



# TestStand Evaluation Guide

## **Worldwide Technical Support and Product Information**

ni.com

### **National Instruments Corporate Headquarters**

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

### **Worldwide Offices**

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,  
Canada (Calgary) 403 274 9391, Canada (Montreal) 514 288 5722, Canada (Ottawa) 613 233 5949,  
Canada (Québec) 514 694 8521, Canada (Toronto) 905 785 0085, China (Shanghai) 021 6555 7838,  
China (ShenZhen) 0755 3904939, Czech Republic 02 2423 5774, Denmark 45 76 26 00, Finland 09 725 725 11,  
France 01 48 14 24 24, Germany 089 741 31 30, Greece 30 1 42 96 427, Hong Kong 2645 3186,  
India 91805275406, Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456,  
Malaysia 603 9596711, Mexico 001 800 010 0793, Netherlands 0348 433466, New Zealand 09 914 0488,  
Norway 32 27 73 00, Poland 0 22 528 94 06, Portugal 351 1 726 9011, Russia 095 2387139,  
Singapore 2265886, Slovenia 386 3 425 4200, South Africa 11 805 8197, Spain 91 640 0085,  
Sweden 08 587 895 00, Switzerland 056 200 51 51, Taiwan 02 2528 7227, United Kingdom 01635 523545

For further support information, see the *Technical Support Resources* appendix. To comment on the documentation, send e-mail to [techpubs@ni.com](mailto:techpubs@ni.com).

© 2002 National Instruments Corporation. All rights reserved.

# Important Information

---

## Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THEREOF PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

## Trademarks

CVI™, LabVIEW™, Measurement Studio™, National Instruments™, NI™, ni.com™, and TestStand™ are trademarks of National Instruments Corporation.

Product and company names mentioned herein are trademarks or trade names of their respective companies.

## Patents

For patents covering National Instruments products, refer to the **Help»About** dialog box in your software (if applicable), the `patents.txt` file on your CD (if applicable), and/or [ni.com/legal/patents](http://ni.com/legal/patents).

## WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS




(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

# Conventions

---

The following conventions are used in this manual:

- » The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
-  This icon denotes an activity that you can complete to practice the concepts given in that section.
-  This icon denotes a note, which alerts you to important information.
-  This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.
- bold** Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
- italic* Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.
- `monospace` Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.

# Contents

---

## Chapter 1

### National Instruments, Test Management Systems, and TestStand

NI Platform .....	1-1
Test Management Systems .....	1-2
TestStand .....	1-3
Related Software Packages .....	1-3
NI Developer Suite .....	1-3
LabVIEW .....	1-4
Measurement Studio .....	1-4

## Chapter 2

### Getting Started with TestStand

About this Evaluation Package .....	2-1
Minimum System Requirements .....	2-1
Installation Instructions.....	2-2
What the Setup Program Installs .....	2-2
Learning TestStand .....	2-3

## Chapter 3

### TestStand Overview

Major Software Components of TestStand.....	3-2
TestStand Engine.....	3-3
TestStand Sequence Editor.....	3-3
TestStand Run-Time Operator Interface .....	3-3
Module Adapters .....	3-4
Process Models.....	3-5

## Chapter 4

### Exploring TestStand

Loading a Sequence File.....	4-1
Running a Sequence.....	4-2
Editing Steps in a Sequence.....	4-2
Inserting a Step.....	4-3
Specifying the Test Module.....	4-3
Configuring Step Properties .....	4-4

*Contents*

Debugging a Sequence .....	4-7
Step Mode Execution .....	4-7
Debugging Tools .....	4-9

**Chapter 5**

**Where to Go from Here**

Tutorials and Related Documentation .....	5-1
Programmer and Reference Information .....	5-1
TestStand Examples .....	5-2
Customer Education .....	5-2
Additional Support Information .....	5-2

**Appendix A**

**Technical Support Resources**

**Glossary**

# 1

---

## National Instruments, Test Management Systems, and TestStand

National Instruments is committed to providing hardware and software for engineers, scientists, and systems integrators who build, maintain, and improve test, measurement, and automation applications. National Instruments developed TestStand to automate a wide variety of test systems. TestStand is a ready-to-run test management environment for organizing, controlling, and executing your automated prototype, validation, or manufacturing test systems.

This chapter discusses the NI platform and provides an introduction to test management systems and TestStand.

### NI Platform

---

National Instruments pioneered the field of virtual instrumentation, which combines advanced hardware, powerful software, and off-the-shelf personal computers to revolutionize the test and measurement industry. This revolution has become the NI platform.

The NI Platform for production test combines TestStand, a high-level test management system, with LabVIEW or Measurement Studio. The test modules you create with LabVIEW or Measurement Studio use instrumentation hardware to take measurements. The NI platform also relies on PXI, a high-performance bus architecture, to control data acquisition devices, vision and motion devices, GPIB, and VXI.

## Test Management Systems

---

Before the introduction of National Instruments TestStand, you had two choices when it came to creating your test applications—write a single test program for each product or write your own in-house test management system. With TestStand, you now have a more powerful choice—a development environment specifically designed for test management.

A test management system is a high-level system used for organizing and executing reusable software modules that perform functional tests on products during the manufacturing process. Test management systems generally have the following two modes: the development mode, for defining test processes and test sequences, and the run-time mode, for executing sequences, displaying results, and generating reports on the sequences tested.

In most test management systems, the actual test programs that control the instrumentation hardware are separate software modules written in a test or general purpose programming language. By following a model that separates the test modules from the execution system, you have a modular architecture that can be easily supported and maintained.

A test management system that controls a production line must be widely integrated with a variety of different subsystems throughout the manufacturing plant. Because testing environments and processes can vary widely between companies, different facilities within a company, and different products within the same manufacturing line, it is challenging to find an off-the-shelf, ready-to-run test management system that exactly meets your requirements.

TestStand encourages collaborative development, globally and locally, through source code control features. TestStand's flexible, modular architecture makes it easier for you to share code, providing more efficient code development. Ultimately, TestStand enables you to lower your test times and reduce the cost of test.

TestStand from National Instruments is a ready-to-run, yet customizable test management system designed to deliver the cost benefits of a purchased product while maintaining the flexibility of a custom-developed solution.



## TestStand

---

National Instruments designed TestStand around a high-performance test executive engine that provides the performance you need to meet increasing demands for faster test development, faster test times, and more intelligent data sharing and test sequencing. The TestStand test executive engine is a multithreaded engine capable of running parallel sequences, which enables you to test a greater quantity of products in less time.

