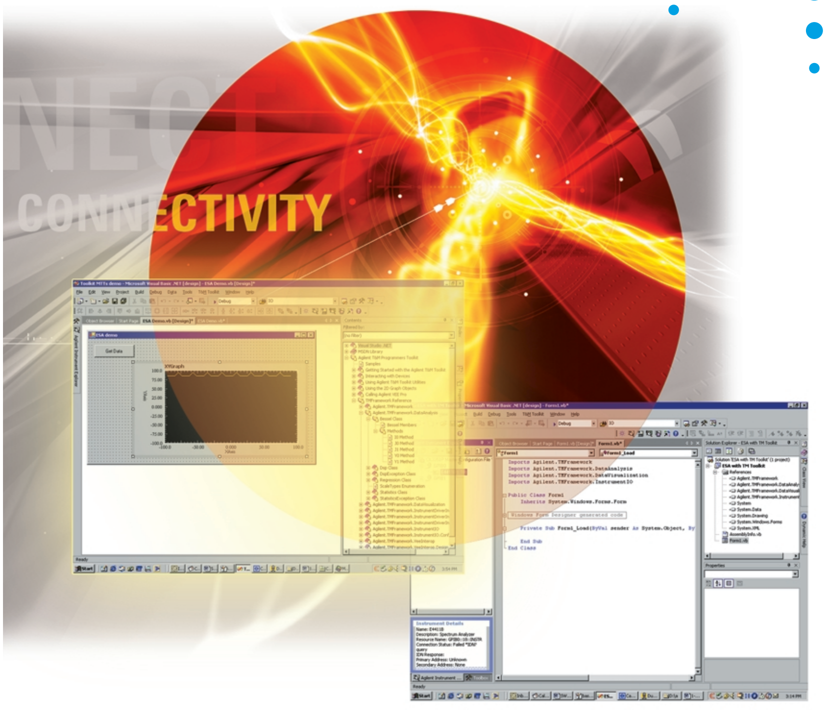
Application Note 1409-2



Agilent T&M Programmers Toolkit

Simplified Instrument
Communication and Programming
Using Textual Programming Languages

Agilent Technologies

Table of Contents

Introduction

At some point, most R&D, manufacturing and test engineers need to program and control test instruments and communicate with them from a PC or laptop. Making that happen should be — but rarely is — a simple task. If you are like most engineers, you have wasted a lot of time and effort dealing with driver issues, writing code, and debugging programs, all in a simple quest to make your test instruments talk to each other and to your PC.

There are some new solutions available to save you time with these connectivity issues so you can spend more time on more productive tasks. One of those solutions is Agilent's T&M Programmers Toolkit. The T&M Programmers Toolkit puts a test-and-measurement "face" on top of a powerful development environment—Microsoft® Visual Studio .NET.

The purpose of this paper is to help you understand how the T&M Programmers Toolkit can help you talk to your instruments more easily, using textual programming languages you already know.

What is T&M Programmers Toolkit?

The T&M Programmers Toolkit brings the benefits of Microsoft's Visual Studio .NET to test and measurement engineers. Visual Studio .NET is Microsoft's latest version of its Visual Studio development environment. The T&M Toolkit extends the Visual Studio .NET platform with integrated, easy-to-use software tools and components — project wizards, class libraries, 2D graph controls and more — to save you time and make it easier to write code for automating measurement tasks and displaying data. Sophisticated instrument-management, bus-monitoring and code-generation wizards eliminate the headaches associated with the traditional methods for connecting to and controlling instruments.

The T&M Toolkit offers many of the test and measurement productivity advantages historically associated with stand-alone proprietary T&M environments. It is completely integrated into Visual Studio .NET to give you easy access to the productivity and flexibility that Microsoft has built into that environment.

The T&M Toolkit lets you easily pull legacy or current code and drivers forward into this open, standard environment. The Toolkit also lets you quickly identify and debug I/O issues during development, a task that has been almost impossible in the past. And to make sure you get up to speed quickly, a fully integrated, online help system shortens your programming learning curve. Using the Toolkit, you can connect to your test instruments and accomplish your instrument programming tasks in a fraction of the time it formerly took.

