

Keysight IO Libraries Suite

Getting Started with VISA.NET / C#

Instrument Control program

- Simple instrument control with C#
- How to use VISA.NET

KEYSIGHT
IO Libraries Suite

The next generation
of instrument control



Application Note

Contents

Introduction.....	1
VISA Library overview.....	1
VISA, VISA COM and VISA.NET Library	1
Keysight VISA.NET Library constitution	1
VISA Address	1
Sample Program operating environment.....	2
Software environment.....	2
Hardware environment.....	2
SCPI (Standard Commands for Programmable Instrument).....	2
Sample program developing procedure.....	2
Sample program overview.....	2
Instrument connection check by Keysight Connection Expert.....	3
Start Visual Studio Express 2017 for Windows Desktop and create a C# project.....	3
Add VISA.NET reference to the project	4
Program coding.....	5
Details of the sample program.....	6
VISA.NET program overview.....	6
Create a VISA.NET session	6
Send command to instrument and receive response from instrument.....	6
Release the VISA.NET session	6
VISA.NET Help file.....	7
Notes.....	7

Introduction

You can control instruments by the program you create. To implement instrument control, you need to connect instruments to PC through interfaces, such as LAN, GPIB, USB, RS232 and you need to create a program which sends commands to instruments and receives responses from instruments. The commands and responses are transferred through interfaces, appropriate communication between the instruments and interfaces are very important. Keysight provides libraries which control the interfaces and ensure proper communication with instruments, as part of the software called IO Libraries Suite.

In this application note, Visual Studio Express Edition 2017 and VISA.NET library included in Keysight IO Libraries Suite are used to communicate with instrument and shows you how to create the most basic instrument control program by C#. The program is a simple program, but you can improve it into a practical application program by adding and modifying commands as necessary.

VISA Library overview

VISA, VISA COM and VISA.NET Library

Keysight provides interface control libraries, such as VISA, VISA COM and VISA.NET, as part of software called IO Libraries Suite. VISA stands for Virtual Instrument Software Architecture. It is possible to send commands to the instrument and receive responses from the instrument by calling one of a VISA, VISA COM, VISA.NET from C#. The specifications of VISA, VISA COM and VISA.NET are defined by the industry standardization organization, IVI Foundation, and Keysight implements VISA, VISA COM and VISA.NET library codes in accordance with the defined specifications.

Although VISA, VISA COM and VISA.NET are implemented differently on Windows OS, the basic functions are same.

Because VISA.NET is installed on Windows as a .NET assembly, VISA.NET can be called in the same way as standard .NET libraries.

Using a VISA, VISA COM or VISA.NET library to control the instrument, you can communicate to the instruments without being aware of interface differences, except you control interface-specific functions.

Keysight VISA, VISA COM and VISA.NET libraries support several interfaces such as GPIB, LAN, USB, RS232, VXI, and PXI.

Keysight VISA.NET Library constitution

Keysight VISA.NET library consists of a part which implements the IVI Foundation specifications and a part which implements Keysight original specifications. By creating a program which implements the IVI Foundation specifications, you can create compatible programs that conform to the IVI Foundation specifications.

The IVI Foundation part is provided by the Ivi.Visa.dll assembly and uses the Ivi.Visa namespace. The Keysight original part is provided by the Keysight.Visa.dll assembly and uses the Keysight.Visa namespace.

VISA Address

VISA, VISA COM and VISA.NET use "VISA address" to establish a connection with the instrument. The VISA address has information about the interface and information to identify the instrument for each interface.

VISA Address examples.

GPIB0::22::INSTR // Instrument with GPIB address 22 connected to the first GPIB interface (GPIB0)

GPIB1::17::INSTR // Instrument with GPIB address 17 connected to the second GPIB interface (GPIB1)

TCPIP0::169.254.4.61::inst0::INSTR // Instrument with IP address 169.254.4.61 connected to the LAN interface (TCPIP0)

Sample Program operating environment

Software environment

- Keysight IO Libraries Suite 17.2 or later (IO Libraries Suite supports VISA.NET later than 17.2)
- Windows7 Service Pack1 or later (IO Libraries Suite 17.2 supports Windows7 Service Pack1 or later)
- Visual Studio / C#

Hardware environment

- PC which can execute the above software environment
- LAN, USB, RS232 Interface attached to the PC
- Keysight GPIB interface products (82350C, 82351B, 82357B, E5810B, etc.)
- Instrument that supports SCPI commands

This application note and the sample program were confirmed to operate with the following configuration.

