TestStand™

TestStand 3.0 Evaluation Guide

July 2003 Edition
Part Number 350863B-01

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 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599, 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, Canada (Vancouver) 514 685 7530, China 86 21 6555 7838, Czech Republic 420 2 2423 5774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11, France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, Greece 30 2 10 42 96 427, India 91 80 51190000, Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400, Malaysia 603 9131 0918, Mexico 001 800 010 0793, Netherlands 31 0 348 433 466, New Zealand 1800 300 800, Norway 47 0 66 90 76 60, Poland 48 0 22 3390 150, Portugal 351 210 311 210, Russia 7 095 238 7139, Singapore 65 6226 5886, Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51, Taiwan 886 2 2528 7227, Thailand 662 992 7519, United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on the documentation, send email to techpubs@ni.com.

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 THERETOFORE 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

Trademarks

CVI™, IVI™, LabVIEW™, Measurement Studio™, National Instruments™, NI™, NI-488™, ni.com™, NI-DAQ™, NI Developer Suite™, NI Developer Zone™, NI-DMM™, 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`.

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 discussed in that section.
	This icon denotes a tip, which alerts you to advisory information.
	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 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

1

National Instruments, Test Management Software, 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. It was this commitment that led National Instruments to pioneer the field of virtual instrumentation, which combines advanced hardware and powerful software with personal computers to revolutionize the test and measurement industry.

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 Integrated Test Framework and introduces test management systems and TestStand.

The NI Integrated Test Framework

The NI Integrated Test Framework for automated test combines a modular test architecture made up of the latest industry-standard test and switch management software tools, along with today's popular application development environments (ADEs), to greatly reduce the development costs and maintenance efforts required to build today's sophisticated automated test systems. NI TestStand test management software and NI Switch Executive switch management software provide seamless integration with NI LabVIEW, NI Measurement Studio for Microsoft Visual Studio .NET, and NI LabWindows™/CVI™. The NI Integrated Test Framework also incorporates the latest high-performance modular hardware platforms, including PXI and Measurement Services driver

software, such as NI-DMM, NI-Scope, NI-DAQ, NI-488, IVI, and others. You can use these platforms to configure and control a wide range of I/O devices including instrumentation, data acquisition devices, vision and motion hardware, GPIB, and VXI.

Test Management Software

Consumer demand for increased quality, state-of-the-art features, growing desires for cutting-edge products, and lower prices has driven substantial changes in industry trends during the past decade. These industry trends equate to many new challenges, such as reducing overall test system costs and developing and delivering test systems in a shorter amount of time in order to satisfy both consumer and corporate demands. Most test system budgets and timelines no longer permit building unique test applications from scratch for each new test system, so test code modularity and code reuse are now crucial for success.

Test management software is designed to help overcome these challenges. It promotes test code modularity and code reuse, which substantially reduces development and maintenance costs. Test management software also permits increased focus on the test routines and algorithms for specific products under test while providing all of the common test components found in nearly every automated test system right off the shelf. These components consist of many capabilities found in previous commercial test executives or in-house tools while also delivering all the additional test functionality required in today's systems such as intelligent branching, automatic code generation, user management, Unit Under Test (UUT) tracking, report generation, database integration, source code control, and deployment tools.

Using test management software, the actual test programs that control the instrumentation and tests are separate software modules written in a test or general purpose programming language such as LabVIEW, LabWindows/CVI, C++, C#, or Microsoft Visual Basic .NET. This type of architecture can be easily reused, supported, and maintained.

Testing environments and processes can vary widely between companies, different facilities within a company, and different products within the same manufacturing line, making it challenging to find an off-the-shelf, ready-to-run test management system that exactly meets your requirements.

TestStand

NI TestStand, named the 2002 Test Product of the Year by the readers of *Test & Measurement World* magazine, is a comprehensive, commercial test management software package that provides the flexibility to meet a diverse set of requirements. TestStand comes with off-the-shelf features such as parallel sequence execution for increased throughput, operator interfaces that simplify operator training, configurable report generation, built-in database logging, and advanced sequencing and branching capabilities. While TestStand provides all of these features right off-the-shelf, the true power and flexibility of the software is its ability to customize most of these features to meet your exact system requirements.

