Agilent T&M Programmers Toolkit for Visual Studio .NET

Version 1.2

Getting Started

Agilent Technologies

Notices

Manual Part Number

5090-6479

Edition

Second Edition, August 2003

Printed in USA

Agilent Technologies, Inc.
815 14th Street SW
Loveland, CO 80537 USA

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

Table of Contents

1 Welcome to the Agilent T&M Programmers Toolkit

This chapter describes the features of the Agilent T&M Toolkit, the system requirements needed to run T&M Toolkit, and how to install, uninstall, and license the Agilent T&M Toolkit.

About the Agilent T&M Toolkit

What is the T&M Toolkit?

The Agilent T&M Toolkit is a set of tools, components, controls, and class libraries that help you communicate with and control test and measurement equipment from within Microsoft’s ® Visual Studio .NET development environment.

What Makes the T&M Toolkit Useful?

The T&M Toolkit helps you cut your instrument programming time, speeds up your product development process, and improves your productivity. The T&M Toolkit extends Microsoft's ® Visual Studio .NET platform with integrated, easy-to-use software tools and components that make Visual Studio .NET the easiest place for you to write code for automating measurement tasks and displaying test data.

With T&M Toolkit, Agilent's extensive measurement expertise is integrated into the Visual Studio .NET framework to eliminate many of the problems traditionally associated with connecting to and controlling instruments. The T&M Toolkit software automatically generates code for you, and Windows® functionality, such as drag and drop, makes many tasks faster and easier. Robust debug, analysis, and tuning tools further streamline your development process. Plus, Agilent's on-line test and measurement help, fully integrated at every step, shortens your programming learning curve.

What is in the T&M Toolkit?

The T&M Toolkit includes:

- Language support for C#, Visual Basic .NET, and Managed C++.
- A wizard, called the New Project Wizard, helps you quickly setup a T&M project.

- Instrument Explorer — Finds instruments attached to your PC or network, and helps you easily manage the instruments and their drivers. Just drag and drop an instrument icon into your work window to generate code to connect with the instrument.
- Interactive IO — Can be used to check instrument connections and to send commands to instruments.
- IO Monitor — Helps you debug instrument control applications by capturing and displaying instrument communication details from different I/O layers.
- The Driver Wrapper Wizard — Provides a way to automatically create a .NET-friendly wrapper object around a traditional VXI*plug&play* driver or IVI-C driver (IVI-COM drivers do not require wrappers). This lets your .NET application use any of the hundreds of drivers available from Agilent and other instrument vendors.
- DirectIO — Gives you an easy way to control instruments using the instrument’s native command set. Supported protocols include: GPIB, Serial, TCP-IP, and USB.
- Data Analysis support — Mathematical, statistical, and digital signal processing support.
- Data Types — Includes support for Complex numbers.
- 2D Graph Objects — A variety of different graphical displays provide multiple ways to view and scale data.
- Special class libraries — Timing, a virtual Waveform Generator, and an engineering number formatter all make your tasks simpler.
- VEE Wrapper Wizard — Provides an easy way for you to call Agilent VEE Pro User Functions from a Visual Studio .NET project.
- Integrated help — T&M Toolkit’s Help system is fully integrated with the Visual Studio .NET help system. The new Dynamic Help feature and IntelliSense shorten the learning curve even further.

How to Use this Manual

This manual has been structured to match a workflow. The workflow model is:

1. Start a new T&M project in Visual Studio
   - See Chapter 2 - The New Project Wizard
2. Discover and configure your instruments
   - See Chapter 3 - Using Instrument Explorer
   - See Chapter 4 - Using Interactive IO
   - See Chapter 5 - Using the IO Monitor
   - See Chapter 6 - Using the Driver Wrapper Wizard
3. Manage your instruments
   - See Chapter 7 - Using the DirectIO Class to Manage your Instruments
4. Analyze your data
   - See Chapter 8 - Analyzing Instrument Data
5. Display your data
   - See Chapter 9 - Using the 2D Graph Objects
6. Optional helpers for numerous areas
   - See Chapter 10 - Virtual Waveforms, Timing Classes, and Number Formatting
7. How to use Agilent VEE with .NET
   - See Chapter 11 - Using Agilent VEE with .NET

Installing Agilent T&M Toolkit

Before installing the Agilent T&M Toolkit, ensure that Microsoft Visual Studio .NET is already installed and is working properly on your computer. Next, ensure no applications, especially .NET, are running when you begin the Agilent T&M Toolkit installation process.

Installing Agilent T&M Toolkit

Place the Agilent T&M Toolkit CD-ROM into the computer's CD-ROM drive. (Do not install directly from the WWW.) Installation should start automatically. However, if it does not, select **Start** > **Run** from the computer's taskbar and enter:

[*The computer's CD-ROM drive letter*]**:\setup.exe**

Select **OK**.

Run Visual Studio .NET and you will notice that the Start Page has a link on the left side with the title: **Agilent T&M Programmers Toolkit.** The main menu also has an item entitled **T&M Toolkit**

Activating Agilent T&M Toolkit

Agilent T&M Programmers Toolkit uses a Product Key to activate it. The Product Key is shipped with the T&M Toolkit. You are prompted for the key when you install T&M Toolkit.

If you are using an evaluation version of T&M Toolkit, the Product Key is "eval" (without the quotes). You can upgrade to a licensed product by purchasing T&M Toolkit and entering the Product Key. The Product Key is entered at **T&M Toolkit** > **Product Key** on the main menu.

Uninstalling Agilent T&M Toolkit

If you open the **Control Panel** > **Add / Remove Programs**, you see **Agilent T&M Programmers Toolkit for Visual Studio .NET**. Select this entry and choose **Remove.** If **Agilent VisaCom** was installed by T&M Toolkit, it is uninstalled. However, if it was installed separately, it is not uninstalled.

When the T&M Toolkit is uninstalled, the next time Visual Studio is run it pauses and displays the message:

“Help is updating to reflect your recent changes. This may take several minutes."

This is caused by the removal of the integrated T&M Toolkit Help from Visual Studio .NET. It is normal, and it does take several minutes.

System Requirements

Hardware Requirements

The hardware requirements listed below include the combined resource needs of Microsoft Visual Studio .NET and the Agilent T&M Toolkit.

Computer / Processor

PC with a Pentium II-class processor, 450 megahertz (MHz)

Minimum Memory (RAM) Requirements

Windows® XP Professional: 160 megabytes (MB) of RAM

Windows® 2000 Professional: 96 MB of RAM

Recommended: 256 MB or more

Free Disk Space:

2.5 GB with 500 MB minimum on the system drive

Agilent T&M Toolkit requires 100 MB on the installation drive

Drive

CD-ROM or DVD-ROM drive

Display

Super VGA (800 x 600) or higher-resolution monitor with 256 colors or more

Input Device

Microsoft mouse or compatible pointing device

Access to the World Wide Web (WWW) is strongly recommended

Supported Instrument Interfaces

One of the following physical connectivity options is required for the PC-to-instrument connection:

- Agilent 82357A USB/GPIB Interface
- Agilent E5810A or E2050A/B LAN/GPIB gateway
- Agilent 82350A or 82341C GPIB interfaces
- Agilent E8491B or E1406B VXI configurations using drivers
- Standard RS-232
- National Instruments I/O hardware using NI 488 version 1.5

Software Requirements

Operating System

- Design Time - Microsoft Windows 2000, XP
- Run Time - Microsoft Windows 98, ME, NT, 2000, XP, and XP Home

Development Environment

- VB .NET Standard Edition
- Visual C# Standard Edition
- Visual Studio .NET Standard and higher

2
The New Project Wizard

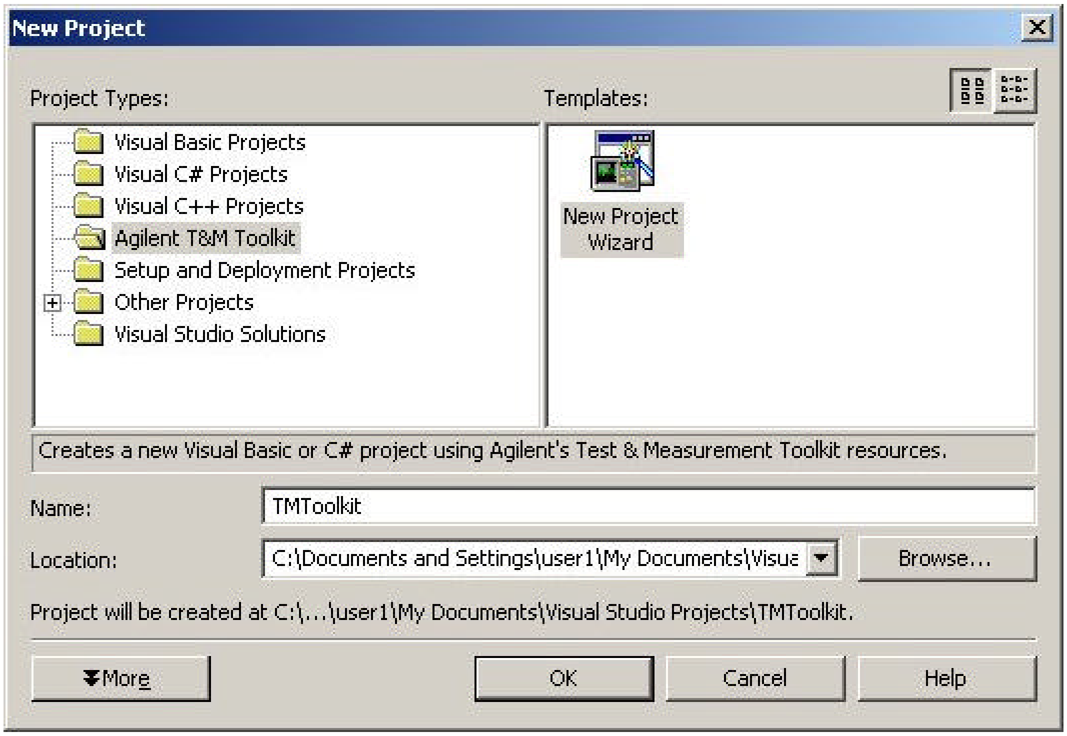
This chapter describes the New Project Wizard. The New Project Wizard walks you through the process of creating a new T&M Toolkit project.

Using the New Project Wizard

The Agilent T&M Toolkit includes a New Project Wizard that you can use to create a new T&M Toolkit project.

Using the New Project Wizard

Step	Action
1	**a** From the Visual Studio main menu, select **File** > **New** > **Project.** **b** As Figure1shows, enter the project name and its home directory. **c** Select **OK** when you are done.



Creates a new Visual Basic or C# project using Agilent's Test & Measurement Toolkit resources.

Name: TMToolkit

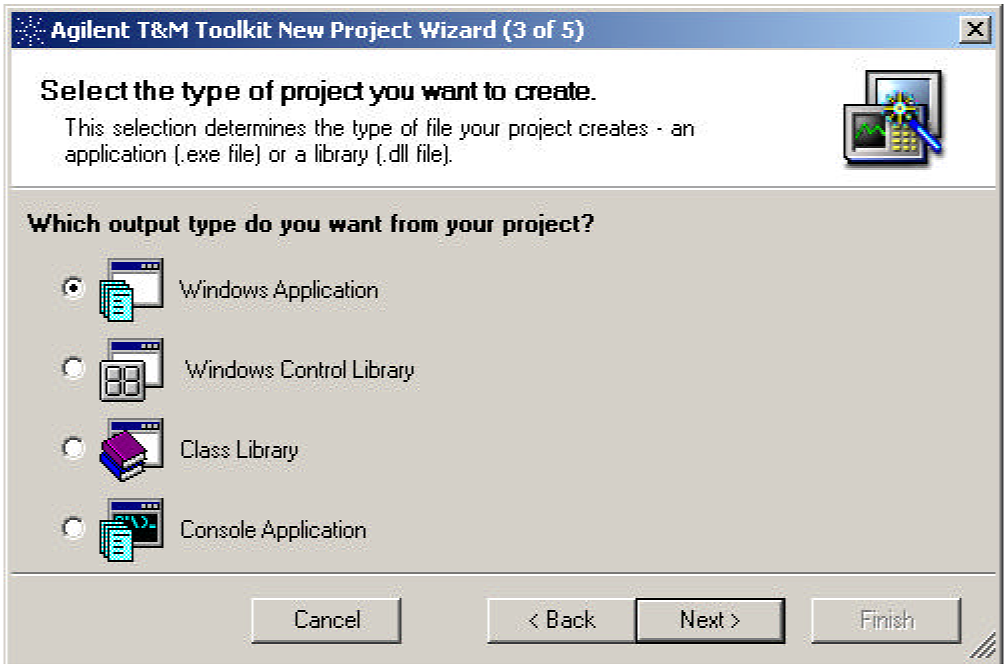
Location: C:\Documents and Settings\user1\My Documents\Visua

Project will be created at C:\...\user1\My Documents\Visual Studio Projects\TMToolkit.

Figure 1 Creating a New Agilent T&M Toolkit Project with .NET

Using the New Project Wizard (continued)

Step	Action
2	Review the welcome screen and select **Next** when you are done.
3	This step only appears if both compilers are installed. **a** Select the programming language for this project by clicking either Visual Basic .NET or C#. For Visual Studio .NET 2003, Managed C++ is also available. **b** Select **Next** when you are done.
4	**a** To create an executable file, select either Windows Application or Console Application. To create a DLL file, select either a Windows Control Library or a Class Library. See Figure2 for an example of the screen. **b** Select **Next** when you are done.


Agilent T&M Toolkit New Project Wizard (3 of 5)

Select the type of project you want to create.

This selection determines the type of file your project creates - an application (.exe file) or a library (.dll file).

Which output type do you want from your project?

- [✓ Windows Application]
- Windows Control Library
- Class Library
- Console Application

Figure 2 Selecting the Project Type

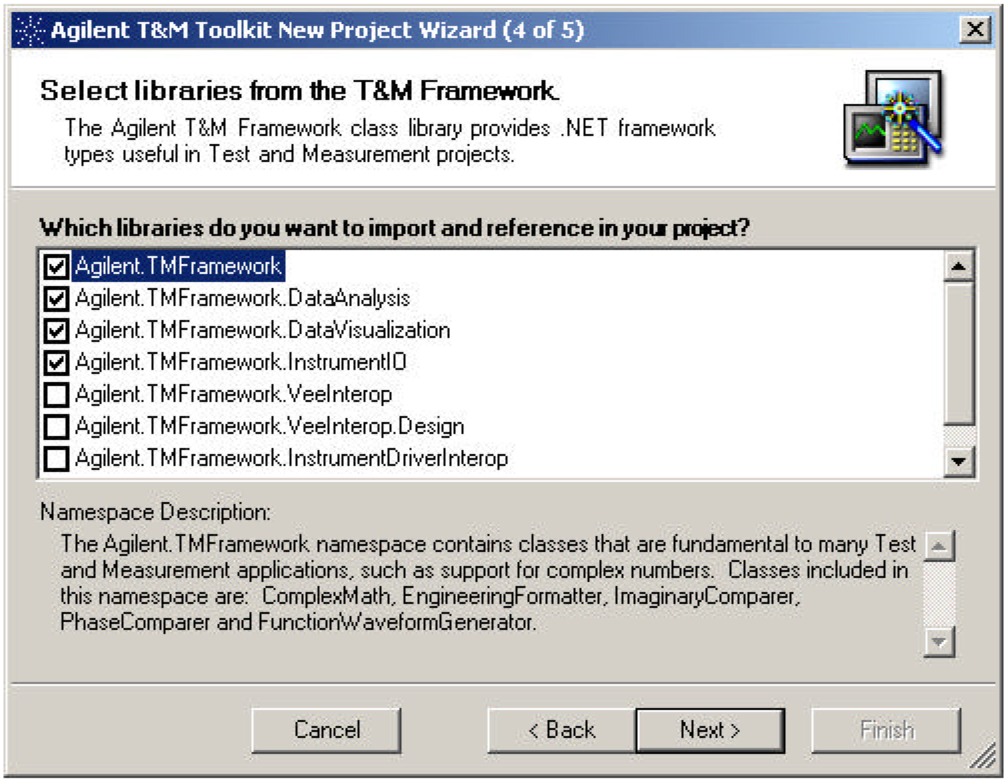
Project Type Definitions

In Figure 2 on page 17, there is a list of project types. The project types are defined as:

- A Windows Application is a stand-alone executable that displays Windows Forms.
- A Windows Control Library is a dynamic link library (DLL) file that contains a library of custom controls for use on Windows Forms.
- A Class Library is a DLL file that contains a library of custom classes.
- A Console Application is a stand-alone executable that does not use Windows Forms and is launched from an operating system prompt.

Using the New Project Wizard (continued)

Step	Action
5	**a** Select the namespaces you need for your program. You can refer to the Namespace Description for an overview of which libraries each namespace provides. You can also add more namespace references after the Wizard is finished. See Figure 3 **b** Select **Next** when you are done.
6	**a** Review the summary. **b** If the summary is correct, select **Finish**.
7	A using statement is added to your C# source code for each of the namespace references you have selected -or- An Imports statement is added to your Visual Basic .NET source code for each of the namespace references you have selected



Select libraries from the T&M Framework

The Agilent T&M Framework class library provides .NET framework types useful in Test and Measurement projects.

Which libraries do you want to import and reference in your project?

- [✓] Agilent.TMFramework
- [✓] Agilent.TMFramework.DataAnalysis
- [✓] Agilent.TMFramework.DataVisualization
- [✓] Agilent.TMFramework.InstrumentIO
- [ ] Agilent.TMFramework.VeeInterop
- [ ] Agilent.TMFramework.VeeInterop.Design
- [ ] Agilent.TMFramework.InstrumentDriverInterop

Namespace Description:

The Agilent.TMFramework namespace contains classes that are fundamental to many Test and Measurement applications, such as support for complex numbers. Classes included in this namespace are: ComplexMath, EngineeringFormatter, ImaginaryComparer, PhaseComparer and FunctionWaveformGenerator.

Figure 3 Selecting Namespaces for the Project

Agilent T&M Framework Class Libraries

The following table summarizes the class libraries available in the Agilent T&M Framework. The classes and their properties and methods can be accessed by adding the appropriate Reference to your Visual Studio project. Using the T&M Project Wizard to create a new project automatically adds the references that are checkmarked in Figure3on page19 .

Table 1 Agilent T&M Framework Namespaces

Namespace	Description	Classes
Agilent.TMFramework.InstrumentIO	Easy access to message based instruments from .NET programs	Direct IO
Agilent.TMFramework	Complex number data type and math support	Data Types
Agilent.TMFramework.DataAnalysis	Mathematical tools to facilitate the analysis of test results and EE data	Data Analysis
Agilent.TMFramework. DataVisualization	T&M-smart graphs for viewing test results and EE data	Data Visualization
Agilent.TMFramework	Generates signals to simulate simple instruments, Precision timing, Formatting exponential numbers, and Engineering Math functions	Function Waveform Generator, Timing, Engineering Formatter, Engineering Math
Agilent.TMFramework. InstrumentDriverInterop	Enables instrument communication using VXIplug&play, IVI-C, and IVI-COM instrument drivers	Driver Wrapper
Agilent.TMFramework.VeeInterop	Access Enables communication between VEE UserFunctions and a .NET application	Callable VEE

3
Using Instrument Explorer

This chapter describes how Instrument Explorer works, how it can help you manage instruments, and how it can simplify your programming.

What is Instrument Explorer?

Instrument Explorer is integrated into Visual Studio and also runs as a standalone application. It provides a visual tree view of instrument connections and easy access to other related tools.

The Instrument Explorer performs the following key tasks:

- Quickly identifying and verifying which instruments are connected to the PC
- Creating and saving files containing instrument connection information to a configuration file for later use.
- Assisting the user in finding and selecting drivers through a Wizard
- Providing access to DirectIO, which enables direct communication with the instrument via ASCII commands
- Providing easy access to the Interactive IO tool, which facilitates sending common, generic commands to instruments
- Generating code that connects to an instrument within a VB .NET, C#, or C++ project using DirectIO or a driver

How to Start the Instrument Explorer

To invoke the standalone version of Instrument Explorer, go to **Start** > **Programs** and select **Agilent T&M Programmers Toolkit 1.2** > **T&M Programming Tools** > **Agilent Instrument Explorer**. To invoke the integrated version, either go to the Visual Studio main menu and select **T&M Toolkit** > **Instrument Explorer** or select the Instrument Explorer icon from the Toolkit toolbar.

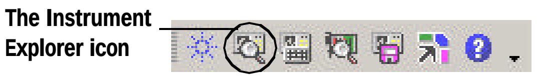


Figure 4 Agilent T&M Toolkit Toolbar

Instrument Explorer appears on the left side of Visual Studio. During runtime, it is hidden.

How Instrument Explorer Discovers Instruments

Instrument Explorer gathers instrument information from three sources to create its instrument configuration. These three sources are described below.

Instrument Explorer and the IVI Config Store

Instrument Explorer reads the IviConfigurationStore.xml file found in [*IVI installation directory*]\IVI\Data directory. From this file, Instrument Explorer extracts a list of IVI-based instruments that have been installed. The list of IVI-based instruments is added to Instrument Explorer's database of instruments.

Instrument Explorer and Active Discovery

Instrument Explorer actively searches the local machine's GPIB and VXI buses for instruments. If an instrument is powered-on and using one of these buses, it is discovered and added to Instrument Explorer's database.

Instrument Explorer and VISA

Instrument Explorer reads the database file created by the IO Config utility of Agilent VISA. All entries in this file are added to the Instrument Explorer database. These entries include entries for instruments that are not active on a bus and entries for LAN based instruments.

Finding Instruments Connected to Your Computer

To find instruments that are connected to your computer:

1 Select the Find Instruments button from the Instrument Explorer toolbar or right click the top entry ( ) of the currently loaded configuration.