- Windows10 64bit
- Visual Studio Express 2017 for Windows Desktop
- IO Libraries Suite 2019
- 82357B USB/GPIB Interface
- 34465A Digital multimeter (GPIB connected)

SCPI (Standard Commands for Programmable Instrument)

SCPI stands for Standard Commands for Programmable Instruments. SCPI is an industry standard specification for instrument commands and defined by industry standardization organization. In principle, using instruments which support SCPI, you can expect to use the same commands with different instruments, if they have the same function.

For example, if you made a SCPI program for a SCPI digital multimeter, then you can replace the digital multimeter to other SCPI digital multimeters without any program change or with minimum program changes.

Sample program developing procedure

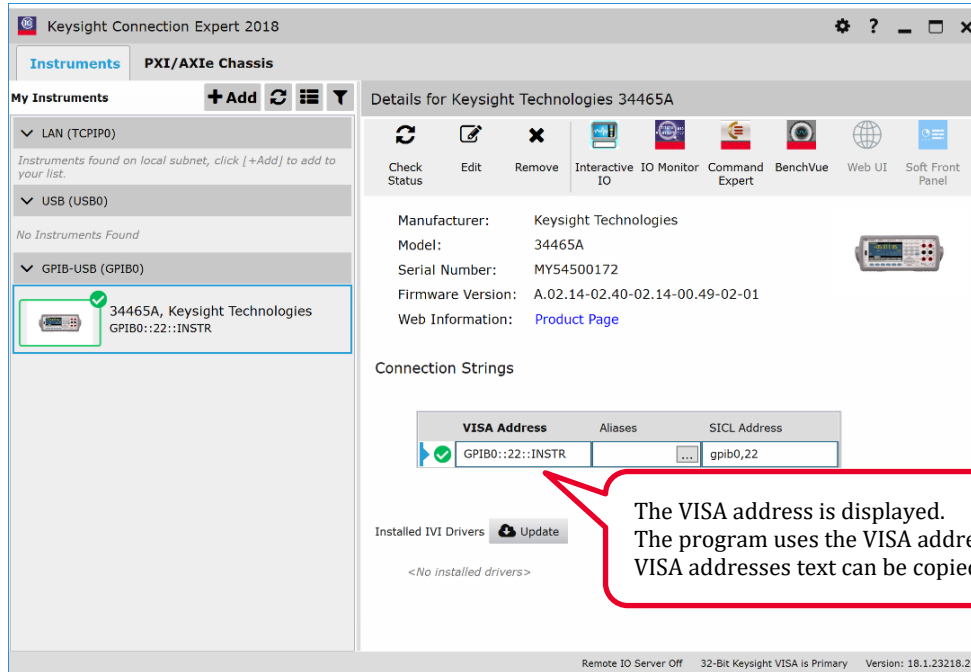
Sample program overview

In this application note, you will make a console application which send a *IDN? command to the instrument and receive a response from the instrument and display the response to the console.

By sending a *IDN? command to an instrument that supports SCPI, the instrument will return information such as manufacturer name, product number, and so on. Although sending *IDN? and receiving a response is a simple transaction, you can confirm bidirectional communication between the PC and the instrument.

Instrument connection check by Keysight Connection Expert

Before creating an instrument control program, it is important to make sure PC and instruments are connected. As a connection check utility program, Keysight provides Connection Expert as a part of IO Libraries Suite. In the following example, you can confirm that the Digital Multimeter 34465A is connected to the PC as the VISA address GPIB0::22::INSTR through GPIB-USB interface.