TestStand is completely customizable, so you can modify and enhance it to meet your specific needs. Modifying the operator interface, generating custom reports, and customizing sequence execution requirements are just a few of the ways in which you can customize TestStand. TestStand features a flexible, modular architecture so you can integrate with existing systems and implement features particular to your test system. Because National Instruments designed TestStand for creating custom test systems, you can use the built-in tools to make the integration with an existing system seamless.

TestStand is compatible with all of the leading test programming languages, including LabVIEW, Measurement Studio, Visual Basic, Visual C++, and HTBasic. TestStand also executes test code compiled as DLLs, ActiveX servers, or Windows executables so you can easily support multiple test programming environments and existing legacy code.

## Related Software Packages

---

TestStand fully integrates with the LabVIEW and Measurement Studio programming environments as well as other National Instrument products. Visit the Product Catalog at [ni.com](http://ni.com) for more information about the following packages.

### NI Developer Suite

For one package that includes all of the tools you need for developing an automated test application, consider NI Developer Suite. The NI Developer Suite Professional Test Edition includes LabVIEW and Measurement Studio for developing test routines and TestStand for managing test execution, sequencing, collecting data, and generating reports. The test edition also includes a comprehensive set of LabVIEW and Measurement Studio add-on tools for Internet connectivity, database communication, signal processing, and code distribution.

## LabVIEW

LabVIEW is an application development environment for measurement and automation that combines easy-to-use graphical development with the flexibility of a powerful programming language. The LabVIEW Full Development System provides tight integration with measurement hardware to facilitate rapid development of data acquisition and analysis, instrument control, and data presentation solutions. It also includes libraries for performing measurement analysis and digital signal processing, generating reports, displaying 3D plots, and calling external code through ActiveX and DLLs.

## Measurement Studio

Measurement Studio offers LabWindows/CVI for ANSI C programmers, C++ classes and tools for Microsoft Visual C++ programmers, and ActiveX controls for Microsoft Visual Basic programmers.

### Measurement Studio LabWindows/CVI

LabWindows/CVI is a complete ANSI C development environment and compiler with built-in libraries for acquisition, analysis, and presentation. LabWindows/CVI provides a simplified drag-and-drop User Interface Editor and automated code generation tools that you can use to interactively test code before adding it to your project. With LabWindows/CVI, you can create localized user interfaces, work with ActiveX components, and create multithreaded applications. LabWindows/CVI delivers unique code generation utilities and the ability to create and control graphical user interfaces, control instruments and plug-in data acquisition hardware from your PC, and develop and debug ANSI C programs.

### Measurement Studio for Visual C++

Measurement Studio delivers an interactive design approach for developing virtual instruments in Visual C++. Measurement Studio tools integrate into the Visual C++ environment so you can use them exactly as you would native Microsoft tools. Use the Measurement Studio AppWizard to create your measurement system. The AppWizard creates a project, according to your specifications, with a code template and the measurement tools you need to design your application. These tools include C++ classes you use to interface with hardware, provide data analysis, and transfer data across the Internet, as well as custom COM-wrapped ActiveX controls you use to create your user interface.

The link between the measurement classes and interface controls are data object classes that seamlessly encapsulate and pass data from acquisition to analysis to presentation.

## **Measurement Studio for Visual Basic**

Measurement Studio provides a collection of ActiveX controls designed for engineers and scientists building virtual instrumentation systems inside Visual Basic or other ActiveX control containers. With Measurement Studio, you can configure plug-in data acquisition boards, GPIB instruments, and serial devices from property pages without writing code. The user interface components enable you to configure knobs, meters, gauges, dials, tanks, thermometers, binary switches, LEDs, and 2D and 3D graphs. With powerful Internet components, you can share live measurement data between applications via the Internet.



# 2

---

## Getting Started with TestStand

This chapter introduces the TestStand Evaluation Package and provides system requirements and installation information.

### About this Evaluation Package

---

The TestStand Evaluation Package contains the TestStand Development software with the following restrictions:

- 30-day expiration.
- 10-minute run-time timeout.
- Consecutive running time of one hour.

### Minimum System Requirements

---

To run TestStand for Windows 2000/NT/XP/Me/9x, you must have the following:

- Windows 2000/XP/Me/9x or Windows NT 4.0 or later. For Windows NT, National Instruments recommends Service Pack 3 or later.
- 266 MHz Pentium class or higher microprocessor.
- SVGA resolution or higher video adapter, with a minimum 800 × 600 video resolution.
- Minimum of 64 MB of memory.
- 100 MB of free hard disk space (250 MB recommended).
- Microsoft-compatible mouse.

## Installation Instructions

---



**Note** National Instruments recommends that you close all open applications before you install TestStand.



**Caution** If you have LabVIEW VIs that you saved with a version of LabVIEW older than 5.1 that call the TestStand API, you must mass compile them in LabVIEW 5.1 or later before installing TestStand 2.0. If you do not do mass compile your VIs, you will have to manually replace every ActiveX diagram node that uses the TestStand API. The TestStand installer displays a message box if it detects an existing LabVIEW installation.

Unless you specify another location during installation, the TestStand installation program copies files to `C:\TestStand Evaluation Version\` after you complete the following steps:

1. Insert the TestStand Evaluation CD into the CD-ROM drive. If the CD startup screen does not appear, select **Run** from the Desktop Taskbar and run `tssetup.exe` from your CD.
2. Follow the instructions that appear in the dialog boxes.

National Instruments recommends that you install the complete TestStand Evaluation program to take full advantage of all the TestStand capabilities. If you choose to install with options, select the options you want to install and follow the directions on the screen. If necessary, you can run the setup program again later and install additional files.

### What the Setup Program Installs

The setup program installs the TestStand development environment and a number of additional files on your system. The full installation includes example files that illustrate many of the features in TestStand and tutorial programs that you use throughout this manual. The installer installs TestStand and the associated files in subdirectories on your hard disk as shown in Table 2-1.

**Table 2-1.** TestStand Subdirectories

Directory Name	Contents
AdapterSupport	Support files for the LabVIEW and C/CVI Standard Prototype Adapters.
Api	TestStand ActiveX automation server libraries for LabWindows/CVI and MFC.