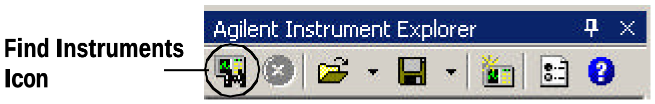
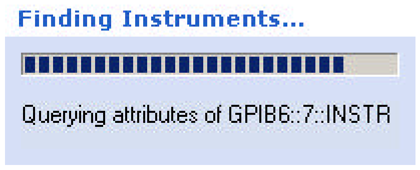


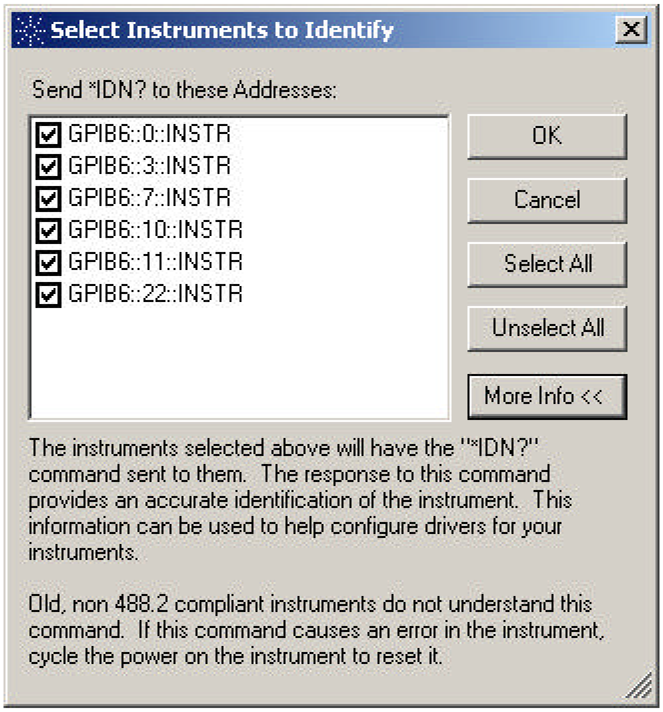
Figure 5 Instrument Explorer Toolbar

2 After instrument discovery is finished (see the progress indicator at the bottom of the Instrument Details area or Figure6), a dialog box appears that prompts you to select which instruments you want to identify using an *IDN query. *IDN provides information, when available, regarding the vendor, model number and version number of the instrument.


Finding Instruments...

Querying attributes of GPIB6::7::INSTR

Figure 6 Progress Bar During Instrument Discovery



Send *IDN? to these Addresses:

- ☑ GPIB6::0::INSTR
- ☑ GPIB6::3::INSTR
- ☑ GPIB6::7::INSTR
- ☑ GPIB6::10::INSTR
- ☑ GPIB6::11::INSTR
- ☑ GPIB6::22::INSTR

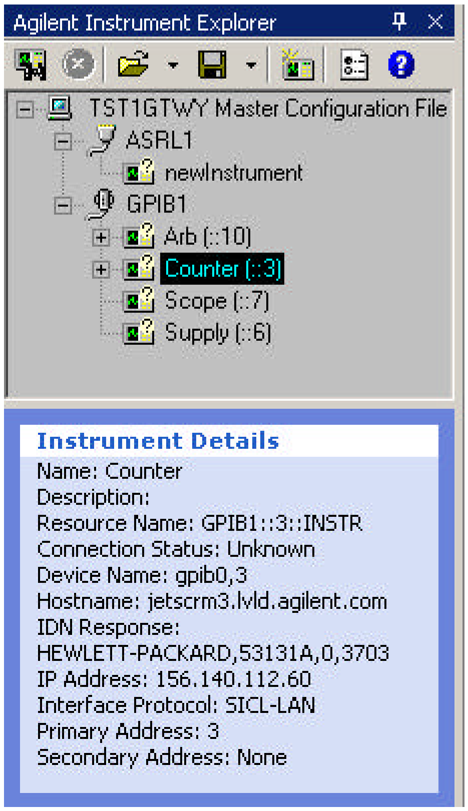
The instruments selected above will have the "*IDN?" command sent to them. The response to this command provides an accurate identification of the instrument. This information can be used to help configure drivers for your instruments.

Old, non 488.2 compliant instruments do not understand this command. If this command causes an error in the instrument, cycle the power on the instrument to reset it.

Figure 7 Selecting Instruments to Query with *IDN

3 To verify the connection and identify the instrument by model and/or vendor using *IDN do the following:
   - a Select the addresses you want to query with an *IDN. See Figure 7.
   - b Select **OK**
   - c If available, additional information about the instrument is displayed.
4 View the list of instruments. Instruments are organized by bus type.
5 Select an instrument and view detailed information about it in the Instrument Details area at the bottom of Instrument Explorer. See Figure8on page27.

NOTE In this release of Agilent T&M Toolkit, the Instrument Explorer does not identify RS-232 instruments.



Instrument Details
Name: Counter
Description:
Resource Name: GPIB1::3::INSTR
Connection Status: Unknown
Device Name: gpib0,3
Hostname: jetscrm3.lvld.agilent.com
IDN Response:
HEWLETT-PACKARD,53131A,0,3703
IP Address: 156.140.112.60
Interface Protocol: SICL-LAN
Primary Address: 3
Secondary Address: None

Figure 8 Instrument Descpition

Instrument Connection Status Icons

Instrument connections are represented iconically in Instrument Explorer as follows:

Instrument connection status unknown

Instrument not found

Instrument found but not identified (*IDN)

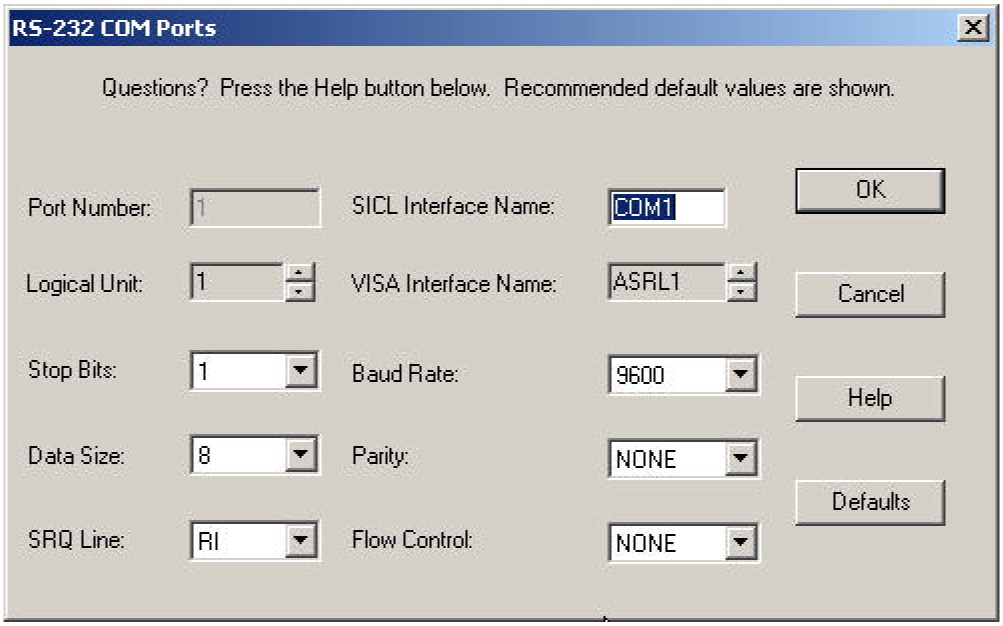
Instrument failed *IDN? query

Instrument identifed with *IDN? query

Finding RS-232 (Serial) Instruments

Agilent T&M Toolkit includes a utility called Interactive IO (see Chapter 4, “Using Interactive IO) that can be accessed from Instrument Explorer. Interactive IO can be used to access any serial port that is properly configured.

You can use the Agilent IO Config utility to configure your serial ports. See Figure9on page29. Using Agilent IO Config, configure the settings for your serial port.


RS-232 COM Ports

Questions? Press the Help button below. Recommended default values are shown.

Port Number: 1
SICL Interface Name: COM1
Logical Unit: 1
VISA Interface Name: ASRL1
Stop Bits: 1
Baud Rate: 9600
Data Size: 8
Parity: NONE
SRQ Line: RI
Flow Control: NONE

Figure 9 Configuration dialogdialog from the Agilent IO Config Utility

From the Visual Studio main menu, select **T&M Toolkit** > **Interactive IO**. Using Interactive IO, connect to the instrument and then query it with an *IDN.

Setting Instrument Properties

Once you have discovered the instruments connected to your computer, they appear as either a list of “New Instrument” or they are identified by name. If you right click on an instrument, you will see the pop-up menu in Figure10. If you select **Instrument Properties**, you see the dialog box in Figure11. Using the **Instrument Properties** dialog box, you can set the instrument’s name, description, and resource name.

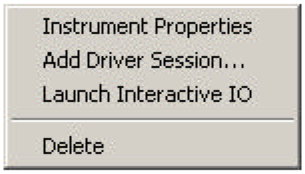
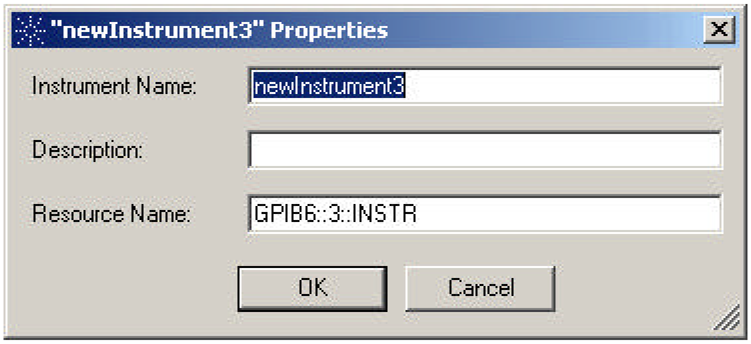


Figure 10 Instrument Right Click Pop-Up Menu



"newInstrument3" Properties

Instrument Name: newInstrument3

Description:

Resource Name: GPIB6::3::INSTR

Figure 11 Setting Instrument Properties

Managing Instrument Configurations

The instrument configuration database for Instrument Explorer is stored in a Master Configuration File. This file is the default configuration and is loaded whenever you start Instrument Explorer.

Opening a Different Configuration File

You can open a different configuration file by selecting the open configuration file icon (see Figure12) and selecting **Open Configuration File.** Use the dialog box that appears to choose a file.

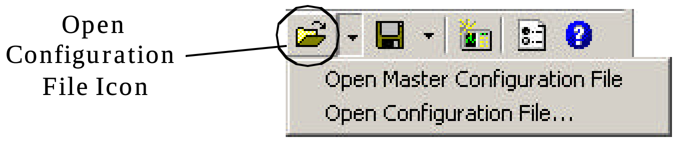

Open Configuration File Icon
Open Master Configuration File
Open Configuration File...

Figure 12 Opening a Different Configuration File

Changing the Default Configuration File

You can load a different configuration file at startup without changing the Master Configuration File. This is a very useful feature when you have a short-term task with a different instrument configuration. To use this feature, select the options icon ( ). Figure13 shows the dialog box that appears. Click **Alternate configuration file** and either enter a fully qualified path to a different configuration file or browse to its location. Click **OK** when you are done.

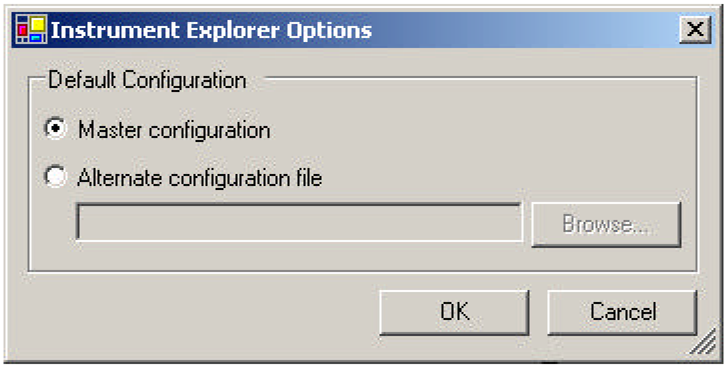


Figure 13 Selecting a Default Configuration File

Creating New Configuration Files

Suppose you have a configuration file loaded and remove several devices. You want to save this new configuration and use it later. You can select the save configuration icon (see Figure14) and save the configuration file with its current name or with a different name.

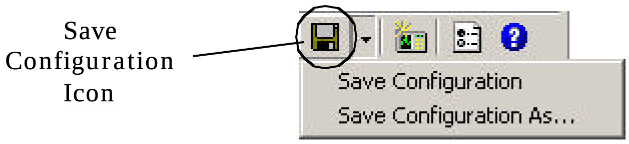
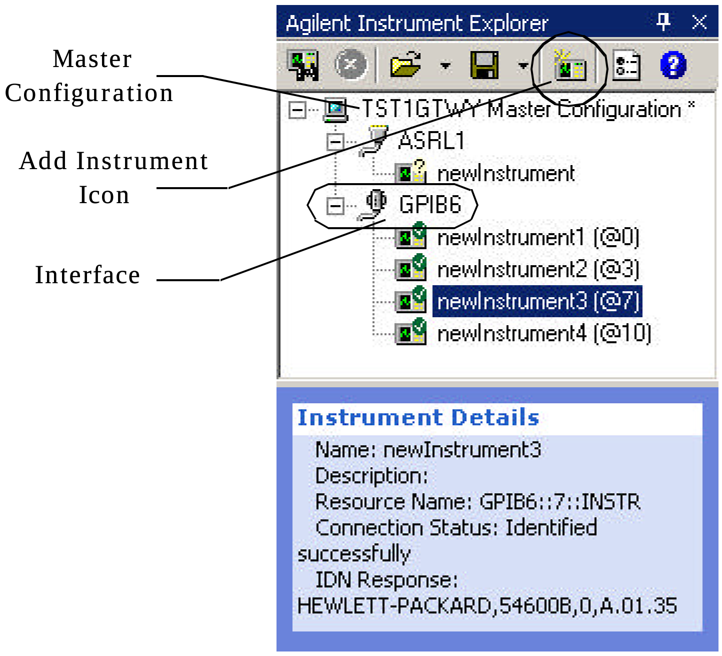


Figure 14 Saving a Configuration

Changing or Adding to the Instrument Configuration

Instrument Explorer is an active instrument configuration manager. It not only reads existing configurations, but it can be used to add or delete items to configurations. You can add more instruments to the database by right clicking an interface (such as GPIB6), by right clicking the Master Configuration, or by clicking the add instrument icon.



Instrument Details
Name: newInstrument3
Description:
Resource Name: GPIB6::7::INSTR
Connection Status: Identified successfully
IDN Response: HEWLETT-PACKARD,54600B,0,A.01.35

Figure 15 Adding an Instrument

NOTE

If you plan to use the configuration, be sure you save it. There is a visual cue to remind you that you have unsaved changes. This appears as a pink icon ( ) next to the Master Configuration file.

Deleting an Instrument

You can also delete an instrument from the Instrument Explorer configuration.

CAUTION

If you delete an instrument and save the configuration, the IviConfigurationStore.xml file and the Instrument Explorer configuration file are updated.

Using the Instrument Session Wizard

An instrument session is the communication connection that Instrument Explorer creates between the instruments in the current configuration and your source code.

There are two types of sessions: a Driver session and a DirectIO session. Driver sessions create a session for a driver type specified by you. The supported types are VXI*plug&play*, IVI-COM, and IVI-C. You can also choose a DirectIO session that allows you to directly communicate with your instrument using a command language such as SICL.

NOTE

DirectIO does not support VXI*plug&play.*

Adding or Creating an Instrument Session

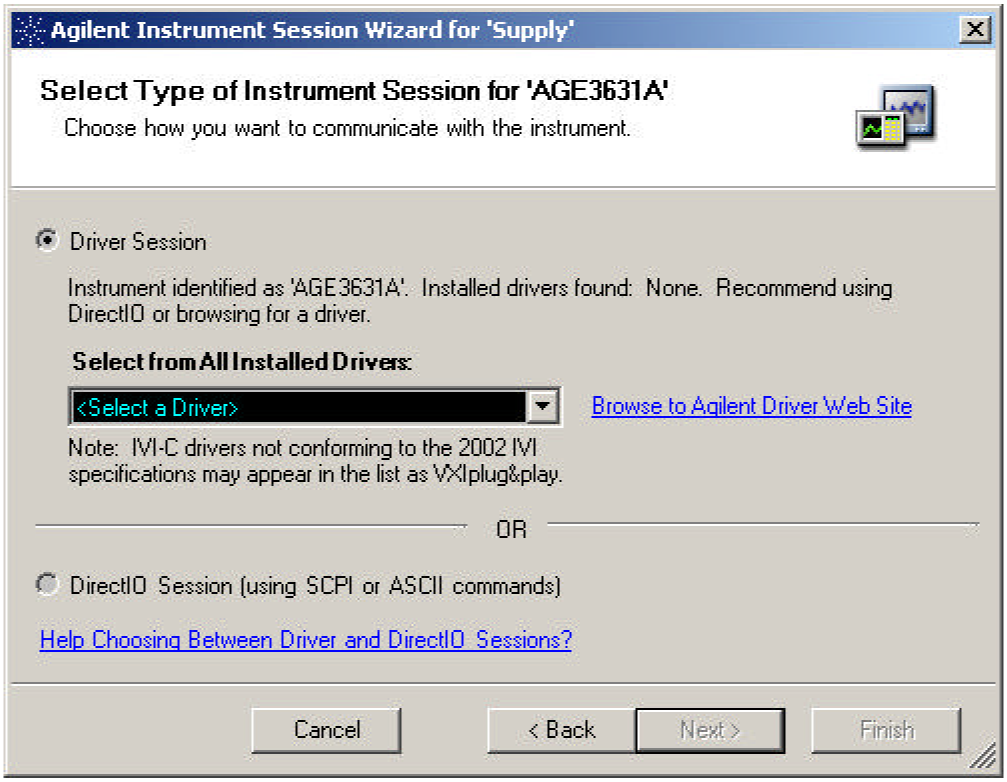
You can generate code and insert it into your existing source code using the instrument session wizard. If you plan to do this, be sure you have a project open before you create the instrument session. To add a new instrument session do the following:

Starting an Instrument Session

Step	Action	Notes
1 Highlight an instrument	Right Click the highlighted instrument	
2 You see a context sensitive pop-up menu	From the pop-up menu, select **Add Instrument Session**	• This starts the Instrument Session Wizard
3 Study the Wizard's overview	Select **Next** to proceed	

Starting an Instrument Session (continued)

Step	Action	Notes
4 Decide between using an instrument driver or DirectIO	**a** To help make that decision, review the matching drivers found by the Wizard. If no matches are found, the following message appears, "No matches may be found if the instrument cannot be identified (*IDN isn't returned). In some cases, the instrument may be turned off."	



Select Type of Instrument Session for 'AGE3631A'

Choose how you want to communicate with the instrument.

Driver Session

Instrument identified as 'AGE3631A'. Installed drivers found: None. Recommend using DirectIO or browsing for a driver.

Select from All Installed Drivers:

<Select a Driver>

Browse to Agilent Driver Web Site

Note: IVI-C drivers not conforming to the 2002 IVI specifications may appear in the list as VXIplug&play.

OR

DirectIO Session (using SCPI or ASCII commands)

Help Choosing Between Driver and DirectIO Sessions?

Figure 16 An Installed Driver Could Not be Found

	b Review the list appearing under “Select from All Installed Drivers”. The Wizard defaults to the first matching driver. Select from the matching drivers.	• There may be no matching drivers. See c.
	c If no driver is available, you may wish to browse the Agilent website to look for a driver for your instrument.	
	d If you do not want to do **b)** or **c)**, you can still communicate with your instrument directly. This requires you to be familiar with an instrument command language supported by the instrument.	• SCPI is an example of a command language supported by many instruments. • More help is available about selecting the proper strategy by selecting **Help on Selecting a Driver**
5 Select the driver or DirectIO	**a** If you choose the driver option, select the appropriate driver **b** Press **Next** when done	

Generating Code

As you finish a pass through the Instrument Session Wizard, you may choose to generate code. The generated code establishes a DirectIO object or a Driver object. All generated code includes fully qualified pathnames. If you choose to reference the libraries by using an Imports statement in VB.NET or a Using statement in C#, you do not need to use fully qualified pathnames.

There are other ways to generate code such as using the pop-up menu or dragging and dropping an instrument object into your source code. Of course, if you would prefer to write the code yourself, you can.

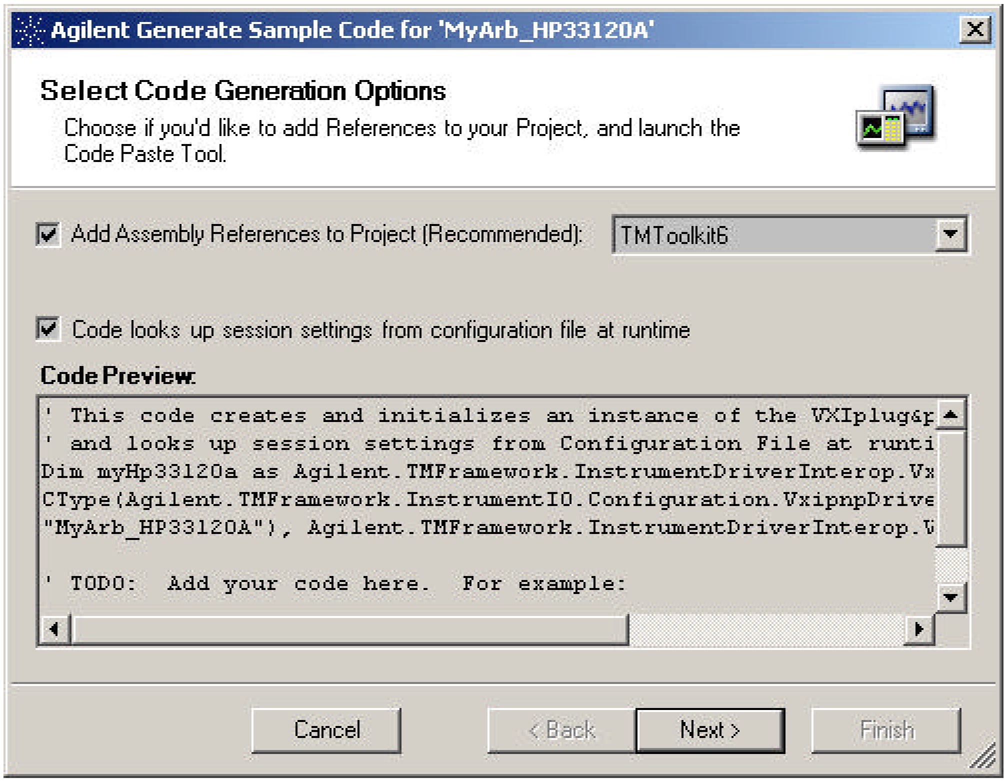
Generating Code Using the Instrument Session Wizard

Step	Action	Notes
1 Specify options	Select the options you want invoked when the code is generated	• There is a preview process, so you can reverse your choices later if you do not like them
2 Preview the generated code	**a** Toggle the checkbox to see how the code looks with different options on or off **b** When you have decided on your options, select **Next**	
3 Confirm your selections	**a** Review the Summary screen **b** Select **Back** to make changes **c** Select **Finish** to proceed	
4 Select a location in your code	**a** The wizard *cannot* know the correct place to put the generated code. Only you can place the cursor at the correct insertion point. **b** If you are not sure where to put the code, click the **Where do I paste my code?** link.	• It may be easier to paste the generated code into your source code and then move it as needed. This could work particularly well if an open section is set aside for this purpose.
5 Paste the generated code	Use CTRL-C to copy and CTRL-V to paste or use the **Paste** button in the paste tool	• The generated code is placed at the cursor insertion point in your source code. • The source code comments are kept at your discretion

Drag and Drop Instrument Objects or Sessions

Another way to create an Instrument Session and generate code is to drag-and-drop an instrument from the Instrument Explorer into your source code. This launches the full version of the Instrument Session Wizard.