Why Visual Studio .NET is so powerful

Visual Studio .NET, Microsoft's latest version of its Visual Studio development toolset, provides an integrated development environment for programmers working with a variety of languages, including C#, C++ and Visual Basic. It offers several advantages that help will you get your job done faster:

- **Eliminate mundane programming** — Visual Studio .NET saves you time by eliminating mundane programming tasks. Visual Studio .NET has an extensive support library (the .NET Framework), which means you write less administrative code, giving you more time to spend on test-and-measurement-specific code. The .NET Framework essentially replaces the C-runtime libraries, Visual Basic Forms libraries, MFC, ATL, and ASP development libraries. The .NET Framework is a single, well-designed class library that gives you the foundation you need and frees you from the mundane tasks of dealing with these libraries on your own.

- **Mix and match languages** — Visual Studio .NET enables multiple development languages to be used together in a single integrated development environment (IDE). Objects and components from all first-class languages in Visual Studio.NET can be used seamlessly together, so you can mix and match languages based on your legacy environment, your expertise and the requirements of your particular task. This is because of the enabling technology of the .NET environment and the common language runtime (CLR) from Microsoft.
- **Easily reuse code** — Visual Studio .NET also gives you the ability to easily reuse code across projects and across programming languages — so you spend less time developing code from scratch. This is possible because Visual Studio .NET provides true object-oriented programming.

As an extension of the powerful built-in functionality available in the .NET Framework, the T&M Toolkit simplifies programmatic access to your instruments and speeds your programming. The T&M Toolkit automatically generates code for you, and Visual Studio .NET functionality integrated into T&M Toolkit — such as drag and drop — makes many instrument communication and measurement automation tasks faster and easier.

Standard versus proprietary languages

Today, a majority of electrical engineers who need to program use standard programming languages, and a large majority of them use Microsoft Visual Studio. Even if you use a proprietary test and measurement language or environment, such as National Instruments LabVIEW or Agilent VEE, you probably use Visual Basic or Visual C++ too. You need to use multiple environments because it is difficult to accomplish everything you need to do in a single programming environment.

There are some distinct advantages to working in a standard language instead of a proprietary language. The skills you develop in a standard language are more portable and give you more flexibility for moving into different work environments. Because more people use standard languages, managers can draw from a larger pool of developers and protect their investment in their intellectual property. The larger developer pool also means you have more places to turn when you need questions answered. Visual Studio .NET is an industry-standard development environment that gives you these advantages. Moving to the open .NET environment allows you to leverage the huge investment Microsoft has made in this platform and also lets you take advantage of the add-ons being built by numerous companies.

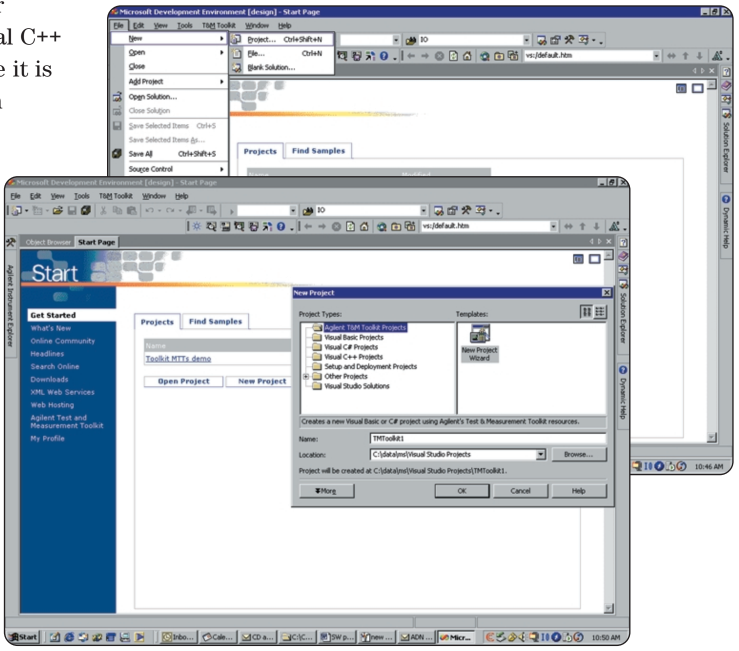
Step-by-step guide to simplifying your tasks with the T&M Toolkit

T&M Toolkit has a variety of features designed to make it easy for you to communicate with and configure your test instruments and test systems, and to acquire, analyze and display data. The Toolkit also makes it easy for you to debug programs and get help when you need it. To show how the T&M Toolkit is integrated into Visual Studio .NET and demonstrate how easy it is to use, let's walk through the steps for building a simple program. In our example, we connect to an ESA spectrum analyzer, send a command requesting a string of data, and then read/display the data.

