

Agilent T&M Toolkit 2.0

Getting Started Guide



Agilent Technologies

Notices

© Agilent Technologies, Inc. 2004

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Manual Part Number

W1140-90026

Edition

Fourth Edition, October 2004

Printed in USA

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

Microsoft® and Windows® are US registered trademarks of the Microsoft Corporation.

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.

WARNING

A **WARNING** 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 personal injury or death. Do not proceed beyond a **WARNING** notice until the indicated conditions are fully understood and met.

Introduction

About the Agilent T&M Toolkit	2
What is the T&M Toolkit?	2
What Makes the T&M Toolkit Useful?	2
What is in the T&M Toolkit?	2
Installing Agilent T&M Toolkit	4
Enabling Agilent T&M Toolkit	4
Uninstalling Agilent T&M Toolkit	4
How to Use this Manual	5
Hardware Requirements	5
Computer/Processor	5
Minimum Memory (RAM) Requirements	6
Free Disk Space	6
Drive	6
Display	6
Input Device	6
Supported Instrument Interfaces	6
Software Requirements	7
Operating System	7
Development Environment	7

1 Preparing Your Instruments

The Agilent IO Libraries Suite	10
Connecting the Instrument	10
Run the Agilent Connection Expert	10
T&M Toolkit and a Bad Connection	12

2 Finding, Defining, and Communicating with Instruments

Introduction	14
How to Start the Instrument Explorer	14
How Instrument Explorer Discovers Instruments	14
Instrument Explorer and the IVI Config Store	15
Instrument Explorer and Agilent's IO Config	15
Instrument Connection Status Icons	17
Other Ways to Add Instruments	18
Using the Instrument Connection Wizard	20

Generating Code	21
Drag-and-Drop Instrument Connections	22
Managing Multiple Instrument Configurations	22

3 The New Project Wizard

Using the New Project Wizard	26
------------------------------------	----

4 Describing the DirectIO Class

Introduction	28
Controlling the Instrument Connection	28
Instrument Connection Strings	28
Data Input and Output	29
Instrument Connection Management	31
Using Low-Level Commands for the GPIB, Serial, TCPIP, and USB Interfaces ...	32

5 Results Management

Introducing Results Management	36
Managing Your Data	36
The Structure of the Results Management Component	36
Drop-On Controls	37
Results Checks	40
Using the Data Table Control	41
The Data Table	41
Features	42
Data Selection	43
Exercise 8-1: Using the Data Table of the Test Automator	43
Using the Graph Control	44
The Available Graphs	44
Exercise 8-1: Continued	44
Features	44
Statistics and Analysis Philosophy	45

6 Programmatically Analyzing Your Data

The Complex Data Type	48
Creating a Complex Number	48
Using Complex Numbers	48

The Complex Type Conversion	48
Comparers	49
The Imaginary Comparer Subclass	49
The Magnitude Comparer Subclass	49
The Phase Comparer Subclass	50
The Waveform Data Type	50
Creating a Waveform	50
Using Waveforms	51
The Phase Spectrum Data Type	51
Creating a Spectrum	52
Using Spectrums	52
Digital Signal Processing Capabilities	53
Fast Fourier and Inverse Fast Fourier Transforms	53
Data Analysis Functions	57
Bessel Functions	58
The I0 Bessel Function	58
The J0 and J1 Bessel Functions	58
The Y0 and Y1 Bessel Functions	58
Statistical Functions	58
Regression Techniques	59

7 Using the 2D Graph Objects

Using Traces to Get Data to the Graph	62
The Default Trace	62
Where are the 2D Graph Controls?	62
Defining New Traces and Markers	63
Setting and Plotting Data	66
What Types of Graphs are Available?	67
Y Graph	68
Strip Chart	69
XY Graph	70
Complex Graph	72
Magnitude Spectrum Graph	73
Phase Spectrum Graph	74
How can I Modify the Appearance of the Graph?	75
Agilent T&M Toolkit 2D Graph Property Categories	75

Panning, Zooming, and the Context Menu	77
--	----

8 Using Agilent VEE with .NET

Using the VEE Wrapper Wizard	79
How do VEE and .NET Communicate?	83
Configuring and Using the Callable VEE Server	84
Can Legacy VEE Code be Adapted to .NET?	86
What can be Accessed in VEE?	86
Naming UserFunctions and Parameters	86

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

Generating Waveforms	90
The Timing Class	92
ProgressUpdater	93
The EngineeringFormatter Class	94
The Engineering Math Class	96

10 Quick Instrument Communication Using Interactive IO

How to Quickly Communicate with Instruments	98
What to Do if an Instrument Does Not Communicate	98
Interactive IO's User Interface	98

11 Using the IO Monitor

Why Monitor Interaction with Instruments?	102
How Can You Monitor Interaction with Instruments?	102
Getting Started Monitoring Interaction with Instruments	104

12 Using the Driver Import Wizard

Using the Driver Import Wizard	108
Expanding the Driver Possibilities	108
Which Drivers are Supported?	108
Starting and Using the Wizard	109
Troubleshooting	111
Parse Errors and Compile Errors	113

13 Product Support

Online Help	116
Dynamic Help	118
F1 Help	120
The Agilent Developer Network	121
Support Included with All T&M Toolkit Products	122
Installation	122
Activation - Product Key	122
Start Up	122
Where You can Find Support	122
User Communities	122

Welcome to the Agilent T&M Toolkit

About the Agilent T&M Toolkit 2

Installing Agilent T&M Toolkit 4

How to Use this Manual 5

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 activate 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 online 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 *VXIplug&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 capabilities including matrix and regression, and digital signal processing support.
- Data Types – Includes support for Complex numbers, the Phase Spectrum data type, and the Waveform data type.
- 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.

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. Finally, ensure you are logged on as a user with administrative privileges.

Place the Agilent T&M Toolkit CD-ROM into the computer's CD-ROM drive. 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 Toolkit**. The main menu also has an item entitled **T&M Toolkit**.

Enabling Agilent T&M Toolkit

Agilent T&M Toolkit uses a Product Key to enable 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 once it is shipped to you. 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 will see **Agilent T&M Toolkit 2.0 for Visual Studio .NET**. Select this entry and choose **Remove**.

How to Use this Manual

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

1 Connecting to Instruments

- See Chapter 1 - Preparing the Instruments
- See Chapter 2 - Using Instrument Explorer

2 Working with Instruments

- See Chapter 3 - The New Project Wizard
- See Chapter 4 - Using the DirectIO Class to Manage your Instruments

3 Analyze and Display your data

- See Chapter 5 - Results Management
- See Chapter 6 - Analyzing Instrument Data
- See Chapter 7 - Using the 2D Graph Objects

4 Optional helpers for numerous areas

- See Chapter 8 - Using Agilent VEE with .NET
- See Chapter 9 - Virtual Waveforms, Timing Classes, and Number Formatting
- See Chapter 10 - Quick Instrument Communication Using InteractiveIO
- See Chapter 11 - Using IO Monitor
- See Chapter 12 - Using the Driver Wrapper Wizard

5 Product Support

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

Large fonts and non-normal DPI settings are not supported. Toolkit only supports small fonts and 96 DPI.

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 or later

Software Requirements

Operating System

- Design Time - Microsoft Windows 2000, XP
- Run Time - Microsoft Windows 98 SE, ME, NT SP6a, 2000 SP4, XP SP1, and XP Home SP1

Development Environment

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

1

Preparing Your Instruments

The Agilent IO Libraries Suite 10
T&M Toolkit and a Bad Connection 12

This chapter describes how to connect your instruments and how to use the Agilent IO Libraries Suite to connect and configure your instrument drivers.

The Agilent IO Libraries Suite

What is the Agilent IO Libraries Suite?

Connection Expert is a software utility that helps you quickly get your instruments connected to your PC and troubleshoot connectivity problems. Its collection of libraries give you the ability to use your instruments from a test and measurement program and provide utilities that help you quickly and easily connect your instruments to your PC.

Connecting the Instrument

Connect the instruments to your PC. Ensure all of the cables are tight and the instruments are powered up.

Run the Agilent Connection Expert

Connection Expert is the key utility you will use in the IO Libraries Suite. You can use it to:

- configure instrument I/O interfaces
- automatically discover instruments that are connected to your PC or to your local area network
- browse the structure and connections of your test system (including your PC, instruments, and interfaces)
- detect and troubleshoot connectivity problems in your test system
- create programming aliases that you can use in place of addresses to improve portability and readability of your test program

Connection Expert includes a task guide (the left pane of the utility's window) that provides shortcuts to common tasks as well as helpful information.

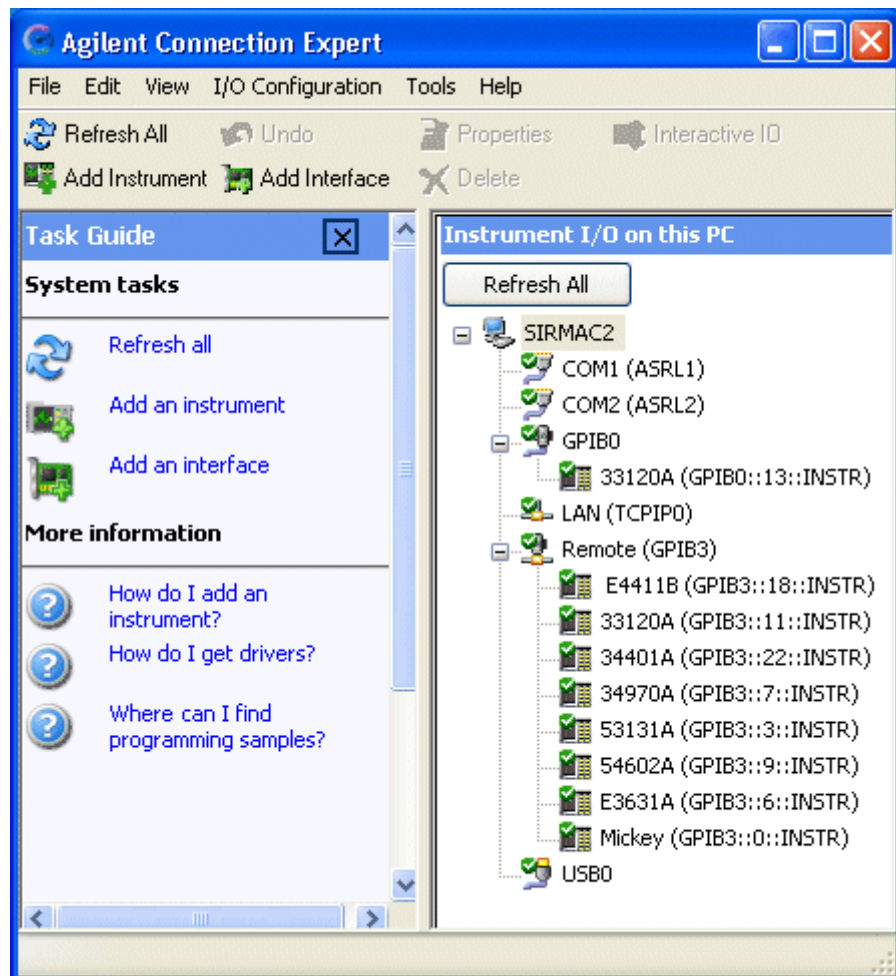


Figure 1 Connection Expert

The Agilent IO Libraries Suite, which includes the Connection Expert as a key utility, is shipped with Agilent T&M Toolkit. For further information about this powerful utility, see the online help for the Connection Expert.

T&M Toolkit and a Bad Connection

If you expect the instrument to respond, and it appears to be inactive, power it down and check the connections. If they appear to be fine, you may have a bad cable, a bad connection interface, or a problem in the program. In this case, T&M Toolkit offers Interactive IO. Interactive IO can be used to send single commands to an instrument. Using it, you can quickly verify whether an instrument is active and, by simply swapping items out, identify the faulty component. See Quick Instrument Communication Using Interactive IO in Chapter 10.

2 Finding, Defining, and Communicating with Instruments

Introduction	14
How to Start the Instrument Explorer	14
Instrument Connection Status Icons	17
Other Ways to Add Instruments	18
Using the Instrument Connection Wizard	20
Generating Code	21
Drag-and-Drop Instrument Connections	22
Managing Multiple Instrument Configurations	22

This chapter describes how to locate, identify, and communicate with your instruments. Managing multiple instrument configurations is also discussed as well as how to generate connection source code directly from the identification process.

Introduction

T&M Toolkit includes a powerful and sophisticated tool called Instrument Explorer to help you find, establish communications with, and identify instruments.

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 identifies and verifies which instruments are connected to the PC
- Creates and saves files containing instrument connection information to a configuration file for later use
- Helps you find and select drivers through a Wizard
- Generates code that connects to an instrument within a Visual Basic .NET, C#, or C++ project using DirectIO or a driver

How to Start the Instrument Explorer

To start the Instrument Explorer, select **T&M Toolkit > Instrument Explorer** from the main menu.

How Instrument Explorer Discovers Instruments

Instrument Explorer creates its instrument configuration by gathering information from two sources. The IVI Config Store contains all previously configured instruments on your system (whether they are still connected or not). Also, the Instrument Explorer can interactively find all live instruments connected to your PC with the assistance of Agilent's IO Libraries Suite.

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 two lists: all of the previously configured instruments on the system, and a list of all IVI drivers that are installed.

Instrument Explorer and Agilent's IO Libraries Suite

Instrument Explorer finds all live instruments with the assistance of Agilent's IO Libraries Suite. All of these instruments are added to the Instrument Explorer database. Defining the LAN connection using IO Libraries Suite allows Instrument Explorer to define and interact with remote instruments.

Exercise 2-1: Finding Instruments Connected to Your Computer

Power up several instruments that are connected to your computer. Run Agilent's IO Libraries Suite and configure the I/O cards. Then follow these steps:

- 1 Select the Find Instruments button from the Instrument Explorer toolbar. Instrument Explorer begins the instrument discovery phase of its operation.



Figure 2 Instrument Explorer Toolbar

- 2 When the discovery phase is finished, 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.

2 Finding, Defining, and Communicating with Instruments

- 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 3. Select **OK**.
 - b If available, additional information about the instrument is displayed.



Figure 3 Using *IDN? with 488.2 Compliant Instruments

- 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 Figure 4 on page 17.

NOTE

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



Figure 4 Displaying Instrument Details

Instrument Connection Status Icons

Instrument connections are represented iconically in Instrument Explorer as follows:

2 Finding, Defining, and Communicating with Instruments

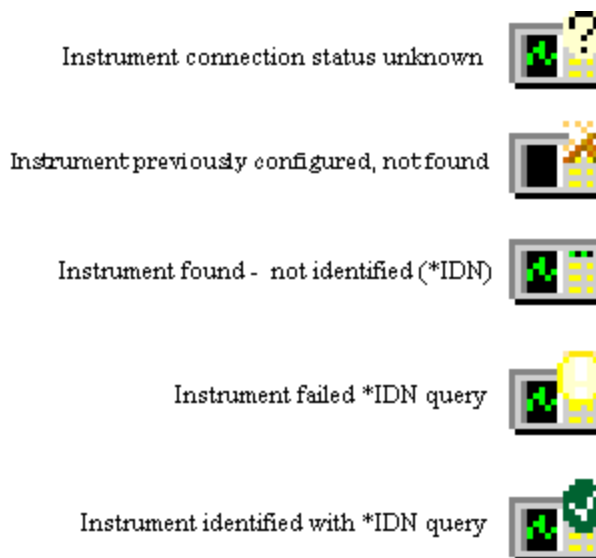



Figure 5 Instrument Connection Status Icons

Other Ways to Add Instruments

Besides the simplicity of the Instrument Explorer's Find instruments ability, there are two other ways to add instruments to your configuration.

You can use the Add Instrument Icon () or right click the bus where you want to add the instrument, see Figure 6.