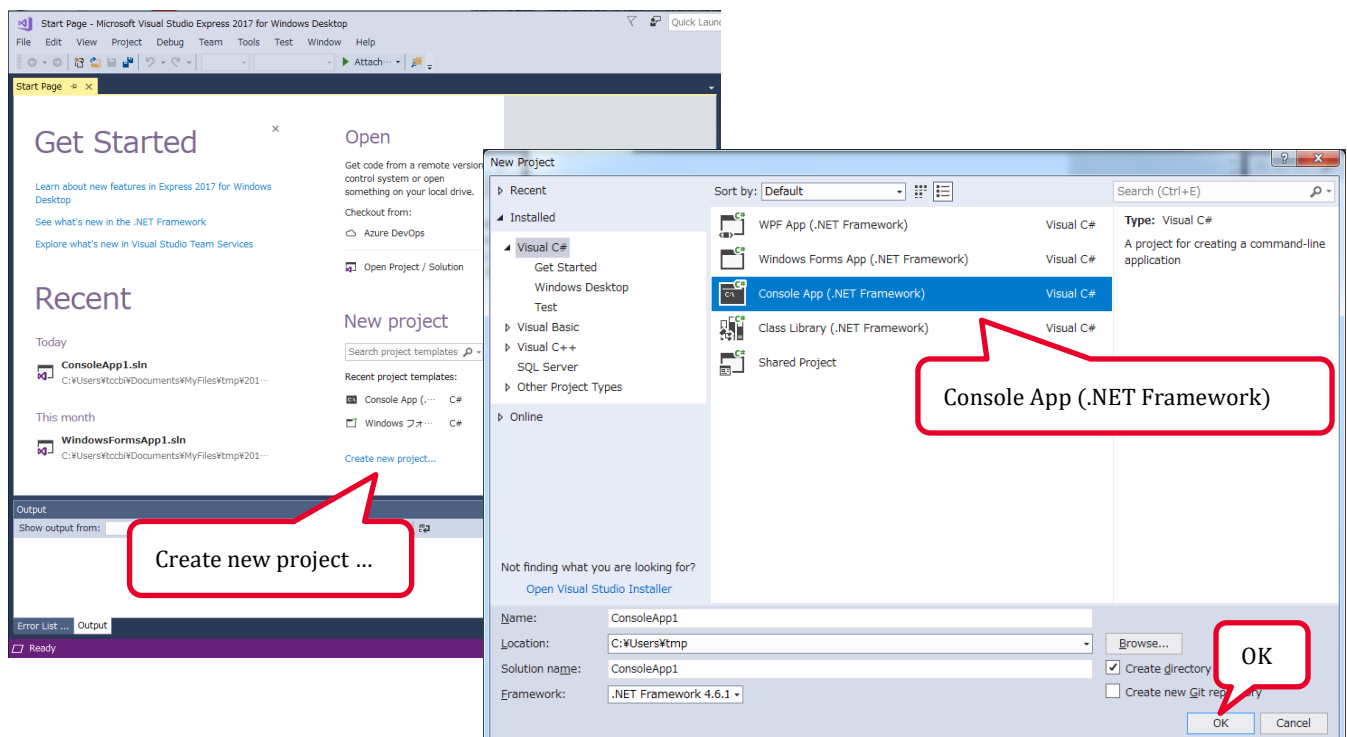
Start Visual Studio Express 2017 for Windows Desktop and create a C# project

Using Windows start menu, start VS Express 2017 for Windows Desktop.

From Visual Studio "Get Start" page, select "Create new project".

Using "New Project" dialog, select "Installed" > "Visual C#" > "Console App (.Net Framework)".

Input Name, Location, then select "OK".