TestStand encourages collaborative development, globally and locally, through source code control features. The modular TestStand architecture makes it easy to share code for more efficient code development, implement features particular to each test system, and use built-in tools to seamlessly integrate with existing systems. TestStand is also completely customizable so that you can adjust it to meet your specific needs. Modifying the operator interfaces, generating custom reports, and customizing sequence execution requirements are a few examples of the many ways that you can customize TestStand. Ultimately, TestStand lowers test times and reduces testing costs.

Related Software Packages

TestStand integrates with all of the leading test programming languages, including LabVIEW, LabWindows/CVI, Measurement Studio, Microsoft Visual Basic .NET, C#, Microsoft Visual C++, and HTBasic. TestStand can also automate test code written in Java, VEE, TCL, Perl, and other test code compiled as DLLs, ActiveX servers, or Windows executables. This allows you to use TestStand with multiple test programming environments and existing legacy code.

TestStand also fully integrates with other National Instrument products such as NI Switch Executive. Visit the National Instruments Product Catalog at `ni.com` for more information about the software packages described in this section.

NI Switch Executive

NI Switch Executive is an intelligent switch management and routing application that you can use with TestStand to interactively configure switch devices from multiple vendors as a single virtual device. You can also specify intuitive names for each channel within the virtual switch device and use the end-to-end routing feature to automatically find switch routes by selecting the channels you need to connect.

NI Switch Executive also increases your test code reuse and system performance with switch programming used in conjunction with TestStand, LabVIEW, LabWindows/CVI, and Measurement Studio. Ultimately, NI Switch Executive simplifies switch system configuration and increases test performance, thus lowering your cost to test.

LabVIEW

LabVIEW 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. LabVIEW also includes libraries for performing measurement analysis and digital signal processing, generating reports, displaying 3D plots, and calling external code through ActiveX and DLLs.

LabWindows/CVI

LabWindows/CVI is a complete ANSI C development environment for creating virtual instrumentation applications with built-in libraries for acquisition, analysis, and visualization. The integrated LabWindows/CVI environment features code generation tools and prototyping utilities for fast and easy C code development. It offers a unique, interactive, ANSI C approach that delivers access to the full power of C with the ease of Microsoft Visual Basic. Because LabWindows/CVI is a programming environment for developing measurement applications, it includes a large set of run-time libraries for instrument control, data acquisition, analysis, and user interfaces. LabWindows/CVI also contains many features that make developing measurement applications easier than developing in traditional C environments.

Measurement Studio

Measurement Studio bridges the gap between standard software development tools and virtual instrumentation, as well as test, measurement, and automation applications. Choose from standard environments such as Microsoft Visual Basic, Microsoft Visual C++, and Visual Studio .NET to create your application, using tools specific to each language. With Measurement Studio, you can write programs quickly and easily and then modify them as your needs change while lowering your application development costs and decreasing your time to market.

Measurement Studio for Visual Studio .NET

Measurement Studio for Visual Studio .NET is an integrated suite of native test, measurement, and control tools and class libraries for Microsoft Visual Studio .NET. Measurement Studio dramatically reduces application development time using wizards, simplified data networking, and .NET user interface controls. New code designers interactively define reusable acquisition tasks and automatically generate code. Advanced analysis libraries and rich object-oriented hardware APIs, such as data acquisition and instrument control, enable the development of sophisticated measurement applications. The highly extensible .NET class libraries in Measurement Studio for Visual Studio .NET deliver unparalleled flexibility to scientists and engineers.

Measurement Studio for Visual C++

Measurement Studio for Visual C++ delivers an interactive design approach for developing measurement and automation systems inside Microsoft Visual C++. All of the tools for Visual C++ integrate into the environment so that you can use them exactly as you would use native Microsoft tools. You can create your measurement system using the Measurement Studio AppWizard, which creates a project according to your specifications, including a code template and the measurement tools required to design your application. These tools include C++ classes to interface with hardware, provide data analysis, and transfer data across the Internet, as well as custom wrapped ActiveX controls for creating your user interface. Data object classes serve as the link between the measurement classes and interface controls in order to seamlessly encapsulate and pass data from acquisition to analysis to presentation.