NOTE

The instrument you insert can be a virtual instrument. In other words, it can exist in the configuration but not be connected or even physically near your computer.

When you choose one of these methods, you will see a dialog box, Figure 7, that helps you set up your new instrument. The address can be any valid address, although you may need to edit the address when the instrument is finally connected.

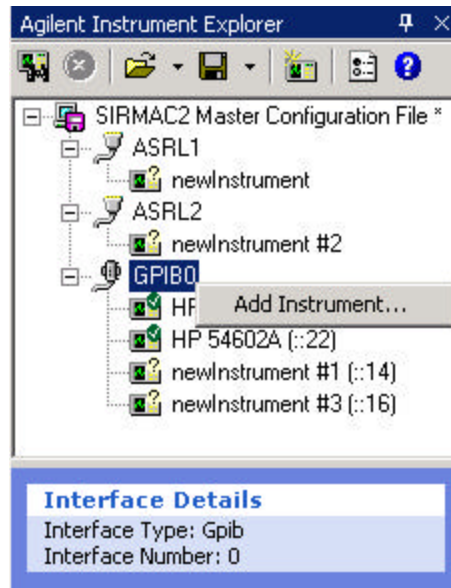


Figure 6 Adding an Instrument

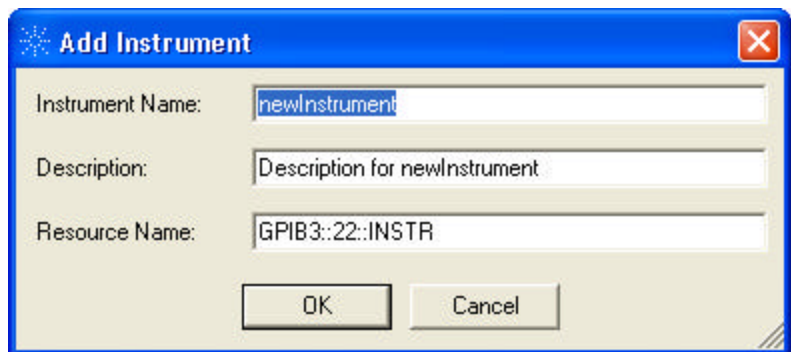


Figure 7 Add Instrument Dialog Box

Using the Instrument Connection Wizard

The Instrument Explorer creates instrument connections. Each connection has a unique name and a defined interface type and can be used in your program once or many times.

There are two types of connections: a Driver connection and a DirectIO connection (more information about DirectIO comes in Chapter 4). Driver connections create a session for a driver type when you use a driver. The supported driver types are *VXIplug&play*, IVI-COM, and IVI-C. You can also choose a DirectIO connection, which does not use drivers at all, that allows you to directly communicate with your instrument using a command language such as SIDL.

You can generate the source code that creates an instrument connection and inserts it into your program using the Instrument Connection Wizard.

Exercise 2-2: Creating an Instrument Connection and Generating Code

- 1 Highlight an instrument in Instrument Explorer and right click on it
- 2 From the pop-up menu, select **Add Instrument Connection**
- 3 Review the Wizard's Welcome screen and then press **Next**
- 4 On this screen, you choose the type of connection you want, either a driver (*VXIplug&play* , IVI-C, or IVI-COM) or a DirectIO connection. If a driver is available for your instrument, choose driver, otherwise choose DirectIO.
- 5 Enter a unique name for the connection. Unless there are changes you need to make, leave the defaults for the rest of the choices. Choose **Next**.

This process now transitions into code generation.

Generating Code

As the Instrument Connection Wizard nears its end, it transitions into a Code Generation tool. You have the option to generate the code you need to create an instrument connection within your program.

- 6 You should be at the Select Code Generation Options screen. From this screen you choose whether to go to the Code Paste Tool or to stop with creating the connection. If you want to stop with the connection, clear the check box for Launch Code Paste Tool after Wizard completes. Choose **Next**.
- 7 If you cleared the Code Paste Tool check box, the connection will be created as described in the summary box you see displayed. Press **Finish**. The sessions are represented iconically in Instrument Explorer as shown in Figure 8.

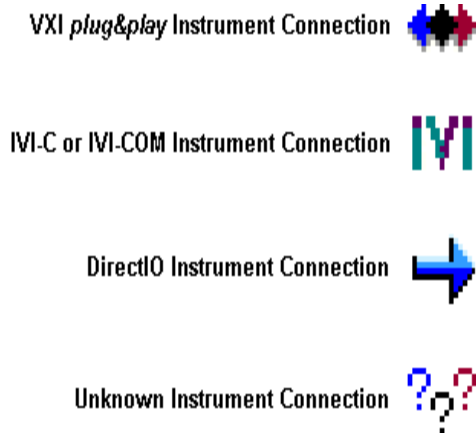


Figure 8 Iconic Representations of the Instrument Sessions

Exercise 2-2: Creating an Instrument Connection and Generating Code - continued

- 8 If you proceed to the Code Paste Tool, place your cursor where you want the connection code pasted and press the **Paste Code at Cursor** button. The Code Paste Tool inserts the code needed to create an instance of the instrument.

Drag-and-Drop Instrument Connections

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

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 drag-and-drop the Instrument Session into your source code, you also established the insertion point for the generated code.

Managing Multiple Instrument Configurations

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

Exercise 2-3: Managing Instrument Configurations

You have opened Instrument Explorer with two instruments defined in your Master Configuration file. You would like to add two more instruments, but you don't want to lose your original configuration. That is, you want two configuration files: the default Master file with two instruments and another file with four instruments.

- 1 Select the **Add Instruments** icon shown in Figure 9. Enter an instrument name, a description, and a resource name. An example of a resource name is GPIB1::7::INSTR. This instrument is not really connected to your system and

may not even be physically present. Nevertheless, you can use it for programming purposes.

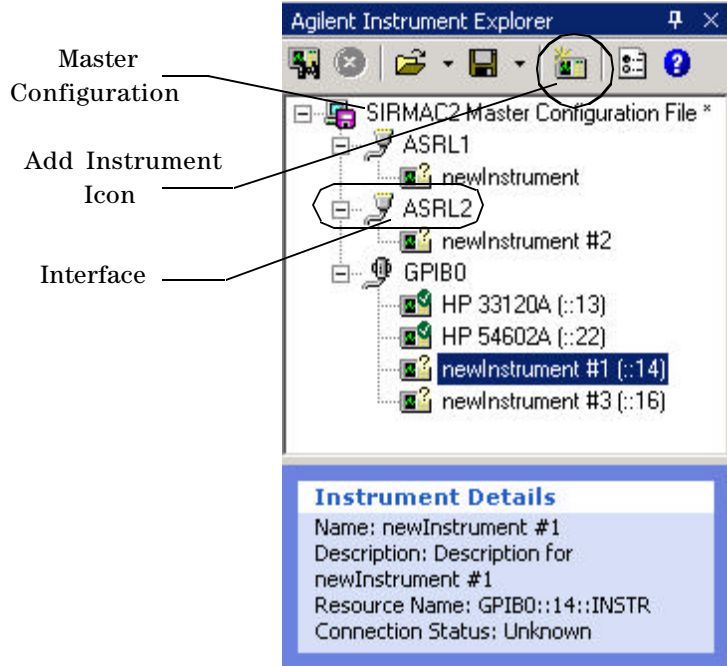



Figure 9 Adding an Instrument

- 2 You want to save this new configuration and use it later, but you don't want to lose your original configuration. Select the save configuration icon (see Figure 10) and save the configuration file as AddingFour.xml.

NOTE

Whenever you plan to use a new configuration, be sure to 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.

2 Finding, Defining, and Communicating with Instruments



Figure 10 Saving a Configuration

Instrument Explorer's versatility is best demonstrated by this feature. The configuration with four added instruments is just as useful as the configuration with two instruments. Instrument Explorer will use either configuration when you are programming. In fact, there is an optional parameter to specify the IVI Configuration filename from the call in your program.

- 3 After saving the configuration with four instruments added, open the Master Configuration file by selecting the open configuration file icon (see Figure 11) and selecting **Open Master Configuration File**. Notice the Master Configuration file only contains the two instruments originally defined. You can now open `AddingFour.xml` to see the configuration with four instruments added.

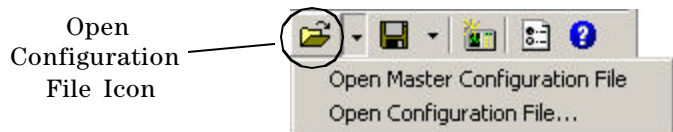


Figure 11 Opening a Different Configuration File

- 4 Highlight one of the instruments you created, right click, and choose **Delete** from the pop-up menu.
- 5 Do not save this new configuration. If you delete an instrument and save the configuration file, the `IVIConfigurationStore.xml` file is also updated. Reload (re-open) the Master Configuration file instead.

3

The New Project Wizard

Using the New Project Wizard 26

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.

Exercise 3-1 Starting a New Project in T&M Toolkit

- 1** Select **File > New > Project**.
- 2** Select Agilent T&M Toolkit Projects and T&M Toolkit Project Wizard.
- 3** Enter a project name and home directory. Select **OK** to proceed.
- 4** Select **Next** at the Welcome screen.
- 5** Select a Language to use. For the examples in this book, Visual Basic is used. Choose **Next**.
- 6** Select the type of project you want. In this case, select **Windows Application** and choose **Next** to continue.
- 7** The New Project Wizard assigns four commonly used namespaces to your project. For a description of these namespaces, refer to the Library Description at the bottom of the library selection window.

While you could select additional namespaces (class libraries), only use the default values for this project. Select **Next** to continue.

- 8** Review the summary screen. Select **Finish** to complete

Your project is initiated with all of the appropriate Imports statements and references to T&M Toolkit namespaces.

4

Describing the DirectIO Class

Introduction 28

Controlling the Instrument Connection 28

This chapter describes how to interact directly with your instruments without the need for drivers.

Introduction

DirectIO allows communication with any message-based instrument. Using the extensive properties and methods of the DirectIO class, you can control the instrument. DirectIO is commonly used when you do not have an instrument-specific driver. Because it allows a tight control over your instrument, you may also prefer it if you are experienced in programming instruments with command sets such as SCPI.

NOTE

DirectIO only supports the GPIB, ASRL, TCP/IP, and USB interface types. It does not support VXI or GPIB-VXI.

Controlling the Instrument Connection

The DirectIO class includes many properties and methods for controlling and configuring your instrument connection.

Instrument Connection Strings

Before you can move forward using DirectIO, you have to create a DirectIO object in your code. This is as simple as “DirectIO arb = new DirectIO(GPIB0::22::INSTR)”.

The *new* operator is used to create objects and invoke constructors. In this case, you are creating a new DirectIO object. You need to call DirectIO with the address of the instrument you are using. In this case, GPIB interface card 0 connecting to an instrument at address 22.

Use the syntax elements found in Table 1 to help you construct your address strings. Items in brackets are optional; italicized words represent items you provide.

NOTE

In the case of USB interfaces, aliases are strongly recommended.

Table 1 Supported Connection Strings

Interface	Connection String
GPIB	GPIB <i>board</i> :: <i>primary address</i> [: <i>secondary address</i>]: INSTR
SERIAL	ASRL[<i>COM port#</i>]:INSTR
TCPIP	TCPIP[<i>board</i>]: <i>host address</i> [: <i>LAN device name</i>]:INSTR
TCPIP	TCPIP[<i>board</i>]: <i>host address</i> :: <i>port</i> :SOCKET
USB	USB[<i>board</i>]: <i>Manufacturer ID</i> : <i>Model Code</i> : <i>Instrument Serial Number</i> ::[<i>USB Interface Number</i>]:INSTR

Data Input and Output

Using Visual Basic .NET to create a new instrument object is as simple as”:

Imports Agilent.TMFramework.InstrumentIO

.

Dim myinst As New DirectIO(“GPIB0:13:INSTR”)

myinst.WriteLine(“*IDN”)

TextBox1.Text=myinst.Read

The power of Intellisense combines with the simplicity of DirectIO to make instrument interaction easy. Toolkit supports WriteLine, Write, and Unbuffered Reads and Writes.

Toolkit’s list of read/write data types is extensive. It includes: arrays, IEEE 488.2 definite binary blocks, integers, bytes, strings, and objects. For more information, see DirectIO Properties and Methods for Managing I/O in the online help.

The following code sample shows how to create a very small and simple application that establishes and instrument connection, issues an *IDN?, resets the instrument, and publishes the *IDN? information.

4 Describing the DirectIO Class

```
static void Main(string[] args)
{
    // Prompt the user for the address of the instrument string
    address;
    Console.WriteLine("Enter address of the instrument (ex:
    GPIB0::22::INSTR):");
    address = Console.ReadLine();
    try
    {
        // Establish the connection with the instrument
        DirectIO instr = new DirectIO(address);
        using (instr)
        {
            // Display instrument information string
            identification;
            instr.WriteLine("*IDN?");
            identification = instr.Read();
            Console.WriteLine(identification);
            // Reset the instrument
            instr.WriteLine("*RST");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    // Wait for user input
    Console.WriteLine("Press ENTER to end program.");
    Console.ReadLine();
}
```

Instrument Connection Management

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

Table 2 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 are read-only.

Table 3 DirectIO Instrument Connection Management

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

Using Low-Level Commands for the GPIB, Serial, TCP/IP, and USB Interfaces

DirectIO provides almost all of the 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. If you move the oscilloscope from a GPIB interface to a LAN interface, all you have to change is the instrument address when you create the new DirectIO object. The program you have written using DirectIO still works.

DirectIO does provide access to lower level commands that are interface specific. However, if you use the lower-level commands 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 classes have properties that 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 sub-class to perform a GPIB only command.

```
Module Module1
    sub Main()
        Dim scope As New
        DirectIO("GPIB0::22::INSTR")
        scope.Gpib.SendLocalLockout()
    End Sub
End Module
```

NOTE

In the `scope.Gpib.SendLocalLockout()` command, `scope` is the instrument, `Gpib` is a property of `scope` that is used to access interface specific properties or methods not available in `DirectIO`, and `SendLocalLockout()` is the method being called. It could just as easily be `scope.Tcpip.HostName`, or `scope.Serial.StopBits`, or `scope.Usb.ManufacturerID`.

4 Describing the DirectIO Class

5

Results Management

Introducing Results Management 36

Managing Your Data 36

Using the Data Table Control 41

Using the Graph Control 44

Results Management is a component that makes it easy to manage, display, and export data from multiple sources in T&M Toolkit.

Introducing Results Management

The T&M Toolkit delivers a new tool to help you with your Test and Measurement tasks, the new T&M Toolkit component called Results Management. The Results Management component is available for use with any solution developed with T&M Toolkit, but Test Automator provides an immediate example of a small part of the Results Management component's potential.

Managing Your Data

The Structure of the Results Management Component

The Results Management component has a data flow as shown in Figure 12 on page 37. This structure is almost identical to Visual Studio's Trace function, and a review of the Trace Function is a good review of how Results Management works.