**Table 2-1.** TestStand Subdirectories (Continued)

Directory Name	Contents
Bin	TestStand sequence editor executable, engine DLLs, and support files.
Cfg	Configuration files for TestStand engine and sequence editor options.
CodeTemplates	Source code templates for step types. This directory contains an NI and a User subdirectory.
Components	Components that come with TestStand and components that you develop. This directory includes callback files, converters, icons, language files, process model files, step types, source files, and utility files. It also contains an NI and a User subdirectory.
Doc	Documentation files.
Examples	Example sequences and tests.
OperatorInterfaces	LabVIEW, LabWindows/CVI, Microsoft Visual Basic, and Delphi operator interfaces with source code. This directory contains an NI and a User subdirectory.
Setup	TestStand installer/uninstaller.
Tutorial	Sequences and code modules that you use in the <i>TestStand Getting Started</i> manual.

## Learning TestStand

---

The best way to familiarize yourself with TestStand is to do the following:

1. Thoroughly read the `doc\readme.txt` file distributed with the TestStand Evaluation CD.
2. Read Chapter 3, *TestStand Overview*, to gain an understanding of the concepts, capabilities, and components of TestStand.
3. Complete the short tutorials in Chapter 4, *Exploring TestStand*.
4. Familiarize yourself with the documentation and online help included on the TestStand Evaluation CD. You can access this documentation by selecting **Start»Programs»National Instruments»TestStand - Evaluation Version»Online Help**.





---

## TestStand Overview

TestStand is a flexible, powerful test management environment that has the following major features:

- Out-of-the-box configuration and components that give you a full-featured test management system that is ready-to-run.
- Numerous ways for you to modify the out-of-the-box configuration and components or to add new components. This scalability enables you to create the test management system that meets your particular needs without modifying the TestStand engine. You can upgrade to newer versions of TestStand without losing your customizations.
- Sophisticated sequencing, execution, and debugging capabilities and a powerful sequence editor that is separate from the run-time operator interfaces.
- Four separate run-time operator interfaces with source code for LabVIEW, LabWindows/CVI, Microsoft Visual Basic, and Borland Delphi.
- Independence from a particular Application Development Environment (ADE). You can create test modules in a wide variety of ADEs and call pre-existing modules or executables. You can create your own run-time operator interface in any language that can control ActiveX automation servers.
- Documentation to help you convert from the LabVIEW Test Executive Toolkit Version 5.0 or the LabWindows/CVI Test Executive Toolkit Version 2.0 to TestStand.
- Comprehensive ActiveX Application Programming Interface (API) for building advanced sequences and code modules that programmatically access or control TestStand.

The remainder of this chapter provides an overview of important TestStand concepts and components.

## Major Software Components of TestStand

This section provides an overview of the major software components of TestStand. Figure 3-1 shows the high-level relationships between elements of the TestStand system architecture.

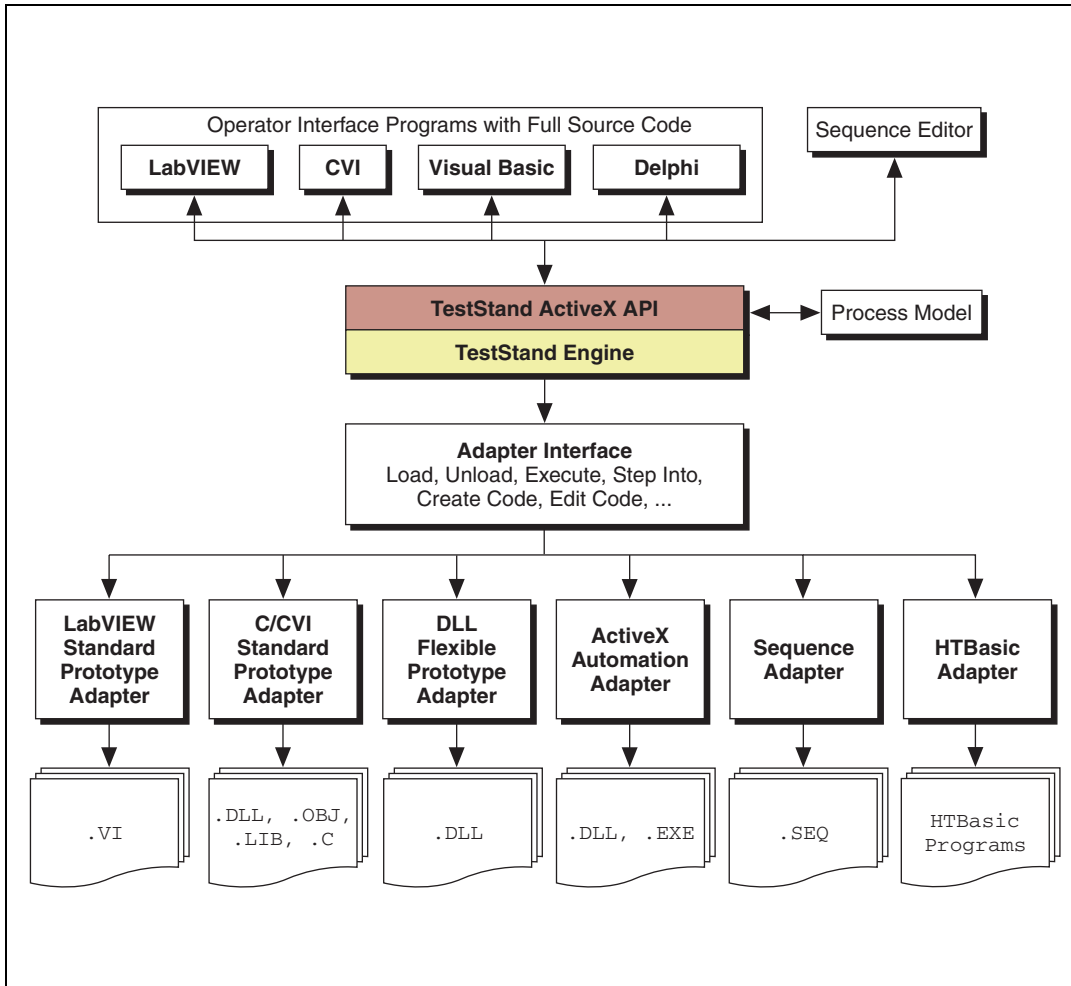


Figure 3-1. TestStand System Architecture

## TestStand Engine

The TestStand engine plays a pivotal role in the TestStand architecture. The TestStand engine is a set of DLLs that export an ActiveX automation API for creating, editing, and debugging sequences. The TestStand sequence editor and run-time operator interfaces use the engine API.

The TestStand engine can run sequences that contain calls to external code modules. By using module adapters that have a standard adapter interface, the TestStand engine can load and execute different types of code modules. You can access the engine API from any programming environment that supports access to ActiveX automation servers; thus, you can even call the engine API from a test module.