Measurement Studio for Visual Basic 6.0

Measurement Studio for Visual Basic 6.0 provides a collection of controls designed to build virtual instrumentation systems inside Microsoft Visual Basic or other ActiveX control containers. Use Measurement Studio ActiveX controls to configure plug-in data acquisition boards, GPIB instruments, and serial devices from property pages without writing code. The ActiveX user interface components enable you to configure real-time 2D and 3D graphs, knobs, meters, gauges, dials, tanks, thermometers, binary switches, and LEDs. With powerful Internet components, you can also share live measurement data between applications across the Web.

NI Developer Suite

If you want a single package that includes all of the tools you need for developing an automated test application, consider NI Developer Suite. NI Developer Suite Professional Test Edition includes TestStand for managing test execution, sequencing, data collection, and report generation, as well as NI Switch Executive for configuring your sophisticated switch layouts and eliminating the need for switch programming. The Professional Test Edition also includes LabVIEW, Measurement Studio, and LabWindows/CVI for quick development of code modules. A comprehensive set of add-on tools for internet connectivity, database communication, signal processing, and code distribution is also included in this package.

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 complete TestStand software. When you run TestStand for the first time, you will be prompted to activate a license for the product. If you do not activate a valid license, TestStand will run in evaluation mode and prompt you to activate a license on each subsequent launch.

When you run TestStand in evaluation mode, the software behaves as a fully-functional Development System for the first 7 days of the evaluation period. After 7 days, TestStand will run with the following restrictions:

- 10-minute run-time timeout
- Consecutive running time limit of one hour

The TestStand Evaluation Package software will expire after 30 days. You can activate your TestStand license at any point during the 30-day evaluation period.

For more information about TestStand licensing options or to purchase a TestStand license, visit `ni.com/TestStand`.

Minimum System Requirements

To run TestStand 3.0, National Instruments recommends that your system meet the following requirements:

- Windows 2000/XP/98 or Windows NT 4.0 Service Pack 6a or later (the TestStand User Interface Controls are not supported in Windows 98)
- 400 MHz Pentium class microprocessor (266 MHz minimum)
- 128 MB of memory (64 MB minimum)
- 500 MB of free hard disk space (100 MB minimum)

- SVGA resolution or higher video adapter, with a minimum 800 × 600 video resolution for small fonts or a minimum 1024 × 768 for large fonts
- Microsoft Internet Explorer version 6.0 or later (5.5 or later required)

TestStand 3.0 is compatible with the following National Instruments application development environments (ADEs):

- LabVIEW 6.1 or later (LabVIEW 7.0 or later is required to use the TestStand User Interface Controls)
- LabWindows/CVI 6.0 or later (LabWindows/CVI 7.0 or later is recommended for use with the TestStand User Interface Controls)
- Measurement Studio (7.0 or later) Enterprise Edition is required for integration with Microsoft Visual Studio .NET 2003 or later

Installation Instructions

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

Note If you have saved LabVIEW VIs that call the TestStand API with a version of LabVIEW earlier than 5.1.1, you must compile and save them in LabVIEW 6.1 or later before installing TestStand 3.0. If you do not compile and save your VIs, the VIs will not load correctly after you install TestStand 3.0.

Unless you specify another location during installation, the TestStand installation program copies files to `<Program Files>\National Instruments\TestStand 3.0` after you complete the following steps:

1. Insert the TestStand CD into the CD-ROM drive. If the CD startup screen is not visible, select **Run** from the Windows **Start** menu and run `setup.exe` from your CD.
2. Follow the instructions in the dialog boxes.

National Instruments recommends that you install the complete TestStand program to take full advantage of the software's capabilities. If you choose to perform a custom installation and do not install all of the TestStand features, you can run the setup program again at a later time to install additional files.

What the Setup Programs Install

The setup program installs the TestStand development software and a number of additional files on your system. The full installation includes example files that illustrate many of the features in TestStand, as well as tutorial programs that you use in TestStand manuals. 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, LabWindows/CVI, .NET, and HTBasic Adapters.
`API`	TestStand ActiveX automation server libraries and utility libraries for several programming languages.
`Bin`	TestStand Sequence Editor executable, TestStand Engine DLLs, and support files.
`Cfg`	Configuration files for the TestStand Engine and TestStand Sequence Editor options.
`CodeTemplates`	Source code templates for step types.
`Components`	Components that are installed with TestStand and components that you develop. This includes callback files, converters, icons, language files, process model files, step types, source files, and utility files.
`Doc`	Documentation files.
`Examples`	Example sequences and tests.
`OperatorInterfaces`	LabVIEW, LabWindows/CVI, Microsoft Visual Basic .NET, C#, and C++ (MFC) Operator Interfaces with source code.
`Redist`	Redistributable engine component for use with the TestStand Deployment Utility.
`Setup`	Support files for the TestStand installer.
`Tutorial`	Sequences and code modules that you use in the tutorial sessions in this manual, as well as in *Using TestStand*, *Using LabVIEW with TestStand*, and *Using LabWindows/CVI with TestStand*.