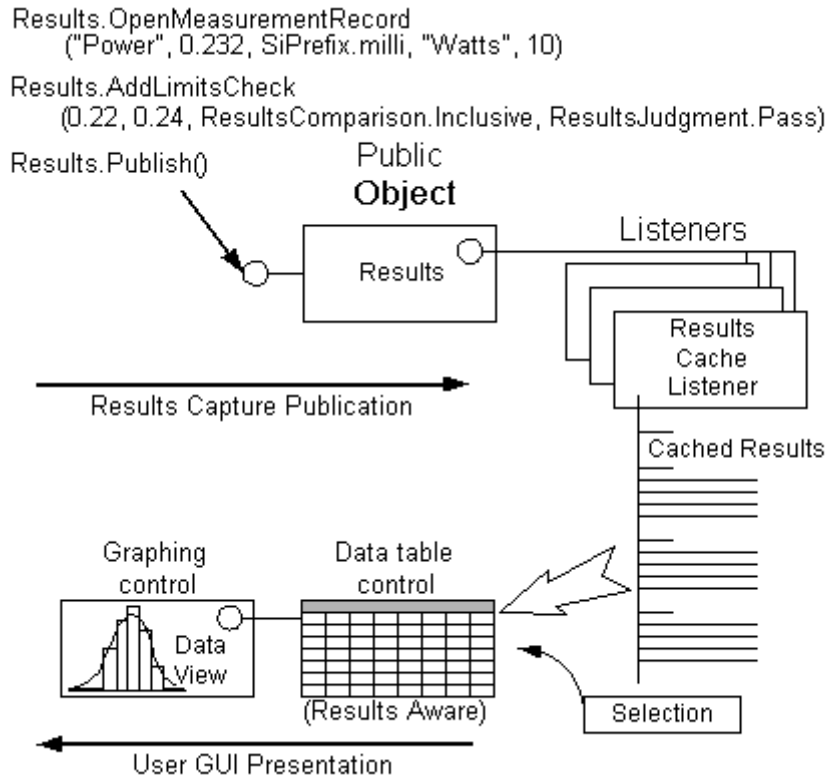


Figure 12 The Results Management Flowchart

Drop-On Controls

The Data Management Graphing Control is a drop-on control. In other words, you can drop the graph control or the results table control onto your form just like you would drop any other control.

5 Results Management

To do this, go to the T&M Toolkit tab of the forms toolbox and right click. From the context menu that appears, select **Add/Remove Items** and scroll to the Agilent.TMFramework.ResultsVisualization Namespace. Select all of the items in this namespace and press OK

Create a new project. Drag and drop the ResultsHistogram control to the form. Add a button and the following code in the button click sub routine.

```
Results.OpenMeasurementRecord("Power", 2.32,  
SiPrefix.none, "Watts", 3)  
Results.AddLimitsCheck(2.2, 2.4,  
ResultsComparison.Inclusive,  
ResultsJudgment.Pass)  
Results.Publish()
```

Run the project and click the button five times. (You need to press it five times to create a small sample set.) You should see the following graphic:

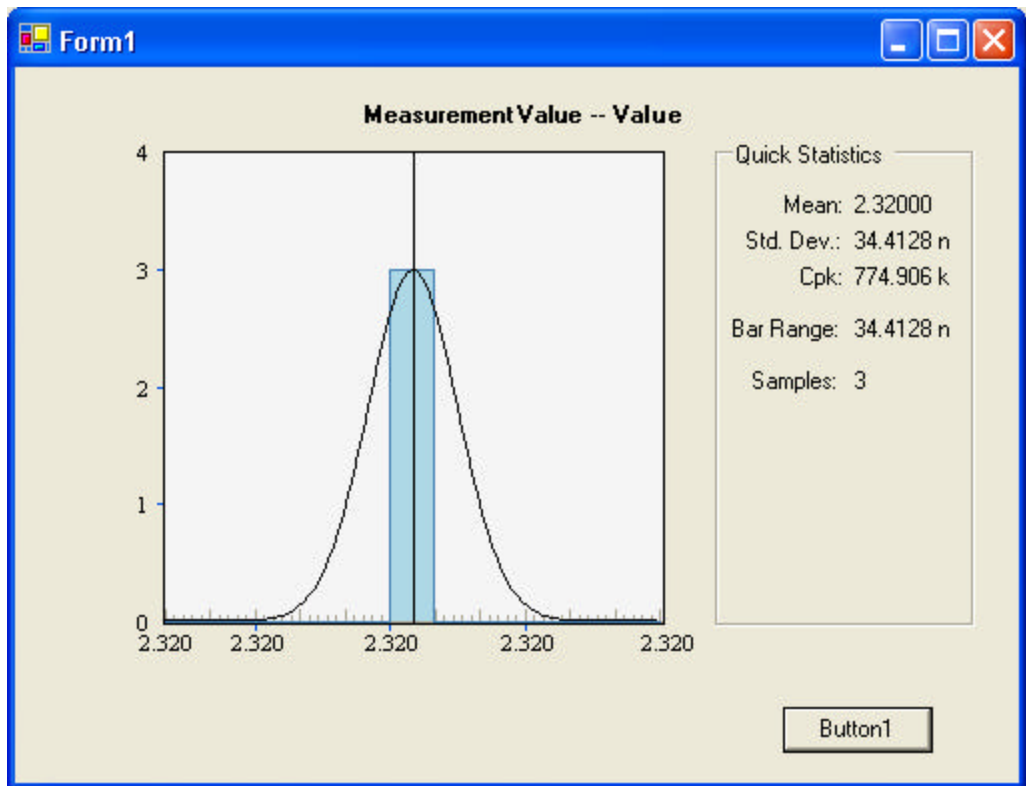
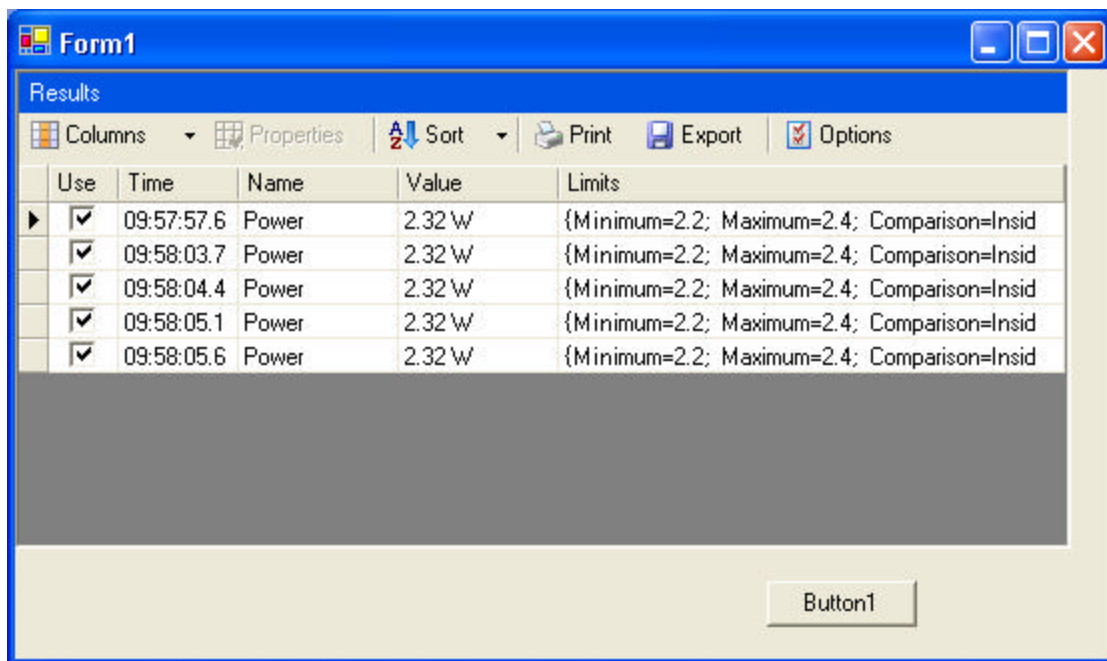


Figure 13 Graph Control Example

Since there is only one value being tracked, all of the responses fall into the same bucket, so there is only one bar.

Delete the ResultsHistogram control and add a ResultsTableViewer control. Run the project again. Click the button five times and you should have a data table that looks like the following graphic:

5 Results Management



Use	Time	Name	Value	Limits
<input checked="" type="checkbox"/>	09:57:57.6	Power	2.32 W	{Minimum=2.2; Maximum=2.4; Comparison=Insid
<input checked="" type="checkbox"/>	09:58:03.7	Power	2.32 W	{Minimum=2.2; Maximum=2.4; Comparison=Insid
<input checked="" type="checkbox"/>	09:58:04.4	Power	2.32 W	{Minimum=2.2; Maximum=2.4; Comparison=Insid
<input checked="" type="checkbox"/>	09:58:05.1	Power	2.32 W	{Minimum=2.2; Maximum=2.4; Comparison=Insid
<input checked="" type="checkbox"/>	09:58:05.6	Power	2.32 W	{Minimum=2.2; Maximum=2.4; Comparison=Insid

Figure 14 Results Table Control Example

You are encouraged to play with the other controls made available by the ResultsVisualization Namespace.

Results Checks

All data in the data results table is generated from the results checks in your sequence definition. You can control the data that appears in the data results table, and subsequently in the graphs, by checking or unchecking the check boxes to the right of the sequence tree as shown in Figure 13 on page 41

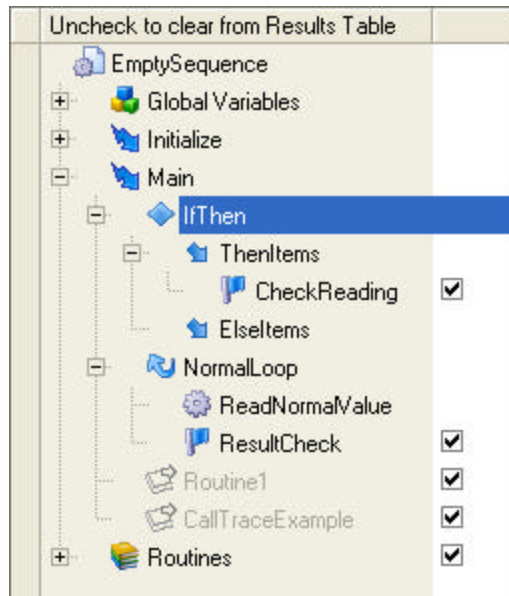


Figure 15 Results Check Check Boxes

Using the Data Table Control


The Data Table


In the data table, only the visible data is used for graphing, export, sorting, and so on. Data that is not visible is considered excluded from all processing and is not acted upon.


In the data table, you can select which rows to use by checking the Use column. This column is a default column for every row and is used specifically to determine which rows are included and which are excluded from the presentation phase of the data output.


Columns can also be excluded; you can select which columns are included and which are not by using a dialog box or an interactive menu. Additionally, some columns have sub-fields (which can also have sub-fields). You can choose to view some sub-fields, all sub-fields, or only the parent column.


Features


 Columns: A dialog box or right click menu that lets you select which columns to include in the Data Table.


 Expand: Used to expand columns that contain sub-fields. The result of this operation is a set of individual columns, one for each sub-field.


 Compress: Used to compress columns that contain sub-fields. The result of this operation is a single parent column.

 Remove: Used to remove a column from the visible data table. The data is not damaged or discarded, only removed from the visible part of the table.


 Rename: Used to rename a column.


 Properties: Used to set up the properties of the data table. For example, column width, format, and data alignment.


 Sort: Used to sort all the columns with the highlighted column(s) as the sort key. Up to 10 different columns can be defined as sort keys.

 Setup: Used to setup the data table page for printing.

 Preview: Used to preview the data table print out.

 Print: Used to print the data table.

 Export: Used to save the data table as an Excel file, a comma separated value (CSV) file, or a text file. Excel is the default.

 Options: Used to set multiple options for the toolbar and the data table. For example, which buttons are displayed on the toolbar, and the colors used in the data table.


Data Selection

With one exception, WYSIWYG is the rule for the data table. If the data is in the table, it is available for analysis, graphing, export, and printing. All data that is not in the table is ignored.

The one exception is the Use column. This is a default column for every row in the data table. You can, if you choose, remove the check mark in this field, and the row associated with the Use field is ignored but still visible. This is one way to eliminate rows from the data table. In this case, however, the row is visible but not considered for analysis, graphing, export, or printing.

The other way to remove data from the data results table is to uncheck the related results check. See Figure 13 on page 41.

Exercise 5-1: Using the Data Table of the Test Automator

- 1 Open the Test Automator application.
- 2 Load the DemoSequence that is shipped with Test Automator.
- 3 Run the Sequence Definition by pressing the run icon () from the main menu.
- 4 Select the results tab.
- 5 The data table is visible at the bottom of the screen. Review the values in the Value column, and note how they cluster around 10. As you will also note, there is an outlier value of 5 that does not fit the pattern at all.
- 6 Select the Use column for this outlier value. Remove the checkmark from the Use field and this eliminates the value from the graphing process.


- 7 You can also uncheck the checkmark next to CheckReading in the sequence definition. You have to expand the If/Then statement to find CheckReading. This demonstrates how the check boxes are nested to the level of the associated results check.

Using the Graph Control





The Available Graphs


Test Automator ships with three default graphs. Using one of these, the Histogram graph, the graphing features will be demonstrated.


Exercise 5-1: Continued


- 8 Highlight the Value column by clicking on the column title.
- 9 With the outlier removed, select the graph icon () , choose **Histogram**, and click on **OK**.
- 10 Note that the histogram displayed is quite symmetrical.
- 11 Select one of the records with outlier values and check the Use column by clicking in it.
- 12 The histogram disappears. Reactivate it by choosing the Value column title once again. Note how the outlier value is now displayed in the histogram.
- 13 Uncheck one of the results checks in the sequence tree and watch how the histogram changes.

Features

-  Graphs: Used to select the graph type.
-  Properties: Used to set the appearance properties of the graph.
-  Setup: Used to setup the page properties for printing.
-  Print Preview: Used to preview the print job before printing it.

 **Print:** Print the graph. Only the visible portion of the graph is printed. This is due to the limitations of the available programming tools.

 **Export:** Save the file in a BMP, GIF, JPEG, PNG, or TIFF format.

 **Options:** Toolbar options. For example, which buttons to show on the Toolbar.

Statistics and Analysis Philosophy

The histogram graph is unique in that it also supplies some statistical values. This statistical analysis functionality is rudimentary and is only intended to provide some basic trend analysis. The use of the data table export feature is strongly encouraged for advanced data analysis.

5 Results Management

6

Programmatically Analyzing Your Data

The Complex Data Type	48
The Waveform Data Type	50
The Phase Spectrum Data Type	51
Fast Fourier and Inverse Fast Fourier Transforms	53
Data Analysis Functions	57
Bessel Functions	58
Statistical Functions	58
Regression Techniques	59

This chapter describes T&M Toolkits Complex data type and the multiple statistical and mathematical functions that T&M Toolkit provides.

The Complex Data Type

Visual Studio .NET does not have a native complex data type, yet the complex data type is commonly used by engineers. 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.

Creating a Complex Number

Complex data is constructed in a similar manner as other .NET objects. For example, to create a complex scalar with real value 2 and imaginary value 3, you could program:

```
Dim cpx As New Complex(2,3).
```

Using Complex Numbers

Mathematics can be performed on Complex numbers using the static methods of the Complex class, and the static methods of the EngMath class. For example:

```
Dim c1 As New Complex(2,3)
Dim c2 As New Complex(5,10)
Dim cSum As Complex
cSum = Complex.Add(c1,c2)
Dim cSin As Complex
cSin = EngMath.Sin(cSin)
```

The Complex Type Conversion

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

Also, either the real or imaginary portion of the Complex data type can be extracted using the "Real" and "Imaginary" properties. There are also methods to retrieve the Phase and Angle of a Complex. Because getting the phase or angle of a complex number is not an instantaneous calculation, Phase and Angle are methods not properties.

When converting a double to a complex number, the double is converted to the real portion and the imaginary value 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. For example, the C# function would be:

```
public static void Sort( Array array );
```

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.

The Waveform Data Type

Visual Studio .NET does not have a native Waveform data type, yet the waveform data type is commonly used by engineers. To meet this need, a waveform data type is included with T&M Toolkit and implemented within .NET as an object. The T&M Toolkit Waveform data type provides a way to work with a common instrument data type. T&M Toolkit also defines helper objects that easily allow the user of Agilent oscilloscopes to scale the raw waveform data returned from the scope using the preamble information returned from the scope.