STEP 1: Get started

The easiest way to get started is to use the T&M Toolkit "New Project Wizard." Project wizards provide instrument and system control program building blocks, so you don't have to start from scratch when you write instrument control applications. Building-block code is automatically generated for either C# or Visual Basic .NET.

From the start screen or the File menu, choose **New > Project.**



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

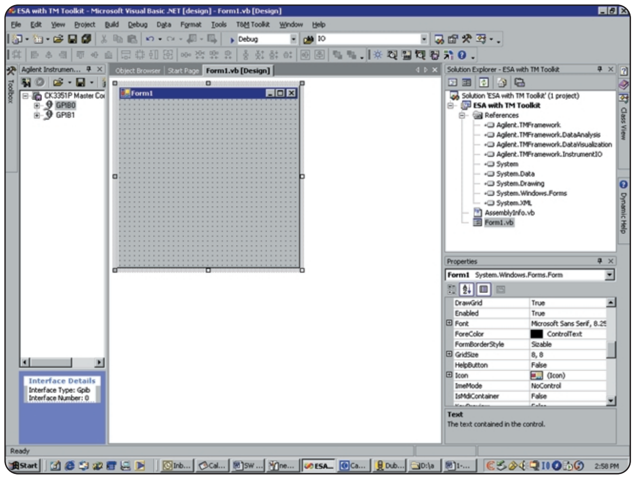
Name: TMToolkit1

Location: C:\data\ms\Visual Studio Projects

Project will be created at C:\data\ms\Visual Studio Projects\TMToolkit1.

Fig. 1. ***The T&M Toolkit project wizard guides you through available options to create the framework of your application.***

The New Project Wizard creates an empty application and Visual Basic form (Figure 2) — with all of the right import statements and references included — in which you can start developing. Dialog boxes prompt you to make the necessary choices and lead you through the steps of creating a project.


ESA with TM Toolkit - Microsoft Visual Basic .NET [design] - Form1.vb [Design]
Form1
Solution Explorer - ESA with TM Toolkit
Agilent.TMFramework
Agilent.TMFramework.DataAnalysis
Agilent.TMFramework.DataVisualization
Agilent.TMFramework.InstrumentIO
Form1.vb
Properties

Interface Details
Interface Type: Gpib
Interface Number: 0

Fig. 2. ***The T&M Toolkit project wizard has automatically placed the correct calls and references within the project.***

After we create the form (Form1) for our application, we need to create the visual design of the application by creating a button to initiate our measurement action. To create a button, select **Toolbox > Windows Forms > Button** and drag it onto the Form1 window (Figure 3). (If the Toolbox is not present on the left hand side of your screen, select **View > Toolbox** from the main menu.) Select **Text** in the Button1 properties list and type “Get data.”

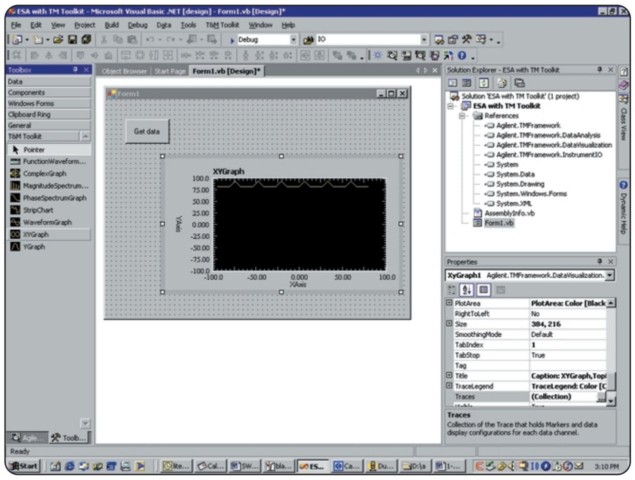


Fig. 3. ***In the form, the user interface and output are in visual layout.***

To create the output area for your application, select **Toolbox > T&M Toolkit > YGraph** (if the T&M Toolkit tab is not viewable, go to the help system for information on how to create this tab). Drag and drop the YGraph object onto Form1.