Learning TestStand

The best way to familiarize yourself with TestStand is to explore the *TestStand Bookshelf*, which contains all of the TestStand documentation in electronic format. To access the TestStand Bookshelf, select **Start»Programs»National Instruments»TestStand 3.0»Online Help»TestStand Bookshelf**.

National Instruments recommends completing the following tasks to get started with TestStand:

- Review the *TestStand System and Architecture Overview Card* to familiarize yourself with the various system components, as well as the flow and structure of sequences, process models, and executions.
- Read Chapter 1, *Introduction to TestStand*, in *Using TestStand* to familiarize yourself with TestStand concepts and features.
- Complete the tutorials in *Using TestStand*.
- Read Chapter 1, *TestStand Architecture*, in the *TestStand Reference Manual* to familiarize yourself with the other chapters in that manual.
- Refer to *Using LabVIEW with TestStand* and *Using LabWindows/CVI with TestStand* to learn how to use TestStand with National Instruments ADEs.

3

TestStand Overview

TestStand is a flexible test development framework that offers the following major features:

- Out-of-the-box configuration and components that give you a ready-to-run, full-featured test management environment.
- Numerous methods for modifying, configuring, and adding new components. These methods provide extensibility and enable you to create a test executive that meets your particular requirements without altering the core engine. You can also 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 operator interfaces.
- User interface controls for creating custom operator interfaces.
- Example operator interfaces with source code for LabVIEW, LabWindows/CVI, Microsoft Visual Basic .NET, C#, and C++ (MFC).
- Open language interface provides support for many ADEs. You can create code modules in a wide variety of ADEs and call preexisting modules or executables. You can also create your own operator interface in any programming language that can host ActiveX controls or control ActiveX automation servers.
- Comprehensive API for building multithreaded test systems and other sophisticated test applications.
- Integration with third-party source code control (SCC) packages.
- Deployment tools to aid in transferring a test system from development to production.

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.