Once a session is created, it appears in the tree view as a branch beneath the parent instrument. You can drag and drop a session into your source code, and an abbreviated version of the Instrument Session Wizard runs. In the abbreviated version, you go directly to the Select Code Generation Options dialog. See Figure17.


Agilent Generate Sample Code for 'MyArb_HP33120A'

Select Code Generation Options

Choose if you'd like to add References to your Project, and launch the Code Paste Tool.

☑ Add Assembly References to Project (Recommended): TMToolkit6

☑ Code looks up session settings from configuration file at runtime

Code Preview:

```
' This code creates and initializes an instance of the VXIplug&p
' and looks up session settings from Configuration File at runti
Dim myHp33120a as Agilent.TMFramework.InstrumentDriverInterop.Vx
CType(Agilent.TMFramework.InstrumentIO.Configuration.VxipnpDrive
"MyArb_HP33120A"), Agilent.TMFramework.InstrumentDriverInterop.V

' TODO:  Add your code here.  For example:
```

Figure 17 The Abbreviated Version of the Instrument Session Wizard.

Once you set your options, you go directly to the summary screen. If you are satisfied with the options you selected, you choose **Finish** and the generated code is immediately dropped into your source code. Note that the paste tool is not used because when you dragged-and-dropped the Instrument Session into your source code, you also established the insertion point for the generated code.

NOTE When you drop the session into your source code, you are establishing the insertion point for the generated code.

CAUTION You can drag-and-drop a session into your source code. However, generated code always includes instrument initialization code. You probably do not want to initialize your instrument more than once. If you drag-and-drop Instrument Sessions, be sure to remove the instrument initialization code if you do not want it.

Instruments Session Icons

Instrument sessions are represented iconically in Instrument Explorer as follows:

VXI*plug&play* Instrument Session

IVI Instrument Session

DirectIO Instrument Session

Unknown Instrument Sessions

4
Using Interactive IO

This chapter describes how to use the Interactive IO tool to quickly and easily control instruments.

Quick and Simple Instrument Control

Why Control Instruments?

The world of test and measurement revolves around the need to make and evaluate measurements. Typically, making a measurement requires setting up the conditions for the measurement, making the measurement, and evaluating the results. This requires regular interaction with instruments.

What Happens if There is a Problem Controlling an Instrument?

What if the instrument being controlled fails to respond correctly, or not at all? You may know that something failed to work, but you may not know why. To troubleshoot this kind of problem, start with simple communication before attempting more complex interaction. Simple communication may be nothing more than sending a few commands to the instrument and verifying its responses. In many cases, it is useful to send and receive these commands without having to create a formal software application.

How Can You Quickly and Simply Control Instruments?

The Interactive IO tool is a standalone software tool that lets you:

- Send a command to an instrument in an industry-standard instrument control language
- Read a response from an instrument
- Send & Read a command in one step
- Quickly recall previously used commands from a list
- View the history of commands and responses

To begin quick and simple instrument control

	Step	Notes
1	Run the Interactive I/O tool.	• Select **T&M Toolkit** from the main menu and select Interactive IO from there. • Click the Interactive IO icon.
2	Specify which instrument to connect to.	• Choose **Connect** > **Resource Name** from Interactive IO's menu bar. Then specify which instrument to use.
3	Create a command.	• Type a command in the **Command** field or choose a command from the typing aid.
4	Execute the command or query.	• Click the **Send Command**, **Read Response**, or **Send & Read** button, as appropriate.

NOTE

When specifying which instrument to use, you must type the address of the instrument into the **Resource Name** field. If you do not know the instrument's address, you can use Instrument Explorer to find the address. If you are using a Serial connection, you must first establish this connection through the Agilent I/O Config tool before starting Interactive IO.

How Instrument Control is Presented

The Interactive IO tool contains several areas of interest, as shown in Figure 18.

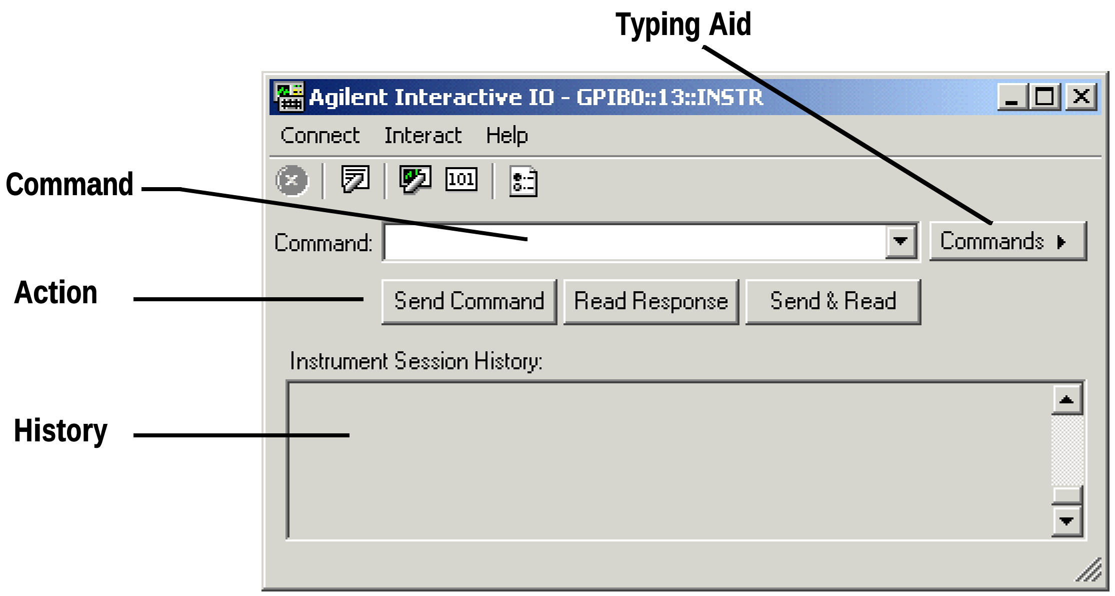


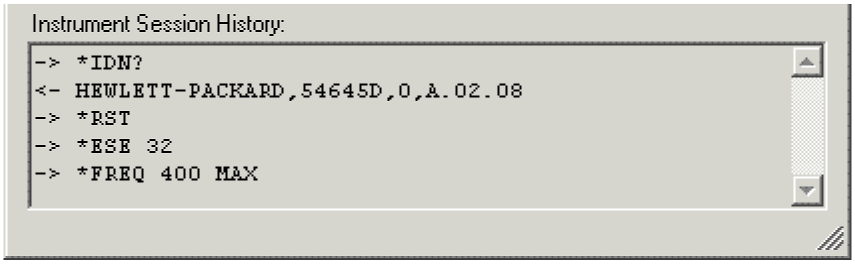
Figure 18 The Interactive IO Main Display

Table2lists the areas of the Interactive IO tool and their purpose.

Table 2 Areas of the Interactive IO Tool

Command	Type your command or query here.
Action	Click whichever button is needed to send the command, read the response, or send and read the response.
History	List of commands/queries/responses sent to or read from the instrument for the current session.
Typing Aid	A list of generic commands you can quickly access.

Your interaction with the instrument is shown in the **Instrument Session History** for the current session, as shown in Figure19.



```
-> *IDN?
<- HEWLETT-PACKARD,54645D,0,A.02.08
-> *RST
-> *ESE 32
-> *FREQ 400 MAX
```

Figure 19 History for a Session

Commands sent to the instrument are preceded by `->`. Responses from the instrument are preceded by `<-`. For example,

```
-> *IDN?
```

sends a command that asks the instrument to identify itself. And

```
<- HEWLETT-PACKARD,54600B,0,A.01.35
```

is the response returned by the instrument.

Commands and Responses vs. Queries

The Interactive IO tool lets you send commands, read responses, or combine the send and read into a single operation. A command is a string sent to an instrument. A response read from an instrument returns data from the instrument.

Shortcuts When Using the Interactive IO Tool

Recall List of Commands

When you type commands or queries, the Interactive IO tool remembers them. You can recall them at any time by dropping down the list, as shown in Figure20.

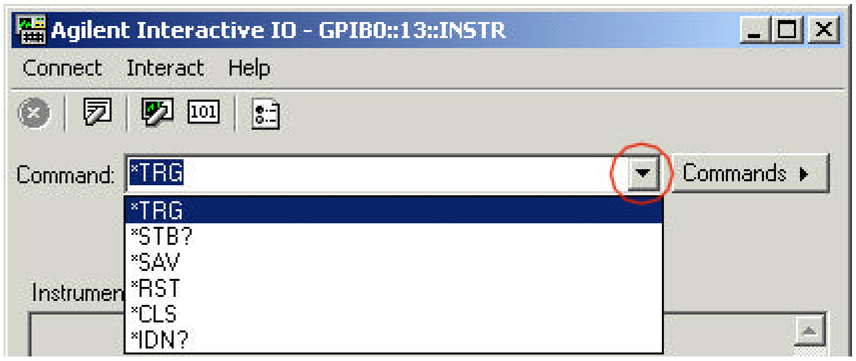


Figure 20 Command History

Typing Aid for Commands

Clicking the button to the right of the drop-down list where you type commands invokes a shortcut menu that contains frequently used commands, as shown in Figure21. Selecting an item in the menu adds it to the command field and saves you some typing.

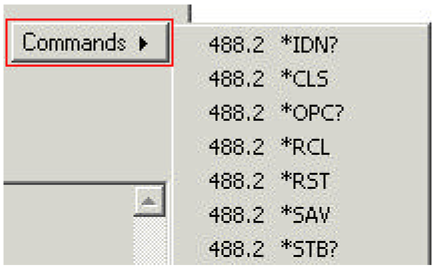


Figure 21 The Typing Aid

5
Using the IO Monitor

This chapter describes how to use the IO Monitor tool to trace, debug, and optimize I/O programs.

Monitoring Interaction with Instruments

Why Monitor Interaction with Instruments?

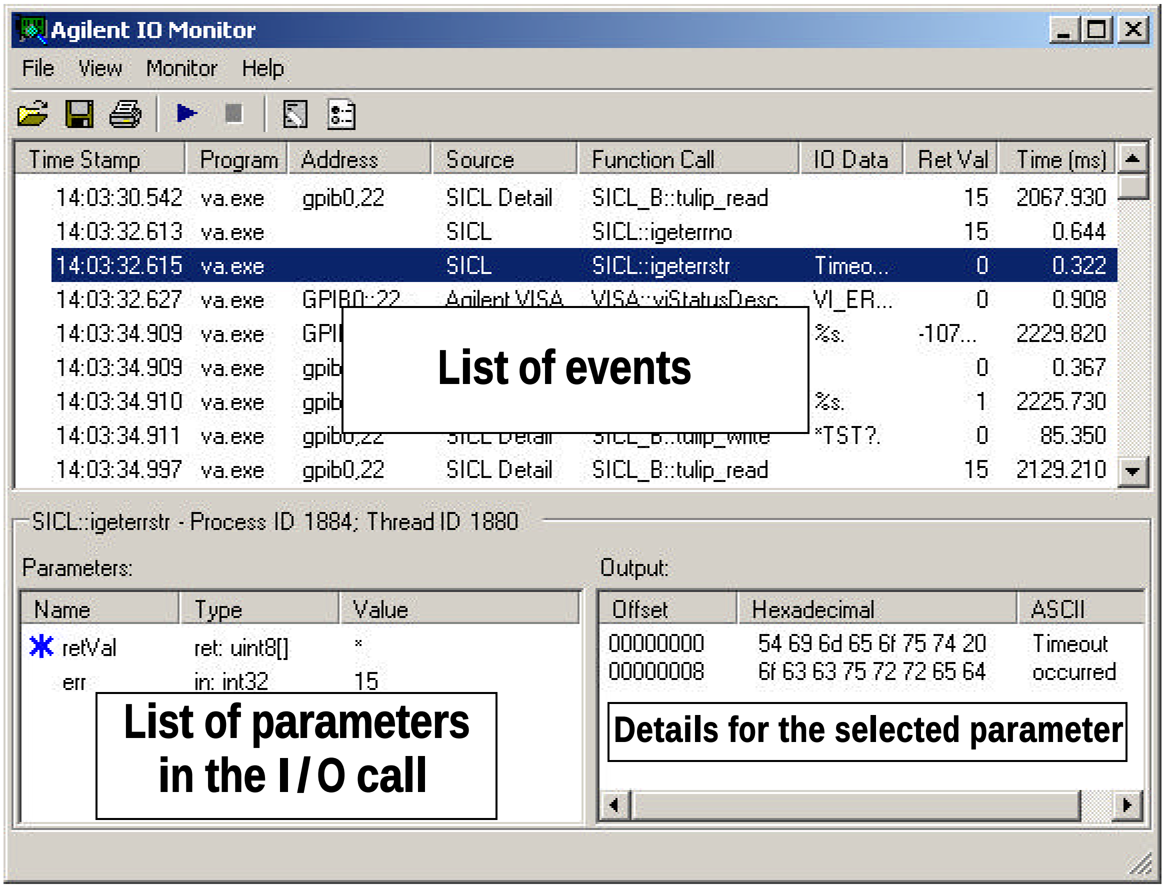
An application you write that communicates with an instrument or other hardware may not work correctly at first. The complex nature of the communication between the application and the instrument—e.g., commands sent via APIs that migrate through layers of software such as I/O libraries—can make it difficult to decide where and how problems arise. What would be helpful is a means to monitor that communication and, ideally, be able to interact with it in a way that helps you debug your application.

How Can You Monitor Interaction with Instruments?

The IO Monitor provides .NET programmers with an easy way to trace and debug I/O programs. The IO Monitor is a standalone software tool that lets you:

- Trace the API call stack of an I/O library
- Selectively enable and disable I/O library layers
- Filter the reported data to control what you see
- Log trace information to a file (for review later or for sending to support personnel)

The IO Monitor keeps track of a sequence of events. An event can be an entry into or exit from an API and an occurrence of an asynchronous interrupt. The IO Monitor filters and formats the events it receives and presents them in several panels. See Figure22.


Agilent IO Monitor
File View Monitor Help
List of events
List of parameters in the I/O call
Details for the selected parameter

Time Stamp	Program	Address	Source	Function Call	IO Data	Ret Val	Time (ms)
14:03:30.542	va.exe	gpib0,22	SICL Detail	SICL_B::tulip_read		15	2067.930
14:03:32.613	va.exe		SICL	SICL::igeterrno		15	0.644
14:03:32.615	va.exe		SICL	SICL::igeterrstr	Timeo...	0	0.322
14:03:32.627	va.exe	GPIB0::22	Agilent VISA	VISA::viStatusDesc	VI_ER...	0	0.908
14:03:34.909	va.exe	GPI			%s.	-107...	2229.820
14:03:34.909	va.exe	gpib				0	0.367
14:03:34.910	va.exe	gpib			%s.	1	2225.730
14:03:34.911	va.exe	gpib0,22	SICL Detail	SICL_B::tulip_write	*TST?.	0	85.350
14:03:34.997	va.exe	gpib0,22	SICL Detail	SICL_B::tulip_read		15	2129.210

SICL::igeterrstr - Process ID 1884; Thread ID 1880

Parameters:

Name	Type	Value
✱ retVal	ret: uint8[]	*
err	in: int32	15

Output:

Offset	Hexadecimal	ASCII
00000000	54 69 6d 65 6f 75 74 20	Timeout
00000008	6f 63 63 75 72 72 65 64	occurred

Figure 22 The Presentation of I/O Data

The IO Monitor also displays the details of a selected event. This additional information includes:

- Application name

- Host PC name
- Contents of the output (write) and input (read) buffers
- Selectable data display format
- Elapsed time in the API (for profiling the performance of I/O applications)

Getting Started Monitoring Interaction with Instruments

When you run the IO Monitor, it automatically attaches itself to the stream of I/O data between an application making I/O calls via Agilent VISA-COM, Agilent VISA/SICL libraries and the instrument being called.

NOTE You can run the IO Monitor at any time while debugging an application so long as your application uses the Agilent VISA-COM and Agilent VISA/SICL libraries to access instruments.

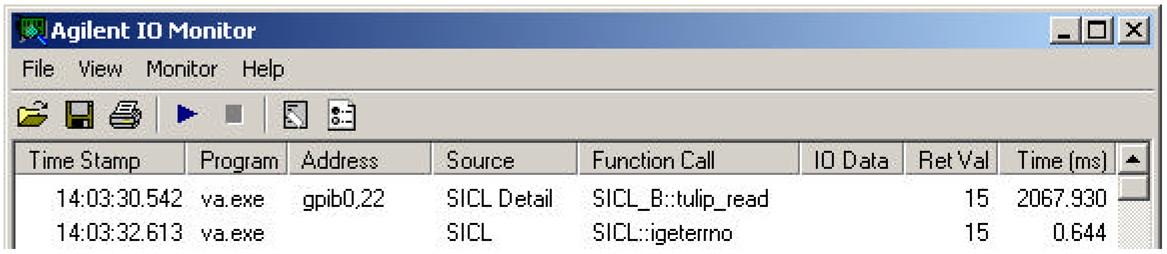
To begin monitoring interaction with instruments

	Step	Notes
1	Run the application that makes I/O calls you wish to monitor.	IO Monitor supports both Agilent and National Instrument interface cards.
2	Run the IO Monitor.	From the main menu, select T&M Toolkit > IO Monitor or Click the IO Monitor icon.
3	Tell the IO Monitor to begin monitoring I/O information.	Choose Start from the Monitor menu (Monitor > Start) in the IO Monitor's menu bar.
4	Switch between your application and the IO Monitor as needed.	Use your application to send I/O data and then use the IO Monitor to examine the data.

How I/O Data is Presented

An Overall View of I/O Data

The uppermost panel lists the I/O events that occur while monitoring I/O operations.



```
Time Stamp    Program  Address   Source       Function Call        IO Data  Ret Val  Time (ms)
14:03:30.542  va.exe   gpib0,22  SICL Detail  SICL_B::tulip_read             15      2067.930
14:03:32.613  va.exe             SICL         SICL::igeterrno                15         0.644
```

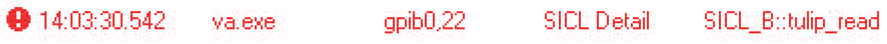
Figure 23 Events Traced by IO Monitor

Table3shows the kinds of I/O data associated with events.

Table 3 I/O Data Associated with Events

Time Stamp	When the event began
Program	Which application caused the event
Address	The I/O bus address used by the event
Source	Which I/O library (API) handled the event
Function Call	The function call sent during the event
IO Data	The buffered I/O data (if any) sent during the event displayed in ASCII format
Ret Val	The value (if any) returned by the event
Time (ms)	How long, in milliseconds, the event took

If the event results in an I/O error, the line where that event appears is preceded by a red warning icon and the event is presented in red, as shown in Figure24.

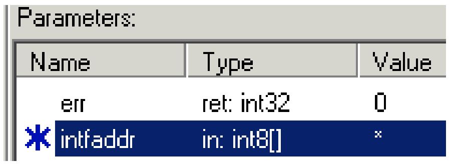


14:03:30.542 va.exe gpib0,22 SICL Detail SICL_B::tulip_read

Figure 24 An Event That Generated an Error

A View of Parameters Passed in I/O Data

When you select an event in the uppermost panel, the lower left panel lists the parameters passed in the I/O call to the instrument. Figure25 shows an output parameter that was passed.



Parameters:

Name	Type	Value
err	ret: int32	0
intfaddr	in: int8[]	*

Figure 25 Parameters Passed in the I/O Call Associated with an Event

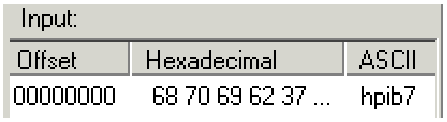
Table4shows the details of the list of parameters.

Table 4 Details of the Parameters Passed in an I/O Call

Name	Name of the parameter
Type	Data type of the parameter (* denotes an array of bytes)
Value	Value of the parameter

A View of I/O Data in a Buffer

If a parameter has an asterisk (*) preceding its name, it means the parameter passed an array of bytes in a buffer during the I/O call. To examine the contents of the buffer, you must select that parameter and look in the bottom right panel. Figure26 shows an example of data in an input buffer.



Input:

Offset	Hexadecimal	ASCII
00000000	68 70 69 62 37 ...	hpib7

Figure 26 Example of Data in a Buffer

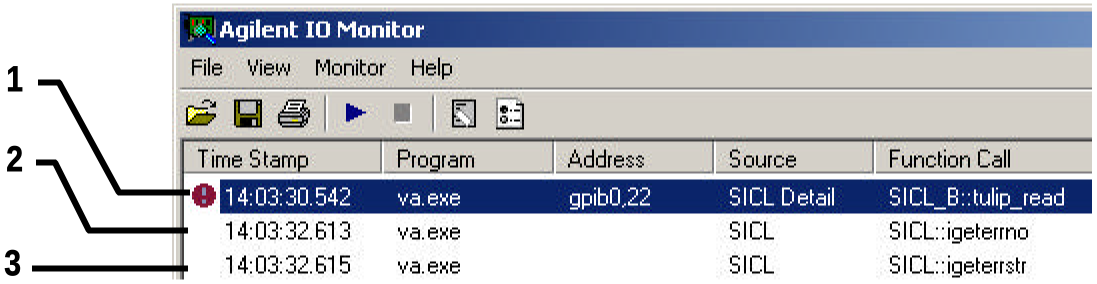
The details of buffer information are shown in Table5

Table 5 Details of Buffer Information

Offset	Offset from the start of the buffer
Hex	The value in the buffer expressed in hexadecimal format
ASCII	The value in the buffer expressed in ASCII format

Tracing the API Call Stack of an I/O Library

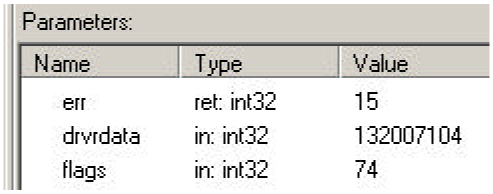
The list of events displayed by the IO Monitor shows calls to APIs in the order that they occur, as well as the data associated with them. This provides you with a view of the call stack, as shown in Figure 27.



