string srcAddress = "GPIB::16";  // GPIB address of N4963A CLOCK SYNTHESIZER13.5
   GHZ
            double srcFreq = 10e9;           // frequency in Hz
            string BERTAddress = "GPIB::5";  // GPIB address of N4962A SERIAL BERT 12.5 GB/S
            double BERTAmplitude = 0.5;      // output amplitude
            double BERTGateTime = 10;        // measurement gate time
            double BERTErrThreshold = 1e-5;  // pre-measurement error threshold
            bool   BERTErrInj = true;        // error injection
            Agilent TechnologiesN4962AErrorInjectionEnum BERTErrInjRate =
                    Agilent TechnologiesN4962AErrorInjectionEnum.
```

```
                                  Agilent TechnologiesN4962AErrorInjection1E1PerSecond;
                Agilent TechnologiesN4962APatternEnum BERTPattern =
                                Agilent TechnologiesN4962APatternEnum.
                                Agilent TechnologiesN4962APatternPRBS31;  // PRBS pattern

            // instantiate variables and connect them to instruments
            //   (check IDN string, perform no RST, use driversetup)
            //    QueryInstrStatus = if SCPI commands will be
            //     followed with *ESR?, this is good practice
            //     to find errors, but will slow down comms
            string driverSetup = "QueryInstrStatus=false";
            Agilent TechnologiesN4963A src = new Agilent TechnologiesN4963A();
            Agilent TechnologiesN4962A BERT = new Agilent TechnologiesN4962A();
            src.Initialize(srcAddress, true, false, driverSetup);
            BERT.Initialize(BERTAddress, true, false, driverSetup);

            // make sure we're connected
            if (!src.Initialized)
            {
                Exception notFound = new Exception("N4963A CLOCK SYNTHESIZER13.5 GHZ not
at " + srcAddress);
                notFound.HelpLink = "Check GPIB address";
                throw notFound;
            }
            if (!BERT.Initialized)
            {
                Exception notFound = new Exception("N4962A SERIAL BERT 12.5 GB/S not at "
+ BERTAddress);
                notFound.HelpLink = "Check GPIB address";
                throw notFound;
            }

            // clear the instrument error queue
            int errorCode = -1;
            string errorMessage = "";
            while (errorCode != 0) src.Utility.ErrorQuery(ref errorCode,
                                                    ref errorMessage);
            errorCode = -1;
            while (errorCode != 0) BERT.Utility.ErrorQuery(ref errorCode,
                                                    ref errorMessage);

            // setup clock
            Console.Write("N4963A CLOCK SYNTHESIZER13.5 GHZ " + srcAddress + " setup..");
            src.Utility.Reset();
            src.Source.Frequency = srcFreq;
            src.Source.set_Amplitude("1", 1.5);
            src.Source.set_Amplitude("2", 1.5);
            src.Source.set_Amplitude("3", 0.8);
            src.Source.set_Amplitude("4", 0.8);
            src.Source.OutputEnable = true;
            Console.WriteLine("done");

            // setup BERT
            Console.Write("N4962A SERIAL BERT 11.3 GB/S " + BERTAddress + " setup..");
            BERT.Utility.Reset();
            BERT.Clock.LowFreqSource = Agilent TechnologiesN4962ASourceEnum.
                                    Agilent TechnologiesN4962ASourceExternal;
            System.Threading.Thread.Sleep(200);
            BERT.Clock.Frequency = srcFreq/1e6;
            BERT.Data.Pattern = BERTPattern;
            BERT.Data.Generator.Amplitude = BERTAmplitude * 1000;
            if (BERTErrInj)
                BERT.Data.Generator.ErrorInjection.Rate = BERTErrInjRate;
```

```csharp
        BERT.Data.Generator.Output = Agilent TechnologiesN4962AOutputStateEnum.
                                     Agilent TechnologiesN4962AOutputStateOn;
    Console.WriteLine("done");

    // autoalign and check for error-free operation
    Console.Write("Autoalign and error check..");
    BERT.Clock.AutoPhase();
    Console.Write(BERT.Clock.Phase + " degrees..");
    BERT.Data.Detector.GateTime = 0.1;
    BERT.Measurement.Start();
    System.Threading.Thread.Sleep(100);
    BERT.Measurement.GetData(ref bitCount, ref errCount,
                             ref elapTime, ref BER);
    if (BER > BERTErrThreshold)
    {
        Console.WriteLine("failed");
        Console.WriteLine(elapTime + ", BER: " + BER + "; bits: " +
                          bitCount + "; errs: " + errCount);
        Environment.Exit(1);
    }
    Console.WriteLine("passed");