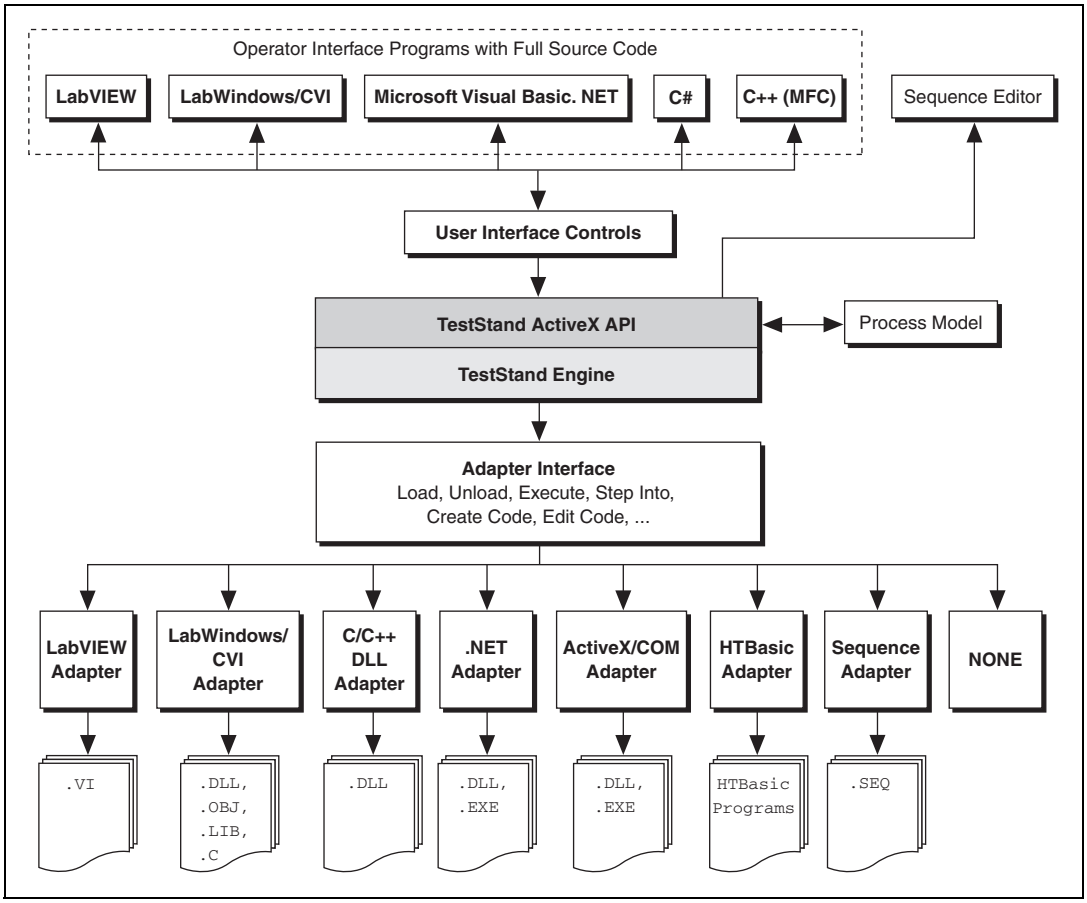


Figure 3-1. TestStand System Architecture

TestStand Engine

The TestStand Engine is a set of DLLs that export an extensive ActiveX Automation API. The TestStand Sequence Editor and User Interface (UI) Controls use the TestStand API, which you can call from any programming environment that supports access to ActiveX automation servers, including code modules you write in LabVIEW and LabWindows/CVI.

You can learn more about the TestStand API in the *TestStand Help*, which you can access from the **Help** menu of the TestStand Sequence Editor or by selecting **Start»Programs»National Instruments»TestStand 3.0» Online Help»TestStand Help**.

TestStand Sequence Editor

The TestStand Sequence Editor is an application program in which you create, edit, execute, and debug sequences. The sequence editor gives you access to all TestStand features, such as step types and process models, and features the debugging tools you are familiar with in ADEs such as LabVIEW, LabWindows/CVI, and Microsoft Visual Studio .NET. These debugging tools include setting breakpoints; stepping into, out of, or over steps; tracing through program executions; displaying variables; and monitoring variables and expressions during executions.

The TestStand Sequence Editor allows you to start multiple concurrent executions, execute multiple instances of the same sequence, and 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. If the execution is suspended, the Execution window displays the next step to execute and provides debugging options.

TestStand Operator Interfaces

TestStand includes several operator interfaces, each of which is a separate application program. These interfaces—developed in LabVIEW, LabWindows/CVI, Microsoft Visual Basic .NET, C#, and C++ (MFC)—are available in both source and executable formats.

The TestStand Operator Interfaces are fully customizable. Like the TestStand Sequence Editor, the operator interfaces allow you to start multiple concurrent executions, set breakpoints, and single-step. However, the operator interfaces do not allow you to modify sequences, and they do not display sequence variables, sequence parameters, step properties, and so on by default.

Refer to the *TestStand System and Architecture Overview Card* and the *TestStand User Interface Controls Reference Poster* for more information about operator interfaces. Refer to Chapter 9, *Creating Custom Operator Interfaces*, in the *TestStand Reference Manual* for more information about customizing operator interfaces.

TestStand User Interface Controls

The operator interfaces use the TestStand User Interface (UI) Controls, a collection of ActiveX controls for creating custom user interfaces in TestStand. These controls simplify common user interface tasks, such as displaying sequences and executions. You can use these controls in any programming environment that can host ActiveX controls.

Refer to the *TestStand Help* and to Chapter 9, *Creating Custom Operator Interfaces*, of the *TestStand Reference Manual* for more information about the TestStand UI Controls. You can also refer to the *TestStand User Interface Controls Reference Poster*, which is included in your TestStand package, for an illustrated overview of the controls.

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 code module, how to call it, and how to pass parameters to it. The different types of code modules include LabVIEW VIs; C functions in source, object, or library modules created in LabWindows/CVI; C/C++ functions in DLLs; objects in .NET assemblies; objects in ActiveX automation servers; and subroutines in HTBasic. TestStand must also know the list of parameters required by the code module. TestStand uses *module adapters* to obtain this knowledge.