The documentation for the engine API is available as online help. You can access the online help through the **Help** menu of the sequence editor or from the TestStand - Evaluation Version program group in your Windows Start menu.

## TestStand Sequence Editor

The TestStand sequence editor is an application program in which you create, modify, and debug sequences. The sequence editor gives you easy access to all of the powerful TestStand features, such as step types and process models. The sequence editor uses debugging tools that you are familiar with in ADEs such as LabVIEW, LabWindows/CVI, and Microsoft Visual C/C++. These include breakpoints, single-stepping, stepping into or over function calls, tracing, a variable display, and a Watch window.

In the TestStand sequence editor, you can start multiple concurrent executions. You can execute multiple instances of the same sequence, and you can execute different sequences at the same time. Each execution instance has its own Execution window. In trace mode, the Execution window displays the steps in the currently executing sequence. When execution is suspended, the Execution window displays the next step to execute and provides single-stepping options.

## TestStand Run-Time Operator Interface

Your TestStand software includes source code for four run-time operator interfaces. Executable formats of the operator interface are available in LabWindows/CVI, Microsoft Visual Basic, and Borland Delphi. Each run-time operator interface is a separate application program. The operator

interfaces differ primarily based on the language and ADE in which each is developed. TestStand ships with run-time operator interfaces developed in LabVIEW, LabWindows/CVI, Visual Basic, and Borland Delphi.

Although you can use the TestStand sequence editor at a production station, the TestStand run-time operator interfaces are less complex and are fully customizable. Like the sequence editor, the run-time operator interfaces allow you to start multiple concurrent executions, set breakpoints, and single-step. However, the run-time operator interfaces do not allow you to modify sequences, and they do not display sequence variables, sequence parameters, step properties, and so on.

## Module Adapters

Most steps in a TestStand sequence invoke code in another sequence or in a code module. When invoking code in a code module, TestStand must know the type of the code module, how to call the code module, and how to pass parameters to the code module. The different types of code modules include LabVIEW VIs, C-based DLLs, ActiveX servers and objects, HTBasic subroutines, and C functions in source, object, or library modules that you create in LabWindows/CVI or other compilers. Using a module adapter, TestStand obtains the list of parameters that the code module requires.

TestStand currently provides six module adapters for the following purposes:

- **DLL Flexible Prototype Adapter**—Calls C functions in a DLL with a variety of parameter types.
- **LabVIEW Standard Prototype Adapter**—Calls any LabVIEW VI that has the TestStand standard G parameter list.
- **C/CVI Standard Prototype Adapter**—Calls any C function that has the TestStand standard C parameter list. The function can be in an object file, library file, or DLL. The function also can be in a source file that is in the project you are currently using in the LabWindows/CVI development environment.
- **Sequence Adapter**—Calls subsequences with parameters.
- **ActiveX Automation Adapter**—Calls methods and accesses the properties of an ActiveX automation object.
- **HTBasic Adapter**—Calls any HTBasic subroutine with no parameters. TestStand and HTBasic exchange data using the TestStand API. TestStand supports HTBasic version 7.2 or later.

Module adapters contain other important information besides the calling convention and parameter lists. If the module adapter is specific to an ADE, the adapter knows how to bring up the ADE, how to create source code for a new code module in the ADE, and how to display the source for an existing code module in the ADE.

## Process Models

Testing a Unit Under Test (UUT) requires more than just executing a set of tests. Usually, the test system must perform a series of operations before and after it executes the sequence that performs the tests. Common operations include identifying the UUT, notifying the operator of pass/fail status, logging results, and generating a test report. These operations define the testing process. The set of such operations and their flow of execution is called a process model.

Some traditional test management systems implement their process model internally and do not allow you to modify them. Other test management systems do not come with a process model at all. TestStand comes with three default process models that you can modify or replace.



**Note** Always make modifications to a copy of the default process model. The copy of the default process model should be placed in the `<TestStand Evaluation Version>\Components\User` directory. If you modify the default process model without making a copy, your edits may be overridden when you install subsequent TestStand releases.

TestStand ships with three fully functional process models: the Sequential model, the Batch model, and the Parallel model. You can use the Sequential model to run a test sequence on one UUT at a time. The Parallel and Batch models allow you to run the same test sequence on multiple UUTs at the same time.

Having a process model is essential to write different test sequences without repeating standard testing operations in each sequence. Because you can modify the process model, you can vary the testing process based on your production line, your production site, or the systems and practices of your company.



# 4

---

## Exploring TestStand

In this chapter, you will learn how to load and run sequence files, add and configure steps, and debug sequences.

### Loading a Sequence File

---

To view the features of the TestStand sequence editor, you must first load a sequence file into the sequence editor.

A sequence is made up of a series of steps. These steps can initialize an instrument, perform a complex test, or change the flow of an execution. Steps also can jump to another step, call a subsequence, call an external code module, or change the value of a variable or property.



**You can complete this activity in 5 minutes.**

1. Launch TestStand by selecting **Start»Programs»National Instruments»TestStand - Evaluation Version»Sequence Editor**.
2. Select **File»Open**.
3. Navigate to the `<TestStand Evaluation Version>\Examples\Demo\C\` subdirectory.
4. Select the `computer.seq` sequence file, and click the **Open** button.



**Note** Located in the `<TestStand Evaluation Version>\Examples\Demo\` directory, you will see the demo written in several programming environments that TestStand supports. Feel free to choose any of the programming environments at this time. If you choose to run the demo using LabVIEW, you must have LabVIEW installed on your machine. The short tutorials that follow apply directly to the LabWindows/CVI programming environment; if you choose another environment, the steps will be similar, though not exact.

The sequence file appears as a separate window within the sequence editor. This window is called the Sequence File window. You can load multiple sequence files into the sequence editor, and the sequence editor displays each in its own Sequence File window.

## Running a Sequence

---

When you run a sequence in the sequence editor, you are initiating an execution. In this section, you will examine the method of running a sequence directly. TestStand also provides the capability of running a sequence using a process model. The process model sequence contains a series of steps that specify the high-level flow of a sequence's execution.



**You can complete this activity in 5 minutes.**

1. Select `MainSequence` in the **View** ring control of the Sequence File window, if it is not already selected.
2. Select **Execute»Run MainSequence**.

When you make this selection, the sequence editor opens a new window. This window is called an Execution window. In an Execution window, you can view the steps as they execute, the values and variables of properties, and the test report when the execution is complete.