Time Stamp	Program	Address	Source	Function Call
14:03:30.542	va.exe	gpib0,22	SICL Detail	SICL_B::tulip_read
14:03:32.613	va.exe		SICL	SICL::igeterrno
14:03:32.615	va.exe		SICL	SICL::igeterrstr

Figure 27 An Ordered List of Calls to the Call Stack

Similarly, the parameters passed to the call stack in a function call are passed in sequence, as shown in Figure28

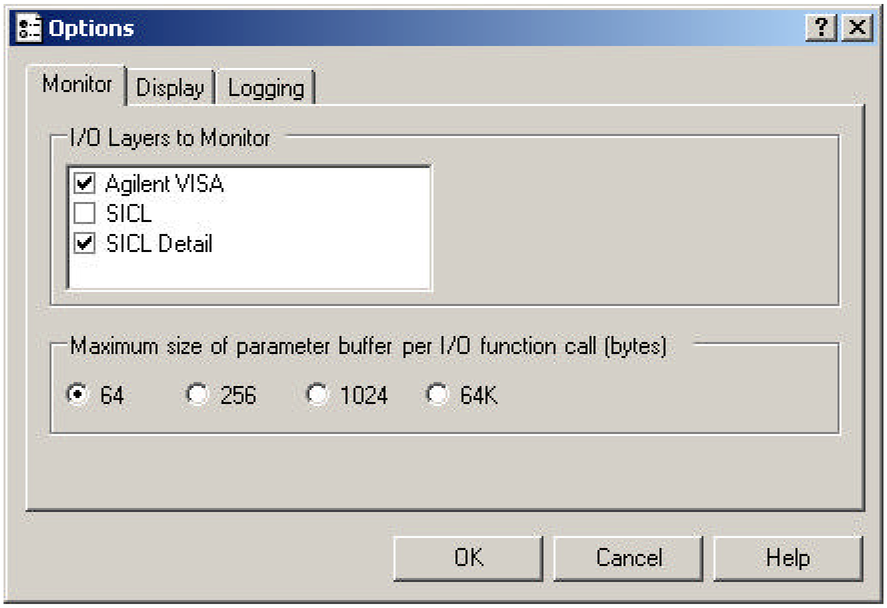

Parameters:

Name	Type	Value
err	ret: int32	15
drvrdata	in: int32	132007104
flags	in: int32	74

Figure 28 An Ordered List of Parameters Passed in a Function Call

Selectively Enabling/Disabling I/O Library Layers

The IO Monitor lets you choose which I/O library layers to monitor, as shown in Figure29 on page 55. If a particular I/O library is not of interest to you, simply deselect it in the list.



I/O Layers to Monitor

- [✓] Agilent VISA
- [ ] SICL
- [✓] SICL Detail

Maximum size of parameter buffer per I/O function call (bytes)

[● 64] ○ 256 ○ 1024 ○ 64K

Figure 29 Enabling/Disabling I/O Layers to Monitor

You select I/O layers based upon the entry point your program is using or upon the entry point being used by an instrument driver you are employing. VISA-COM, VISA, and SICL are all valid entry points for accessing your instruments. SICL Details offers you a look at the lowest level of activity. See Figure30 for a diagram of where the trace points are located.

You can change these setting from another location. From the main menu, select View > View by IO Layers. This selection process is identical to the selection process you see in Figure29's Options > Monitor tab.

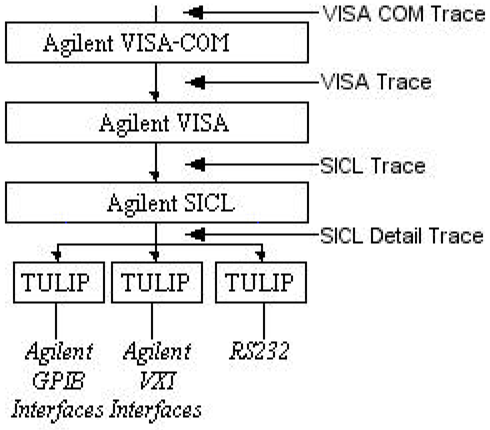
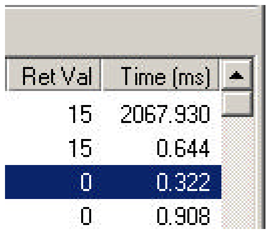


Figure 30 The Location of Trace Points in IO Monitor

Optimizing the Performance of I/O Operations

As shown in Figure 31, the Time column associated with each I/O event shows how long (in milliseconds) the event took. You can examine the Time values to determine where bottlenecks exist in your I/O code. For example, if one I/O operation takes much longer than the others, that operation may be a good place to begin optimizing your code. After you have optimized the code, you can rerun the I/O application and examine its events to verify improvements.

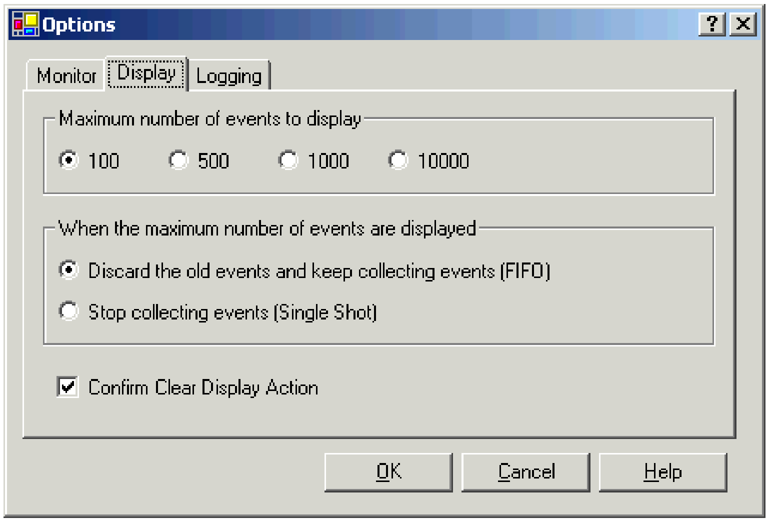


Ret Val	Time (ms)
15	2067.930
15	0.644
0	0.322
0	0.908

Figure 31 Identifying the Time Spent in I/O Events

Controlling How Many Events are Displayed

You can control the amount of data being presented as shown in Figure32. The more events you choose to display, the more data you see.



Maximum number of events to display

[✓ 100] 500 1000 10000

When the maximum number of events are displayed

[✓ Discard the old events and keep collecting events (FIFO)]

Stop collecting events (Single Shot)

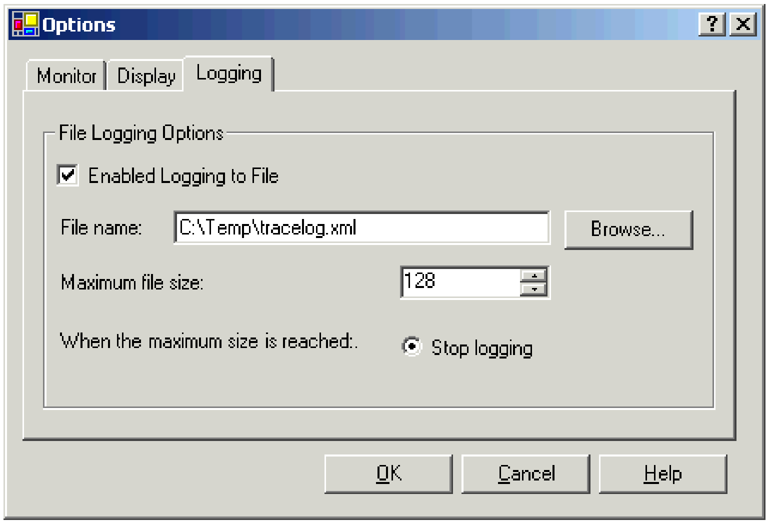
[✓ Confirm Clear Display Action]

Figure 32 Filtering Data by the Number of Events

Logging Trace Information

If desired, you can save the trace information accumulated by the IO Monitor to a file. This information is saved in XML format. Later, you can open the file and examine its contents.

Figure33 shows the options you can specify when logging to a file. These options include the location of the file, its maximum size, and what action to take if that size is exceeded.



[✓] Enabled Logging to File

File name: C:\Temp\tracelog.xml

Maximum file size: 128

When the maximum size is reached:. ● Stop logging

Figure 33 Specifying the Options for Logging to a File

6
Using the Driver Wrapper Wizard

This chapter describes the Driver Wrapper Wizard, how to use it, the drivers supported by it, and some of the typical problems with wrapped drivers.

Using the Driver Wrapper Wizard

Expanding the Driver Possibilities

There are hundreds of existing instrument drivers. These drivers were written to meet industry standards, such as VXI*plug&play*, that existed before .NET and the T&M Toolkit. The Driver Wrapper Wizard is for the programmer who is comfortable with VXI*plug&play* drivers and wants to quickly capture VXI*plug&play* functionality for T&M Toolkit. With the Driver Wrapper Wizard, you can create a .NET and Agilent T&M Toolkit-compliant wrapper around the same standards-compliant drivers that you have relied upon for years. This gives you the extra power of the latest software development tools without limiting your access to legacy technology.

If you are new to VXI*plug&play* or want to start quickly with your instruments, T&M Toolkit's Instrument Explorer offers similar functionality through the Driver Session Wizard. The Instrument Explorer rapidly identifies your instrument configuration and provides quick communication with a variety of drivers including VXI*plug&play*. It is a great tool to use while you build your expertise with device drivers.

Which Drivers are Supported?

The Driver Wrapper Wizard wraps VXI*plug&play* and IVI-C drivers. The IVI-C wrapper is based on the IVI Foundation's IVI-C Driver standard, version 1.0, approved in 2002. There are some IVI-C drivers available that are based on an earlier draft standard. Due to the significant changes between this earlier draft standard and the final standard, Agilent T&M Toolkit recognizes these drivers as VXI*plug&play* drivers.

The VXI*plug&play* wrapper is based on the most recent version of the VXI*plug&play* driver standard. Because VXI*plug&play* is a mature standard, Agilent is shipping more than a hundred of these drivers with the T&M Toolkit; many of these are already wrapped.

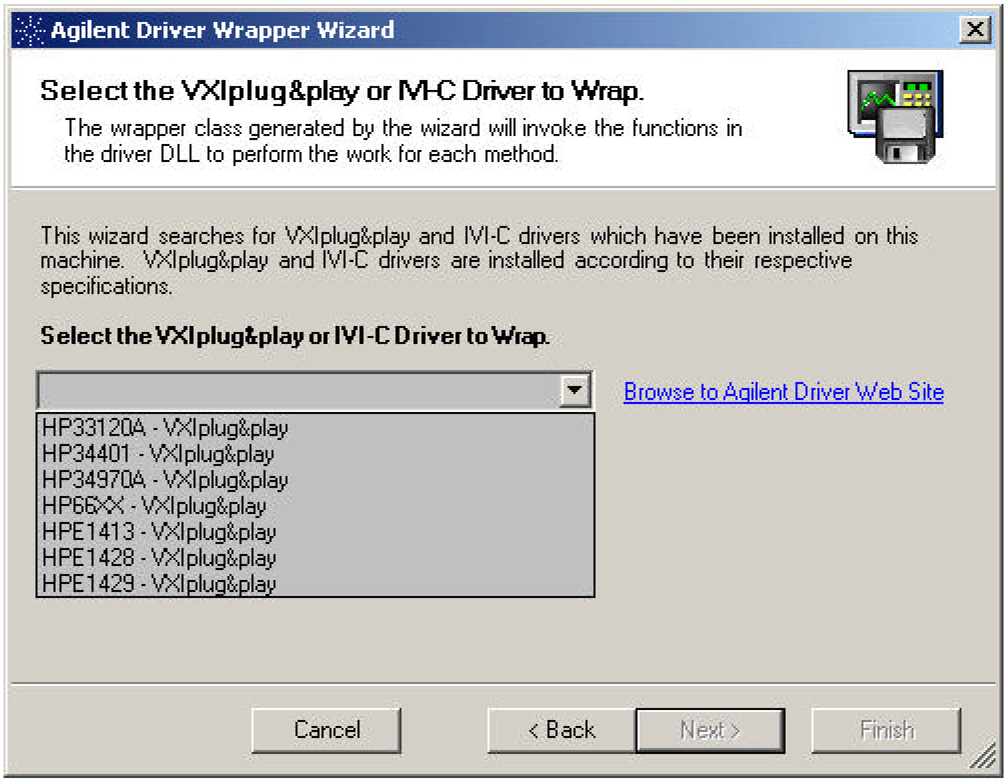
Starting and Using the Wizard

NOTE Before discussing the Driver Wrapper Wizard, it is important to draw a distinction between the wrapper and the driver. The wrapper does not replace the driver. The wrapper exists separately, and your source code indirectly uses the driver.dll file by using the wrapper assembly. If the driver is uninstalled, the wrapper does *not* work.

The Driver Wrapper Wizard takes you through several steps as you create the wrapper. There is no set procedure for the Driver Wrapper Wizard because the steps adjust dynamically depending upon the driver you have selected. In general, the steps are as follows:

1 A welcome screen explaining what you are about to do and what you are required to provide.

2 Choose from a list of VXI*plug&play* or IVI-C drivers installed on your machine. You can also download additional drivers from the Web. See Figure 34.


Agilent Driver Wrapper Wizard

Select the VXIplug&play or IVI-C Driver to Wrap.

The wrapper class generated by the wizard will invoke the functions in the driver DLL to perform the work for each method.

This wizard searches for VXIplug&play and IVI-C drivers which have been installed on this machine. VXIplug&play and IVI-C drivers are installed according to their respective specifications.

Select the VXIplug&play or IVI-C Driver to Wrap.

HP33120A - VXIplug&play
HP34401 - VXIplug&play
HP34970A - VXIplug&play
HP66XX - VXIplug&play
HPE1413 - VXIplug&play
HPE1428 - VXIplug&play
HPE1429 - VXIplug&play

Figure 34 Selecting a VXI*plug&play* or IVI-C Driver to Wrap

3 Customize the wrapper generation process by choosing to generate XML. The XML is generated automatically and is used later to create IntelliSense documentation for the wrapper.

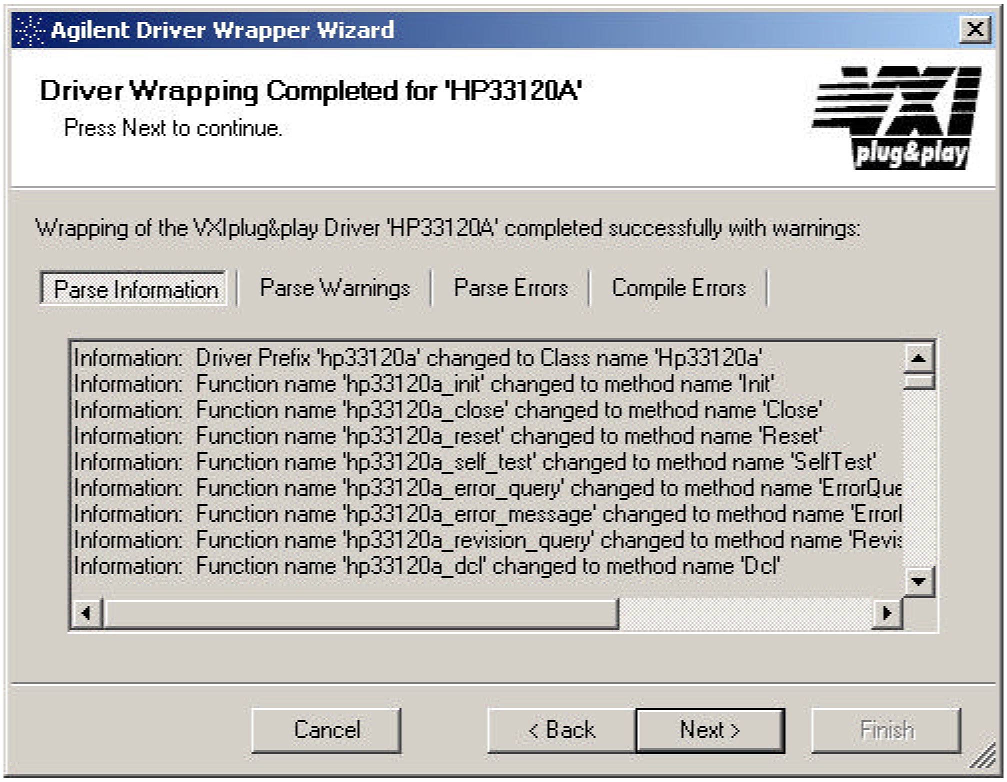
4 The driver is parsed and wrapped. You can review lists of errors or warnings generated by the parsing or compilation.

5 At this point, you have the option to have the Driver Wrapper Wizard include the necessary Project References and generate sample code. This step is optional but strongly recommended.

6 Project References are added to your project and sample code is available from a Code Paste Tool for you to paste into your project.

Troubleshooting

When the Driver Wrapper Wizard finishes parsing and compiling the wrapper, a screen similar to Figure35 appears. Using the tabs, you can review any messages, warnings, or errors that were generated during the parse and/or compile processes.



Wrapping of the VXIplug&play Driver 'HP33120A' completed successfully with warnings:

```
Information:  Driver Prefix 'hp33120a' changed to Class name 'Hp33120a'
Information:  Function name 'hp33120a_init' changed to method name 'Init'
Information:  Function name 'hp33120a_close' changed to method name 'Close'
Information:  Function name 'hp33120a_reset' changed to method name 'Reset'
Information:  Function name 'hp33120a_self_test' changed to method name 'SelfTest'
Information:  Function name 'hp33120a_error_query' changed to method name 'ErrorQue
Information:  Function name 'hp33120a_error_message' changed to method name 'ErrorI
Information:  Function name 'hp33120a_revision_query' changed to method name 'Revis
Information:  Function name 'hp33120a_dcl' changed to method name 'Dcl'
```

Figure 35 Troubleshooting Wrapper Creation

The Driver Wrapper Wizard translates the driver's function names to .NET-compatible method names. It is common to see many of these translations on the **Parse Information** tab textbox.

The Driver Wrapper Wizard also verifies that all of the driver functions exist in both the driver's **.h** and **.fp** files. If a function does not exist in both locations, the driver vendor probably did not intend the function to be visible. In this case, the Driver Wrapper Wizard does not wrap the function. The list of functions that were not wrapped is shown in the **Parse Warnings** tab text box. Most of these warnings are also normal.

Parse Errors and Compile Errors

As the Driver Wrapper Wizard parses a driver to create a wrapper, the driver is verified for compliance with the standards set by its governing body. If a driver fails any of these tests, the Driver Wrapper Wizard refuses to create a wrapper and, in almost all instances, tells you why the driver failed. For an example of how this screen might look, see Figure35 on page63. Contact the driver manufacturer for help in this situation.

You are also prompted to send the failure information to Agilent. While Agilent cannot resolve issues with drivers, problems with the Driver Wrapper Wizard are of great interest and every effort is made to resolve them.

If the driver wrapper is created and fails when you use it, this is almost always a problem with the instrument setup or configuration. The most common error is **VI_ERROR_RSRC_NFOUND**, which is generated by VISA and is usually caused by an incorrect address or hardware setup. This error appears as an exception when the program is run.

If the physical connections all appear to be correct and the addressing also looks accurate, other errors may be found in either the **VISA.h** or the <*PREFIX*>**.h** driver specific file.

7 Using the DirectIO Class to Control an Instrument

This chapter describes the features of the Direct IO class library found in Agilent.TMFramework.InstrumentIO.

Describing the DirectIO Class

The DirectIO class provides easy ways to move data between your instruments and your PC. In this chapter, almost all of the properties and methods of the DirectIO class are discussed.

Reading and Writing Data

The many read and write methods of the DirectIO class convert instrument data formats into programmatic data types while reading and writing between the instrument and your PC. These methods have, as most T&M Toolkit methods, to be as similar to the .NET syntax as possible, so your .NET learning is easily transferred to the T&M Toolkit environment.

There are three instrument formats: a Number, a List, and an IEEE Block.

- A Number is a numeric value.
- A List is a set of delimited values.
- The IEEE Block is an IEEE 488.2 definite binary block.

Table6shows a conversion matrix. For any listed data type, you can look at the instrument format columns and determine if there is a DirectIO Read or Write method that does a conversion.

Table 6 A Conversion Matrix for Instrument Formats and Data Types

	Instrument Format		
Data Type	**Number**	**List**	**IEEE Block**
Byte	Read/Write		
Double	Read/Write		
Int16	Read/Write		
Int32	Read/Write		
Single (float)	Read/Write		
String		Read	Read/Write

Table 6 A Conversion Matrix for Instrument Formats and Data Types

	Instrument Format		
Data Type	**Number**	**List**	**IEEE Block**
Object	Write		
ObjectArray		Write	
ByteArray		Read/Write	Read/Write
DoubleArray		Read/Write	Read/Write
Int16Array		Read/Write	Read/Write
Int32Array		Read/Write	Read/Write
SingleArray		Read/Write	Read/Write
StringArray		Read/Write	Read/Write

The DirectIO class also supports the **Write** and **WriteLine** methods, the **UnbufferedRead** method, and the **UnbufferedWrite** method. The **Write** and **WriteLine** methods vary from other write methods because they provide string formatting. **WriteLine** also appends a newline. The **Newline** property defaults to an ASCII linefeed (LF).

The **UnbufferedRead** and **UnbufferedWrite** methods transmit and receive data directly between the instrument and the PC without the use of system buffers.

Reads can be terminated with a character. You can define the character using the **TerminationCharacter** property. However, the termination character is not active until you set the **TerminationCharacterEnabled** property to **True**.

Controlling the Instrument Session

The DirectIO class includes many properties and methods for controlling and configuring your instrument session. Some of these are gateways to low-level control of the instrument.

Buffer Management

By default, the **AutoFlush** property is **True**, which automatically flushes the buffer after every write. If you want finer control of the buffers, you can call the **FlushRead** or **FlushWrite** methods to clear the read and write buffers.

The size of the buffers is controlled by the **SetReadBufferSize** and **SetWriteBufferSize**. The default size is 1,000 bytes.