Creating a Waveform

The Waveform data type comprises an array of double values, the Y data, along with an implicit X axis (start time, time between points, number of points). The primary use case for the Waveform data type is to treat the combination of Y data and X axis (time-domain data) as a singular unit. This allows you to pass this information around as a single, encapsulated unit to math routines that understand a waveform like an FFT to graph objects that know how to display a waveform.

Waveform data is constructed in a manner similar to other .NET objects. For example, to create a simple waveform with start time of 1 and time between points of .02, you would program:

```
Dim array() As Double = {1, 2, 3, 4, 5}
Dim startTime As Double = 1
Dim timeBetweenPts As Double = 0.02

Dim waveform As Waveform = New Waveform(array,
    startTime, timeBetweenPts)
```

Using Waveforms

Many mathematical operations can be performed on waveforms using the methods in the Waveform class. For example:

```
Dim array1() As Double = {1, 2, 3, 4, 5}
Dim array2() As Double = {6, 7, 8, 9, 10}

Dim startTime As Double = 1
Dim timeBetweenPts As Double = 0.02

Dim waveform1 As Waveform = New Waveform(array1,
    startTime, timeBetweenPts)

Dim waveform2 As Waveform = New Waveform(array2,
    startTime, timeBetweenPts)

Dim waveformSum As Waveform

waveformSum = waveform1 + waveform2
```

You can perform basic mathematical operations on waveforms, you can clone them, you can convert them to strings, you can compare two waveforms, and much more.

The Phase Spectrum Data Type

Visual Studio .NET does not have a native Spectrum data type, yet the Spectrum data type is commonly used by engineers. To meet this need, a Spectrum data type is included with T&M Toolkit and implemented within .NET as an object. The T&M Toolkit Spectrum data type provides a way to work with a common instrument data type. The Spectrum provides a way to model the spectral information from spectrum or network analyzers, Vector Signal Analyzers, and other instruments used to measure spectral data.

Creating a Spectrum

The Spectrum data type is comprised of an array of Complex values, the Y data, along with X axis information (start frequency, stop frequency, and number of points). The primary use case for the Spectrum data type is to treat the combination of Y data and X axis (frequency-domain data) as a singular unit. This allows you to pass this information around as a single, encapsulated unit to math routines that understand a spectrum like an FFT, and to graph objects that know how to display a spectrum, like Magnitude vs. Frequency, Smith Chart, and so on.

Spectrum data is constructed in a similar manner as other .NET objects. For example, to create a simple spectrum with startFrequency of 1 and stopFrequency of 10, you would program:

```
Dim c1 As New Complex(2, 3)
Dim c2 As New Complex(4, 5)
Dim array() As Complex = {c1, c2}
Dim startFreq As Double = 1
Dim stopFreq As Double = 10
Dim spectrum As Spectrum = new Spectrum(array, startFreq, stopFreq)
```

Using Spectrums

Many mathematical operations can be performed on spectrums using the methods in the Spectrum class. For example:

```
Dim array1() As Complex = {c1, c2}
Dim array2() As Complex = {c2, c1}
Dim startFreq As Double = 1
Dim stopFreq As Double = 10
```

```
Dim spectrum1 As Spectrum = new Spectrum(array1,
startFreq, stopFreq)
```

```
Dim spectrum2 As Spectrum = new Spectrum(array2,
startFreq, stopFreq)
```

```
Dim spectrumSum As Spectrum
```

```
spectrumSum = spectrum1 + spectrum2
```

You can perform basic mathematical operations on spectrum, you can clone them, you can convert them to strings, you can compare two spectrums, and much more.

Digital Signal Processing Capabilities

The Agilent T&M Toolkit includes a strong set of digital signal processing (DSP) methods. FFT and IFFT using double precision 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 accepts waveform data and returns spectrum data and the IFFT accepts spectrum data and returns waveform data. Both 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 input array data. 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 have prime or large prime factors.

The following exercise relies on a utility called the function waveform generator. This utility produces waveform data sets and is discussed in chapter 9. This exercise is also used to demonstrate 2D graphing, so it ends somewhat abruptly.

Exercise 7-1 Using Complex Numbers and FFT

- 1** Using the T&M Toolkit New Project Wizard, create a new Visual Basic .NET project.
- 2** From the Toolbox's T&M Toolkit tab, select FunctionWaveformGenerator and drag it onto the form
- 3** Go to the Properties Window of the FunctionWaveformGenerator1 and change the name to FWG1.
- 4** Configure the Properties of the FWG1 as shown in Figure 14 on page 55.

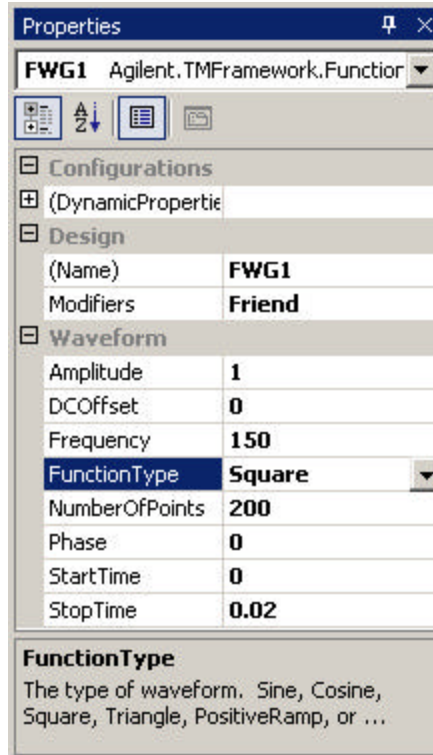


Figure 14 FWG1 Property Settings

- 5 From the Toolbox's T&M Toolkit tab, select Magnitude Spectrum Graph and drag it onto the form.
- 6 Add two Button Controls to the bottom of the Form. Change the name of one to Acquire and the other to Close. Double-click the Acquire button.
- 7 Enter the following code:

Try

```
Dim data(), DataArray() As Double
Dim startFrequency, stopFrequency As Double
Dim i As Integer

'Setup the FWG1
```

6 Programmatically Analyzing Your Data

```
DataArray = FWG1.GenerateYData

'Introduce random noise into the first 75 values
data = DataArray
For i = 0 To 75
    Randomize()
    data(i) = (data(i) * Rnd())
Next

'Apply either the Bartlett, Blackman, Hamming,
'Hanning, or Rectangular filter here

'Using the Hamming filter
Dim filteredData() As Double=Dsp.Hamming(data)

'Using FFT and the Complex data type
Dim fftArray() As Complex=Dsp.FFT(filteredData, _
FWG1.TimeBetweenPoints, startFrequency, _
stopFrequency)

'Set and Plot data on the the Magnitude Graph
MagnitudeSpectrumGraph1.Traces(0).SetData(fftArray, _
startFrequency, stopFrequency)

MagnitudeSpectrumGraph1.XAxis.YIntercept = -100
MagnitudeSpectrumGraph1.PlotAll()
MagnitudeSpectrumGraph1.AutoScaleXY()

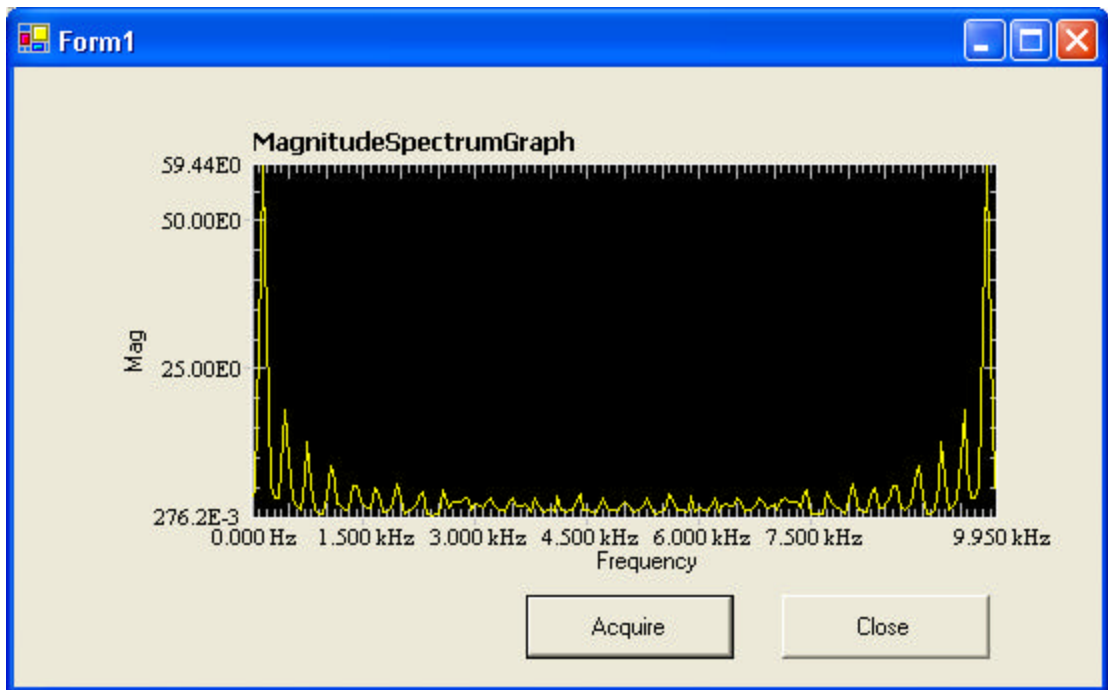
Catch ex As Exception
    MessageBox.Show(Me, "Unknown error occurred: " + _
ex.Message + vbCrLf + vbCrLf + "Stack Trace:" + _
vbCrLf + ex.StackTrace, Me.Text, _
MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try
```

- 8 From the Forms Designer, double-click the close button and enter the following code:

```
Application.Exit()
```

- 9 Run the program, press the **Acquire** button, and you will have a graphic that looks like the following:



10 Press the **Close** button to stop the application.

Data Analysis Functions

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 data array 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 T&M Toolkit includes a Statistics Class. This Class has the statistical tools you most often use. The examples include permutations, factorial, root mean squared, and median.

Regression Techniques

Agilent T&M Toolkit supports five regression methods: Linear, Logarithmic, Exponential, Power, and Polynomial. 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.

6 Programmatically Analyzing Your Data

7

Using the 2D Graph Objects

Using Traces to Get Data to the Graph	62
Where are the 2D Graph Controls?	62
Defining New Traces and Markers	63
What Types of Graphs are Available?	67
How can I Modify the Appearance of the Graph?	75
Panning, Zooming, and the Context Menu	77

This chapter describes the multiple graphs the T&M Toolkit provides and how to use them.

Using Traces to Get Data to the Graph

Traces are the fundamental building block for the 2D Graph Object. Every graph has at least one trace, called the DefaultTrace, and usually several other traces, 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. The DefaultTrace is designed for situations where you want a quick, one trace plot.

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.

See Figure 15 to review the T&M Toolkit tab.

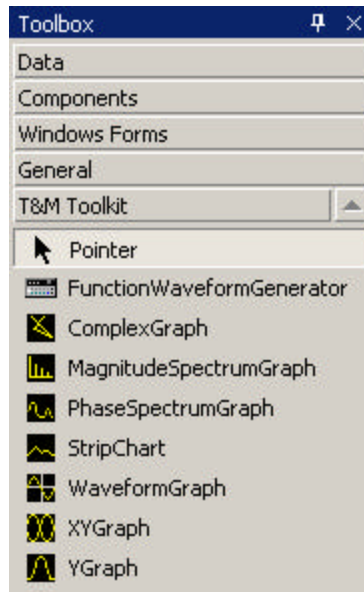


Figure 15 The Agilent T&M Toolkit Toolbox Tab

Defining New Traces and Markers

Exercise 7-1 Creating a Trace

- 1 Start the T&M Toolkit New Project Wizard
- 2 Select a Visual Basic .NET project and accept the defaults for all other Wizard queries.
- 3 If the Toolbox is not visible, select **View > Toolbox** from the Main Menu.
- 4 Open the T&M Toolkit tab in the Toolbox and Drag-and-drop a Waveform graph on the form.
- 5 Since Traces is the default property for all of the 2D Graph Objects, just click the graph and then click the Traces ellipsis in the Properties listing to go to the Trace Collection Editor.

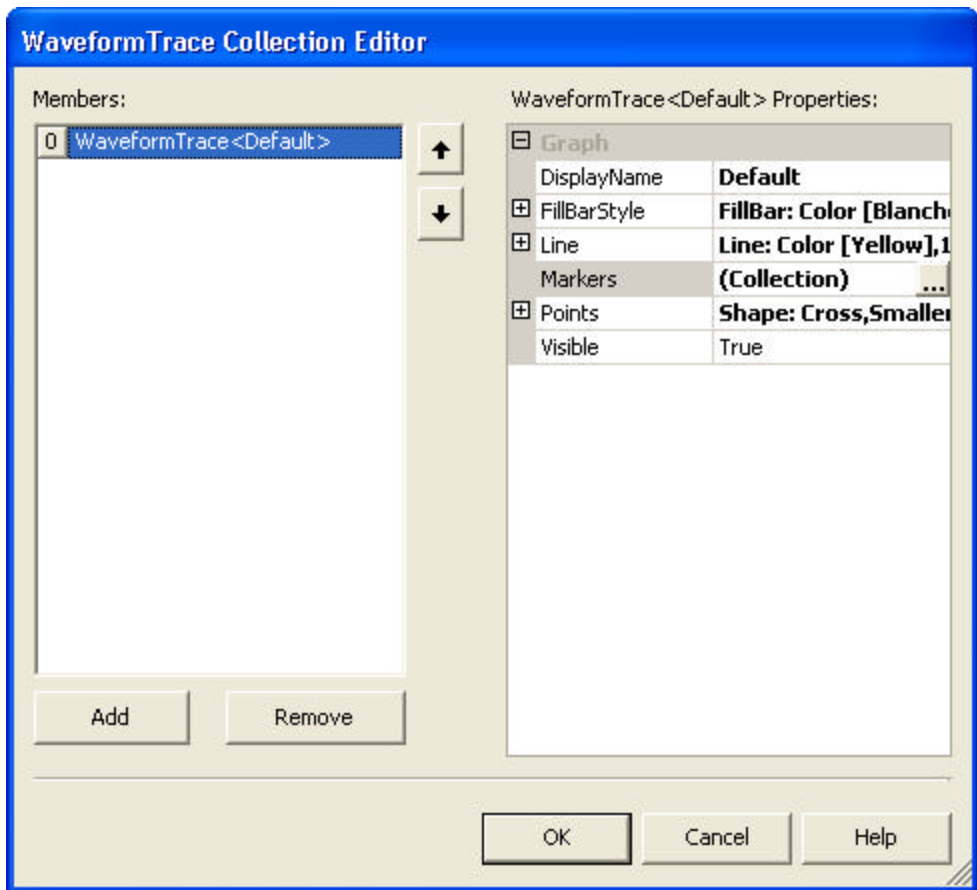


Figure 16 The Trace Collection Editor for the Waveform Trace

6 Add a Trace and choose **OK**.

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 Figure 17.