3. The first step in the sequence calls a code module that displays a Test Simulator dialog box. Select the **RAM** test by clicking its checkbox to enable it.
4. Click the **Done** button. Observe the Execution window as it traces through the execution.
5. After the execution completes, close the Execution window.



**Note** If you were not able to see the execution pointer, the tracing option may be disabled. Select **Configure»Station Options** and go to the Execution tab. Select the **Enable Tracing** and **Allow Tracing into Setup/Cleanup** options. You can adjust the speed of the tracing by using the **Speed** slide control. Click **OK** to close the Station Options dialog box and select **Execute»Run MainSequence** to rerun your execution.

## Editing Steps in a Sequence

---

TestStand contains a set of predefined step types. Step types define a list of standard properties and behaviors and give you the ability to call code modules using module adapters.



## Inserting a Step

TestStand comes with five built-in step types that you can use with any module adapter: Action, Pass/Fail Test, Numeric Limit Test, Multiple Numeric Limit Test, and String Value Test. When you insert a step into a sequence, TestStand binds the step to the adapter that is currently selected in the Adapter ring control on the sequence editor toolbar. The selected adapter applies only to the step types that can use the module adapter. The icon for the selected adapter appears as the icon for the step.



You can complete this activity in 5 minutes.

1. To insert a step that uses a code module, you must specify which module adapter the step uses. Using the **Adapter** ring control, select the **C/CVI Standard Prototype Adapter**.
2. Right-click the `RAM` step in the Sequence File window and select **Insert Step»Tests»Pass/Fail Test** from the context menu that appears. After you make this selection, a Pass/Fail Test step is inserted after the `RAM` step. The C/CVI Standard Prototype Adapter icon should appear to the left of the new step.
3. By default, the new step is named `Pass/Fail Test`. After you insert the step, the Pass/Fail Test step is highlighted, type `Video Test` and press enter to rename the step.

To rename the step later, right-click the step name and select **Rename** from the context menu.

## Specifying the Test Module

After you add a new step to the sequence, you must specify the test module that the step executes.



You can complete this activity in 5 minutes.

1. Right-click the `Video Test` step and select **Specify Module** from the context menu.

When you select **Specify Module** from the context menu, the sequence editor displays an Edit Call dialog box specific to the module adapter associated with that step.

2. Click the **Browse** button next to the Module Pathname control and select `<TestStand Evaluation Version>\Examples\Demo\C\computer.dll`. Click **OK**.

3. A dialog box may appear with options for the file path. Choose the option **Use a relative path for the file you selected**. This will make the path to the code source relative to the sequence file.
4. Select the `VideoTest` function in the **Function Name** ring control by using the arrow or the scrollbar to the right of the control.

Figure 4-1 shows the completed Edit C/CVI Module Call dialog box.

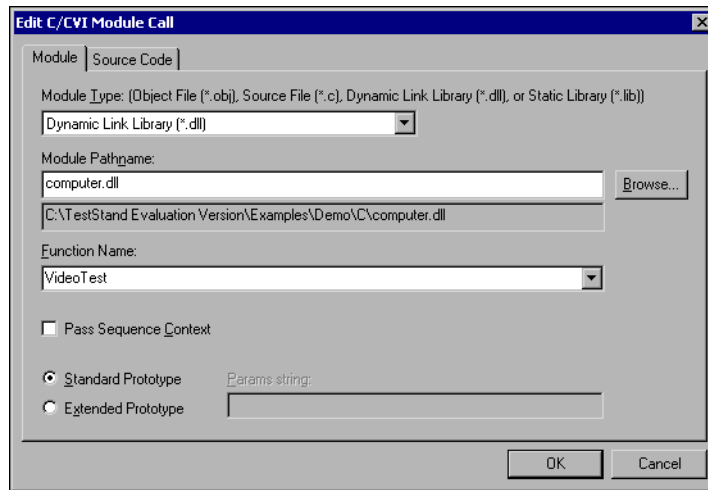


Figure 4-1. Edit C/CVI Module Call Dialog box

5. Click **OK** to close the Edit C/CVI Module Call dialog box and return to the Sequence File window.

## Configuring Step Properties

Each step in a sequence contains properties. The type of step determines the set of properties that a step has. All steps have a common set of properties that determine the following:

- When to load the step.
- When to execute the step.
- What information TestStand examines to determine whether a test passes or fails.
- Whether TestStand executes the step in a loop.
- What conditional actions occur upon step completion.

You can change the common properties of steps using the tabs located in the Step Properties dialog box.



**You can complete this activity in 10 minutes.**

1. Right-click the `Video Test` step and select **Properties** from the context menu.



**Note** When you make this selection, TestStand displays the Step Properties dialog box. The title of this dialog box is specific to the step that you select. For example, the `Video Test` step above has a Step Properties dialog box named `Video Test Properties`.

2. Click the **Preconditions** button in the Step Properties dialog box. A precondition specifies the conditions that must be evaluated as True for TestStand to execute a step during the normal flow of execution.
3. For the `Video Test` step, define a precondition so that the step executes only if the `Power On` step passes. To do this, complete the steps below.
  - a. In the **Insert Step Status** section of the Preconditions dialog box, click the `Power On` step in the list of names for the Main step group.
  - b. Add a condition to the preconditions list by clicking the **Insert Step Pass** button.
  - c. Click **OK** to close the Preconditions dialog box and return to the Step Properties dialog box.

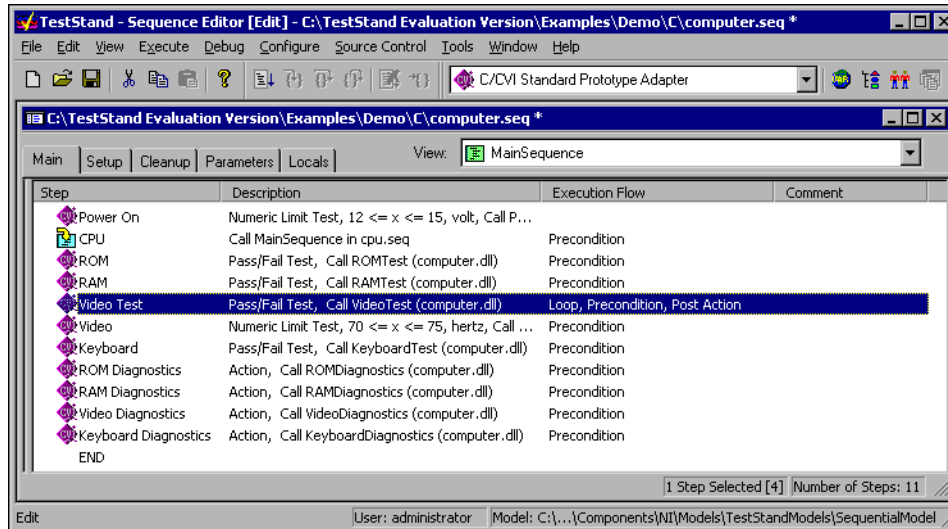
With these precondition settings, the `Video Test` step executes only if the `Power On` step passes.