Instrument Session Management

The DirectIO class has several properties and methods that can be used to set instrument configurations and control their behavior.

Table 7 DirectIO Properties for Instrument Session Management

Property Name	Description
Address	Gets the address of the current instrument
HardwareInterfaceType	Gets the hardware interface type such as GPIB or ASRL
InstrumentByteOrder	Sets the instrument byte order to either big endian or little endian
Locked	Gets the status of the session. If the session is locked, the status is **True**
waitForLockTimeout	Gets or sets the millisecond value for the I/O timeout during the creation of the instrument object

NOTE

The Address, Locked, and waitForLockTimeout properties can be set when the instrument is first created as an object. At all other times, these properties get the assigned value.

Table 8 DirectIO Instrument Session Management

Method Name	Description
AssertTrigger()	Sends a trigger to the instrument.

Table 8 DirectIO Instrument Session Management (continued)

Method Name	Description
ReadStatusByte()	Reads the status byte of instruments.
Clear()	Clears an instrument.
Lock()	Locks an instrument session and does not allow other session to access the instrument.
Unlock()	Disables exclusive access to an instrument.

The VISA Support Class

The VISA Support Class contains a comprehensive but not complete listing of VISA properties, enumerations, and methods. It is also used to validate whether a VISA Resource Name is well formed.

NOTE While the VISA Support Class tests whether a VISA Resource Name is well formed, it does not verify whether the address is valid or not.

Using Low Level Gpib, Serial, Tcpip, and USB

DirectIO provides the basic features needed for interacting with your instruments. If you work within the DirectIO class, the interface for the instrument becomes transparent. For example, suppose you have written some source code for an oscilloscope. If you use DirectIO to interact with the oscilloscope, it does not matter what type of interface you are using. However, if you use the lower-level methods by accessing the Gpib, Serial, Tcpip, or USB properties, your source code is tied to a specific interface and has to be edited whenever the interface changes. This reduces the source code’s reusability and your efficiency.

The Gpib, Serial, Tcpip, and USB properties are access points to low-level instrument control that you sometimes need to use. In the following code snippet, the Gpib property is used to access the GpibInstr subclass.

```
Module Module1
   Sub Main()
      Dim scope As New DirectIO(“GPIB0::22::INSTR”)
      scope.Gpib.SendLocalLockOut()
   End Sub
End Module
```

Low-Level Access through DirectIO.Gpib

You can access several properties and methods through the DirectIO.Gpib property. These properties and methods simplify the low-level configuration and control of instruments accessed through the GPIB protocol. The connect string for a instrument with a GPIB connection is:

GPIB[*board*]::*primary address*[::*secondary address*][::INSTR]

Items in square brackets, [], are optional and items in *italics* are supplied by you. All other entries are required.

Addressing

There are several addressing related properties.

Table 9 Properties Related to Addressing

Property Name	Description
HardwareInterfaceNumber	Gets the zero based number of the GPIB board.
PrimaryAddress	Gets the primary address of the GPIB board.
RepeatAddressing	Gets whether the same address should be used for the next read or write operation.
SecondaryAddress	Gets the secondary address of the GPIB board.
UnAddressing	Gets whether the same address is not used for the next read or write operation.

Remote Operation

There are several methods and one property that affect remote operations.

- **RenState** is a property that gets the status of the Remote Enable Line.

Table 10 Methods for Remote Operations

Method Name	Description
ControlRen	Controls the state of the Remote Enable Line as well as the local lockouts.
SendGoToLocal	Configures the instrument to assume local front panel control.
SendLocalLockout	Locks the front panel of the current instrument.

Low-Level Access through DirectIO.Serial

There are many properties and methods that simplify the low-level configuration and control of instruments accessed through the serial protocol. The connect string for a instrument with a serial connection is:

ASRL[*board*][::INSTR]

Items in square brackets, [], are optional and items in *italics* are supplied by you. All other entries are required.

Data Flow Control

There are many properties that support several different serial transmission flow control techniques.

Table 11 Properties for Flow Control

Property Name	Description
BaudRate	Gets or sets the rate at which data bits are transmitted and received.
BytesAvailable	Gets the number of bytes available in the global receive buffer.
CleartoSendState	Gets the current status of the clear to send line.
DataBits	Gets or sets the number of data bits sent in each data frame.
DataCarrierDetectState	Gets the current status of the data carrier detect line.
DataSetReadyState	Gets the current state of the data set ready line.
DataTerminalReadyState	Gets the current state of the data terminal ready line.
EndIn	Gets or sets the terminator for serial read operations.
EndOut	Gets or sets the terminator for serial write operations.

Table 11 Properties for Flow Control (continued)

Property Name	Description
FlowControl	Gets or sets the serial flow control technique.
Parity	Gets or sets the type of parity being used in the flow control technique.
ReplacementCharacter	Gets or sets a character used to replace a damaged incoming character.
RequestToSendState	Gets or sets the request to send output signal.
RingIndicatorState	Gets the current state of the ring indicator input signal.
StopBits	Gets or sets the number of stop bits used to indicate the end of a signal.
XoffCharacter	Gets or sets the XOFF character.
XonCharacter	Gets or sets the XON character.

There are also two methods that support flow control.

- **Flush** is a method used to clear the read and write buffers.
- **SetBufferSize** is a method used to define the read and write buffer size

Low-Level Access through DirectIO.TCPIP

Many Agilent instruments support network connectivity and, for those that do not, there are converter boxes now available that make this connectivity quick and easy. The connect string for an instrument with a TCPIP connection is:

TCPIP[*board*]::*host address*[*::LAN Device Name*]::INSTR

Items in square brackets, [], are optional and items in *italics* are supplied by you. All other entries are required.

Table 12 Tcpip Properties

Property Name	Description
DeviceName	Gets the LAN device name used during the connection.
HostName	Gets the hostname of the instrument.
IPAddress	Gets the TCPIP address of this instrument.

Low-Level Access through DirectIO.USB

Several Agilent instruments support USB connectivity and, for those that do not, there are converter boxes now available that make this connectivity quick and easy. The connect string for an instrument with a USB connection is:

USB[*board*]::*Manufacturer ID*::*Model Code*::*Instrument Serial Number*::*[USB Interface Number*]::INSTR

Items in square brackets, [], are optional and items in *italics* are supplied by you. All other entries are required.

Table 13 USB Properties

Property Name	Description
Is4882Compliant	Device supports 488.2 standard.
ManufacturerID	The identification number of the manufacturer.
ManufacturerName	The manufacturers name.
ModelCode	The model number for the device.
ModelName	The name of the model.
HardwareInterfaceNumber	The interface number for the device.
UsbInterfaceNumber	The USB interface number.
UsbSerialNumber	The serial number of the USB device.

Table 14 USB Methods

Method Name	Description
SendLocalLockOut	Disables the front panel controls of the device.
SendGoToLocal	Enables the front panel of the device and passes control to it.

Table 14 USB Methods

Method Name	Description
ControlRen	Controls the state of the GPIB REN interface line and, optionally, the remote / local state of the device.

8
Analyzing Instrument Data

This chapter describes the complex data type, the digital signal processing abilities, the statistical functions, mathematical tools, and regression techniques that are available in the Agilent.TMFramework, Agilent.TMFramework.DataAnalysis, and Agilent.TMFramework.DataAnalysis.DSP class libraries.

The Complex Data Type

Visual Studio .NET does not have a native complex data type, yet the complex data type is essential for your work. To meet this need, a complex data type is included with T&M Toolkit and implemented within .NET as an object. The T&M Toolkit includes complex mathematics, comparers, and a data type converter.

The Complex Class

The Complex class has many useful features. In the following tables, the properties, methods, and conversions are discussed.

The Complex Class Properties

Table 15 Properties of the Complex Class

Property Name	Description
Real	Gets or sets the real part of a complex number.
Imaginary	Gets or sets the imaginary part of a complex number.
Conjugate	Gets the Complex Conjugate of this complex number.
ImaginaryComparer	Create a new instance of ImaginaryComparer.
MagnitudeComparer	Create a new instance of MagnitudeComparer.
PhaseComparer	Create a new instance of PhaseComparer.

The Complex Class Methods

Table 16 Methods of the Complex Class

Method Name	Description
Add	Adds two complex values.
Divide	Divides two complex values.

Table 16 Methods of the Complex Class (continued)

Method Name	Description
Equals	Determines whether two instances of complex are equal.
FromPolarComplex	Creates a new complex number from a given phase and magnitude and expresses it in radians.
GetMagnitude	The magnitude, or absolute value, of this complex number.
GetPhase	The phase, or argument, of this complex number expressed in radians.
IComparable.CompareTo	Compares the current instance with another object of the same type.
IsInfinity	A value indicating if either the real or the imaginary part of the specified complex number evaluates to either negative or positive infinity.
IsNaN	A value indicating if either the real or the imaginary part of the specified complex number evaluates to Not-a-Number (NaN).
IsNegativeInfinity	A value indicating if either the real or the imaginary part of the specified complex number evaluates to negative infinity.
IsPositiveInfinity	A value indicating if either the real or the imaginary part of the specified complex number evaluates to positive infinity.
Multiply	Multiplies two specified complex values.
Negate	Negates the value of a specified complex number.
Parse	Converts the specified string representation of a complex number to its complex equivalent.
Subtract	Subtracts two complex values.
ToString	Converts the value of this instance to its equivalent string representation using the specified format and culture-specific format information.

The Complex Type Conversion

The T&M Toolkit Complex data type can only be converted to a string.

However, the real portion of the Complex data type can also be converted to a double but, when this is done, the imaginary portion is set to zero. Likewise, when converting a double to a complex number, the double is converted to the real portion and the imaginary number is set to zero.

Comparers

The complex class supports three comparers. These comparers are accessed through the **ImaginaryComparer**, **MagnitudeComparer**, and **PhaseComparer** properties.

The **ImaginaryComparer**, **MagnitudeComparer**, and **PhaseComparer** properties are access points to comparer methods for imaginary numbers. In the following code snippet, the **ImaginaryComparer** property is used to access the **ImaginaryComparer** subclass.

```
Module Module1
   Sub Main()
      Dim d As Object
      Dim result As New _
         Agilent.TMFramework.ImaginaryComparer()
      Dim a As Complex = New Complex(2, 2)
      Dim b As Complex = New Complex(3, 1)
      d = result.Compare(a, b)
      Console.WriteLine(d)
   End Sub
End Module
```

These subclasses can also be used with the **Array.Sort** method to specify array sorting based on imaginary values.

The Imaginary Comparer Subclass

Compares two complex numbers and returns a value indicating whether one complex is less than, equal to, or greater than the second complex.

The Magnitude Comparer Subclass

Compares two complex numbers and returns a value indicating whether the magnitude of one complex is less than, equal to, or greater than the magnitude of the second complex.

The Phase Comparer Subclass

Compares two complex numbers and returns a value indicating whether the phase of one complex is less than, equal to, or greater than the phase of the second complex.

Digital Signal Processing Capabilities

The Agilent T&M Toolkit includes a strong set of digital signal processing (DSP) methods. FFT and IFFT using double and complex numbers are supported. There are also five data filters Bartlett, Blackman, Hamming, Hanning, Rectangular, and two other DSP processing functions Convolution and CrossCorrelation.

Fast Fourier and Inverse Fast Fourier Transforms

The FFT and the IFFT methods accept an array of complex numbers or doubles. Additionally, FFT provides parameters for the input of a time interval and the return of a start frequency and a stop frequency. IFFT provides parameters for the input of frequency interval and the return of a start time and a stop time. These extra parameters make it very easy to graph the resulting data set using the T&M Toolkit's 2D Graphing Objects.

NOTE The execution time for FFT and IFFT depends upon the length of the transform. It is fastest for powers of two. It is almost as fast for lengths that have only small prime factors. It is typically several times slower for lengths that are prime or which have large prime factors.

Windowing Functions

Five of the data filters accept the same parameters. The Bartlett, Blackman, Hamming, Hanning, and Rectangular windows all accept arrays of doubles and arrays of complex numbers.

For each of these filters, you can set the window to narrow, wide, or fixed. A narrow window is useful for narrow band measurements. When the narrow window is used, the peak value at a given on-bin frequency is preserved. For wide band measurements, use the wide window. When the wide window is used, the energy within a given frequency band is preserved. Finally, when looking at windowed time domain data and when windowing the coefficients of a FIR filter, the fixed window should be used. This scaling preserves the peak value of the input data.

When submitting data sets to these windows, you can also submit a portion of an array by specifying how many array elements to use. The starting point for these subsets is always the first element of the array.

Data Set Comparison Functions

There are two methods for data set comparison. The Convolve method filters a data array by multiplying it with another data array. While the two data sets can be of different lengths, they must contain equally spaced data. The CrossCorrelate method also filters a data array by multiplying it against another but uses a different process for determining which values to multiply together.

Bessel Functions

The Agilent T&M Toolkit has a modified Bessel function, a Bessel function of the first kind, and a Bessel function of the second kind. These Digital Signal Processing filters are used to smooth waveform data.

The I0 Bessel Function

The I0 Bessel function is a modified version of the Bessel function of the first kind. It accepts values less than 700 or greater than negative 700. Values outside this range return an exception.

The J0 and J1 Bessel Functions

The J0 Bessel function computes the Bessel function of the first kind of order zero (J0) of the vector array. The J1 Bessel function computes the Bessel function of the first kind of order one (J1) of the vector array.

The Y0 and Y1 Bessel Functions

For both the Y0 and the Y1 Bessel functions, inputs that are less than or equal to 0 return a zero. The Y0 Bessel function computes the Bessel function of the first kind of order zero (Y0) of the vector array. The Y1 Bessel function computes the Bessel function of the first kind of order one (Y1) of the vector array.

Statistical Functions

The Agilent T&M Toolkit has three groups of statistical functions available for your use: Discrete Mathematics, Probability, and Special Functions.

Discrete Mathematics

The discrete mathematical functions included with T&M Toolkit are contained in the following table.

Table 17 Discrete Mathematics

Method Name	Description
Beta	Calculates the beta of *x*.
Binomial	Calculates the number of combinations of *n* taken *b* at a time.
Combination	Calculates how many combinations of *n* things taken *r* at a time.
Factorial	Calculates the number of ways that *n* objects can be permuted.
Gamma	Calculates the gamma function for doubles.
Permutation	Calculates the permutations of *n* numbers taken *r* at a time.

Probability

The probability functions included with T&M Toolkit are contained in the following table.

Table 18 Probability

Method Name	Description
RMS (Root Mean Squared)	A standard way of calculating the effective voltage or current of an AC wave.
Standard Deviation	Calculates how far a data point is from the mean value of the data set.

Table 18 Probability (continued)

Method Name	**Description**
Variance	Calculates the spread of a distribution.

Special Statistical Functions

The special statistical functions included with T&M Toolkit are contained in the following table.

Table 19 Special Statistical Functions

Method Name	**Description**
Mean	Calculates the average value for a set of data.
Mode	Returns the most commonly found value within the data set.
Median	Returns the middle value of a data set.

Engineering Math

Your engineering math needs are easily met by the EngMath class. This class is a super-set of System.Math and is found in the Agilent.TMFramework class library. In the following table, the methods are described. Many of these methods also have overloads that accept the complex data type.

Table 20 Engineering Math Methods

Method Names	Description
Abs	Returns the absolute value of a number.
Acos	Returns the angle whose cosine is the specified number.
Acosh	Returns the hyperbolic arccosine of *c*.
Acot	Returns the angle whose cotangent is *x*.
Acoth	Returns the hyperbolic arctangent of *d*.
Asin	Returns the angle whose sine is the specified number.
Asinh	Returns the hyperbolic arcsine of *d*.
Atan	Returns the angle whose tangent is the specified number.
Atan2	Returns the angle that corresponds to the inverse tangent of the quotient of two specified numbers.
Atanh	Returns the hyperbolic arctangent of *d*.
Ceiling	Returns the smallest whole number greater than or equal to the specified number.
Conjugate	Returns the conjugate of a number.
Cos	Returns the cosine of the number.
Cosh	Returns the hyperbolic cosine of the specified angle.
Cot	Returns the cotangent of *d*.
Coth	Returns the hyperbolic cotangent of *d*.
Exp	Returns the natural antilogarithm of a number.

Table 20 Engineering Math Methods (continued)

Method Names	Description
Floor	Returns the largest whole number that is less than or equal to the number passed to the method.
IEEERemainder	Returns the remainder resulting from the division of a specified number by another specified number. Note, see the Microsoft documentation for more detail.
Imaginary	Computes the imaginary portion of a complex number input in phase and magnitude.
Log	Returns the natural logarithm of a number.
Log10	Returns the common logarithm of a number.
Magnitude	Returns the Magnitude of a complex number.
Max	Returns the larger of the two parameters.
Min	Returns the smaller of the two parameters.
Phase	Returns the phase of a complex number in radians.
Pow	A number raised to a power.
Real	Computes the real portion of a complex number that has been input as phase and magnitude.
Round	Returns a number nearest the number passed in the parameter.
Sign	For a number X, return (X / ABS(X))
Sin	Returns the sine of the number.
Sinh	Returns the hyperbolic sine of the specified angle.
Sqrt	The square root of a specified number.
Tan	Returns the tangent of the number.
Tanh	Returns the hyperbolic tangent of the specified angle.
ToDegrees	Converts radians to degrees.
ToRadians	Converts degrees to radians.

Regression Techniques

Agilent T&M Toolkit supports four regression methods: Linear, Logarithmic, Exponential, and Power. The Logarithmic, Exponential, and Power regression methods throw exceptions if they are passed values less than or equal to zero.

The regression methods accept the following data sets:

- A one-dimensional array of Y values, the X start value, and the X interval.
- A one-dimensional array of XY values where X is always the first value and the pattern is XYXYXY.
- A one-dimensional array of X values and a one-dimensional array of Y values.
- A two-dimensional array of XY values where the array can be dimensioned [n,2].

The Linear regression method produces both coefficients, the goodness of fit, and Y fitted data. All of the other regression methods produce both coefficients, the goodness of fit, Y fitted data, and X fitted data.

9
Using the 2D Graph Objects

This chapter provides an overview of the available graphs, of how to get data to the graph, and categories of properties that are used to configure a graph's appearance. All of these features are found in the Agilent.TMFramework.DataVisualization class library.

Using the 2D Graph Objects

Where are the 2D Graph Controls?

From the Windows Forms Designer, open the Toolbox by selecting **View** > **Toolbox** or enter ALT-CTL-X. The Toolbox appears on the left side of your display. You should have already created a T&M Toolkit Tab and added the Agilent components to it. However, if you have not, review the “How Do I Add the T&M Toolkit Tab to the Toolbox?” topic in the **Frequently Asked Questions** book of the online Help.

The T&M Toolkit tab looks like:

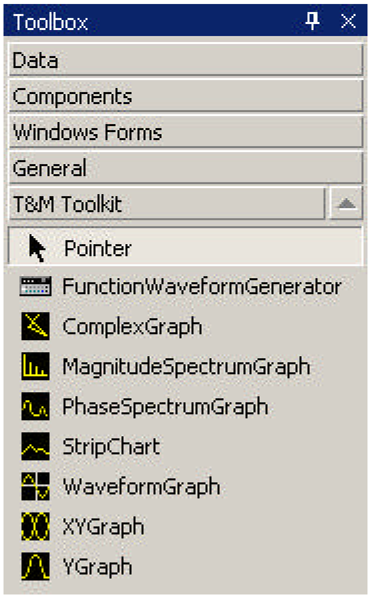


Figure 36 The Agilent T&M Toolkit Toolbox Tab

To use a graph control such as the Complex Graph, just drag and drop it onto the upper left corner of your form.

What Types of Graphs are Available?

There are seven types of graphs.

XY Graph

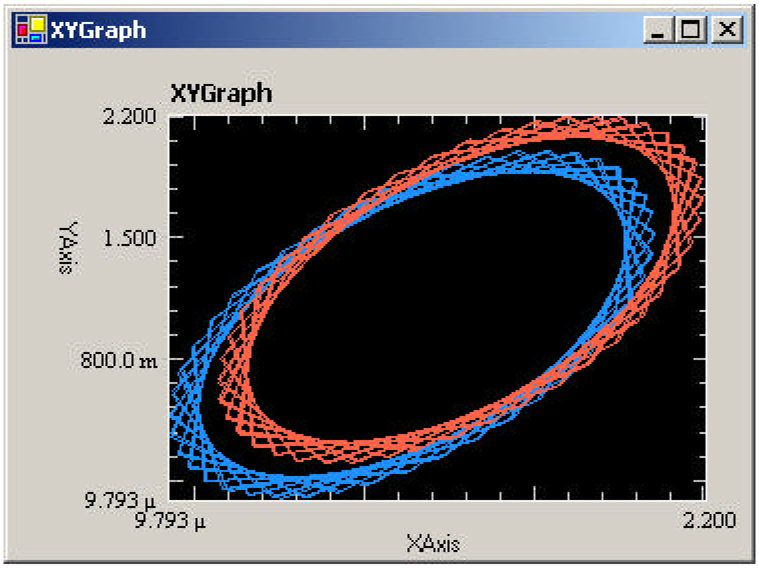


Figure 37 XY Graph

The XY Graph displays traces fed by dual-channel data (X and Y data) which can be arbitrarily or functionally related to each other. In Figure37, there are two traces. Each trace consists of a one-dimensional array of X,Y pairs. The XY Graph also accepts two-dimensional arrays.

NOTE Whenever X,Y data pairs are loaded into an array of any size, the 2D Graph Object always assumes the first value read is the X value of the X,Y pair.

Waveform Graph

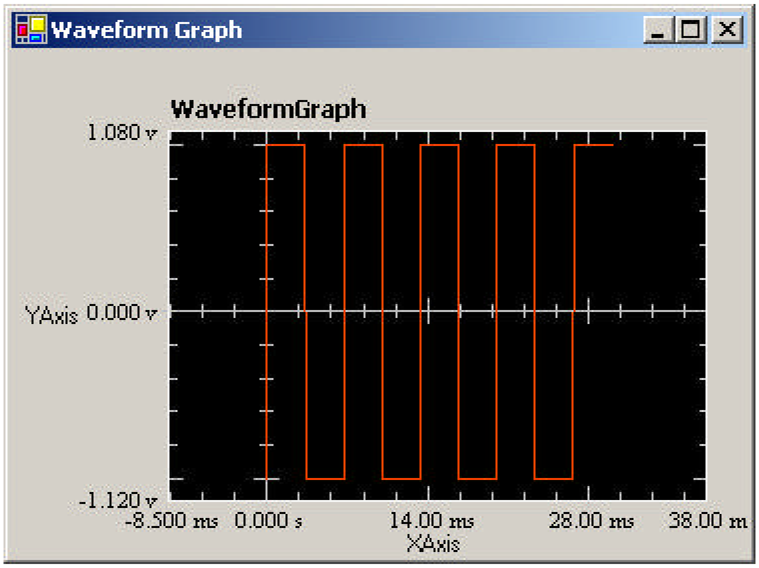


Figure 38 Waveform Graph

The Waveform Graph displays time domain data given Y values and a start time/stop time for the Y values.