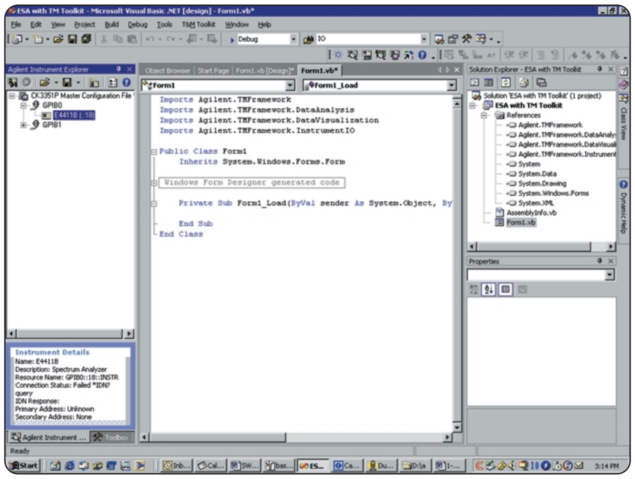
Now you have the basic structure of your program in place.

STEP 2: Configure your interface and validate your PC-instrument connection

For this example, we will use the Agilent 82357A USB/GPIB converter. Plug the converter into the computer and click to accept the automatic configuration settings.

Before you can write the actual code, you need instrument connection information. The T&M Toolkit *Instrument Explorer* helps you discover and identify the instruments available to the computer on the different I/O buses like USB, GPIB, VXI and LAN.

To find connected instruments, go to the T&M Toolkit menu and choose the Instrument Explorer (Figure 4). The Instrument Explorer identifies test instruments from any vendor on the various interfaces and provides information about them. For this example, we will use the Agilent E4411B Spectrum Analyzer at GPIB address 18.


ESA with TM Toolkit - Microsoft Visual Basic .NET [design] - Form1.vb*
Agilent Instrument Explorer
CK3351P Master Configuration File
GPIB0
E4411B (:18)
GPIB1
Form1
Form1_Load
Solution Explorer - ESA with TM Toolkit

```
Imports Agilent.TMFramework
Imports Agilent.TMFramework.DataAnalysis
Imports Agilent.TMFramework.DataVisualization
Imports Agilent.TMFramework.InstrumentIO

Public Class Form1
    Inherits System.Windows.Forms.Form

 Windows Form Designer generated code 

    Private Sub Form1_Load(ByVal sender As System.Object, By

    End Sub
End Class
```

Instrument Details
Name: E4411B
Description: Spectrum Analyzer
Resource Name: GPIB0::18::INSTR
Connection Status: Failed *IDN? query
IDN Response:
Primary Address: Unknown
Secondary Address: None

Fig. 4. ***Instrument explorer searches and displays instruments connected to your PC.***

After using Instrument Explorer to find the instrument, we know to use address GPIB 18 with the code we are about to write.

STEP 3:
Communicate with your instrument

There are several different ways you can communicate with your instrument. One option is to use an instrument driver that is written for a specific instrument, such as a VXI*plug&play* instrument driver or an interchangeable virtual instruments IVI-COM or IVI-C instrument driver. Or you can program the instrument – with the T&M Toolkit's DirectIO object – using the instrument's native command set by sending it standard commands for programmable instrumentation (SCPI) strings. The sidebar (right) describes each of these methods and discusses how the T&M Toolkit helps you complete your task.

Since our example involves a small and very specific measurement, we will use DirectIO.

Configure the instrument by creating an "instrument session." Right click on the instrument from the Instrument Explorer and walk through the wizard steps presented. Template code is generated for each instrument based on the parameters you enter in the instrument session. You can just drag and drop an instrument session icon into your work window to generate code to connect with the instrument.

STEP 4:
Program your instrument

Next, we will create an application that will get data from the instrument, and create a YGraph to plot the returned data points.

First we need to create an I/O object for the instrument. You can use the drag and drop functionality as described in Step 3 or you can right click on the instrument icon in instrument explorer and paste the code into your code window. The code will read:

```
myhpesa = New DirectIO("GPIB1::18::INSTR")
```

As you are working in the code window, you will notice that the T&M Toolkit suggests alternatives to end code lines. The Toolkit takes advantage of Microsoft's IntelliSense to help you write code.