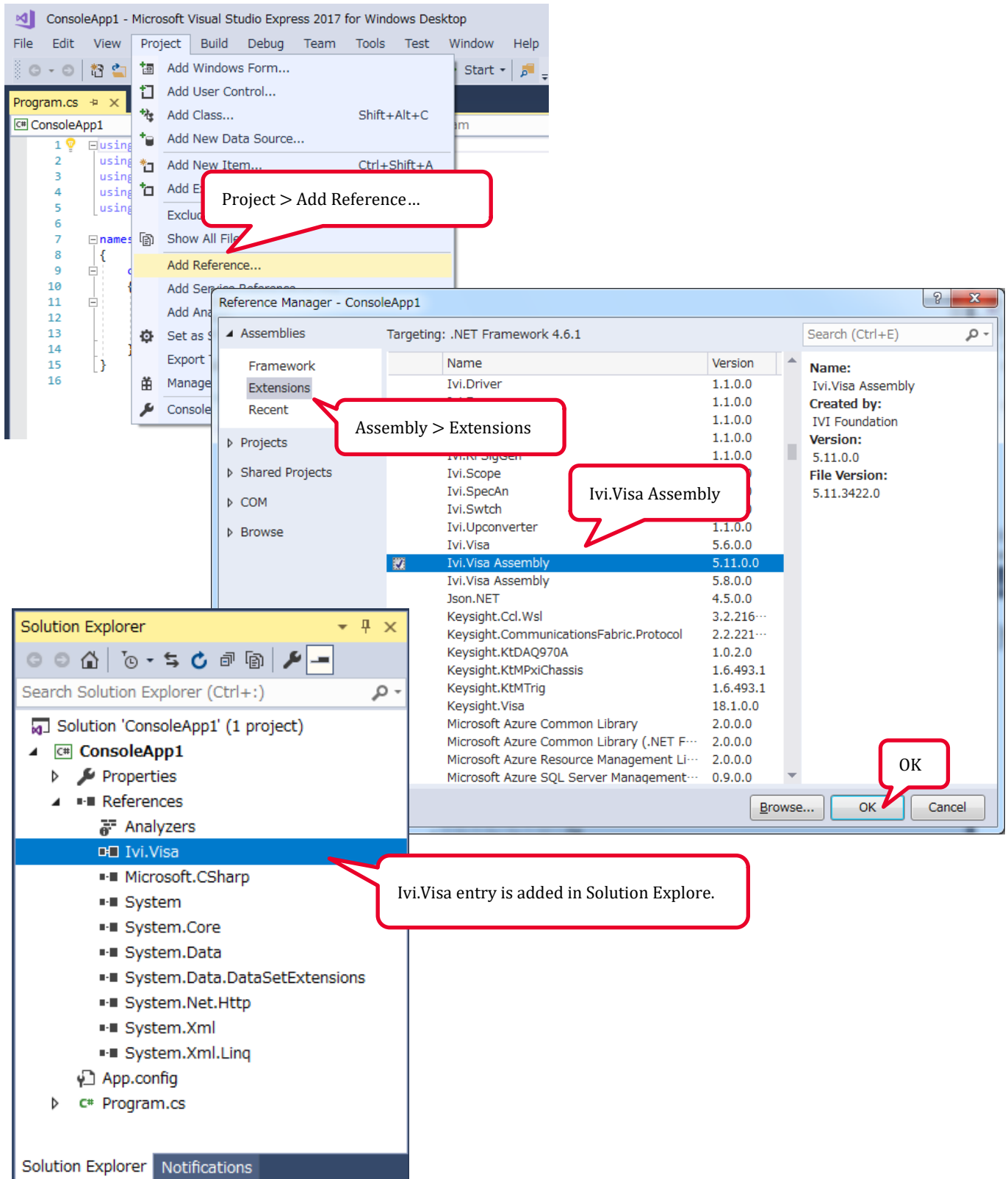


Add VISA.NET reference to the project

Using Visual Studio menu, select “Project” > “Add Reference...”.