    // start measurement
    Console.WriteLine(BERTGateTime.ToString() + "s BER measurement");
    BERT.Data.Detector.GateTime = BERTGateTime;
    BERT.Measurement.Start();
    bool running = true;
    while (running)
    {
        running = BERT.Measurement.Running;
        System.Threading.Thread.Sleep(1000);

        BERT.Measurement.GetData(ref bitCount, ref errCount,
                                 ref elapTime, ref BER);
        Console.WriteLine(elapTime + ", BER: " + BER + "; bits: " +
                          bitCount + "; errs: " + errCount);
        if (elapTime == BERTGateTime) running = false;
    }
        }
    }
}
```

5.5 Useful code snippets for programming the N4965A-CTR multi-channel BERT controller

Once you've referenced the IVI driver in your Visual Studio project, assigned a using command
simplify the object namespace, initialized the driver object, and connected to an instrument, now what?

To turn on the TG5P1A generator pod attached to Channel 1 of the N4965A-CTR multi-channel BERT controller
instrument, set the amplitude of the output to 1.0V, change the pattern to PRBS15, and set 2.5dB of de-emphasis:

```
IAgilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT CONTROLLERChannelGenerator gen =
   pcbDriver.Channels.get_Item("1").Generator;
gen.State = Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT CONTROLLERStateEnum.
   Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT CONTROLLERStateOn;
gen.Output.Amplitude = 1.0;
gen.Pattern.Name = Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT
   CONTROLLERGeneratorPatternEnum.
               Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT
               CONTROLLERGeneratorPatternPRBS15;
gen.Output.Deemphasis = 2.5;
```

To turn on the N4965A-200 12.5 Gb/s error detector remote head attached to Channel 2 of the N4965A-CTR
multi-channel BERT controller instrument, and set the pattern to PRBS15:

```
IAgilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT CONTROLLERChannelDetector det =
   pcbDriver.Channels.get_Item("2").Detector;
det.State = Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT CONTROLLERStateEnum.
   Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT CONTROLLERStateOn;
det.Pattern.Name = Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT
   CONTROLLERDetectorPatternEnum.
               Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT
               CONTROLLERDetectorPatternPRBS15;
```

To perform a clock/data alignment on the N4965A-200 12.5 Gb/s error detector remote head, to align the
sampling point with the most open portion of the eye, and to start a 10 second measurement:

```
det.Eye.AutoAlign();
det.Gate.Period = 10;
det.Gate.State = Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT CONTROLLERStateEnum.
   Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT CONTROLLERStateOn;
```

To capture the number of bits, number of errors, BER, and elapsed time calculated while the measurement is running:

```
double[] fetchAllAry = new double[4];
double numBits, numErrors, BER, ElapsedTime;
do
{
  // get the new gated results
  det.Fetch.All(ref fetchAllAry);
  numBits = fetchAllAry[0];
  numErrors = fetchAllAry[1];
  BER = fetchAllAry[2];
  ElapsedTime = fetchAllAry[3];
}
while (det.Gate.State == Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT
   CONTROLLERStateEnum.Agilent TechnologiesN4965A-CTR MULTI-CHANNEL BERT
   CONTROLLERStateOn);
```

Other examples are available. Please contact Agilent Technologies for more information.

Troubleshooting

If you run into problems, or if you have any questions not answered in this document, please don't hesitate to contact an Agilent Technologies application engineer.

Appendix A: End User License Agreement

LEGAL NOTICE: PLEASE READ THESE TERMS BEFORE INSTALLING OR OTHERWISE USING THE LICENSED MATERIALS. ALL USE OF THESE LICENSED MATERIALS IS SUBJECT TO THE LICENSE TERMS SET FORTH BELOW. "LICENSED MATERIALS" INCLUDES THE SOFTWARE, ANY WHOLE OR PARTIAL COPIES, AND ANY ACCOMPANYING INSTRUCTIONS, DOCUMENTATION, TECHNICAL DATA, IMAGES, RECORDINGS AND OTHER RELATED MATERIALS.

FOR LICENSED MATERIALS DOWNLOADED OR AVAILABLE ON-LINE:
TO DOWNLOAD AND INSTALL THE LICENSED MATERIALS, YOU MUST FIRST AGREE TO THE FOLLOWING TERMS BY CLICKING ON THE "ACCEPT" BOX BELOW. IF YOU DO NOT AGREE TO ALL OF THESE TERMS, CLICK ON THE "DO NOT ACCEPT" BOX BELOW. NOTWITHSTANDING ANYTHING TO THE CONTRARY IN THIS NOTICE, INSTALLING OR OTHERWISE USING ANY OF THE LICENSED MATERIALS INDICATES YOUR ACCEPTANCE OF THESE TERMS.

FOR LICENSED MATERIALS PROVIDED ON MEDIA OR BUNDLED WITH ANOTHER PRODUCT:
USING THE LICENSED MATERIALS INDICATES YOUR ACCEPTANCE OF THE LICENSE TERMS. IF YOU DO NOT AGREE TO ALL OF THESE TERMS, YOU MAY RETURN ANY UNOPENED LICENSED MATERIALS FOR A FULL REFUND. IF THE LICENSED MATERIALS ARE BUNDLED OR PRE-LOADED WITH ANOTHER PRODUCT, YOU MAY RETURN THE ENTIRE UNUSED PRODUCT FOR A FULL REFUND.

AGILENT TECHNOLOGIES LICENSE TERMS

The following License Terms govern your use of the Licensed Materials unless you have a separate written agreement with Agilent Technologies, in which case, that written agreement will control and take precedence.

Readers of this document are requested to submit their comments, notification of any relevant patent rights or other intellectual property rights of which they may be aware which might be infringed by any use of this intellectual property, software, or specification (the "Intellectual Property"), as appropriate, and to provide supporting documentation to Agilent Technologies, Inc., Legal Department.

Attention is drawn to the possibility that some of the elements of this Intellectual Property may be the subject of patent or other intellectual property right (collectively, "IPR") of third parties. Agilent Technologies shall not be responsible now or in the future for identifying any or all such IPR.

License Grant. Agilent Technologies grants you a non-exclusive license to use one copy of the Licensed Materials. With respect to the software portion of the Licensed Materials, "use" means to install, store, display, execute and use the software on the computer or device, or on the class or series of equipment, for which you have paid the corresponding license fee. If no fee is required, you may use the software on one computer or device. If the software is licensed for concurrent or network use, you may not allow more than the maximum number of authorized users to access and use the software concurrently. You may copy, modify and translate the Licensed Materials for your own internal use.

License Restrictions. You may make copies or adaptations of the Licensed Materials only for archival or internal purposes as granted above, or only when copying or adaptation is an essential step in the authorized use of the Licensed Materials. You must reproduce all copyright notices in the original Licensed Materials on all permitted copies or adaptations. You may not copy the Licensed Materials onto any public or distributed network or service bureau. In addition, you may not lease, rent or sublicense the Licensed Materials without Agilent Technologies' prior written consent.

Upgrades. This license does not entitle you to receive upgrades, updates or technical support. Such services may be purchased separately. If the Licensed Materials include an upgrade to previously licensed material, your license in that material automatically terminates and you should destroy the previous content and any copies or adaptations.

Ownership. The Licensed Materials are owned and copyrighted by Agilent Technologies and/or its third party suppliers. Your license confers no title to, or ownership in, the Licensed Materials and is not a sale of any rights in the Licensed Materials. Agilent Technologies' third party suppliers may protect their rights in the event of any violation of these License Terms.

No Disassembly. You may not disassemble or decompile the Licensed Materials unless you obtain Agilent Technologies' prior written consent

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

High Risk Activities. The Licensed Materials are not specifically designed, manufactured or intended for use in the planning, construction, maintenance or direct operation of a nuclear facility, nor for use in on-line control or fail safe operation of aircraft navigation, control or communication systems, weapon systems or direct life support systems.

Transfer. You may transfer the license granted to you here provided that you deliver all the Licensed Materials to the transferee along with these License Terms. The transferee must accept these License Terms as a condition to any transfer. Your license to use the Licensed Materials will terminate upon transfer.

Termination. Agilent Technologies may terminate your license upon notice for failure to comply with any of these License Terms. Upon termination, you must immediately destroy the Licensed Materials, together with all copies, adaptations and merged portions in any form.

Export Requirements. The Licensed Materials may be subject to export control laws, including the U.S. Export Administration Regulations and other export laws and regulations of other countries. You may not export or re-export the Licensed Materials or any copy or adaptation in violation of any applicable laws or regulations. You certify that you are not on the U.S. Department of Commerce's Denied Persons List, the U.S. Department of Treasury's Specially Designated Nationals list or other government list prohibiting you from receiving the Licensed Materials.

U.S. Government Restricted Rights. If the Licensed Materials are licensed for use in the performance of a U.S. Government prime contract or subcontract, you agree that they have been developed entirely at private expense. You further agree that they are licensed as "commercial computer software" as defined in DFARS 252.227-7014 (Jun 1995), as a "commercial item" as defined in FAR 2.101(a), or as "Restricted computer software" as defined in FAR 52.227-19 (Jun 1987) (or any equivalent agency regulation or contract clause), whichever is applicable. You agree that you acquire only those rights provided for such Licensed Materials by the applicable FAR or DFARS clause or the Agilent Technologies standard license agreement for the product involved.