Next we format the trace data. Type in:

```
'format trace data to ASCII
myhpesa.WriteLine("form:data ASC")
```

Instrument Communication Alternatives

The Agilent T&M Programmers Toolkit allows you to communicate with the instrument either directly through Toolkit DirectIO or with a vendor-supplied driver. DirectIO is an easy-to-use I/O interface to directly program the instrument with its built-in commands (for example, SCPI commands). The vendor supplied drivers for specific instruments provide libraries that are customized to a particular instrument so you can program the instrument using conventional programmatic approaches instead of sending strings to control the instrument. Toolkit directly supports the IVI-COM, VXI*plug&play*, and IVI-C driver standards.

Generally, a vendor supplies a single driver with their instrument, so you will use this if it meets your needs. DirectIO is primarily used when there is no instrument-specific driver, when the instrument-specific drivers do not meet your needs, or when it's the most convenient.

The following driver standards are supported by Toolkit:

- **IVI-COM:** These drivers provide an easy way to communicate with your instruments that takes advantage of the new technologies in Microsoft application development environments. IVI-COM presents the driver as a COM object. With it, you get a more robust driver with all the benefits of the development environment. The IVI-COM drivers are well supported in the .NET environment. COM interoperation in .NET is handled well because it is a much more native .NET component than either VXI*plug&play* or IVI-C. This means that the driver will have full capabilities for each base class, as well as for the customized portion of the drivers.
- **IVI-C:** This is a C-based instrument driver. With the T&M Toolkit, you can use the .NET Wrapper Wizard to generate a .NET object around a IVI-C driver. This is actually the same wizard used for VXI*plug&play* drivers.
- **VXI*plug&play*:** With the T&M Programmers Toolkit, you can use the VXI*plug&play* .NET Wrapper Wizard to automatically create a .NET-friendly wrapper object around existing VXI*plug&play* instrument drivers. This improves the usability of the drivers in .NET by exposing the C-style interface as a .NET object.

Then, to return the trace data from the instrument, we type in:

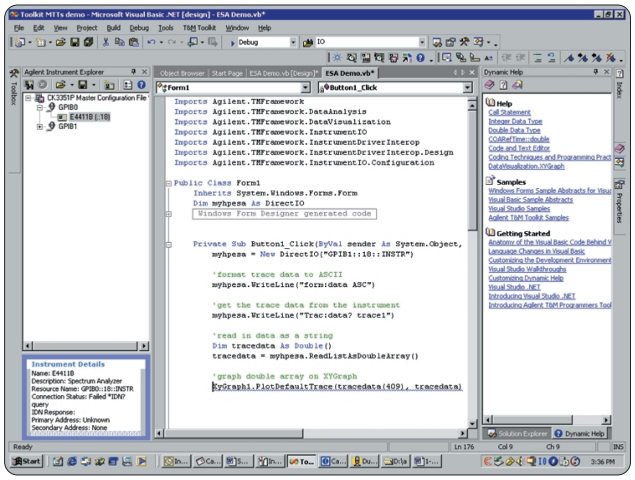
```
myhpesa.WriteLine("Trac:data? trace1")
```

In order to read the data, we need to create a double array and then read the data into that array:

```
'read in data as a string
Dim tracedata As Double()
tracedata = myhpesa.ReadListAsDoubleArray()
```

There are many options for displaying data in the T&M Toolkit. The data visualization namespace has class libraries available for mouse-based zoom capabilities, full color control, size, position, various chart types, multiple markers, various axes, grid display styles and more. For this example, we will simply graph the data on a YGraph plot. To do so, we type in:

```
'graph double array on YGraph'
YGraph.PlotDefaultTrace(tracedata)
```



```
Imports Agilent.TMFramework
Imports Agilent.TMFramework.DataAnalysis
Imports Agilent.TMFramework.DataVisualization
Imports Agilent.TMFramework.InstrumentIO
Imports Agilent.TMFramework.InstrumentDriverInterop
Imports Agilent.TMFramework.InstrumentDriverInterop.Design
Imports Agilent.TMFramework.InstrumentIO.Configuration

Public Class Form1
    Inherits System.Windows.Forms.Form
    Dim myhpesa As DirectIO
    Windows Form Designer generated code


    Private Sub Button1_Click(ByVal sender As System.Object,
        myhpesa = New DirectIO("GPIB1::18::INSTR")

        'format trace data to ASCII
        myhpesa.WriteLine("form:data ASC")

        'get the trace data from the instrument
        myhpesa.WriteLine("Trac:data? trace1")

        'read in data as a string
        Dim tracedata As Double()
        tracedata = myhpesa.ReadListAsDoubleArray()

        'graph double array on XYGraph
        XyGraph1.PlotDefaultTrace(tracedata(409), tracedata)
```