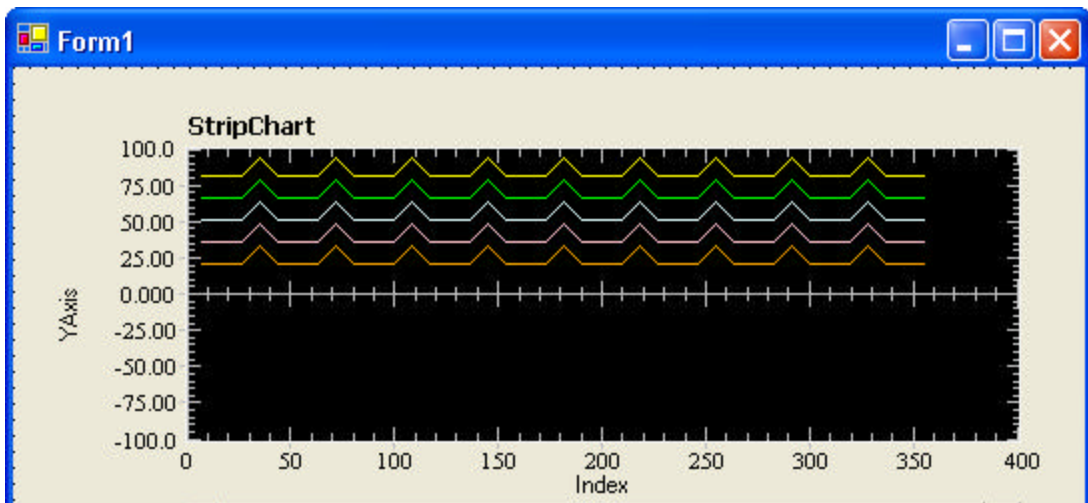


Figure 17 Trace Representations

Notice in Figure 16 that the Marker Collection is accessible from the Trace Collection Editor.

- 7** Click the ellipsis next to Markers. The Markers Collection Editor is very similar to the Traces Collection Editor. Add a Marker.

Having created the graph, the trace, and a marker, the data source is necessary. The FunctionWaveformGenerator was mentioned previously and it will be used again here.

It is a utility that creates data sets that can, among other things, be used very effectively with the T&M Toolkit's 2D graphics.

- 8** Drag-and-drop a FunctionWaveformGenerator onto the form. In the properties listing, rename it FWG1.
- 9** Add a Button control to the form and double-click it.
- 10** Enter the following few lines of code:

```
WaveformTrace1.SetData(FWG1.GenerateYData(), _
    FWG1.StartTime, FWG1.StopTime)

WaveformGraph1.PlotAll()
```


7 Using the 2D Graph Objects

11 Run the program.

12 Press the button to start the graph. The graph should look like Figure 18.

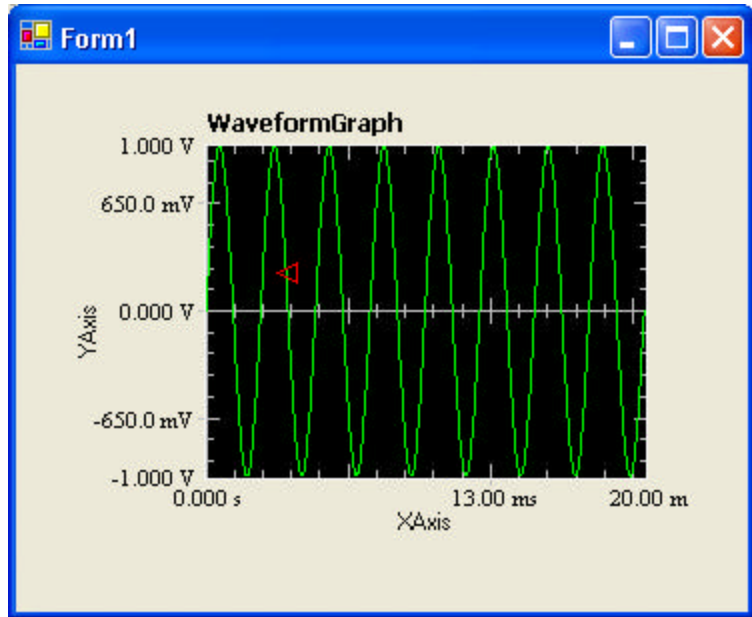


Figure 18 Waveform Graph

13 On the far left side of the graph, there is a Marker. Drag the Marker from point-to-point.

14 Save this project as WaveformGraph

Setting and Plotting Data

You have created a trace and a marker. Using the FunctionWaveformGenerator, you generated the data and then programmatically connected the data to the Waveform Graph. To plot the data, there are three 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 a single trace gives you a quick, simple graph.
- Whenever you have multiple data sets, which means multiple traces, a **SetData** method is used to load data for 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.

What Types of Graphs are Available?

There are seven types of graphs. You have already seen the Waveform Graph, the other six are:

Y Graph

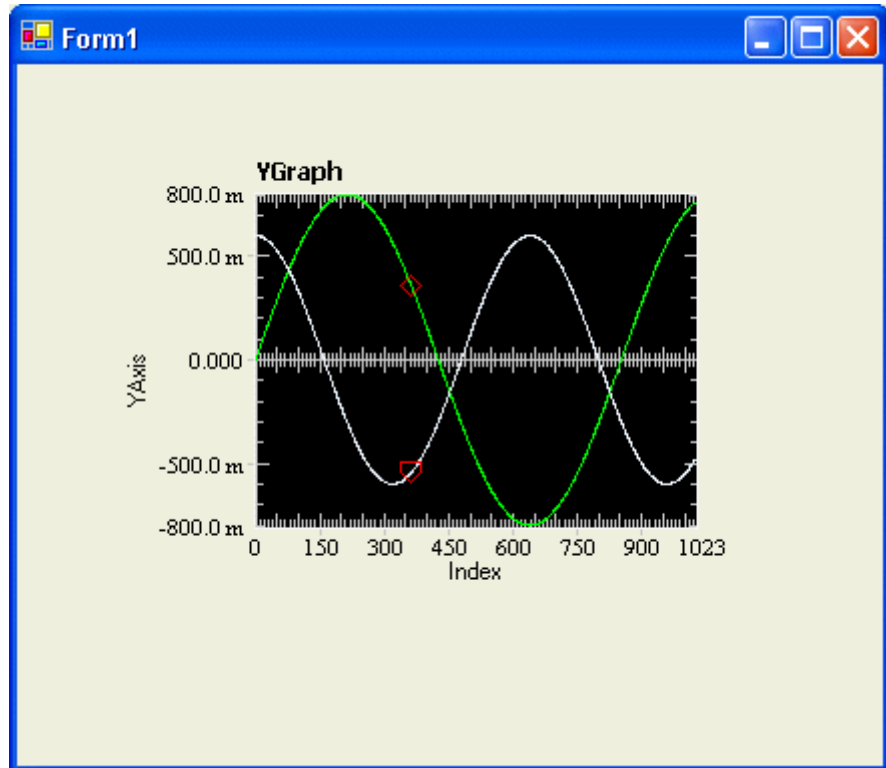


Figure 19 Y Graph

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

Strip Chart

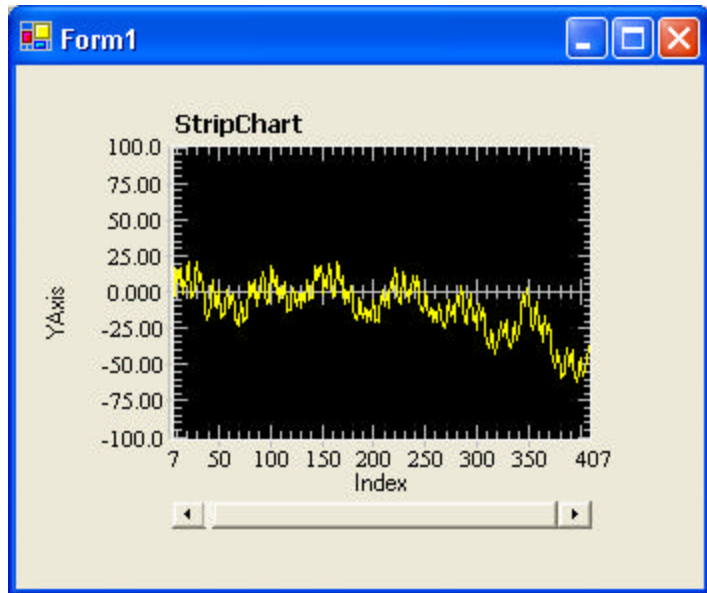


Figure 20 Strip Chart

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 place a timer on your Form. Enable the timer and connect it directly to the system timer. In your source code, this looks like **StripChart1.TimeInterval = Timer1.Interval**. You then manipulate the Timer properties to control the StripChart time interval. See the StripChart example shipped with this product.

XY Graph

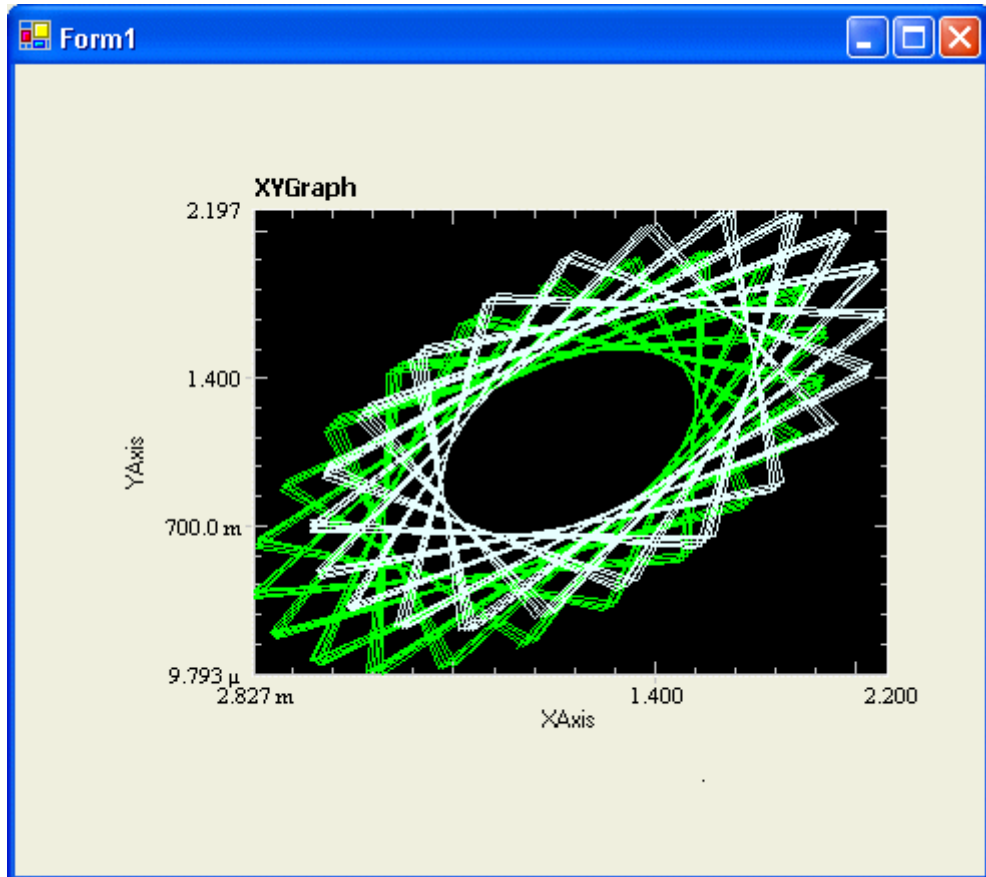


Figure 21 XY Graph

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 Figure 21, there are two traces. Each trace consists of a one-dimensional array of X,Y pairs. The XY Graph also accepts two-dimensional arrays.

Exercise 7-2 Creating an XY Graph

- 1 Using the Agilent T&M Toolkit New Project Wizard, create a new Visual Basic .NET project.
- 2 From T&M Toolkit tab on the Toolbox, drag-and-drop the XY Graph onto your form and add two traces to the graph.
- 3 Add a Windows Forms Button control and double-click it.
- 4 Enter the following source code.

```

Dim i as Int16 = 0

'Array length must be divisible by 2, so these arrays are
'defined as 0 to 199 inclusive long

Dim XYCoords1(199) As Single
Dim XYCoords2(199) As Single

For i = 0 To 199
    XYCoords1(i) = Math.Sin(i) + 1
    XYCoords2(i) = Math.Cos(i) + 1.2
Next

XyTrace1.SetData(XYCoords1)
XyTrace2.SetData(XYCoords2)

XyGraph1.PlotAll()

```

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.

- 5 Run the application and double-click the button to see a graph similar to the one in Figure 21.

Complex Graph

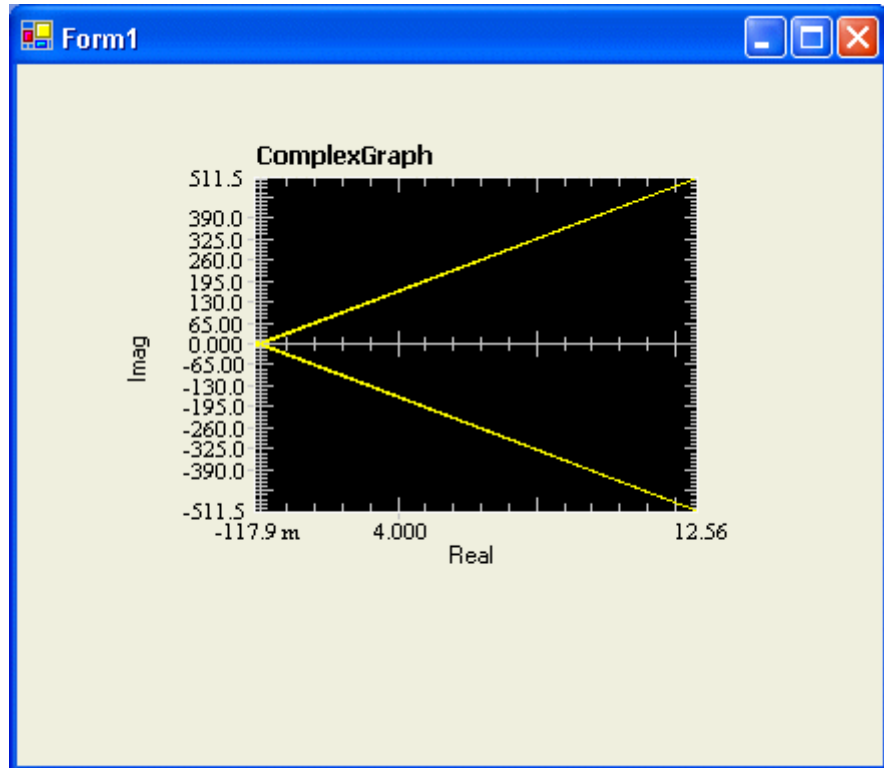


Figure 22 Complex Graph

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.