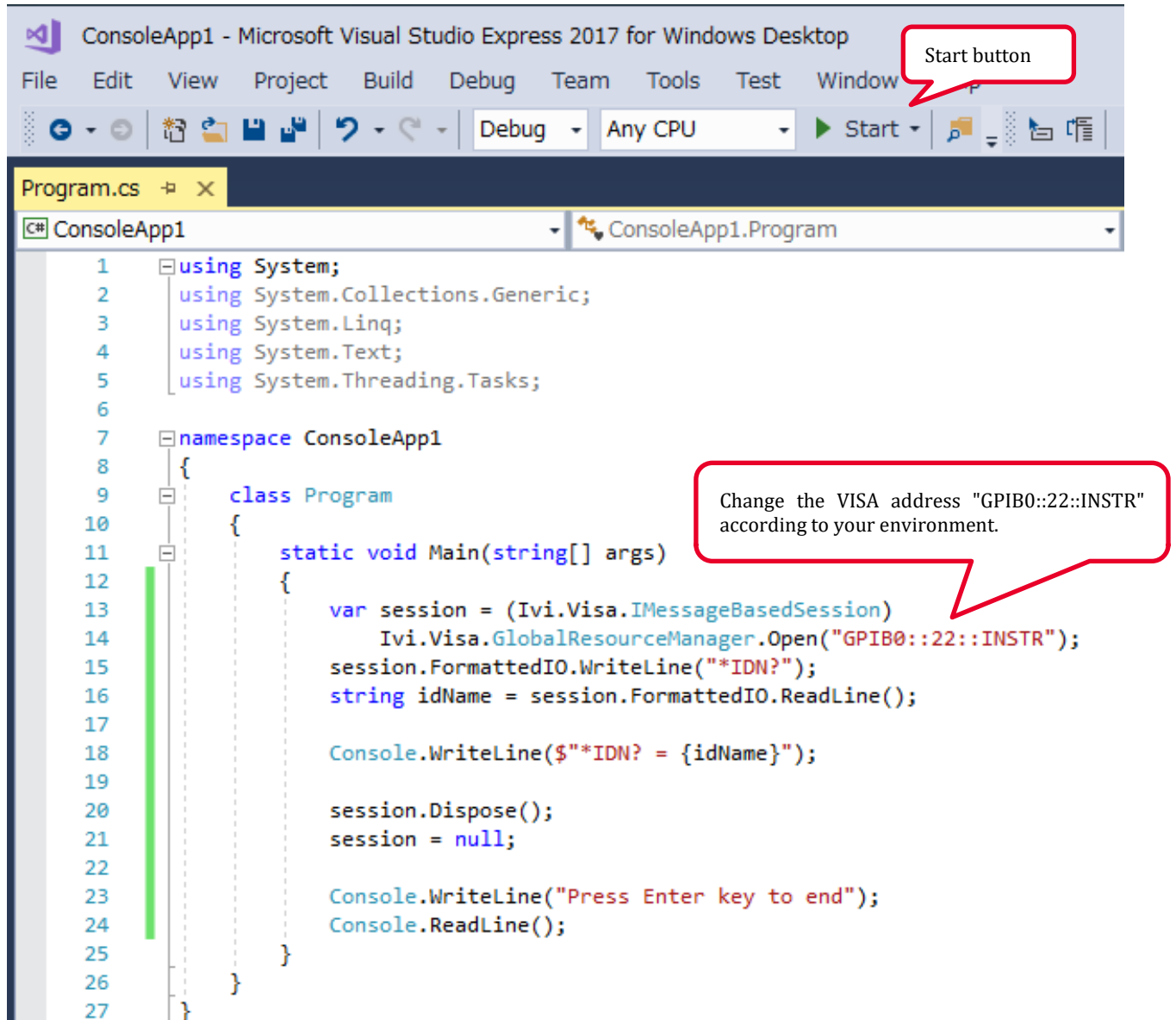
Using “Reference Manager” dialog, select “Assemblies” > “Extensions” and select “Ivi.Visa Assembly”. If you find multiple Ivi.Visa Assembly entries, it is recommended to select the latest version.


Once you had added the Ivi.Visa Assembly reference, then you can find the Ivi.Visa reference in “Solution Explorer”.



Program coding

Using Visual Studio program editor, input the following codes.

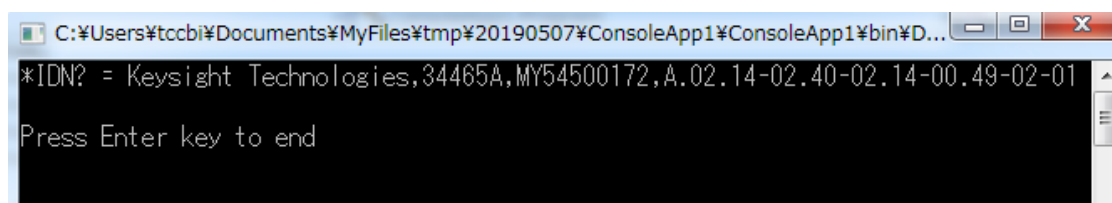


After input the program codes, select "Start button"  to start the program.

Visual Studio starts to build the program and if the build completed without errors, then Visual Studio starts the program.