Instrument Details
Name: E4411B
Description: Spectrum Analyzer
Resource Name: GPIB0::18::INSTR
Connection Status: Failed *IDN? query
IDN Response:
Primary Address: Unknown
Secondary Address: None

Fig. 5. ***Code has been added behind the user interface.***

Now we will build and run the code from within Visual Studio. Just press the Run button, which is a triangle on the toolbar pointing to the right.

To send our commands and receive the data, press the "Get data" button in the application you wrote.

This completes the programming portion. Steps 5 and 6 mention features of the T&M Toolkit.

STEP 5:
Debug I/O problems and performance

The T&M Toolkit has an IO Monitor you can use as a diagnostic tool for debugging I/O-related problems. It provides a listing of the activity in several different I/O layers—just select the one you want to see. You can use the IO Monitor (Figure 6) for debugging, to see I/O "traffic" happen in each I/O layer, and to improve performance by identifying process bottlenecks. To turn the IO Monitor on, go to the T&M Toolkit menu and select **IO Monitor.**

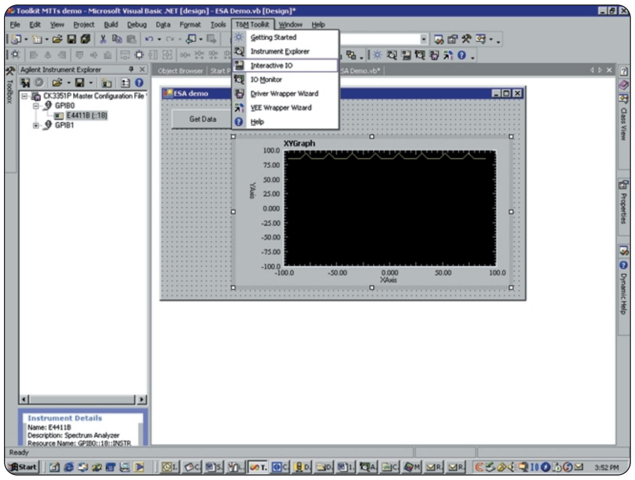


Fig. 6. ***The IO Monitor keeps a record that can be used real time, saved to a file or to sent to a colleague for collaboration.***

STEP 6:
Analyze data

In the case of our simple example, analysis is unnecessary. However, the T&M Toolkit does provide tools to support analysis, such as curve fitting routines and signal analysis routines (FFT). It also includes extensive mathematical and statistical libraries, including signal processing, trigonometry, matrix functions, and more. (Figure 7.)

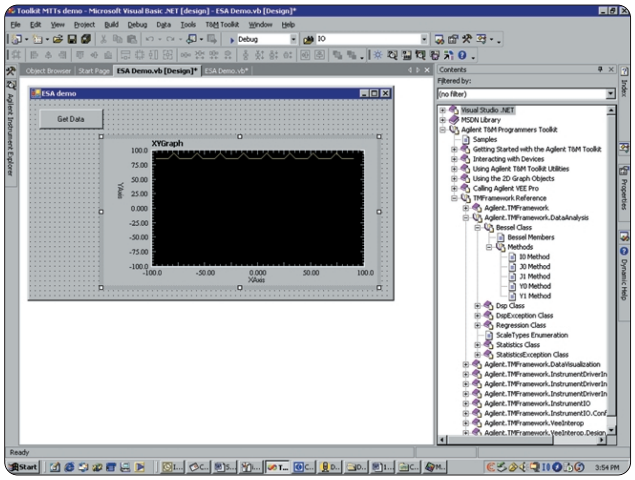
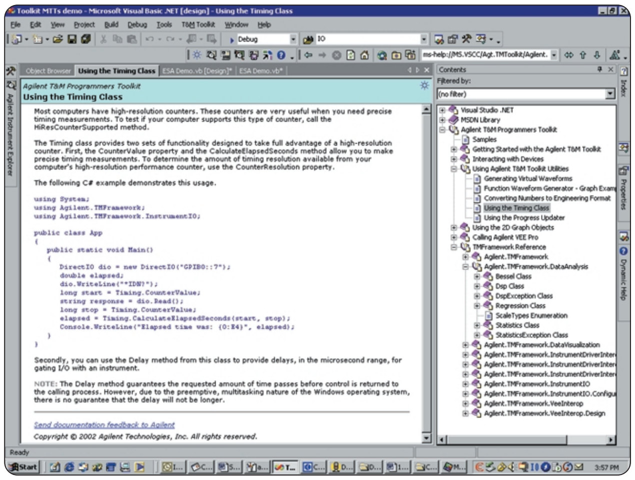