There is a tutorial in online help entitled “Using the 2D Graph Objects”. The tutorial uses the Waveform Graph to describe traces. Every graph has at least one trace, the DefaultTrace, as well as others that you may add.

Traces, such as the one displayed in Figure38, visually represent your data. They are the fundamental building block to graphing with the T&M Toolkit. Traces are discussed in more detail later in this chapter.

Y Graph

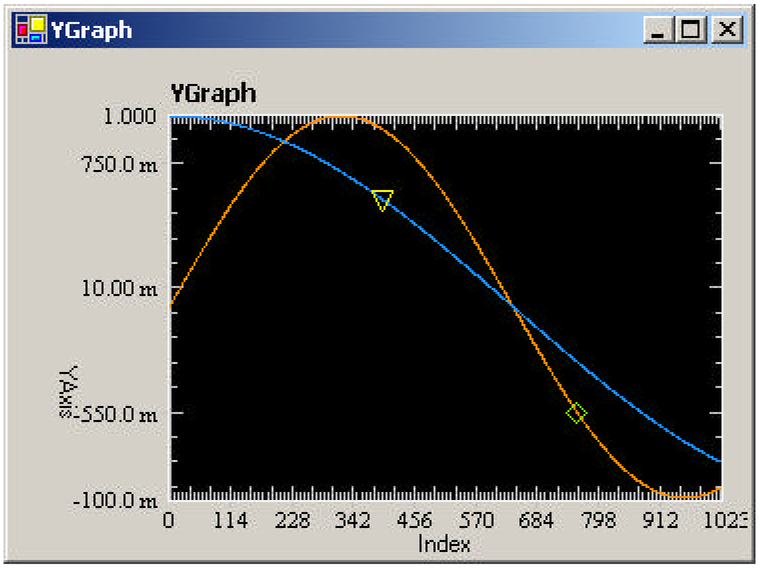


Figure 39 Y Graph

The Y graph displays traces fed by a single-channel Y data source, where X data is the index for the data point.

The Y graph in Figure 39 shows two different traces and two different markers. The markers appear as a diamond shape near the end of the trace with the shorter cycle time and an upside-down triangle about one-third of the way along the other trace. Markers move from data point to data point along the trace.

StripChart

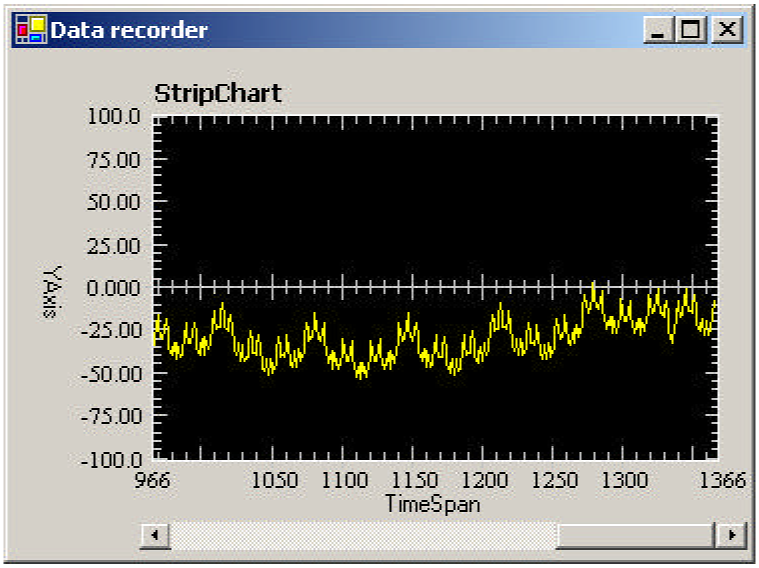


Figure 40 Strip Chart

A Strip Chart displays live, continuous traces fed by single-channel Y data with TimeSpan as the X axis. The data is accumulated in a user-configurable internal buffer.

NOTE

The recommended way to manage the Strip Chart time interval is to connect it directly to the system timer. In your source code, this looks like `StripChart1.TimeInterval = System.Windows.Forms.Timer1.Interval.` You then manipulate the Timer properties to control the StripChart time interval.

Complex Graph

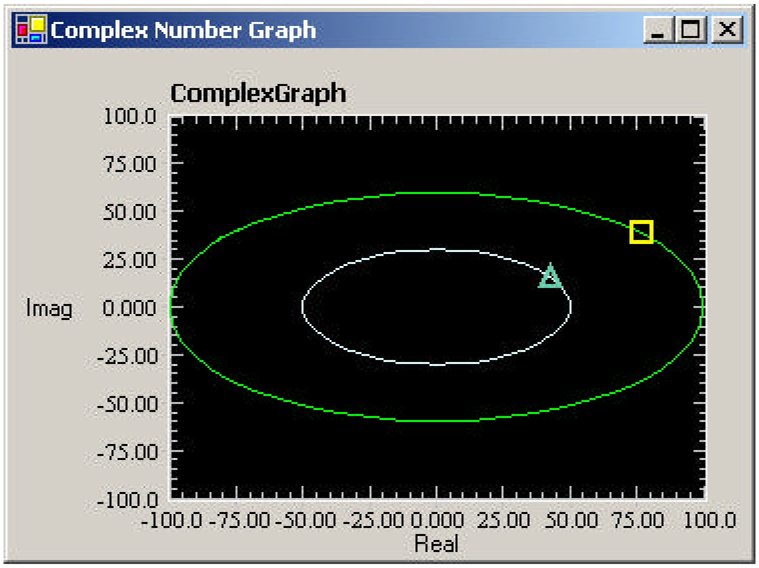


Figure 41 Complex Graph

A Complex Graph is a special-purpose graph for complex numbers where the real portion of the complex number is the X axis, and the imaginary portion of the complex number is the Y axis.

The Complex Graph in Figure 41 has two markers. These markers are linked so that when the marker on the outside trace is moved, the marker on the inner trace moves in unison with it. There are no limits to the number of linked markers on a graph, and you can link one marker to an infinite number of other markers. However, the links are always unidirectional. For example, if you move the marker on the inner trace in Figure41, there is no corresponding movement from the marker on the outer trace.

Magnitude Spectrum Graph

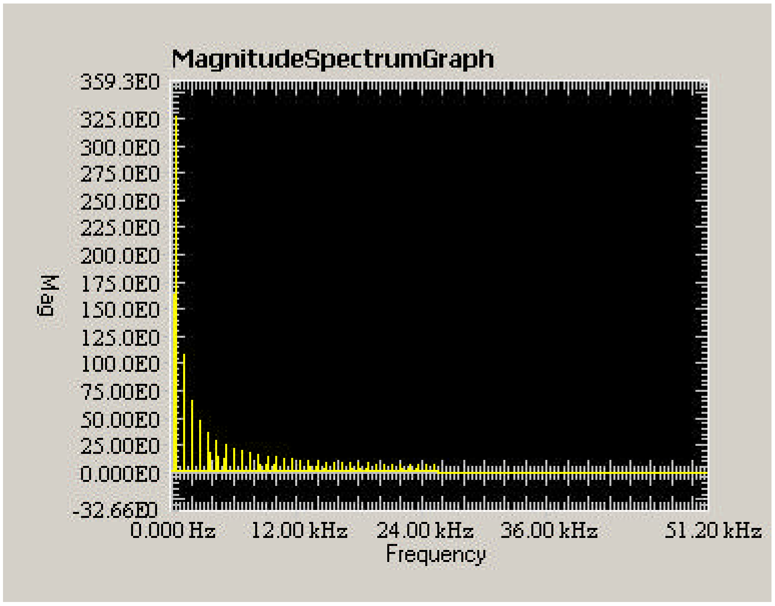


Figure 42 Magnitude Spectrum Graph

A Magnitude Spectrum Graph displays one-dimensional, complex data of a frequency domain. When the data is prepared, a method in the Agilent T&M Framework is used to capture the real portion (the magnitude) of the complex number. The magnitude is then displayed on the Y axis and frequency is displayed on the X axis.

Phase Spectrum Graph

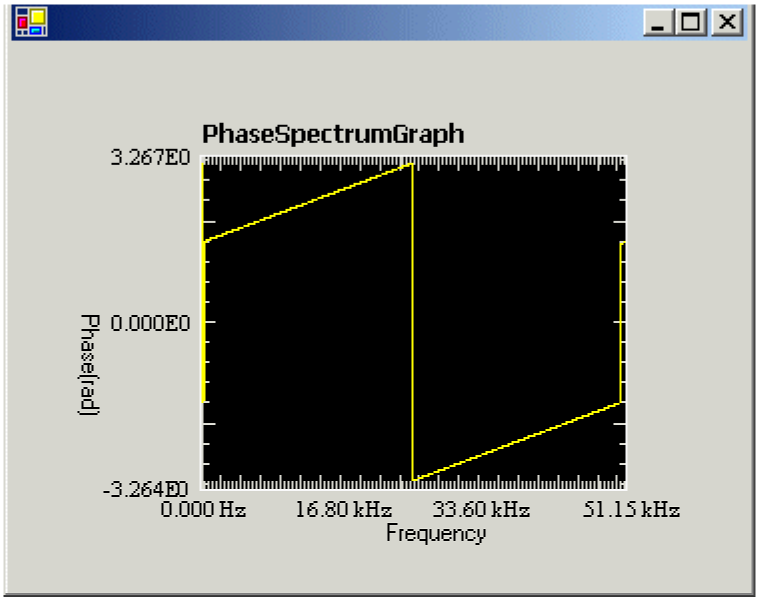


Figure 43 Phase Spectrum Graph

A Phase Spectrum Graph displays one-dimensional, complex data based on a phase angle. When the data is prepared, a method in the Agilent T&M Framework is used to capture the imaginary portion (the phase) of the complex number. The phase is then displayed on the Y axis and frequency is displayed on the X axis.

Using Traces to Get Data to the Graph

As has been mentioned, traces are the fundamental building block for the 2D Graph Object. Every graph has at least one trace, called the DefaultTrace, and usually several others which you have added.

The trace holds the data points you have generated. This could be a continuous stream of data plotted on the Y axis with time on the X axis (a Strip Chart), a two-dimensional array of X,Y pairs (an XY Graph), or a large array of complex numbers (a Complex Graph). In each of these cases, traces function in the same way and with the same descriptive properties.

The Default Trace

Every graph object in the 2D Graph Object starts with one DefaultTrace. You can delete the DefaultTrace, but, should you also delete all other traces, a DefaultTrace is automatically re-created.

The DefaultTrace is designed for situations where you want a quick, single trace plot. The **PlotDefaultTrace()** method supports the use of the DefaultTrace for single trace plots. The **PlotDefaultTrace()** method loads and plots the DefaultTrace in a single step. The other plotting method is designed for multiple traces and requires a separate step to load data into the trace.

Defining New Traces and Markers

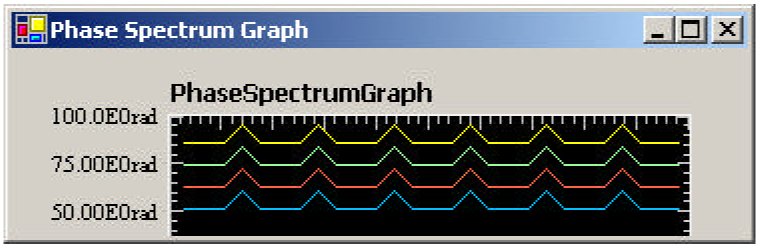
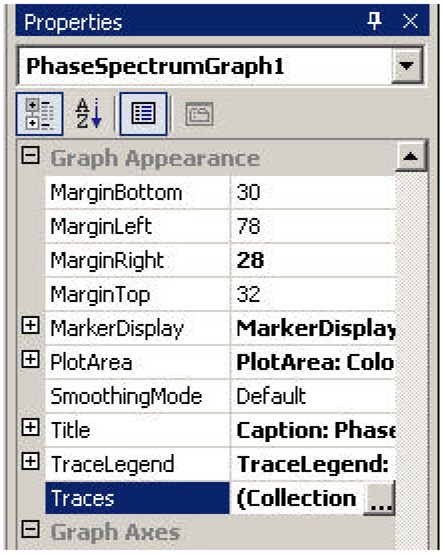


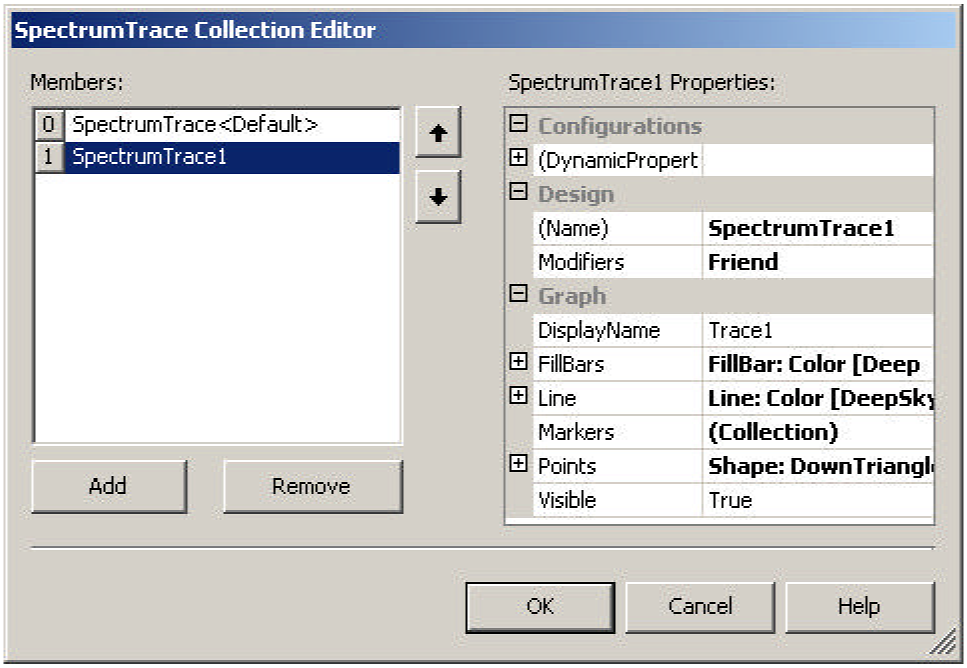
Figure 44 Trace Representations

To create traces, you need to access the Trace Collection. See Figure45 for an example of how the Graph Appearance category looks. Since Traces is the default property for all of the 2D Graph Objects, you can also just click the graph and click the ellipsis to go to the Trace Collection Editor.



MarginBottom	30
MarginLeft	78
MarginRight	28
MarginTop	32
MarkerDisplay	MarkerDisplay
PlotArea	PlotArea: Colo
SmoothingMode	Default
Title	Caption: Phase
TraceLegend	TraceLegend:
Traces	(Collection ...

Figure 45 Locating the Traces Collection in the Property Listing



0	SpectrumTrace<Default>
1	SpectrumTrace1

SpectrumTrace1 Properties:

⊟ Configurations	
⊞ (DynamicPropert	
⊟ Design	
(Name)	**SpectrumTrace1**
Modifiers	**Friend**
⊟ Graph	
DisplayName	Trace1
⊞ FillBars	**FillBar: Color [Deep**
⊞ Line	**Line: Color [DeepSky**
Markers	**(Collection)**
⊞ Points	**Shape: DownTriangl**
Visible	True

Figure 46 The Trace Collection Editor for the SpectrumTrace

All traces are listed in the left column and, by choosing one of the traces, you can set the properties for it from the property listing in the right column. As you change the properties, your changes are reflected on the trace representation as seen in Figure44.

Notice in Figure 46 that the Marker Collection is accessible from the Trace Collection Editor. To create markers or change their properties do the following:

Accessing the Markers Collection Editor

Step	Description
1	Launch the Trace Collection Editor

Accessing the Markers Collection Editor (continued)

Step	Description
2	Select the trace that contains the marker you want to edit
3	Select the Marker Collection
4	Select and edit the marker

Markers, however, do not appear on the trace representations seen in Figure44.

Setting and Plotting Data

You have created all the traces and markers you need for your graph. The properties are set so that the appearance and behavior of your graph is going to be just what you want. All that remains is to generate the data and then programmatically connect your graph with the data set. To accomplish this within the 2D Graph Object Set, there are just three simple methods you need to know.

- The **PlotDefaultTrace Method** is *only* used for the DefaultTrace. It loads data from a source you have provided and plots it directly to the DefaultTrace. One data source connected to one trace gives you a quick, simple graph.
- Whenever you have multiple data sets, which means multiple traces, the **SetData Method** is used to load data into each trace. The **SetData Method** is always used in conjunction with the **PlotAll Method**.
- The **PlotAll Method** is always used as the final step in a series of **SetData Methods.** It plots all of the traces that have been loaded, supports multiple data types, one-dimensional arrays, and a two-dimensional array for the XY graph.

How can I Modify the Appearance of the Graph?

The appearance of your graph is modified by the extensive property set within the Microsoft .NET Form Designer and the features of the 2D Graph Objects provided by the Agilent T&M Toolkit.

Agilent T&M Toolkit 2D Graph Property Categories

The 2D Graph Objects are made available through the seven graph objects previously mentioned and the three graph property categories used to control their appearance. A description of these three categories follows:

- The **Graph Appearance Category**
    - The **Bottom**, **Left**, **Right**, and **Top Margin** property settings control the distance, in pixels, between the plot area and edge of the graph control. The plot area is the area where your traces appear.
    - The **Marker Display** is a legend for the markers. Using this group of properties, you can define how the Marker Display appears.

NOTE The Marker Display not only presents an example of each marker’s appearance, it also captures the relationships between markers that are linked.

-
    - The **Plot Area** is the area where your traces appear. You can control its appearance with this property set.
    - The **Smoothing Mode** property specifies the rendering quality for the graphic.
    - The main **Title** for the graph control. You can control its appearance with this property set.
    - The **Trace Legend** displays the trace’s name, color, whether fill bars are visible, and whether the data points are visible.
- The **Graph Axes Category**

- The **Graticule** group of properties includes the creation and configuration of a grid for your graph. You can also set a property called **TickStyle**. **TickStyle** is used to define which of the plot area boundaries have tick marks.
- The **XAxis** and **YAxis** groups of properties are identical in nature. You can control the appearance of the axis, major ticks, and minor ticks; or the caption text and appearance. You can also control the scaling by adjusting the minimum and maximum values; and several other items such as **CoordStyle(Linear,Log)**.

NOTE There is an item of particular interest on the **XAxis** and **YAxis** property subsets. The **AutomaticScaling** property defaults to automatically scaling both axes (except in the Strip Chart) and is used for design-time work. For run time, the **AutoScaling** method is strongly recommended. The **AutoScaling** method supports the **UndoZoom** feature of the context menu whereas the **AutomaticScaling** property does not.

- The **Graph Behavior Category**
  - The 2D Graph Objects have a very useful context menu that is activated by placing the cursor in the plot area and right clicking. This feature is discussed in more detail in “The Context Menu" on page110.
  - The **KeyboardEnabled** property activates several handy shortcut keys for zooming and panning. This feature is discussed in more detail in “Keyboard Shortcuts " on page 108.
  - The **MouseEnabled** property activates the mouse wheel. The mouse wheel, used in conjunction with the keyboard, can provide the full range of zooming and panning. This feature is discussed in more detail in “Mouse Shortcuts" on page108.

Panning, Zooming, and the Context Menu

Once your graph is drawn, you may want to zoom-in on specific areas and pan from location to location. There are several ways to do this with the 2D Graph Objects. You can use your keyboard, mouse, or a context menu, whichever is the most convenient for you.

Keyboard Shortcuts

To use keyboard shortcuts, hold the **CTRL** or **ALT** key down (depending upon whether you are panning or zooming) and then hold down the second key until you have achieved the desired result. See the following table for key combinations and their corresponding action.

Table 21 Keyboard Shortcuts for Panning and Zooming

Key Combination	Action
`CTRL-RightArrow`	Pans right
`CTRL-LeftArrow`	Pans left
`CTRL-UpArrow`	Pans upward
`CTRL-DownArrow`	Pans downward
`ALT-RightArrow`	Zooms in on the X axis
`ALT-LeftArrow`	Zooms out on the X axis
`ALT-UpArrow`	Zooms in on the Y axis
`ALT-DownArrow`	Zooms out on the Y axis

Mouse Shortcuts

When using the mouse shortcuts, the action is not completed until after you release the buttons.

Table 22 Mouse Shortcuts for Panning and Zooming

Key / Mouse Combination	Action
Hold down the left mouse button and draw a rectangle inside the plot area.	Zooms the area within the rectangle. (Press the ESC key before releasing the mouse's left button to cancel the zoom.)
Hold down the CTRL key and the left mouse button simultaneously. Drag the mouse.	Pans the graph display in the direction you drag the mouse.
Hold down the X key and move the mouse wheel forward or backward.	Zooms the X axis in (mouse wheel forward) or out (mouse wheel backward).
Hold down the Y key and move the mouse wheel forward or backward.	Zooms the Y axis in (mouse wheel forward) or out (mouse wheel backward).
Hold down the Z key and move the mouse wheel forward or backward.	Zooms the X and Y axis in (mouse wheel forward) or out (mouse wheel backward).
Hold down the H key and move the mouse wheel forward or backward.	Pans the display to the left (mouse wheel forward) or to the right (mouse wheel backward).
Hold down the V key and move the mouse wheel forward and backward.	Pans the display up (mouse wheel forward) or down (mouse wheel backward).
Hold down the E key and move the mouse wheel forward and backward.	Pans the display towards the upper right corner (mouse wheel forward) or the lower right corner (mouse wheel backward).
Hold down the W key and move the mouse wheel forward and backward.	Pans the display to the upper left corner (mouse wheel forward) or to the lower left corner (mouse wheel backward)

The Context Menu

There is also a context menu available that provides quick access to several useful features. To access this menu, place your mouse pointer in the plot area and right click.