Magnitude Spectrum Graph

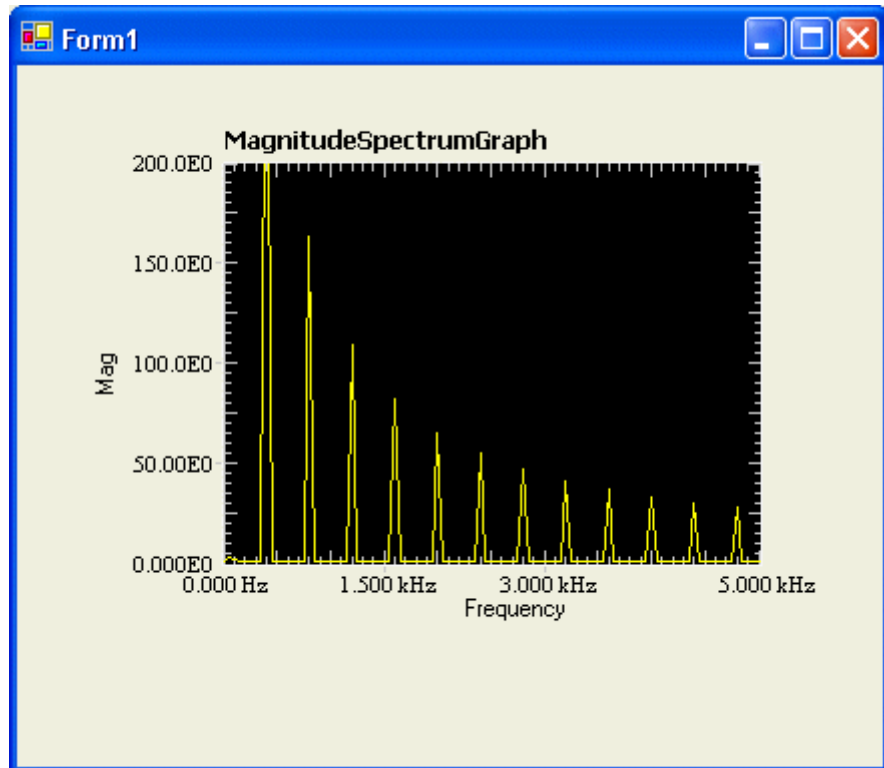


Figure 23 Magnitude Spectrum Graph

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

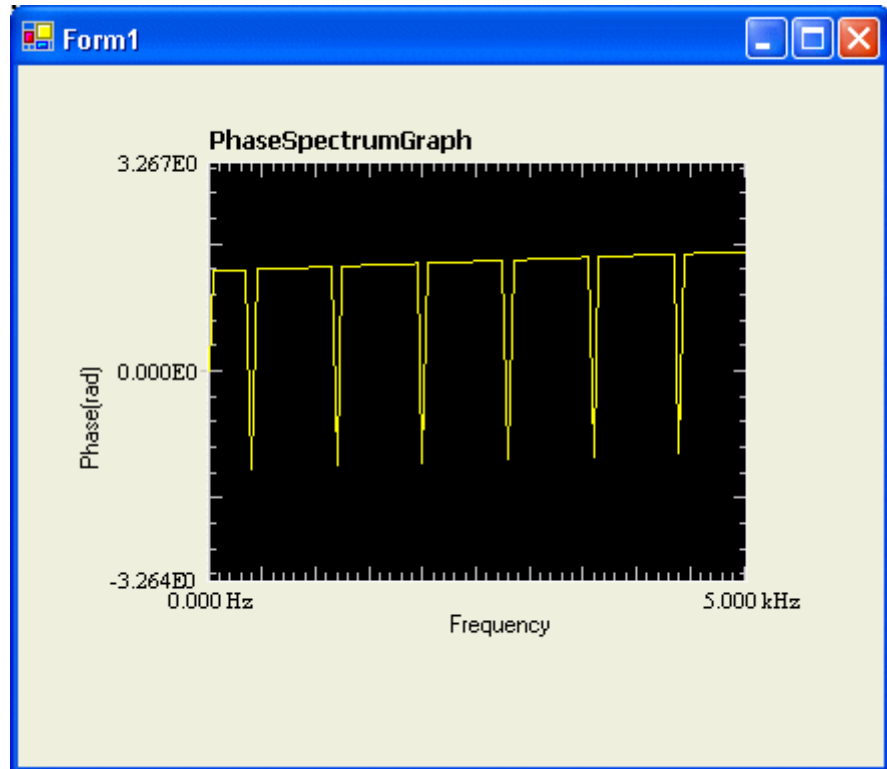


Figure 24 Phase Spectrum Graph

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.

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.

Before reviewing these categories, highlight the graph so the properties are visible and make sure the properties are sorted by category and not alphabetically. Review the following three categories:

1 The Graph Appearance Category

- The **Bottom**, **Left**, **Right**, and **Top Margin** property settings control the distance, in pixels, between the plot area and the 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.

2 The **Graph Axes Category**

- The **Gaticule** 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; and 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 **AutoScale** method is strongly recommended. The **AutoScale** method supports the **UndoZoom** feature of the context menu whereas the **AutomaticScaling** property does not.

3 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.
- The **KeyboardEnabled** property activates several handy shortcut keys for zooming and panning.
- 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.

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.

Exercise 7-3 Panning and Zooming

- 1 Open the Waveform Graph project you previously saved.
- 2 Run the application.
- 3 Using the Waveform Graph, use the keyboard by selecting ALT-RightArrow to zoom into the graphic.
- 4 Hold down the CTRL key and the left mouse key. Drag your mouse in the direction you want to pan.
- 5 Right click in the plot area and use the context menu to View Marker. View Marker is handy when you have zoomed to the point that the marker is no longer visible. View Marker moves the Marker to the left edge of your viewing area.

See Online Help for descriptions and tables of all other keyboard, mouse, and context menu shortcuts.

7 Using the 2D Graph Objects

8

Using Agilent VEE with .NET

Using the VEE Wrapper Wizard	79
How do VEE and .NET Communicate?	83
Can Legacy VEE Code be Adapted to .NET?	86

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 the VEE Wrapper Wizard

Agilent VEE is 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	<p>a From the Visual Studio .NET main menu, select T&M Toolkit.</p> <p>b From the T&M Toolkit menu, select VEE Wrapper Wizard.</p> <p>c Click Next to continue or Cancel to quit the wizard.</p>	<ul style="list-style-type: none"> • Before starting the VEE Wrapper Wizard, start VEE and open your VEE project.
2 Point the VEE Wrapper Wizard to your .vee file.	<p>a Either browse to or type in the full path to the .vee file you are using.</p> <p>b Click Next to continue, Cancel to quit the wizard, or Back to return to the previous screen.</p>	
3 The Wrapper Wizard parses the file.	Click Next to continue, Cancel to quit the wizard, or Back to return to the previous screen.	<ul style="list-style-type: none"> • 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.

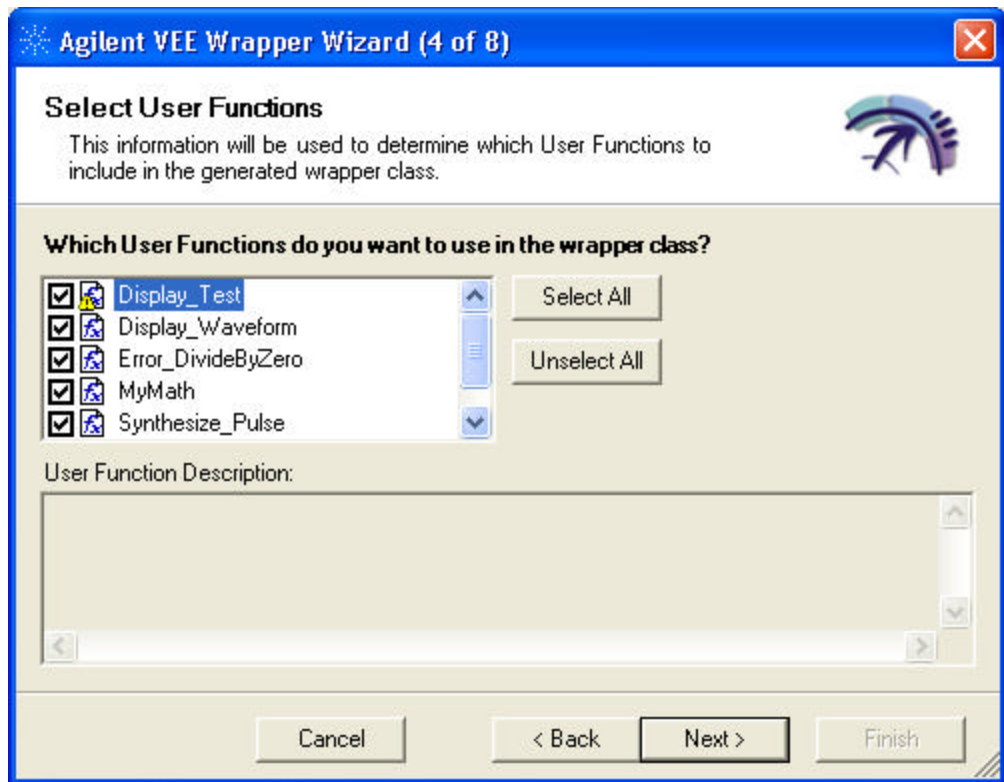


Figure 25 Selecting User Functions

Using the VEE Wrapper Wizard (continued)

Steps	Actions	Notes
4	<p>Select the User Functions to include in your VEE Wrapper.</p> <p>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.</p> <p>b Click Next to continue, Cancel to quit the wizard, or Back to return to the previous screen.</p>	<ul style="list-style-type: none"> The VEE Wrapper Wizard <i>only</i> 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.

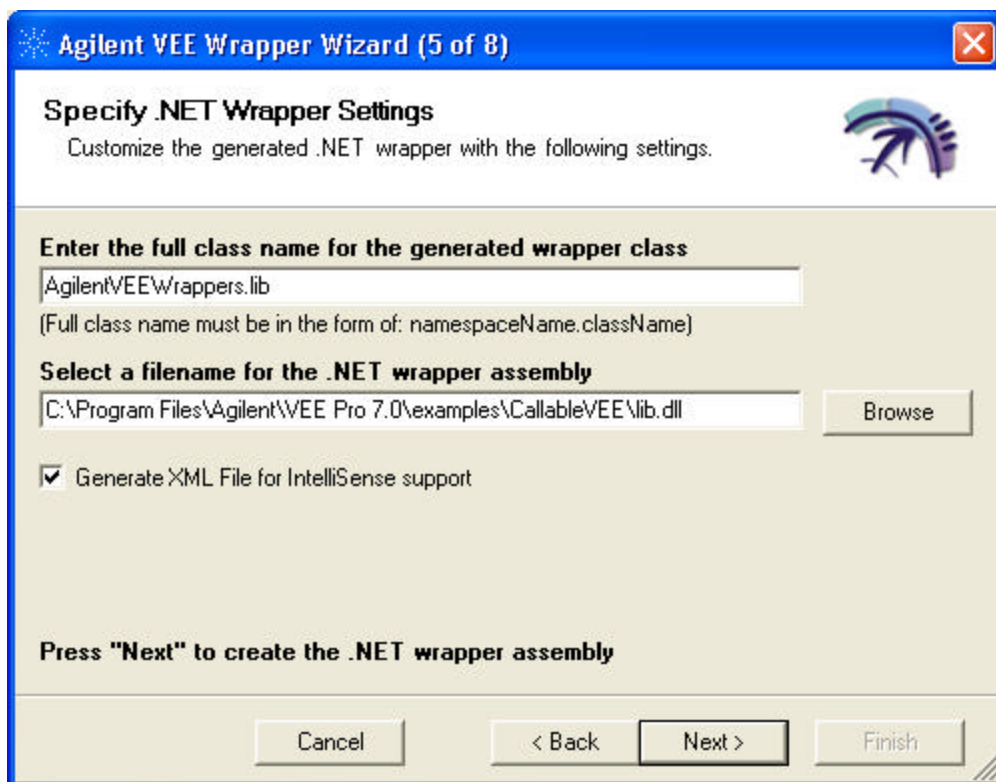


Figure 26 Specify .NET Wrapper Settings

Using the VEE Wrapper Wizard (continued)