Toolkit MTTs demo - Microsoft Visual Basic .NET [design] - ESA Demo.vb [Design]*
ESA demo
Get Data
XYGraph
YAxis
XAxis
Contents
Filtered by:
(no filter)
Visual Studio .NET
MSDN Library
Agilent T&M Programmers Toolkit
Samples
Getting Started with the Agilent T&M Toolkit
Interacting with Devices
Using Agilent T&M Toolkit Utilities
Using the 2D Graph Objects
Calling Agilent VEE Pro
TMFramework Reference
Agilent.TMFramework
Agilent.TMFramework.DataAnalysis
Bessel Class
Bessel Members
Methods
I0 Method
J0 Method
J1 Method
Y0 Method
Y1 Method
Dsp Class
DspException Class
Regression Class
ScaleTypes Enumeration
Statistics Class
StatisticsException Class
Agilent.TMFramework.DataVisualization
Agilent.TMFramework.InstrumentIO
Agilent.TMFramework.VeeInterop

Fig. 7. ***Analysis capabilities included with the T&M Toolkit are fully integrated with Visual Studio .NET.***

Get help when you need it

Along with the tools in the Toolkit, there is extensive documentation and help provided, including sample code you can cut and paste directly into your work environment (Figure 8). This help is integrated into the Visual Studio environment so it is available when and where you need it. Help features like Intellisense and Dynamic Help provide help for both Toolkit-specific and general Visual Studio .NET functionality.



Toolkit MTTs demo - Microsoft Visual Basic .NET [design] - Using the Timing Class
File Edit View Project Build Debug Tools T&M Toolkit Window Help
Debug
IO
ms-help://MS.VSCC/Agt.TMToolkit/Agilent.
Object Browser
Using the Timing Class
ESA Demo.vb [Design]*
ESA Demo.vb*
Agilent Instrument Explorer
Contents
Filtered by:
(no filter)
Visual Studio .NET
MSDN Library
Agilent T&M Programmers Toolkit
Samples
Getting Started with the Agilent T&M Toolkit
Interacting with Devices
Using Agilent T&M Toolkit Utilities
Generating Virtual Waveforms
Function Waveform Generator - Graph Exam
Converting Numbers to Engineering Format
Using the Timing Class
Using the Progress Updater
Using the 2D Graph Objects
Calling Agilent VEE Pro
TMFramework Reference
Agilent.TMFramework
Agilent.TMFramework.DataAnalysis
Bessel Class
Dsp Class
DspException Class
Regression Class
ScaleTypes Enumeration
Statistics Class
StatisticsException Class
Agilent.TMFramework.DataVisualization
Agilent.TMFramework.InstrumentDriverInter
Agilent.TMFramework.InstrumentDriverInter
Agilent.TMFramework.InstrumentDriverInter
Agilent.TMFramework.InstrumentIO
Agilent.TMFramework.InstrumentIO.Configu
Agilent.TMFramework.VeeInterop
Agilent.TMFramework.VeeInterop.Design
Index
Properties
Dynamic Help
Ready
Start
3:57 PM

Agilent T&M Programmers Toolkit

Using the Timing Class

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

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

The following C# example demonstrates this usage.

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

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

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

NOTE: The Delay method guarantees the requested amount of time passes before control is returned to the calling process. However, due to the preemptive, multitasking nature of the Windows operating system, there is no guarantee that the delay will not be longer.