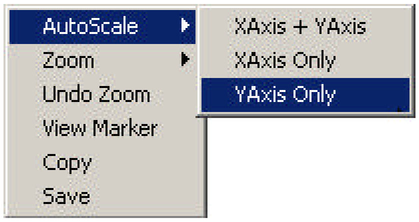


Figure 47 Context Menu - AutoScale

- If the current graph has already been AutoScaled, the AutoScale menu option is greyed out. If you choose AutoScale, a sub-menu containing entries for each axis or both axes appears. AutoScale tells the graph object to automatically scale the chosen axis(es) to match the minimum and maximum values of the traces.
- Choosing Zoom also initiates a sub-menu containing entries for the axis or axes and an additional sub-menu that prompts for a zoom factor.

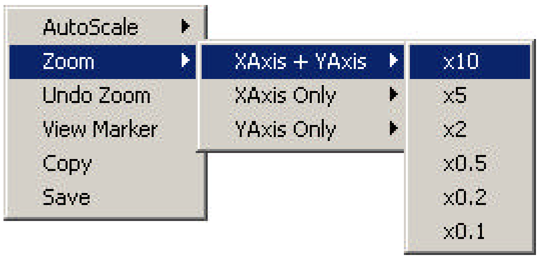


Figure 48 Context Menu - Zoom

- UndoZoom is disabled until you have done a Zoom operation. At that point, choosing UndoZoom reverses the zoom process by returning to the previous zoom point.
- If a marker is not defined for a trace on the graph object, the ViewMarker menu choice is disabled. If you have a marker defined, choosing ViewMarker brings a marker back into the display area. For example, suppose you had a trace with a marker defined for it. You zoomed the trace several times and the marker was now on a part of the trace that was no longer in the display area. By choosing this menu option, the marker would be moved to the edge of the display area.
- Choosing the Copy option places a copy of the graph control (not the Forms Designer form) onto the Windows Clipboard.
- Choosing Save initiates a File Save dialog box. You can save a copy of the graph control (not the Forms Designer form) in a .jpg, .bmp, or .gif format.

10 Virtual Waveforms, Timing Classes, Number Formatting, and Engineering Math

This chapter describes a component and several additional classes that are very useful to engineers. All of these classes are found in the Agilent.TMFramework class library.

Generating Waveforms

Have you ever been in a situation where you just need to create a simple sine wave to test some source code you have written, but you don’t want to start up the instruments? The T&M Toolkit’s FunctionWaveformGenerator is made-to-order.

The FunctionWaveformGenerator creates data for six different waveform functions: sine, cosine, triangle, square, positive ramp, and negative ramp. The properties for each of these functions are fully configurable by you, so you can control the characteristics of the waveform and match it directly to your needs.

Of course, it is nice to have a tool that creates fully configurable data sets that can be used to simulate different waveforms, but the FunctionWaveformGenerator has a great deal more convenience to offer. You can drag and drop it right onto your Windows Form, adjust its properties at design time, and run your program. The FunctionWaveformGenerator is only visible at design time.

In Figure49 you see a YGraph with a FunctionWaveformGenerator that has been dragged from the T&M Toolkit tab of the Forms Designer Toolbox and dropped onto the form. No programming has been done yet.

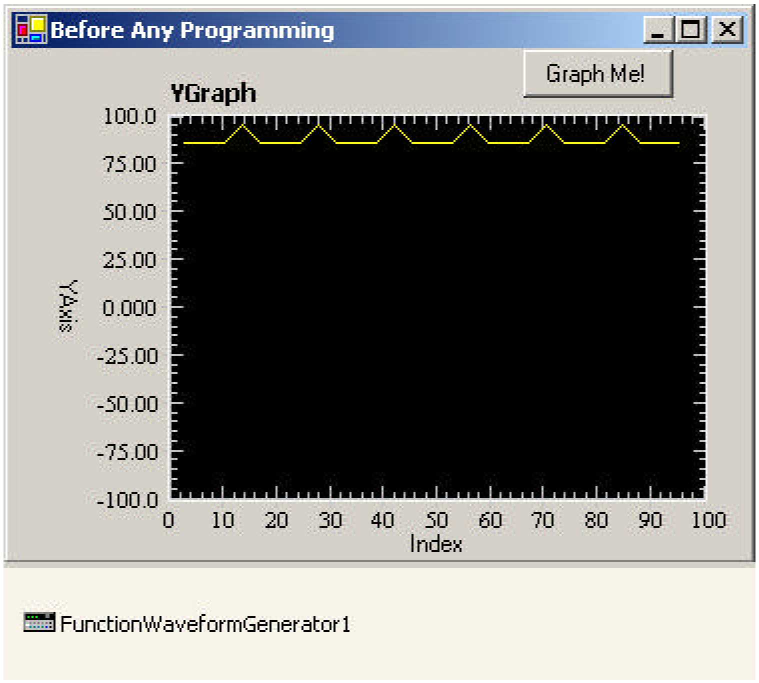
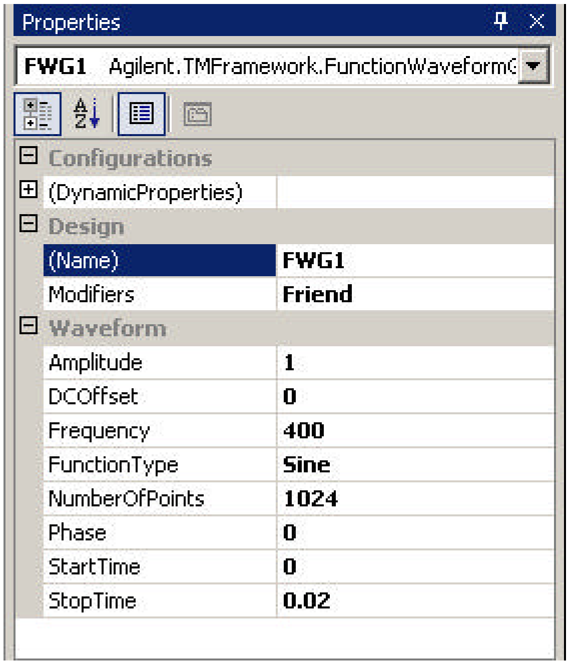


Figure 49 A YGraph with FunctionWaveformGenerator Component

In Figure50 you see the properties for the FunctionWaveformGenerator. Notice that the name has been changed to FWG1 to simplify typing later. At this point, you are still in the Windows Forms Designer and, due to the unique features of the FunctionWaveformGenerator, you are able to control the data for your graph as well as its appearance.



Configurations	
(DynamicProperties)	
Design	
(Name)	FWG1
Modifiers	Friend
Waveform	
Amplitude	1
DCOffset	0
Frequency	400
FunctionType	Sine
NumberOfPoints	1024
Phase	0
StartTime	0
StopTime	0.02

Figure 50 Properties for the FunctionWaveformGenerator

The graph is ready, but it does require some programming to take advantage of the FunctionWaveformGenerator and create a graph of the sine wave. Switch to the code window by double-clicking on the “Graph Me!” button. You should find yourself in the Button1_Click event. Enter the following line of code:

```
YGraph1.PlotDefaultTrace(FWG1.GenerateYData)
```

Run the program and click **Graph Me!** to produce the graph shown in Figure 51.

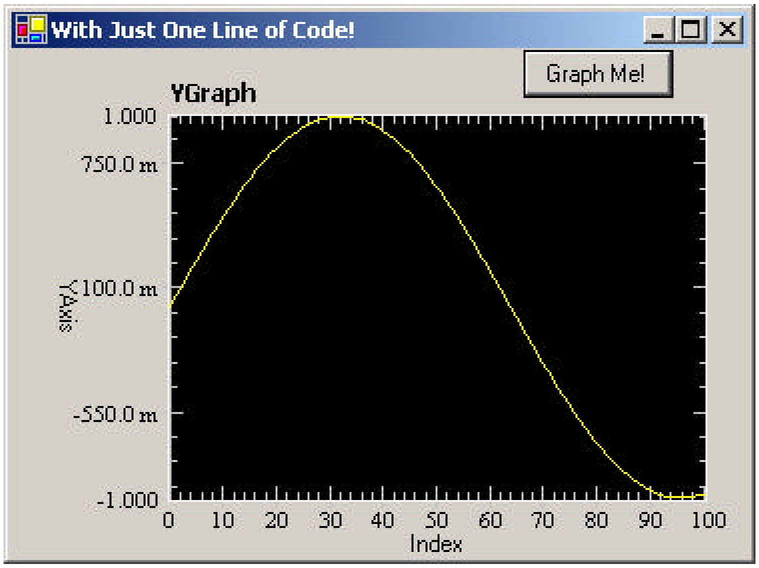


Figure 51 Graphed Sine Wave Using Data from FWG1

Properties

Table 23 FunctionWaveformGenerator Properties

Property Name	Description
Amplitude	Gets or sets the amplitude of the generated waveform.
DCOffset	Gets or sets the DC offset of the waveform.
Frequency	Gets or sets the frequency of the generated waveform.
FunctionType	Gets the wave type. Sets the wave type to **Sine**, **Cosine**, **Square**, **Triangle**, **PositiveRamp**, or **NegativeRamp**.

Table 23 FunctionWaveformGenerator Properties

Property Name	Description
NumberOfPoints	Gets or sets the number of points in the generated waveform.
Phase	Gets or sets the phase (in degrees) of the generated waveform.
StartTime	Gets or sets the start time of the waveform.
StopTime	Gets or sets the stop time of the waveform.

Methods

Table 24 FunctionWaveformGenerator Methods

Method Name	Description
GenerateXData	Generates the X axis values.
GenerateYData	Generates the Y axis values.

Timing Classes

The Timing Class

Most computers have high-resolution counters. These counters are very useful when you need precise timing measurements. To test if your computer supports this type of counter, call the **HiResCounterSupported** method.

The Timing class provides two sets of functionality designed to take full advantage of a high-resolution counter. First, the **CounterValue** property and the **CalculateElapsedSeconds** method allow you to make precise timing measurements. To determine the amount of timing resolution available from your computer's high-resolution performance counter, use the **CounterResolution** property.

The following C# example demonstrates this usage.

```
using System;
using Agilent.TMFramework;
using Agilent.TMFramework.InstrumentIO;

public class App {
   public static void Main() {
     DirectIO dio = new DirectIO("GPIB0::7");
     double elapsed;
     dio.WriteLine("*IDN?");
     long start = Timing.CounterValue;
     string response = dio.Read();
     long stop = Timing.CounterValue;
     elapsed = Timing.CalculateElapsedSeconds(start, stop);
     Console.WriteLine("Elapsed time was: {0:E4}", elapsed);
   }
}
```

Secondly, you can use the **Delay** method from this class to provide delays, in the microsecond range, for gating I/O with an instrument.

NOTE

The **Delay** method guarantees the requested amount of time passes before it returns. However, due to the preemptive, multitasking nature of the Windows operating system, there is no guarantee the delay is not longer.

ProgressUpdater

The **ProgressUpdater** class is quite useful in at least two instances. If you are programmatically building a driver wrapper with T&M Toolkit’s Wrapper Generator and want notification of the progress, the **ProgressUpdater** properties can do this for you. You can also use this class for any program you are developing where you want to monitor the progress of an event.

The **ProgressUpdater** class has two properties:

- **Message** is a property that gets the message associated with this event.
- **PercentComplete** is a property that gets the percent complete value for this event.

The EngineeringFormatter Class

The **EngineeringFormatter** class addresses the number formatting needs of engineers. It converts any number into engineering format and, for numbers with exponent values between E+024 and E-024 (inclusive), it can substitute the abbreviated or full SI prefix for the exponent.

Any formatting method you use that takes an **IFormatProvider** as a parameter can also use the **EngineeringFormatter** to format numbers. For example, the **System.String.Format** method can accept a **FormatProvider** (an object that implements **IFormatProvider**). This method is used as follows for C#:

```
string result =
String.Format(EngineeringFormatter.Default,
"{0:M6}", 123456);
```

The result of this operation is "123.456E+003". If you had used the standard .NET number format provider with the format code of "E5", the result would have been "1.23456E+005". Note how the **EngineerFormatter** adjusts the position of the decimal point so the exponent is always a multiple of three. This is true even when you display the exponent value as an SI prefix instead of a number.

The general format for an engineering format specifier is: <formatCharacters><significantDigits>. The significant digits portion of the format specifier is optional. If the significant digits are not specified, a default of four significant digits is used.

CAUTION When specifying the number of significant digits, use at least 3 in your format specifier. For example, the code snippet `String.Format("{0:mt5}", 123456)` specifies five significant digits and produces the string "123.46e3". If only one significant digit is specified using `"{0:mt1}"`, the result is "100e3", which is not very informative.

The following table describes the different format specifiers used to apply different engineering formats. Any format specifier that is not recognized is passed through to the default .NET **FormatProvider**. The description of each of the following format codes shows an example of the format based on the number 123456.

Table 25 Format Specifiers and Their Action

Format Specifier	Description
M or m	Engineering Multiple Mode. Exponent value is always a multiple of three. Exponent character is displayed in either upper case (E) or lower case (e) depending on the case of the format character. **M** displays "123.5E+003". **m** displays "123.5e+003".
Mt or mt	Engineering Multiple Terse Mode. Same as **M** and **m** except that extraneous exponent characters are removed to make the display as terse as possible. Leading zeros in the exponent are removed as well as the "+" character. In the case of a negative exponent, the "-" character is displayed. **Mt** displays "123.5E3". **m** displays "123.5e3". If the number were 0.123456, then **M** would yield "123.5E-3". In all cases, the t must be in lower case.
A	SI Abbreviation Mode. Exponents between E+024 and E-024 are replaced with the appropriate SI prefix abbreviation. Exponent values that fall outside of this range are not changed since there are no SI prefixes for these values. **A** displays "123.5 k".
At	SI Abbreviation Terse Mode. Same as **A** except that the space between the number and the SI prefix abbreviation is removed. **At** displays "123.5k". In all cases, the **t** must be in lower case.
S	SI Prefix Mode. Same as **A** except that the full SI prefix name is used instead of the abbreviation. **S** displays “123.5 kilo".
St	SI Prefix Terse Mode. Same as **S** except that the space between the number and the SI prefix is removed. **St** displays "123.5kilo". In all cases, the **t** must be in lower case.

A Visual Basic .NET sample for each formatting code appears in the following list.

```
Dim formatProvider as IFormatProvider = EngineeringFormatter.Default
Console.WriteLine(String.Format(formatProvider, "Case 1:{0:M}", 3.123456E-001))
Console.WriteLine(String.Format(formatProvider, "Case 2:{0:m3}",  3.123456E+001))
Console.WriteLine(String.Format(formatProvider, "Case 3:{0:Mt}",  3.123456E-001))
Console.WriteLine(String.Format(formatProvider, "Case 4:{0:mt3}", 3.123456E+001))
Console.WriteLine(String.Format(formatProvider, "Case 5:{0:A}",   3.123456E-001))
Console.WriteLine(String.Format(formatProvider, "Case 6:{0:S}",   3.123456E-001))
Console.WriteLine(String.Format(formatProvider, "Case 7:{0:At}",  3.123456E-001))
Console.WriteLine(String.Format(formatProvider, "Case 8:{0:St}",  3.123456E-001))
Console.WriteLine(String.Format(formatProvider, "Case 9:{0:A5}S", 3.123456E-001))

/* Outputs:
Case 1: 312.3E-003
Case 2: 31.2e+000
Case 3: 312.3E-3
Case 4: 31.2e0
Case 5: 312.3 m
Case 6: 312.3 milli
Case 7: 312.3m
Case 8: 312.3milli
Case 9: 312.35 mS
*/
```

The Engineering Math Class

The Engineering Math class is a superset of math functions consisting of the Microsoft Math class and some additional methods provided by Agilent that are specific to test and measurement users.

The Microsoft library is well documented within Microsoft's help, so those methods are not discussed. Additionally, Agilent has added a Complex data type method to many of the Microsoft methods. Since the nature of the method is identical, these methods are not discussed here either.

The following table contains a list of methods Agilent has provided to enhance the Engineering Math Class.

Table 26 Table of Engineering Math Methods

Method	Description
Asinh	Returns the hyperbolic sine of the parameter passed to the method. This is an overload supporting the complex and double data types.
Acosh	Returns the hyperbolic cosine of the parameter passed to the method. This is an overload supporting the complex and double data types.
Acoth	Returns the hyperbolic cotangent of the parameter passed to the method. This is an overload supporting the complex and double data types.
Atanh	Returns the hyperbolic tangent of the parameter passed to the method. This is an overload supporting the complex and double data types.
Atan2	Returns the angle (the inverse tangent), as measued in radians, that is equal to the first parameter passed to the method divided by the second parameter passed to the method. This is an overload supporting the complex and double data types.
Conjugate	This is accomplished by negating the imaginary portion of the complex number. This is an overload supporting the complex and double data types.

Table 26 Table of Engineering Math Methods

Method	Description
Magnitude	The absolute value of the parameter passed to the method. This is an overload supporting the complex and double data types.
Phase	Returns the argument of a Complex number expressed in radians. This is an overload supporting the complex and double data types.
Real	Computes the real portion of a Complex number that is input as phase and magnitude
Imaginary	Computes the imaginary portion of a Complex number that is input as phase and magnitude
ToRadians	Converts Degrees to Radians
ToDegrees	Converts Radians to Degrees

11
Using Agilent VEE with .NET

This chapter describes how to use the VEE Wrapper Wizard and offers many tips and techniques for making new and old VEE source code work well with .NET.

NOTE

The VEE Wrapper Wizard only works with VEE versions 6.03 or higher. If you are using version 6.0, the VEE support page on the WWW (www.agilent.com/find/vee) provides a free 6.0 to 6.01 update under the VEE Support category, Update Patches link. A version 6.01 to 6.03 update is included with the Agilent T&M Toolkit product. Versions of VEE earlier than 6.0 require an upgrade.

Using Agilent VEE with .NET

Using the VEE Wrapper Wizard

Agilent VEE is a a powerful graphical programming environment for fast measurement analysis results. The VEE Wrapper Wizard is designed to allow the VEE programmer to continue working in the VEE environment but be able to use Agilent T&M Toolkit and Visual Studio .NET as well.

Using the VEE Wrapper Wizard

Step	Action	Notes
1 Launch the Wrapper Wizard	**a** From the Visual Studio .NET main menu, select T&M Toolkit. **b** From the T&M Toolkit menu, select VEE Wrapper Wizard. **c** Click **Next** to continue or **Cancel** to quit the wizard.	• Before starting the VEE Wrapper Wizard, start VEE and open your VEE project.
2 Point the VEE Wrapper Wizard to your **.vee** file.	**a** Either browse to or type in the full path to the **.vee** file you are using. **b** Click **Next** to continue, **Cancel** to quit the wizard, or **Back** to return to the previous screen.	
3 The Wrapper Wizard parses the file.	Click **Next** to continue, **Cancel** to quit the wizard, or **Back** to return to the previous screen.	• Warnings may appear. These are typically related to mismatches between the variable naming conventions of .NET and the more relaxed standards of VEE. • The VEE Wrapper Wizard automatically converts naming differences between the VEE and .NET. You should review each warning.

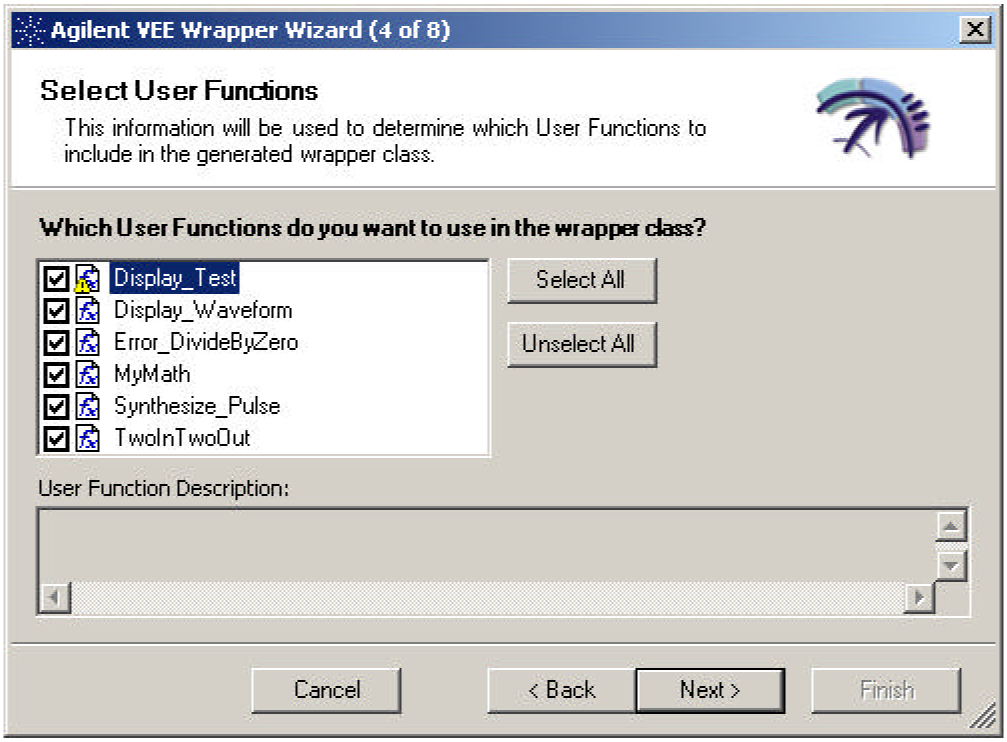
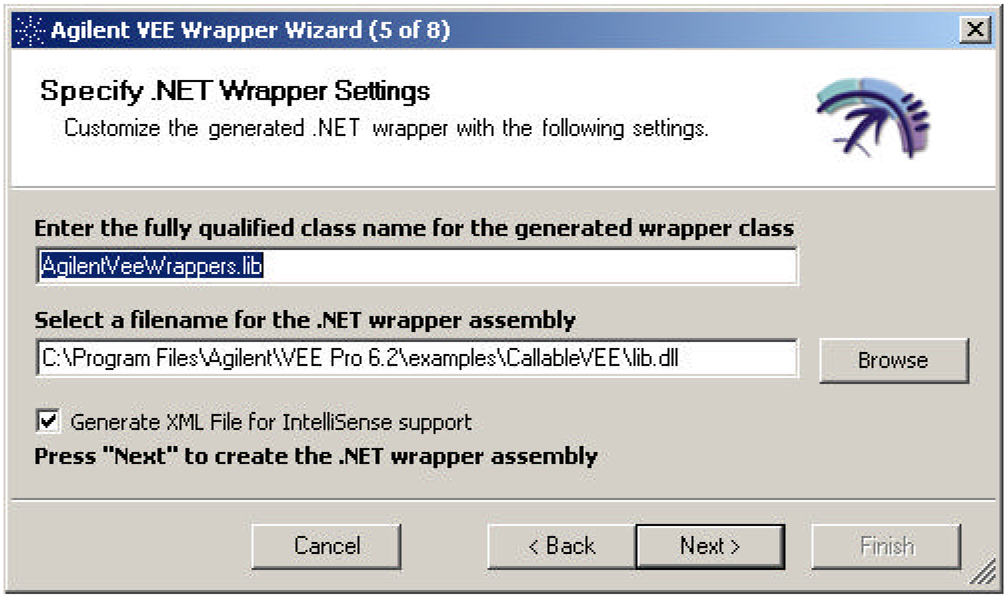

Agilent VEE Wrapper Wizard (4 of 8)
Select User Functions
This information will be used to determine which User Functions to include in the generated wrapper class.
Which User Functions do you want to use in the wrapper class?
☑ Display_Test
☑ Display_Waveform
☑ Error_DivideByZero
☑ MyMath
☑ Synthesize_Pulse
☑ TwoInTwoOut
Select All
Unselect All
User Function Description:
Cancel
< Back
Next >
Finish

Figure 52 Selecting User Functions

Using the VEE Wrapper Wizard (continued)

Steps	Actions	Notes
4 Select the User Functions to include in your VEE Wrapper.	**a** Review the list of available VEE User Functions and select the ones you want to include and clear the ones you do not want to include. **b** Click **Next** to continue, **Cancel** to quit the wizard, or **Back** to return to the previous screen.	• The VEE Wrapper Wizard *only* includes User Functions. • The description that appears in the **User Function Description** text box comes directly from the VEE Description text box for the User Function.