Steps	Actions	Notes
5 Specify the .NET Wrapper settings.	<ul style="list-style-type: none"> 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.	
7 Select code generation options.	<ul style="list-style-type: none"> 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 with VEE 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
```

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 data types in intellisense as the generic type "object."

9

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

Generating Waveforms	90
The Timing Class	92
ProgressUpdater	93
The EngineeringFormatter Class	94
The Engineering Math Class	96

This chapter describes multiple small classes that provides valuable resources for your development processes.

Generating Waveforms

You need 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` has already been used several times in this manual. It produces 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 another great convenience to offer. You can drag and drop it right onto your Windows Form (see Figure 27), adjust its properties at design time, and run your program.

In Figure 28 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 can directly configure the settings for the waveform you want to generate.

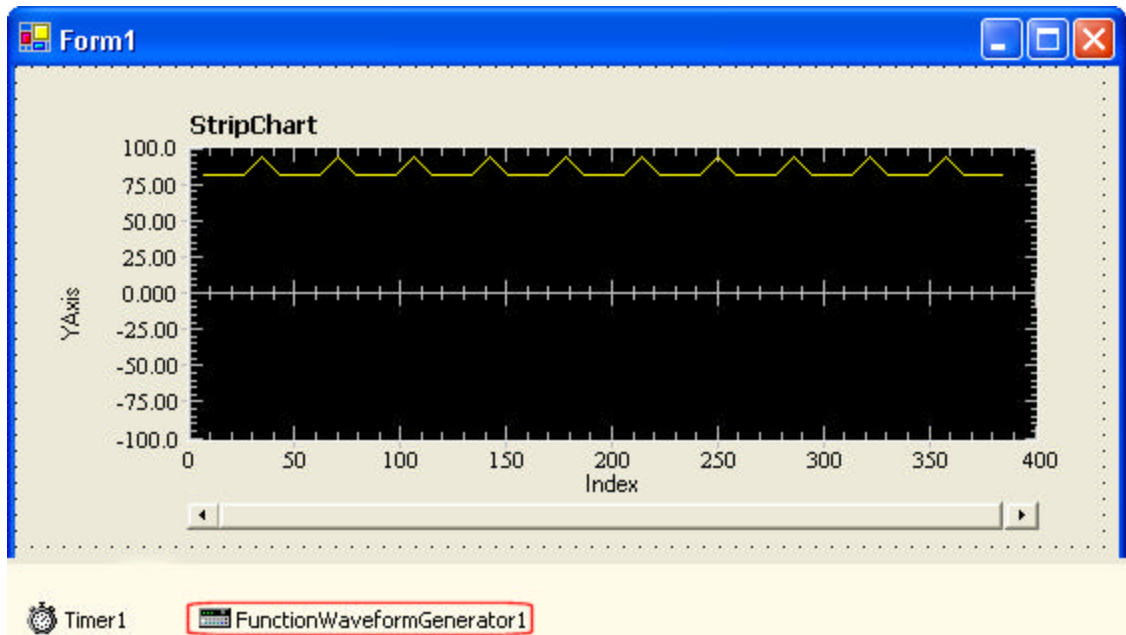


Figure 27 Function Wave Generator Placement

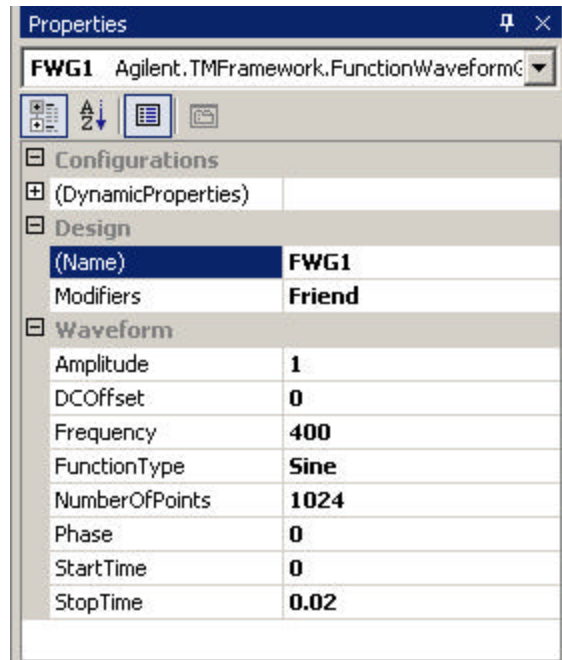


Figure 28 Properties for the FunctionWaveformGenerator

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 on this class. The simplest way to do this is to create a button and a textbox. For the button event, set the textbox equal to `Timing.HiResCounterSupported`. Run the form and click the button. The textbox displays **True** or **False** depending upon whether you have a `HiResCounterSupported` or not.

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 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 (such as k for kilo) 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 **EngineeringFormatter** 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 4 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". mt 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 providing constants and static methods for trigonometric, logarithmic, and other common mathematical functions. Due to the exceptional size of this Class, please refer to the online documentation of the members.

10

Quick Instrument Communication Using Interactive IO

How to Quickly Communicate with Instruments 98
What to Do if an Instrument Does Not Communicate 98
Interactive IO's User Interface 98

This chapter describes how to quickly communicate simple commands to an instrument, and, in the process, test connectivity with the instrument.

How to Quickly Communicate with Instruments

Agilent's 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

What to Do if an Instrument Does Not Communicate

What if the instrument being controlled fails to respond correctly, or not at all? To troubleshoot this kind of problem, you want to start with basic communication before attempting more complex interaction.

Basic communication may be nothing more than sending a few commands to the instrument and verifying the responses. In most cases, it is useful to send and receive these commands without having to create a formal software application.

Interactive IO's User Interface

The Interactive IO tool is described in Table 5 and shown in Figure 29.

Table 5 Areas of the Interactive IO Tool

Command	Type your command or query here.
History	List of commands/queries/responses sent to or read from the instrument for the current session.
Common Command Selection	A list of common commands you can quickly access.

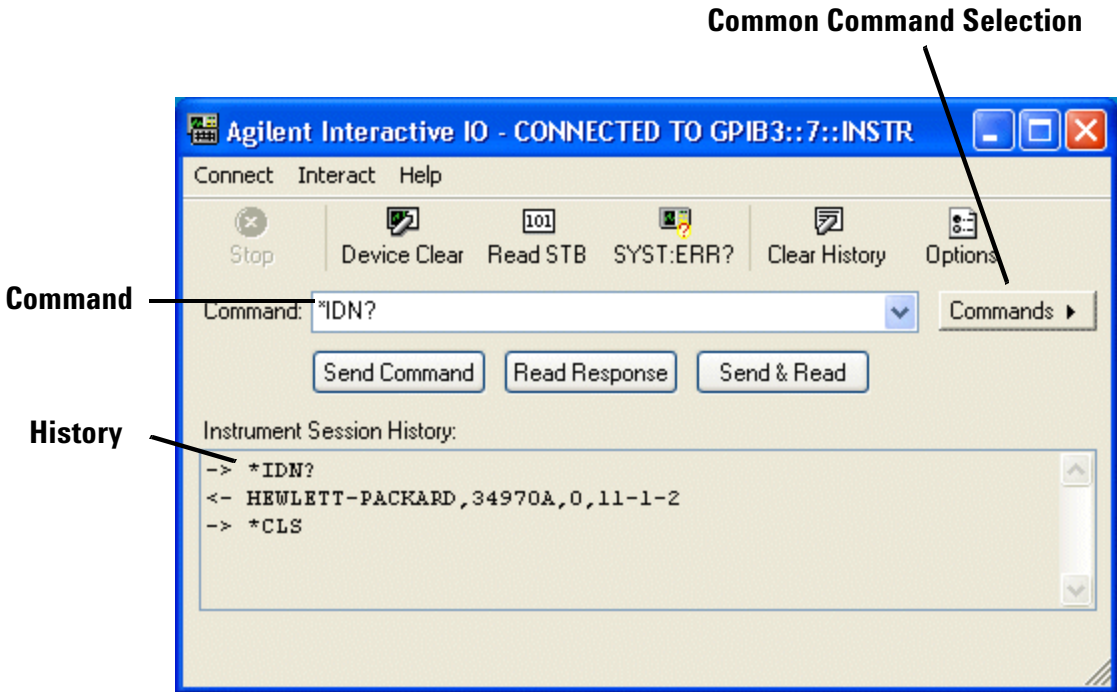


Figure 29 The Interactive IO Main Display

Exercise 10-1: Sending and Reading Simple Commands

Assumptions: You have an instrument connected.

- 1 Run the Interactive IO tool by selecting it from the T&M Toolkit menu.
- 2 Select **Connect > Connect** from the main menu. Using the table of connection strings that follows, build a matching string for the instrument. For the example, the string could be GPIB0::13::INSTR.

Table 6 Supported Connection Strings

Interface	Connection String
GPIB	GPIB[board:primary address[:secondary address]]: INSTR

10 Quick Instrument Communication Using Interactive IO

Table 6 Supported Connection Strings

Interface	Connection String
SERIAL	ASRL[<i>COM port#</i>]::INSTR
TCPIP	TCPIP[<i>board</i>]::host address[: <i>LAN device name</i>]::INSTR
TCPIP	TCPIP[<i>board</i>]::host address[: <i>port</i>]::SOCKET
USB	USB[<i>board</i>]:: <i>Manufacturer ID</i> : <i>Model Code</i> : <i>Instrument Serial Number</i> ::[<i>USB Interface Number</i>]::INSTR

Table 6 presents each supported interface's connection string. Optional items are in brackets [], and the information you provide is in italics.

USB also supports the use of aliases. Aliases are provided due to the long length of the USB connection strings. They create a transparent logical connection to a well formed USB address.

3 Type *IDN? in the Command field and choose **Send Command**.

4 Now choose **Read Command**. You should see the standard response to *IDN in the History area of Interactive IO

11

Using the IO Monitor

Why Monitor Interaction with Instruments? 102
How Can You Monitor Interaction with Instruments? 102
Getting Started Monitoring Interaction with Instruments 104

This chapter describes how to use the IO Monitor tool to trace and debug Agilent I/O layers.

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 I/O layers—can make it difficult to determine where and how problems arise. What would be helpful is a means to monitor the 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 is a standalone software tool that lets you:

- Trace the API call stack of an Agilent I/O library
- Enable and disable the viewing of selected Agilent 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 Figure 30.

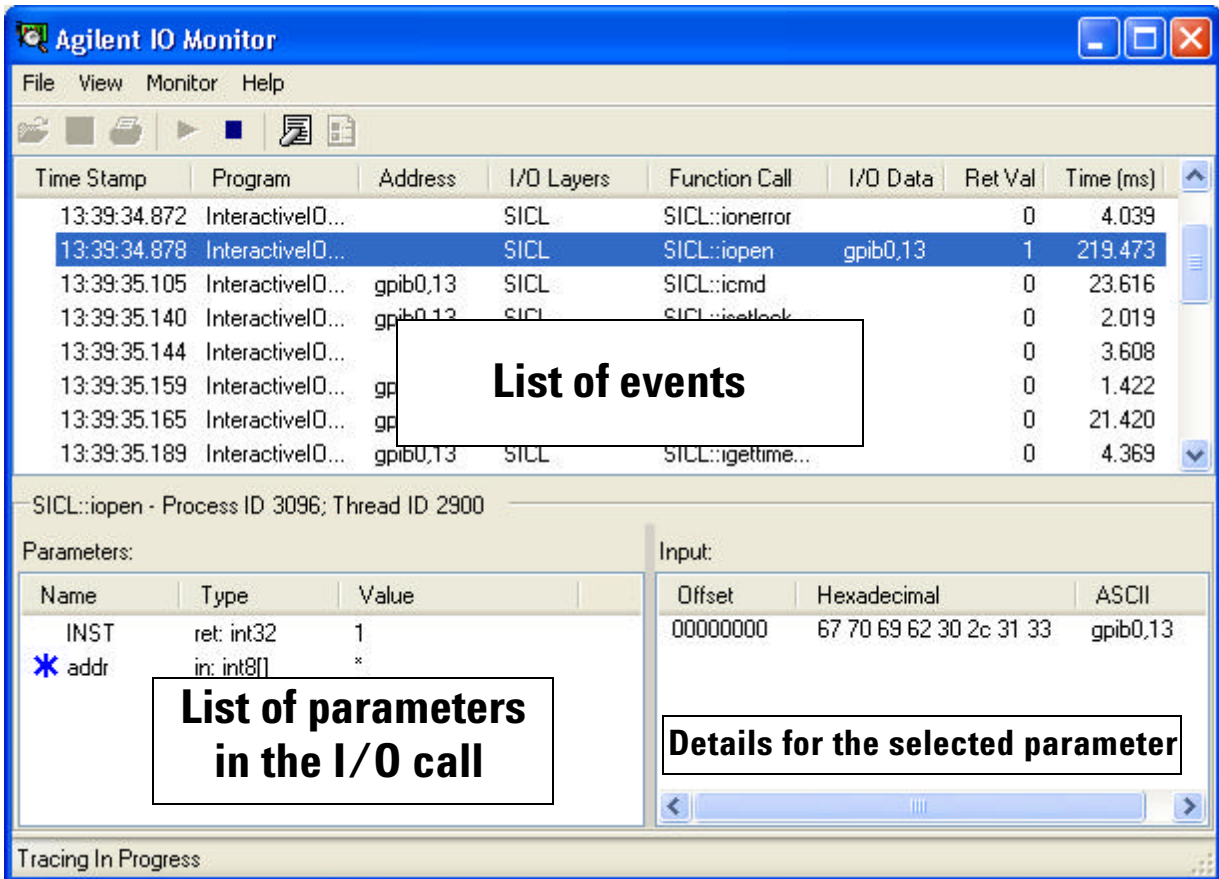


Figure 30 The IO Monitor's Main Screen

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

- Application name
- Contents of the output and input buffers

- Selectable data display format
- Elapsed time in the API

Getting Started Monitoring Interaction with Instruments

When you start 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.

Exercise 11-1 Using IO Monitor

- 1 Run the application that makes I/O calls you wish to monitor. If you don't have one, start Interactive IO. Use a dummy resource address for your instrument.
- 2 Run IO Monitor.
- 3 Tell the IO Monitor to begin monitoring I/O information by selecting **Monitor > Start**.
- 4 Start your application or issue commands from Interactive IO.
- 5 Watch each event as it is traced by IO Monitor.
- 6 Create a log of the traced events by going to **File > Log to File**. You could save the log file also and send it to a technical support person for review.
- 7 Select **View** from the main menu. Pick one of the I/O layers to trace.
- 8 Run the application and note that only the selected I/O layer is reported.

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 Figure 31 for a diagram of where the trace points are located.

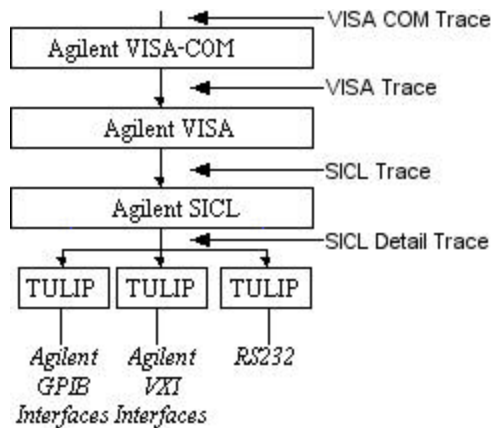


Figure 31 The Location of Trace Points in IO Monitor

- 9 The Time column associated with each I/O event shows how long (in milliseconds) the event took. Examine the Time values to determine if any 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 studying your code. After you have adjusted the code, you can rerun the I/O application and examine its events to verify any improvements.

NOTE

IO Monitor is not a performance measuring tool simply because it takes so many machine cycle just for itself. It is suggest that you use the `Agilent.TMFramework.Timing` class to time the different function calls without having the IO Monitor active.

11 Using the IO Monitor

12

Using the Driver Import Wizard

Using the Driver Import Wizard 108

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

Using the Driver Import Wizard

Expanding the Driver Possibilities

There are hundreds of existing instrument drivers. These drivers were written to meet industry standards, such as *VXIplug&play*, that existed before .NET and the T&M Toolkit. The Driver Import Wizard is for the programmer who is comfortable with *VXIplug&play* drivers and wants to quickly capture *VXIplug&play* functionality for T&M Toolkit. With the Driver Import 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 *VXIplug&play* or want to start quickly with your instruments, T&M Toolkit's Instrument Explorer offers similar functionality through the Driver Connection Wizard. The Instrument Explorer rapidly identifies your instrument configuration and provides quick communication with a variety of drivers including *VXIplug&play*. It is a great tool to use while you build your expertise with instrument control.

Which Drivers are Supported?

The Driver Import Wizard wraps *VXIplug&play* and IVI-C drivers. (IVI-COM drivers do not require wrappers.) 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 *VXIplug&play* drivers.

The *VXIplug&play* wrapper is based on the most recent version of the *VXIplug&play* driver standard. Because *VXIplug&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 Import 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.