4. Click the Post Actions tab. The Post Actions tab specifies what type of action occurs after the step executes. Set the **On Fail** post action control to **Terminate execution**.
5. Click the Loop Options tab. You can use the Loop Options tab to configure an individual step to run repeatedly in a loop when it executes. Change the following controls in the Loop Options tab.

<b>Loop Type</b>	Fixed number of loops
<b>Number of Loops</b>	10
<b>Loop result is Fail if</b>	< 80%

With these settings, TestStand executes the `Video Test` step ten times and sets the overall status to failed if less than eight of the ten iterations pass.

- Click the **OK** button to close the Step Properties dialog box. The Execution Flow column in the Sequence File window, shown in Figure 4-2, shows that the `Video Test` step contains a Loop, Precondition, and Post Action.



**Figure 4-2.** Execution Flow Column Containing Step Properties

- Select **File»Save As**. Save the sequence in the `<TestStand Evaluation Version>\Examples\Demo\C\` directory as `computer2.seq`.
- Run the sequence by selecting **Execute»Single Pass**. Notice that if you select the `Video Test` to fail, the sequence immediately terminates after calling the `Video Test` step ten times in a loop.
- Close the Execution window.

## Debugging a Sequence

---

TestStand has several features you can use to debug a sequence. These features include tracing, breakpoints, single-stepping, the sequence context browser, and watch expressions.

### Step Mode Execution

In this section of the tutorial, you learn how to run your sequences in step mode execution.



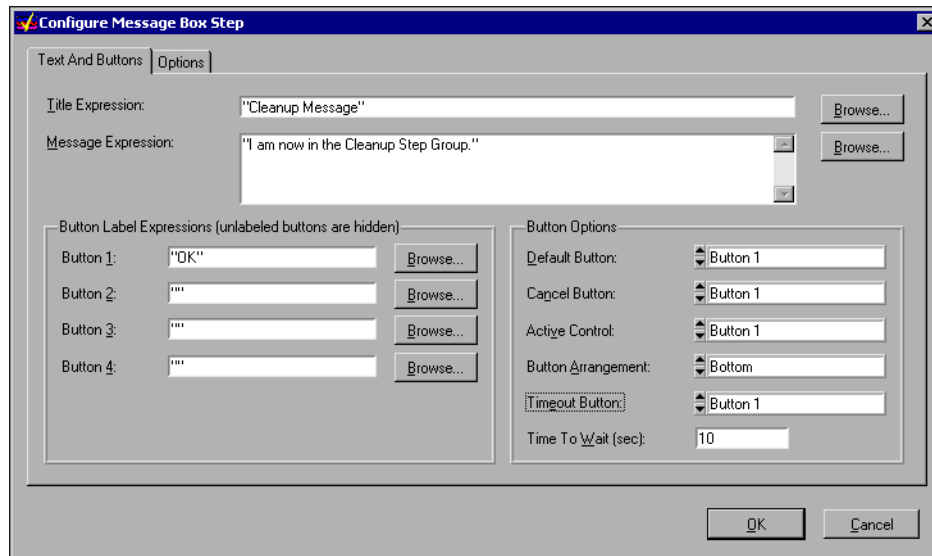
**You can complete this activity in 10 minutes.**

1. Open the file <TestStand Evaluation Version>\Examples\Demo\C\computer2.seq, which you created in the previous tutorial.
2. Select **Execute»Break At First Step**. You use this command to suspend an execution on the first step that TestStand executes. When enabled, this command has a checkmark beside it in the **Execute** menu.
3. Select the Cleanup tab in the Sequence File window.  
TestStand executes the Cleanup step group after the Main step group executes, regardless of whether the sequence completes successfully. If a Setup or Main step causes a run-time error to occur, the flow of execution stops and jumps to the Cleanup step group.
4. Complete the following steps to add a step to the Cleanup step group that displays a message popup.
  - a. Right-click in the empty step list of the Cleanup step group and select **Insert Step»Message Popup** from the context menu.
  - b. By default, the step is named `Message Popup`. After you insert the step, the Message Popup step is highlighted, type `Cleanup Message` and press **Enter** to rename the step.
  - c. Right-click the `Cleanup Message` step and select **Edit Message Settings** from the context menu. When you make this selection, the sequence editor displays the Configure Message Box Step dialog box.
  - d. Enter the text "Cleanup Message" into the Title Expression string control. Enclose the string with double quotation marks (" "), this indicates to TestStand that the expression is a literal string.

- e. Enter the text "I am now in the Cleanup Step Group.", including the quotation marks in the Message Expression string control.
- f. In the button options area, use the **Timeout Button** control to select **Button 1**. The Timeout Button specifies which message box button activates automatically after a timeout period expires.
- g. Enter 10 into the **Time to Wait (sec)** control. When the sequence is executed, the Cleanup Message step displays a notification message that automatically dismisses itself if the user does not make an entry within 10 seconds. This is useful if an operator is not present to acknowledge a non-critical message that displays during testing.

Figure 4-3 shows the Text and Buttons tab of the completed Configure Message Box Step dialog box.

- h. Click **OK** to close the Configure Message Box Step dialog box.



**Figure 4-3.** Configure Message Box Step Dialog Box



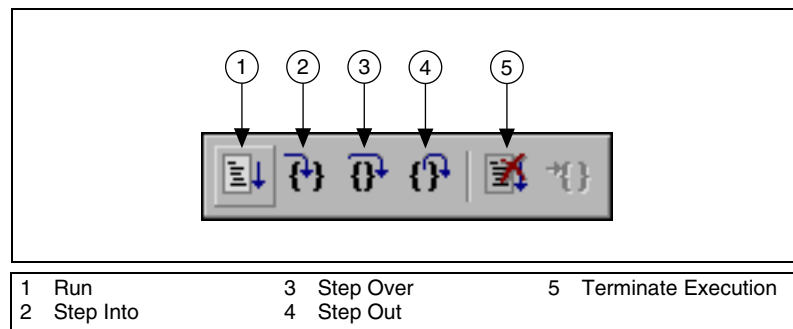
**Note** The Options tab of the Configure Message Box Step dialog box allows you to enable a response text box for operator input, position the message popup using coordinates, and make the message popup modal with respect to the application. A modal dialog box is a dialog box that you must dismiss before you can operate other application windows. For this example, leave all settings on the Options tab at their default values.