Both the program build and the program execution are completed without errors, then the following window will be displayed.

To stop the program and close the window, press the Enter key.



Details of the sample program

VISA.NET program overview

At first, the program creates a VISA.NET session to establish a connection with the instrument. Next, using the VISA.NET session, the program sends commands to the instrument and receives responses from the instrument. At last, the program releases the VISA.NET session when control of the instrument ends.

Create a VISA.NET session

```
var session = (Ivi.Visa.IMessageBasedSession)
    Ivi.Visa.GlobalResourceManager.Open("GPIB0::22::INSTR");
```

To create a VISA.NET session, pass the instrument's VISA address to the Open() method of Ivi.Visa.GlobalResourceManager. The return type of the Open() method of Ivi.Visa.GlobalResourceManager is Ivi.Visa.IVisaSession. Since the program wants to use the format conversion functions to communicate with instrument, it casts the return type to Ivi.Visa.IMessageBasedSession and assigns it to a variable "session".

You need to create one VISA.NET session for one instrument you want to control. For example, if you control two instruments, you need to create two VISA.NET sessions for each instrument.

Send command to instrument and receive response from instrument

```
session.FormattedIO.WriteLine("*IDN?");
string idName = session.FormattedIO.ReadLine();
```

First, this program outputs "*IDN?" command to the instrument using WriteLine() method of FormattedIO property of Ivi.Visa.IMessageBasedSession. The WriteLine() method adds a "line feed character" to the end of the string data passed as an argument and outputs it to the instrument.

Next, this program receives response from the instrument using ReadLine() method of FormattedIO property of Ivi.Visa.IMessageBasedSession. The ReadLine() method reads the response from the instrument as a string data and returns it as a return value.

The FormattedIO property of Ivi.Visa.IMessageBasedSession provides a number of methods with format conversion functions, which are useful with instrument control applications. Please refer to the VISA.NET Help file for more information.

Release the VISA.NET session

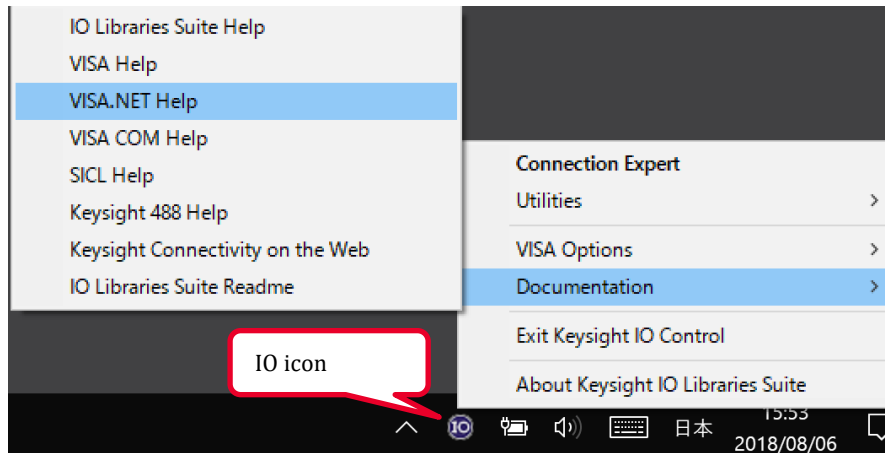
```
session.Dispose();
session = null;
```

Using Dispose() method of Ivi.Visa.IMessageBasedSession, this program releases VISA.NET session after completion of the instrument control.

VISA.NET Help file

The IO Libraries Suite provides VISA.NET Help files, which contains detail explanation about VISA.NET interfaces, methods and more. To view the VISA.NET Help file, click the "IO" icon and select "Documentation"> "VISA.NET Help" from the menu.

VISA.NET provides many features for use in instrument control. Please refer to the Help file of VISA.NET to create more practical programs.



Notes

The copyright of this application note and the sample program are owned by Keysight Technologies Inc. You may use, modify, copy and distribute the sample programs listed in this application note. Please do not reprint or distribute this application note. Our company and any group or organization to which we belong are not responsible for any failure, damage, trouble caused by using this application note and the sample program. Although this application note and sample program have been verified by us, they are not guaranteed to operate under the other environment.