Before beginning this exercise, you should note that there is no set procedure for the Driver Import Wizard. The steps adjust dynamically depending upon the driver you have selected. In general, the steps in this exercise conform to the usual process.

Exercise 12-1: Creating a Driver Wrapper

- 1 Using the T&M Toolkit New Project Wizard, open a Visual Basic .NET or C# project.
- 2 From the main menu, select **T&M Toolkit > Driver Import Wizard**.
- 3 Read the welcome screen explaining what you are about to do and what you are required to provide. Select **Next**.
- 4 Choose a driver for your instrument from the list of *VXIplug&play* or IVI-C drivers installed on your machine. You can also download additional drivers from the Web.
- 5 Once you have selected a driver (shown in Figure 32) you might see the message in Figure 33. If you see this message, T&M Toolkit has shipped with a wrapper for this instrument. Cancel the driver definition process and use Instrument Explorer to establish your instrument connection using this driver and the existing wrapper. If you don't see this message box, choose **Next**.

12 Using the Driver Import Wizard

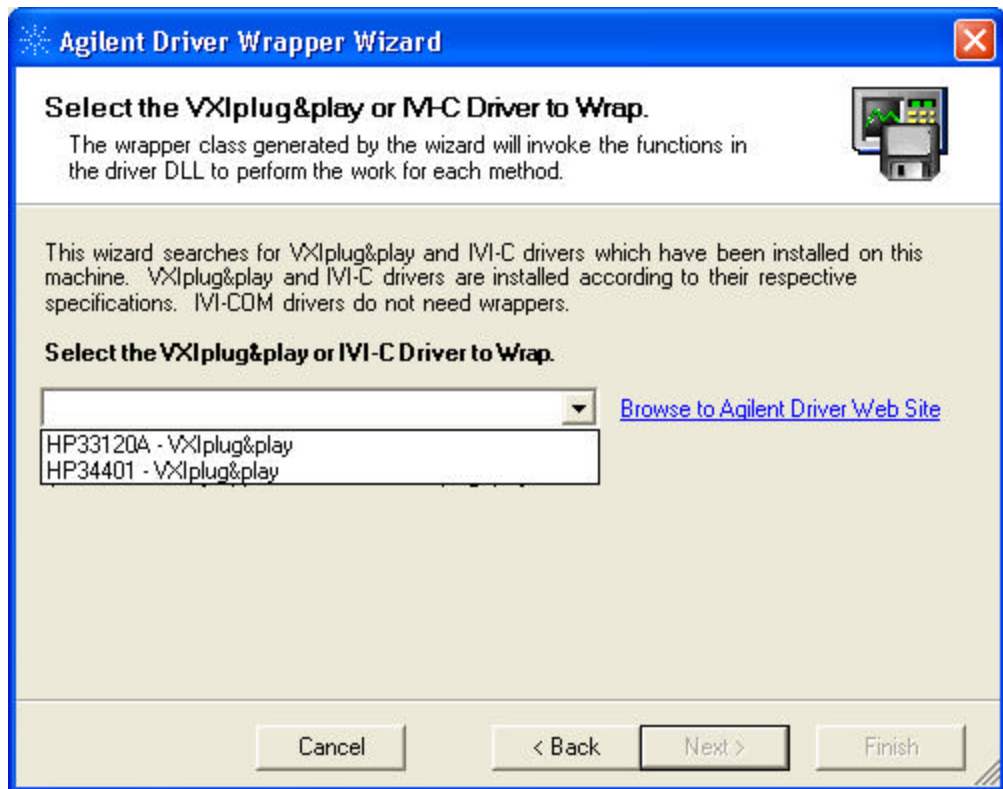


Figure 32 Selecting a VXIplug&play or IVI-C Driver to Wrap

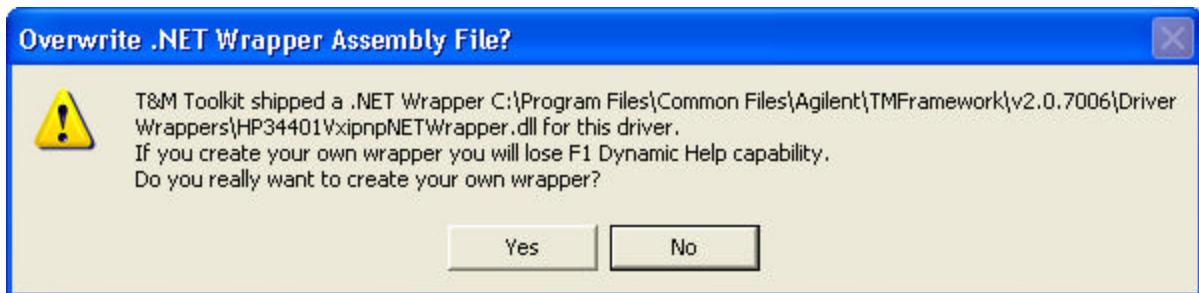


Figure 33 T&M Tooling Driver Wrapper Exists

- 6 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. Choose **Next**.
- 7 The driver is parsed and wrapped. Parse Warnings are displayed. Select the Parse Errors and Compile Errors to review any listed errors or warnings generated by the parsing or compilation. See "Troubleshooting" on page 111.
- 8 Review the screen that appears. The screen is almost identical to a screen used in Instrument Explorer's Connection Wizard. In fact, from this point on, you are following a set of screens that were established in the Instrument Explorer Connection Wizard. Choose **Next** to continue.
- 9 Review the Generate Code summary screen. The References have been chosen for you automatically and should need no editing. Choose **Finish**.
- 10 The Paste Code window is the final window of this process. Place your cursor where you want the code to be inserted in your program and select **Paste Code at Cursor**.

Troubleshooting

In Step 7 of Exercise 6-1, the Driver Import Wizard finishes parsing and compiling the wrapper and a screen similar to Figure 6 appears. Using the tabs, you can review any messages, warnings, or errors that were generated during the parse and/or compile processes.

12 Using the Driver Import Wizard

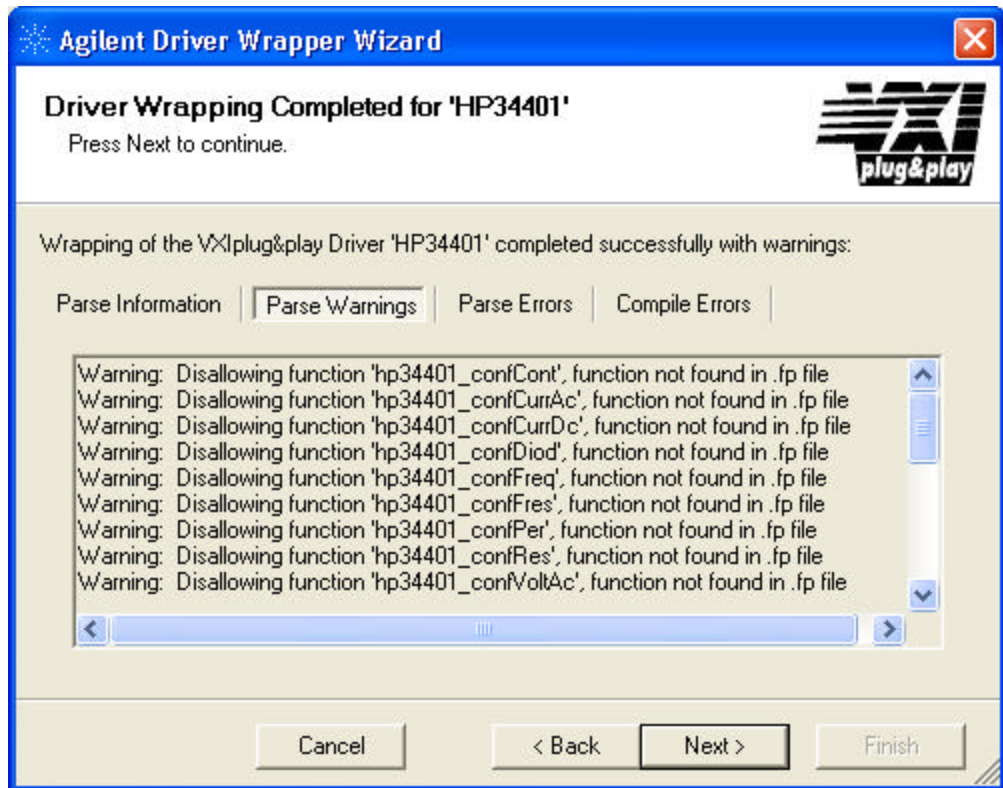


Figure 34 Troubleshooting Wrapper Creation

The Driver Import 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 Import 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 Import Wizard will 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 Import 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 Import 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 Figure 34 on page 112. 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 Import 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.

12 Using the Driver Import Wizard

13

Product Support

Online Help	116
Dynamic Help	118
F1 Help	120
The Agilent Developer Network	121
Support Included with All T&M Toolkit Products	122
Where You can Find Support	122
User Communities	122

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

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 Figure 35 on page 117.

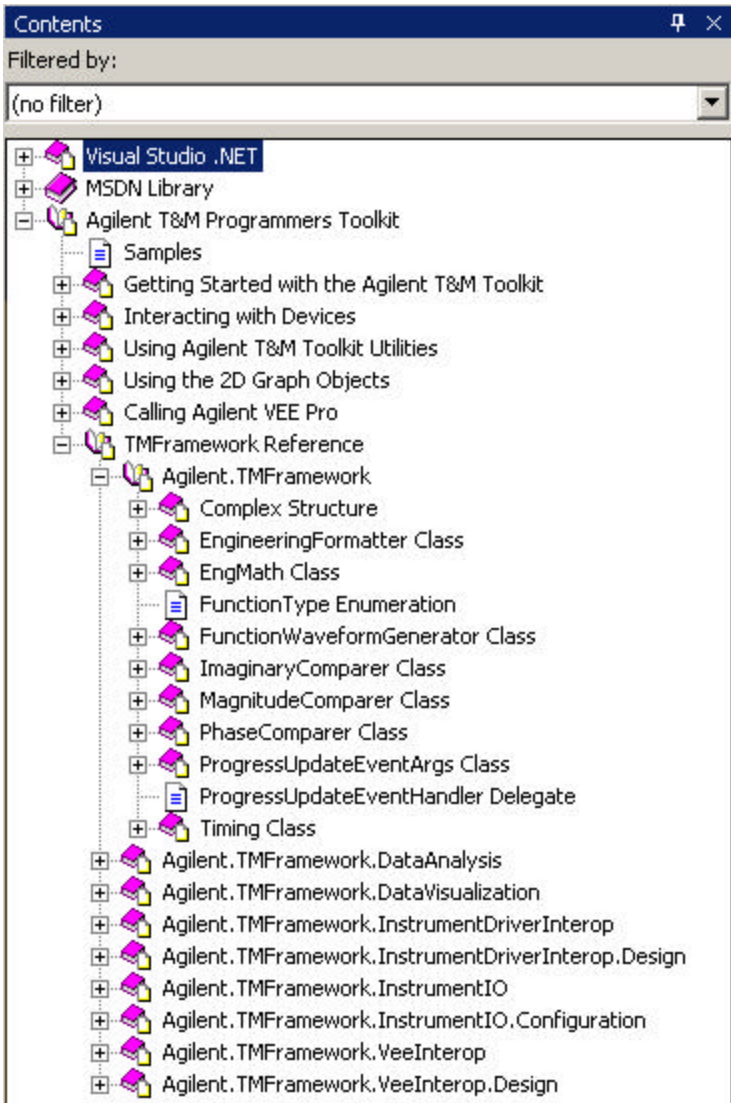


Figure 35 Agilent T&M Toolkit Help Contents

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 Figure 36 on page 119, 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.

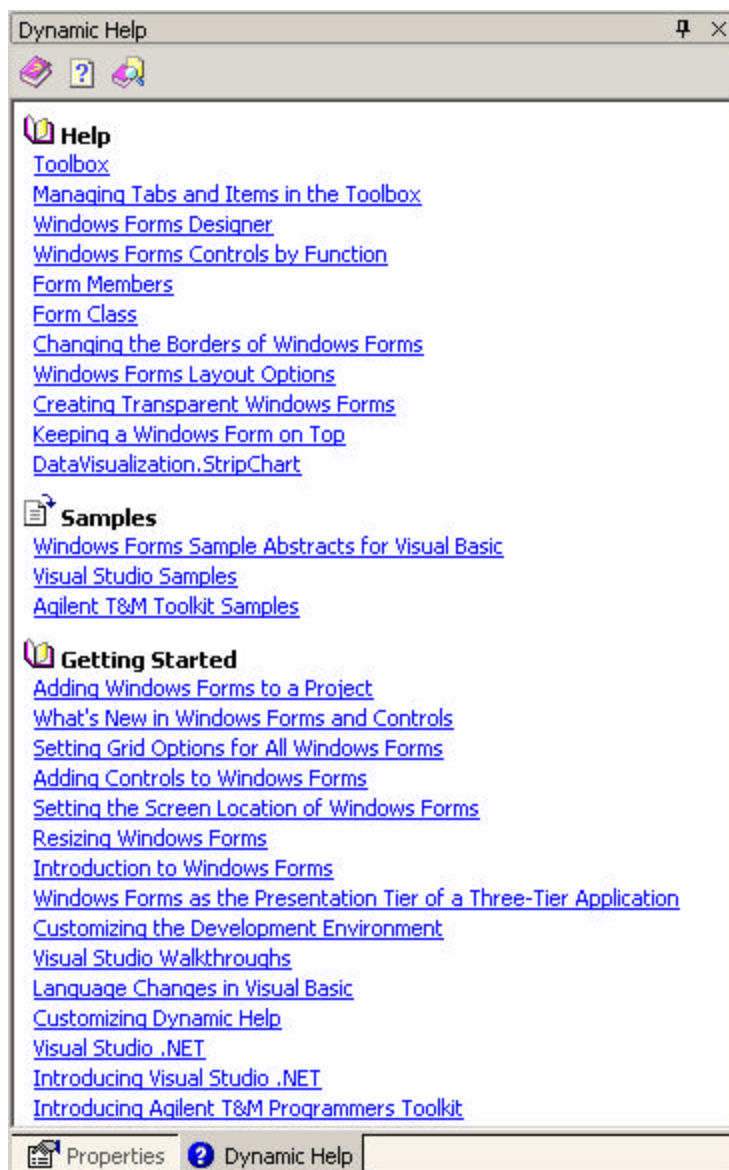


Figure 36 Visual Studio .NET Dynamic Help

F1 Help

Agilent T&M Toolkit supports F1 help. This means you can highlight the word or control and press F1 and, if a help topic exists for the control, word, phrase, etc., the help topic is displayed. See Figure 37.



Figure 37 With XYGraph Active, F1 is Pressed

The 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 access ADN at <http://www.agilent.com/find/adn>.

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.com/find/assist>.

User Communities

Agilent VEE has a large and very active group of supporters who share information and support each other. You can participate in this group by selecting VEE Help > Agilent VEE on the Web > VEE Users Group (vrf) from the main menu.

A
adapting legacy VEE code
for .NET 86

adding
a new trace 63

ADN 121

Agilent Developer Network
121

arrays
X,Y pairs 71

B

Bessel function

I0 58

J0 58

J1 58

modified version of the first
kind 58

of the first kind 58

of the second kind 58

Y0 58

Y1 58

C

collection editor

markers 65

traces 64

complex

converting to other data
types 48

imaginary comparer 49

magnitude comparer 49

not native to .NET 48

phase comparer 50

Complex graph 72

context menu 77

creating a new project 26

creating traces 63

D

default trace 62

diagnosing instrument con-
nection problems 98

DirectIO

buffer management 29
converting instrument
formats to data types 28
defined 28
instrument session
management

API methods 31

API properties 31

instrument sessions 29

DirectIO session 20

discovering instruments 14

discrete mathematics 58

beta 58

binomial 58

combination 58

factorial 58

gamma 58

permutation 58

displaying I/O traces 103

drag and drop

an instrument session 22

driver session 20

Driver Wrapper Wizard

checking for standards

compliance 113

reviewing any parse or

compile errors 113

steps to create a wrapper
109

supported driver standards
108

supported drivers 108

troubleshooting 111

dynamic help 118

graphic 119

E

engineering formatter

format specifiers 95

general format 94

significant digits 94

examples

accessing the DirectIO.Gpib
49

engineering number
formatting 96
function waveform
generator properties 92
using the high resolution
timer 93

exponential regression 59

exponents 94

F

F1 help 120

Fast Fourier Transform 53

FFT and IFFT

performance 53

formatting numbers 94

Fourier 53

G

generating code 21

graph

appearance 75

a marker legend 75

a trace legend 76

axes 76

grid 76

margins 75

plot area 75

ticks 76

title 75

min/max values 76

smoothing mode 75

H

hardware requirements 5

help

dynamic 118

F1 120

online 115

technical support 122

high resolution counters 92

I

I/O streams

SICL 104

VISA 104

- VISA-COM 104
- IO in the Bessel function 58
- imaginary comparer 49
- installing
 - how to 4
- instrument configuration
 - deleting from 22
- Instrument Explorer
 - and RS-232 17
 - defined 14
 - generating code 21
 - instrument discovery 14
 - key tasks of 14
- Instrument Session Wizard 20
 - abbreviated version 22
 - generating code 21
- instrument sessions
 - adding or creating 20
 - managing 31
- Interactive IO
 - main display 99
 - what it does 98
- Inverse Fast Fourier Transform 53
- IO Config
 - running 10
- IO Monitor
 - displaying results 103
 - events 103
 - I/O streams monitored 104
 - improving I/O performance 105
 - trace points 105
- IVI-C
 - using in .NET 108
- IVIplug&play
 - using in .NET 108
- J**
- J0 in the Bessel function 58
- J1 in the Bessel function 58

- K**
- keyboard shortcuts
 - enabling 76
- L**
- linear regression 59
- logarithmic regression 59
- M**
- magnitude comparer 49
- Magnitude Spectrum graph 73
- managing instrument configurations 22
- markers
 - linking 73
- mouse shortcuts
 - enabling 76
- N**
- number formats 94
- number formatting specifiers 95
- O**
- online help
 - T&M Toolkit 117
- online help, about 115
- P**
- panning 77
- phase comparer 50
- Phase Spectrum graph 74
- PlotAll method 67
- power curve regression 59
- progress updater 93
 - how to use 93
 - properties 93
- R**
- regression
 - exponential 59
 - linear 59
 - logarithmic 59
 - power curve 59

- running IO Config 10
- S**
- SetData method 67
- setting and plotting data 66
- smoothing mode 75
- software requirements 7
- Strip Chart 69
 - setting the timing interval 69
- support 122
- system requirements
 - hardware requirements 4
 - software requirements 7
- T**
- T&M Toolkit
 - defined 2
 - installing 4
 - reasons to use 2
 - uninstalling 4
 - what it contains 2
- technical support
 - all products 122
- timing
 - delay 93
 - limits of delay 93
 - precision of 92
 - progress updater 93
- traces
 - collection editor 64
 - creating 63
 - default trace 62
 - full description 62
 - introduction to 68
 - representations 65
 - setting properties 64
- troubleshooting
 - driver wrappers 111
- U**
- uninstalling 4
- using existing drivers 108

V

VEE

- adapting legacy code for .NET 86
- communication basics 84
- configuring the callable VEE server 84
- verifying your communication configuration 86

VEE Wrapper Wizard

- avoiding naming problems 86
- how to use 80
- I/O configuration concerns for .NET 85
- what can be read by .NET 86

W

windowing functions

- data set comparison cross correlate 57

wizards

- Instrument Session 20
- the Driver Wrapper Wizard 108
- the New Project Wizard 26
- the VEE Wrapper Wizard 79

X

X,Y data and arrays 71

XY graph 70

Y

Y graph 68

Y0 in the Bessel function 58

Y1 in the Bessel function 58

Z

zooming 77