5. Select **File»Save As**. Save the sequence as `computer3.seq` in the `<TestStand Evaluation Version>\Examples\Demo\C\` directory.
6. Select **Execute»Run MainSequence**.  
After the execution starts, the sequence editor immediately pauses at the first step of the sequence. This occurs because you enabled the **Break At First Step** option.

In the left corner of the sequence editor status bar, you should see the running state of the execution. The Execution window indicates this as well with [Pause] in the title bar and a red button in the lower left corner of the window. The running state should now be paused.

## Debugging Tools

When an execution is paused, you can single-step through the sequence using the **Step Into**, **Step Over**, and **Step Out** commands in the Debug section of the toolbar, as shown in Figure 4-4. These tools also can be found in the **Debug** menu of the sequence editor.



**Figure 4-4.** Single-Stepping Toolbar Buttons



**You can complete this activity in 10 minutes.**

1. Click the **Step Over** button to execute the `Display Dialog` step. This step displays the Test Simulator dialog box.
2. Select the **RAM** test to fail, and click **Done**. After the Test Simulator dialog box closes, the sequence editor suspends the sequence execution at the end of the Setup step group on `END`.
3. Leaving the Execution window suspended, activate the `computer3.seq` Sequence File window by selecting it from the **Window** menu.

4. Right-click the CPU sequence call step in the Main step group, and select **Toggle Breakpoint** from the context menu. Notice that a red stop sign icon appears to the left of the CPU sequence call step.
5. Return to the Execution window by selecting it from the **Window** menu.
6. Select **Debug»Resume** to continue the execution. After you make this selection, the sequence editor suspends the execution when it reaches the CPU sequence call step.
7. Click the **Step Into** button to step into the subsequence.

Usually, when a step invokes a subsequence, the sequence that contains the calling step waits for the subsequence to return. The subsequence invocation is nested in the invocation of the calling sequence. The chain of active sequences that are waiting for nested subsequences to complete is called the call stack. The last item in the call stack is the most nested sequence invocation.

The Call Stack pane in the lower left half of the Execution window displays the call stack for the execution. A yellow pointer icon appears to the left of the most nested sequence invocation. The call stack shows that the `MainSequence` in `computer3.seq` is calling the `MainSequence` in `cpu.seq`.

When execution suspends, you can view a sequence invocation in the call stack by clicking its radio button.

8. Click each radio button in the Call Stack pane to view the status of each sequence invocation. Return to the most nested sequence invocation in the call stack.
9. Click the **Step Into** button to start stepping through the subsequence.
10. Before reaching the end of the sequence, select the **Step Out** button. TestStand resumes the execution through the end of the current sequence and suspends the execution at the next step or breakpoint, whichever comes first.
11. Continue single-stepping through the sequence using the **Step Over** button until the execution completes. The last step executed is the Cleanup Message step that you added to the Cleanup step group.
12. Click **OK** to close the Cleanup Message dialog box before you complete the execution. The Execution window dims when the execution completes. Do not close the Execution window.
13. Rerun the execution by selecting **Execute»Restart**. The Execution window must be the active window to restart the sequence.



14. After the sequence editor suspends the execution on the first step, select **Debug»Terminate**.

Notice that TestStand displays the Cleanup Message dialog box even though you terminated the sequence execution. When an operator or run-time error terminates the execution, it proceeds immediately to the steps in the Cleanup step group.

15. Click **OK** to close the Cleanup Message dialog box.
16. Rerun the execution again by selecting **Execute»Restart**.
17. After the sequence editor suspends the execution on the first step, select **Debug»Abort**. Notice that the execution of the sequence immediately stops, and TestStand does not execute any steps in the Cleanup step group.
18. Close the Execution window.
19. Save and close the sequence.



---

## Where to Go from Here

Visit the TestStand Web site at [ni.com/teststand](http://ni.com/teststand) and NI Developer Zone at [ni.com/zone](http://ni.com/zone) for the most up-to-date information.

---

## Tutorials and Related Documentation

The TestStand Evaluation CD contains PDF files for the following documents in the `<TestStand Evaluation Version>\Doc` directory.

- *TestStand Getting Started Manual*
- *TestStand User Manual*
- *TestStand Release Notes*
- *IVI Step Types*



**Note** Use Adobe Acrobat Reader 3.0 or later to view any TestStand PDF file. You can download a free version of the Reader from [www.adobe.com](http://www.adobe.com) or install a copy from your TestStand Evaluation CD.

---

## Programmer and Reference Information

The *TestStand Programmer Help* is located online. To view the help, select **Start»Programs»National Instruments»TestStand - Evaluation Version»Online Help»TestStand Programmer Help** or select it from the **Help** menu in the TestStand sequence editor toolbar. The *TestStand Programmer Help* contains the following sections:

- **TestStand API Overview**—Describes the basics of using an ActiveX automation server and, more specifically, the TestStand ActiveX automation server API.
- **Writing an Application with the API**—Provides information on how to develop an application using the TestStand API.
- **Navigating the API**—Contains a hierarchy of the classes, methods, properties, constants, and enumerations in the TestStand API.

- **Property and Method Reference**—Provides detailed information on each class, method, property, constant, and enumeration in the TestStand API.
- **Example Programs**—Contains the full directory path and description of each TestStand example.

## TestStand Examples

---

TestStand installs example programs that will assist you in the development of your own test sequences. You also can visit NI Developer Zone at [ni.com/zone](http://ni.com/zone) for the latest examples.



**Tip** Examples are an excellent way to get started. Look for an example similar to what you need and modify it to meet your specifications.

## Customer Education

---

National Instruments offers a complete selection of introductory and advanced training classes to build your proficiency with TestStand. These classes are designed to help you master the TestStand environment and develop successful test systems.

The following Fundamentals courses teach you the important features of TestStand so that you can get started right away.

- *TestStand I: Introduction*
- *TestStand II: Customization*

The Advanced TestStand course, *Advanced System Design*, teaches you several alternative design strategies and methodologies.