Specify .NET Wrapper Settings

Customize the generated .NET wrapper with the following settings.

Enter the fully qualified class name for the generated wrapper class

AgilentVeeWrappers.lib

Select a filename for the .NET wrapper assembly

C:\Program Files\Agilent\VEE Pro 6.2\examples\CallableVEE\lib.dll

☑ Generate XML File for IntelliSense support

Press "Next" to create the .NET wrapper assembly

Figure 53 Specify .NET Wrapper Settings

Using the VEE Wrapper Wizard (continued)

Steps	Actions	Notes
5 Specify the .NET Wrapper settings.	**a** Enter the fully qualified class name for the generated wrapper class. **b** Select a file name for the .NET wrapper assembly. **c** Select whether to automatically generate XML. It is recommended that you do generate XML. **d** Click **Next** to continue, **Cancel** to quit the wizard, or **Back** to return to the previous screen.	
6 Compile the .NET Wrapper Assembly.	Click **Next** to continue, **Cancel** to quit the wizard, or **Back** to return to the previous screen.	

Using the VEE Wrapper Wizard (continued)

Steps	Actions	Notes
7 Select code generation options.	**a** Select whether to add the new References to the project and select the project. **b** Select whether to launch the Code Paste Tool. **a** Click **Next** to continue, **Cancel** to quit the wizard, or **Back** to return to the previous screen.	
8 Generate code	Click **Next** to continue, **Cancel** to quit the wizard, **Back** to return to the previous screen, or Finish.	
9 Paste Code	If you chose to launch the Code Paste Tool, paste the code from the Code Paste Tool into your source code.	

How do VEE and .NET Communicate?

VEE Pro provides an ActiveX Automation Server that makes VEE UserFunctions available for use by other applications. Those UserFunctions are stored in a **.vee** file. The Agilent T&M Toolkit provides classes, methods, and properties that enable communication with the Automation Server in VEE. These VEE communication tools are available when you add a GeneratedWrapper class to your .NET application. The GeneratedWrapper class is created when you run the VEE Wrapper Wizard.

Several classes are involved withVEE Pro’s communication. An overview of the most important classes follows:

- The **VeeLibrary** object knows the name and location of your **.vee** program file. It has a collection of all the UserFuctions available in that program. It creates and owns an instance of the VeeCallServer to handle the actual communication with VEE Pro.
- The **VeeCallServer** object controls VEE's configuration (such as the window geometry).

- The **GeneratedWrapper** is the principal object you use to write code that communicates with VEE Pro. For programming convenience, the GeneratedWrapper presents each VEE UserFunction as a separate method, complete with IntelliSense help. Because the GeneratedWrapper class inherits from the VeeLibrary class, it also provides a way to access all the public members of VeeLibrary and VeeCallServer.

Configuring and Using the Callable VEE Server

VEE's Callable VEE Server is an ActiveX Automation Server available in version 6.x and greater. Some helpful hints for configuring and using the Callable VEE Server follow.

For the Server Host, either the VEE Pro development environment or VEE Pro RunTime can host the Callable VEE Server. If VEE Pro RunTime is installed on a system without the development environment, VEE Pro RunTime registers itself as the Callable VEE Server. On systems containing both VEE Pro RunTime and the development environment, you can change the Callable VEE Server host from the command line. To make VEE Pro RunTime the Callable VEE Server, execute:

<*vee_run_install_dir*>`\veerun.exe /regserver`

To make the VEE Pro development environment the Callable VEE Server, execute:

<*vee_pro_install_dir*>`\vee.exe /regserver`

When in the operator mode, the Callable VEE Server does not always display a VEE window. In many cases, the VEE server performs its tasks without displaying a window. The following conditions cause the VEE server to display a window when it runs:

- The DebugEnabled property is **True**.
- The called UserFunction uses a Panel view as its operator interface and specifies Show Panel on Execute.
- A dialog box (from the Data menu) executes, requesting operator input.

There are some I/O configuration items you should consider:

- The Callable VEE Server does *not* support loading embedded I/O configurations. This means that the Callable VEE Server only uses the global I/O configuration file. VEE searches for the global I/O configuration file as follows:
  1. If an environment variable named HOME exists, VEE always looks for the I/O configuration file there. A blank I/O configuration file is created if one doesn't exist in the %HOME% directory.
  2. If no HOME environment variable is defined, then VEE looks for the I/O configuration file in:

<%USERPROFILE%> **\Local Settings\Application Data\Agilent\VEE Pro**

- The remote instance of the Callable VEE Server is also affected if HOME is not defined as an environment variable. In this case, the "Identity" selection of the "dcomcnfg" utility affects which I/O configuration file is used.
  - When using "The interactive user" setting as the Identity Selection, the current user's I/O configuration file is used. For example, if JDoe is logged on at the time the Callable VEE Server started, the Callable VEE Server looks for the I/O configuration file:

..\JDoe\Local Settings\Application Data\Agilent\VEE Pro\vee.io

  - If you use "The launching user" setting or "This user" as the Identity Selection, Windows NT normally loads the "Default User" profile. If you use either of these settings, you should copy the "vee.io" and "vee.rc" files from your profile directory:

<%USERPROFILE%> **\Local Settings\Application Data\Agilent\VEE Pro**

to

..\Default User\Local Settings\Application Data\Agilent\VEE Pro\vee.io

NOTE If you are using VEE Pro RunTime to host the Callable VEE Server, change all directory references above to **VEE Pro RunTime**.

To verify your configuration settings, install VEE Pro 6.03 or higher on the client computer. Then, using a command prompt window, change directory to:

*<vee_pro_install_dir>***\examples\CallableVEE\VBScript**

and execute:

```
cscript GetServerInfo.vbs \\<remote_host_name>
```

If this script fails, see the Troubleshooting section of online Help.

Can Legacy VEE Code be Adapted to .NET?

Absolutely! The T&M Toolkit can generate a wrapper that lets you access the UserFunctions in any (Version 6.03 or greater) VEE Pro program, without requiring you to modify the VEE program. However, if you are willing to revisit your legacy VEE programs, there are some steps you can take to enhance the interaction between your VEE code and your Visual Studio .NET applications.

What can be Accessed in VEE?

Keep in mind that the Callable VEE Server cannot access the "Main" portion of a VEE program. Only UserFunctions and things embedded within UserFunctions are accessible when calling VEE from a .NET application. Structure your VEE program so that all the features you want to access are built as UserFunctions.

Naming UserFunctions and Parameters

When naming UserFunctions and their input and output terminals, follow the .NET naming rules. In summary: the first character of a name should be a letter, and a name should contain only letters, numbers, and underscores. It is also helpful to use meaningful names. The VEE Wrapper Wizard automatically alters names that do not conform to .NET naming rules. You are not required to change non-conforming names, but you might prefer your own alterations to the automatic alterations.

If you know the expected data type for your input parameters, it is helpful to make those constraints part of your program. In VEE, double-click the input terminal to open the **Input Terminal Information** dialog. Set the **Required Type** and **Required Shape** to the expected values. This lets your .NET application see the actual data type. Without this constraint, .NET applications see the terminals as the generic type "object."

12
Product Support

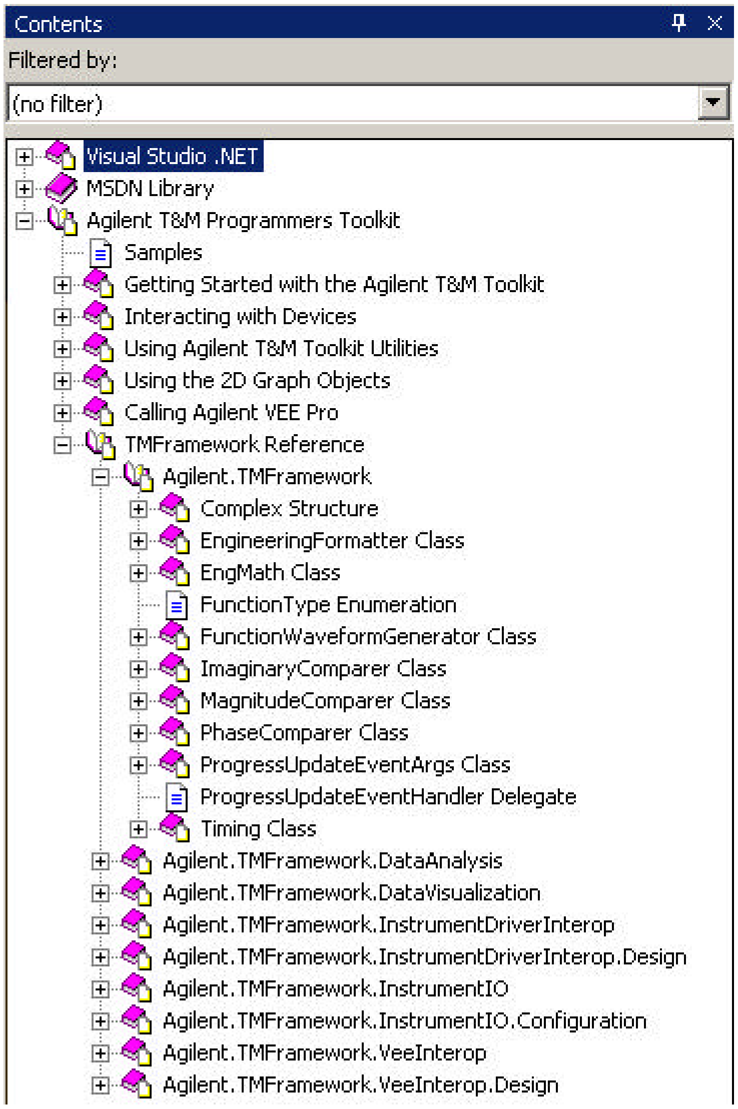
This chapter describes the sources of help within Agilent T&M Toolkit, the Agilent Developer Network, and the support options.

Help Sources

Online Help

Agilent T&M Toolkit has an extensive set of Help files. These files include tutorials, samples, and documentation for all of the class libraries and stand-alone products.

The Agilent T&M Toolkit Help follows the Visual Studio .NET help guidelines, have the same look-and-feel, and are tightly integrated into the .NET help system. See Figure54 on page139.



Contents

Filtered by:

(no filter)

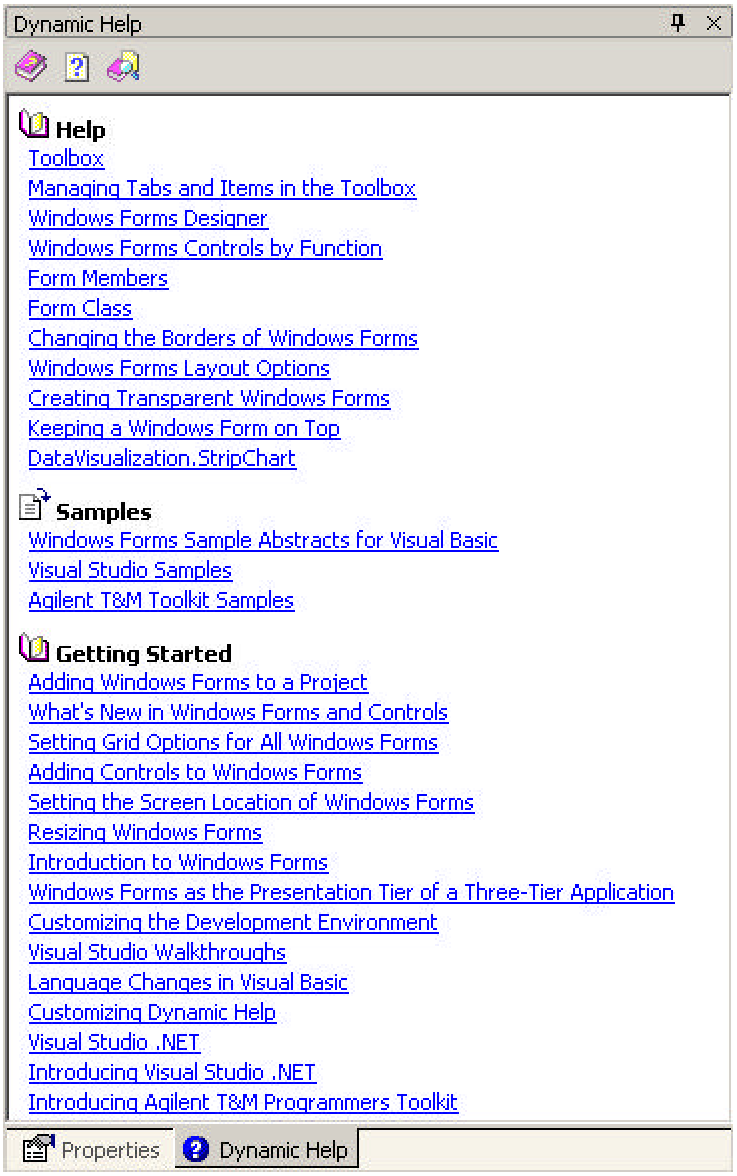
- Visual Studio .NET
- MSDN Library
- Agilent T&M Programmers Toolkit
  - Samples
  - Getting Started with the Agilent T&M Toolkit
  - Interacting with Devices
  - Using Agilent T&M Toolkit Utilities
  - Using the 2D Graph Objects
  - Calling Agilent VEE Pro
  - TMFramework Reference
    - Agilent.TMFramework
      - Complex Structure
      - EngineeringFormatter Class
      - EngMath Class
      - FunctionType Enumeration
      - FunctionWaveformGenerator Class
      - ImaginaryComparer Class
      - MagnitudeComparer Class
      - PhaseComparer Class
      - ProgressUpdateEventArgs Class
      - ProgressUpdateEventHandler Delegate
      - Timing Class
    - Agilent.TMFramework.DataAnalysis
    - Agilent.TMFramework.DataVisualization
    - Agilent.TMFramework.InstrumentDriverInterop
    - Agilent.TMFramework.InstrumentDriverInterop.Design
    - Agilent.TMFramework.InstrumentIO
    - Agilent.TMFramework.InstrumentIO.Configuration
    - Agilent.TMFramework.VeeInterop
    - Agilent.TMFramework.VeeInterop.Design

Figure 54 Agilent T&M Toolkit Help Contents

Dynamic Help

With Visual Studio .NET, Microsoft has released a new help feature, Dynamic Help. This feature is both powerful and simple to use. Dynamic Help is typically docked in the lower right corner of your display. If it is not there, go to the .NET main menu, select **Help** > **Dynamic Help**.

Dynamic Help contains a list of help references for the features you are currently using. In Figure55 on page141, you see a Dynamic Help window with three books, Help, Samples, and Getting Started. Under the Help book is a reference to the 2D Graph Control, DataVisualization.StripChart. The cursor was on this control and, since a help topic is available for this control, dynamic help provides quick access to it. As the cursor shifts to a different control, feature, or word, Dynamic Help dynamically changes.



Dynamic Help

Help

Toolbox
Managing Tabs and Items in the Toolbox
Windows Forms Designer
Windows Forms Controls by Function
Form Members
Form Class
Changing the Borders of Windows Forms
Windows Forms Layout Options
Creating Transparent Windows Forms
Keeping a Windows Form on Top
DataVisualization.StripChart

Samples

Windows Forms Sample Abstracts for Visual Basic
Visual Studio Samples
Agilent T&M Toolkit Samples

Getting Started

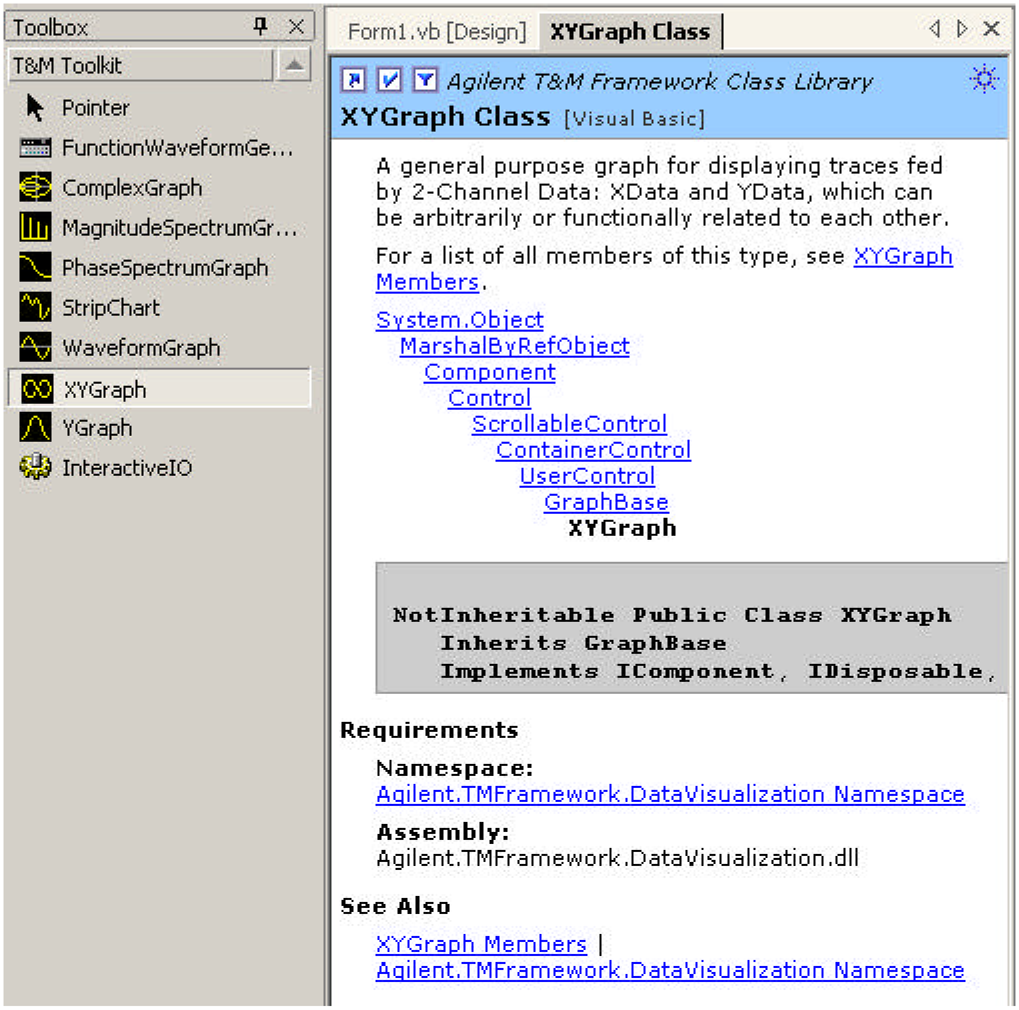
Adding Windows Forms to a Project
What's New in Windows Forms and Controls
Setting Grid Options for All Windows Forms
Adding Controls to Windows Forms
Setting the Screen Location of Windows Forms
Resizing Windows Forms
Introduction to Windows Forms
Windows Forms as the Presentation Tier of a Three-Tier Application
Customizing the Development Environment
Visual Studio Walkthroughs
Language Changes in Visual Basic
Customizing Dynamic Help
Visual Studio .NET
Introducing Visual Studio .NET
Introducing Agilent T&M Programmers Toolkit

Properties | Dynamic Help

Figure 55 Visual Studio .NET Dynamic Help

F1 Help

Agilent T&M Toolkit supports F1 help. This means you can press F1 and, if a help topic exists for the control, word, phrase, etc., the help topic is displayed. See Figure56.



Agilent T&M Framework Class Library

XYGraph Class [Visual Basic]

A general purpose graph for displaying traces fed by 2-Channel Data: XData and YData, which can be arbitrarily or functionally related to each other.

For a list of all members of this type, see XYGraph Members.

```
System.Object
   MarshalByRefObject
      Component
         Control
            ScrollableControl
               ContainerControl
                  UserControl
                     GraphBase
                        XYGraph
```

```
NotInheritable Public Class XYGraph
   Inherits GraphBase
   Implements IComponent, IDisposable,
```

Requirements

Namespace: Agilent.TMFramework.DataVisualization Namespace

Assembly: Agilent.TMFramework.DataVisualization.dll

See Also

XYGraph Members | Agilent.TMFramework.DataVisualization Namespace

Figure 56 With XYGraph Active, F1 is Pressed

Agilent Developer Network

The Agilent Developer Network (ADN) gives you access to the connectivity resources you need, in one place, easily accessible, available 24 hours a day, 7 days a week. You'll find drivers, tools, sample code, documentation, and expertise, as well as a Web-based community of T&M professionals to help you get your job done. You save time, maintain your productivity, and get results.

You can register for a FREE membership in ADN at http://www.agilent.com/find/adn.

Support

Support Included with All T&M Toolkit Products

Installation

Agilent ensures the software is correctly installed.

Activation - Product Key

Agilent ensures the software is correctly activated.

Start Up

Agilent ensures that, once the software is installed and activated, it works correctly subject to the terms of the Warranty enclosed with your software.

Where You can Find Support

If you do not know where the nearest support center is, use the World Wide Web to go to http://www.agilent.co/find/assist.