Send documentation feedback to Agilent

Copyright © 2002 Agilent Technologies, Inc. All rights reserved.

Fig. 8. ***The comprehensive T&M Toolkit help system is built into the Visual Studio .NET help system for easy reference and look up.***

If you need help with other aspects of connectivity beyond the scope of the T&M Toolkit, visit the Agilent Developer Network Web site at **http://www.agilent.com/find/ADN.**

Conclusion

You no longer need to waste valuable time and energy struggling with connectivity issues. You can use the Agilent T&M Programmers Toolkit for Visual Studio .NET to communicate with and program instruments that support industry standards such as IVI, VXI*plug&play*, GPIB, etc. Agilent also provides hardware connectivity solutions and in-depth connectivity support to eliminate connectivity hassles and free you up for more productive tasks.

Glossary

ADE — application development environment

API — application programming interface — You use the API to access functionality embedded in classes.

C# — (pronounced "C sharp") — a new C++-like, component-oriented language that was built from the ground up to run on the .NET Framework class library and therefore take advantage of the productivity improvements associated with it

Class — a software module that provides methods and properties that you can incorporate into your code by calling its properties, methods and interfaces. When you are writing code, instantiating an object as a member of a class lets the object use all of the class' functionality.

Class library — a group of classes

CLR — common language runtime — The CLR handles a variety of functions including memory management, security, verifying type safety, and exceptions. Unlike Java, the CLR is specifically not the runtime for a particular language. In the Visual Studio .NET environment, Visual Basic .NET, Managed C++ and the new C# language all emit intermediate language code (MSIL) that is just-in-time compiled and executed under the control of the CLR.

CLS — common language specification — CLS-compliant languages can interact with Visual Studio .NET

First-class languages — A first class language has a language compiler that supports the .NET Framework. First class languages include Visual Basic .NET, C#, Managed C++, COBAL, Perl, Eiffel and many more. See the current list at **www.msdn.microsoft.com/vstudio/partners**

IDE — integrated development environment

IVI — The IVI (Interchangeable Virtual Instrument) Foundation is a consortium dedicated to creating standards that simplify programming instruments.

IVI-C — IVI-C refers to the standards established by the IVI Foundation for instrument drivers based upon the C programming language. These standards are derived from the legacy VXI*plug&play* standards.

IVI-COM — IVI-COM refers to the standards established by the IVI Foundation for instrument drivers based upon the Microsoft COM standard. These drivers are optimized for COM environments and can easily be used in Visual Studio .NET.

.NET Framework —The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The .NET Framework class library is an object-oriented API that simplifies application development for Microsoft Windows.®

SCPI — standard commands for programmable instrumentation — SCPI defines a standard set of commands to control programmable test-and-measurement devices in instrumentation systems. Learn more at **www.scpiconsortium.org**

VISA — virtual instrument software architecture

VXI*plug&play* — a hardware and software standard that allows interoperability between instruments made by different manufacturers. Learn more **www.vxipnp.org**

Agilent Technologies' Test and Measurement Support, Services, and Assistance

Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise

Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

Your Advantage

Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent T&M Software and Connectivity

Agilent's Test and Measurement software and connectivity products, solutions and developer network allows you to take time out of connecting your instruments to your computer with tools based on PC standards, so you can focus on your tasks, not on your connections. Visit **www.agilent.com/find/connectivity for more information.**

For more assistance with your test & measurement needs or to find your local Agilent office go to **www.agilent.com/find/assist**

By internet, phone, or fax, get assistance with all your test & measurement needs.

Online assistance:
www.agilent.com/find/assist

Phone or Fax

United States
(tel) 1 800 452 4844

Canada
(tel) 1 877 894 4414
(fax) (905) 282 6495

China
(tel) 800 810 0189
(fax) 800 820 2816

Europe
(tel) (31 20) 547 2323
(fax) (31 20) 547 2390

Japan
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Korea
(tel) (82 2) 2004 5004
(fax) (82 2) 2004 5115

Latin America
(tel) (305) 269 7500
(fax) (305) 269 7599

Taiwan
(tel) 0800 047 866
(fax) 0800 286 331

Other Asia Pacific Countries
(tel) (65) 6375 8100
(fax) (65) 6836 0252
Email: tm_asia@agilent.com

Printed in USA
August 1, 2002

5988-6617EN

Agilent Technologies