## Additional Support Information

---

For additional information on TestStand, refer to the following Web resources:

- **Support**—Visit [ni.com/support](http://ni.com/support) for access to the following:
  - KnowledgeBase—Search a database of tips, common questions, and more.
  - Troubleshooting Wizards.
  - Application notes and white papers.

- **Developer Resources**—Visit [zone.ni.com](http://zone.ni.com) for access to the following:
  - **Resource Library**—View example programs, technical presentations, and tutorials.
  - **Developer Exchange**—Participate in discussion forums and exchange code with measurement and automation developers around the world.
  - **Product Advisor**—Find the sensors, motors, cameras, and more to complete your next system. Explore detailed technical specifications from over 800 suppliers.
  - **Measurement Encyclopedia**—Find information on measurement principles, standards organizations, and a wide range of technology and measurement terms.





---

# Technical Support Resources

## Web Support

---

National Instruments Web support is your first stop for help in solving installation, configuration, and application problems and questions. Online problem-solving and diagnostic resources include frequently asked questions, knowledge bases, product-specific troubleshooting wizards, manuals, drivers, software updates, and more. Web support is available through the Technical Support section of [ni.com](http://ni.com).

## NI Developer Zone

---

The NI Developer Zone at [ni.com/zone](http://ni.com/zone) is the essential resource for building measurement and automation systems. At the NI Developer Zone, you can easily access the latest example programs, system configurators, tutorials, technical news, as well as a community of developers ready to share their own techniques.

## Customer Education

---

National Instruments provides a number of alternatives to satisfy your training needs, from self-paced tutorials, videos, and interactive CDs to instructor-led hands-on courses at locations around the world. Visit the Customer Education section of [ni.com](http://ni.com) for online course schedules, syllabi, training centers, and class registration.

## System Integration

---

If you have time constraints, limited in-house technical resources, or other dilemmas, you may prefer to employ consulting or system integration services. You can rely on the expertise available through our worldwide network of Alliance Program members. To find out more about our Alliance system integration solutions, visit the System Integration section of [ni.com](http://ni.com).

## Worldwide Support

---

National Instruments has offices located around the world to help address your support needs. You can access our branch office Web sites from the Worldwide Offices section of [ni.com](http://ni.com). Branch office Web sites provide up-to-date contact information, support phone numbers, e-mail addresses, and current events.

If you have searched the technical support resources on our Web site and still cannot find the answers you need, contact your local office or National Instruments corporate. Phone numbers for our worldwide offices are listed at the front of this manual.



# Glossary

---

## A

adapter	<p>If an adapter is specific to an Application Development Environment (ADE), the adapter knows how to open the ADE, how to create source code for a new code module in the ADE, and how to display the source for an existing code module in the ADE. Some adapters support stepping into the source code of the ADE while executing the step from the TestStand sequence editor.</p> <p>The following adapters are available in TestStand: DLL Flexible Prototype Adapter, LabVIEW Standard Prototype Adapter, C/CVI Standard Prototype Adapter, ActiveX Automation Adapter, HTBasic Adapter, and the Sequence Adapter. The general name for all specific adapters is module adapter.</p>
Application Development Environment (ADE)	<p>A programming environment such as LabVIEW, LabWindows/CVI, or Microsoft Visual Basic, in which you can create test modules and run-time operator interfaces.</p>
Application Programming Interface (API)	<p>A set of classes, methods, and properties that you use to control a specific service, such as the TestStand engine.</p>

## B

breakpoints	<p>An interruption in the execution of a program.</p>
-------------	---

## C

call stack	<p>The chain of active sequences that are waiting for the nested subsequences to complete.</p>
code module	<p>A program module, such as a Windows Dynamic Link library (*.dll) or LabVIEW VI (*.vi), that contains one or more functions that perform a specific test or other action.</p>

## *Glossary*

container property	A property that contains no values, but typically contains multiple subproperties. Container properties are analogous to structures in Microsoft Visual C/C++ and to clusters in LabVIEW.
context menu	Context menus are accessed by right-clicking on an object. Menu options in a context menu pertain specifically to the object you have selected.

## **D**

DLL	Dynamic Link Library.
-----	-----------------------

## **E**

engine	A module or set of modules that provide an API for creating, editing, executing, and debugging sequences, as well as sharing data between sequences. A sequence editor or run-time operator interface uses the services of the engine.
execution	An object that contains all of the information TestStand needs to run a sequence, its steps, and any subsequences it calls. Typically, the TestStand sequence editor and operator interface create a new window for each execution.
Execution window	A window in the sequence editor that displays the steps an execution runs. When execution is suspended, the Execution window displays the next step to execute and provides single-stepping options. You also can view variables and properties in any active sequence context in the call stack.

## **M**

module adapter	A module adapter knows how to load and call a code module, how to pass parameters to a code module, and how to return values and status from a code module.
----------------	---

## **N**

nested	Called by another step or sequence. If a sequence calls a subsequence, the subsequence is nested in the invocation of the calling sequence.
--------	---

**P**

process model	A series of operations before and after a test executive executes the sequence that performs the tests. Common operations include identifying the UUT, notifying the operator of pass/fail status, generating a test report, and logging results.
property	A property can be of type number, string, Boolean, container, ActiveX reference, a user-defined data type, or an array of these types. Properties are objects that store and maintain settings or attributes. Each property has a name and a comment. Only a container property has the ability to contain any number of subproperties.

**R**

run-time operator interface	A program that provides a graphical user interface for executing sequences on a production station.
-----------------------------	---

**S**

sequence	Located within a sequence file, a sequence contains a series of steps that you specify to execute in a particular order. When and if a step is executed can depend on the results of previous steps.
sequence editor	A program that provides a graphical user interface for creating, editing, and debugging sequences.
sequence file	A file that contains the definition of one or more sequences.
Sequence File window	A separate window within the sequence editor in which a sequence file appears.
step	An element that you can insert into a sequence that performs an action, such as calling a step module to perform a specific test. Typically, a sequence contains a series of steps that define your test and execution flow.

*Glossary*

step types                      A component that defines a set of custom step properties and standard behavior for each step of that type. All steps of the same step type have the same properties, but the values of the properties can differ. Step types define their standard behaviors using substeps.

subsequence                    A sequence that a step calls. You specify a subsequence call as a step in the calling sequence.

**T**

test module                    A code module that performs a test.

**U**

Unit Under Test (UUT)        The device or component that you are testing.

**V**

variable                        A property that you can freely create in a certain context. You can have variables that are global to a sequence file or local to a particular sequence. You also can have station global variables.

**W**

Watch window                 A window that shows the values of user-selectable variables and expressions that are currently active.