TestStand includes the following module adapters:

- **LabVIEW Adapter**—Calls LabVIEW VIs with a variety of connector panes.
- **LabWindows/CVI Adapter**—Calls C functions with a variety of parameter types. The functions can be in object files, library files, or DLLs. They can also be in source files that are in the project you are currently using in LabWindows/CVI.
- **C/C++ DLL Adapter**—Calls functions or methods in a DLL with a variety of parameter types, including National Instruments Measurement Studio classes.
- **.NET Adapter**—Calls methods and accesses the properties of objects in a .NET assembly.

- **ActiveX/COM Adapter**—Calls methods and accesses the properties of objects in an ActiveX server.
- **HTBasic Adapter**—Calls HTBasic subroutines.
- **Sequence Adapter**—Calls other TestStand sequences with parameters.

The module adapters contain other important information in addition to the calling convention and parameter lists. If the module adapter is specific to an 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.

Refer to Chapter 5, *Module Adapters*, in the *TestStand Reference Manual* for more information about the module adapters included in TestStand.

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 ships with three fully functional process models that you can modify or replace: 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.

Process models allow you 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 start TestStand, load and run sequence files, add and configure steps, and debug sequences.

Starting TestStand

Complete the following steps to start the TestStand Sequence Editor.

1. Launch the TestStand Sequence Editor by selecting **Start»Programs»National Instruments»TestStand 3.0»Sequence Editor**. After the sequence editor displays the main window, it launches the Login dialog box, as shown in Figure 4-1.

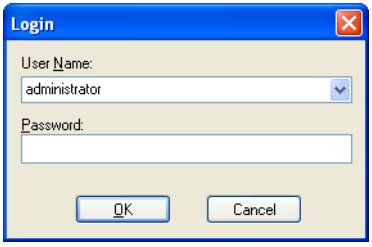
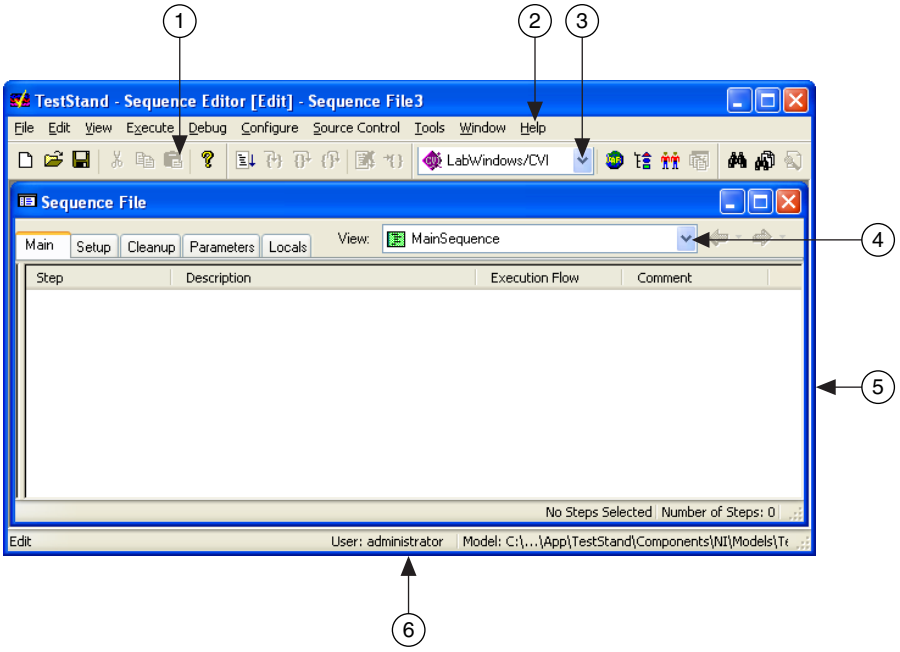


Figure 4-1. Login Dialog Box

2. Select the default user name, `administrator`, from the User Name ring control. In this default case, leave the **Password** field empty.
3. Click **OK** to begin. You should now see the Sequence Editor window, as shown in Figure 4-2.



1	Toolbar	3	Adapter Ring Control	5	Development Workspace
2	Menu Bar	4	View Ring Control	6	Status Bar

Figure 4-2. Sequence Editor Window

Loading a Sequence File

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

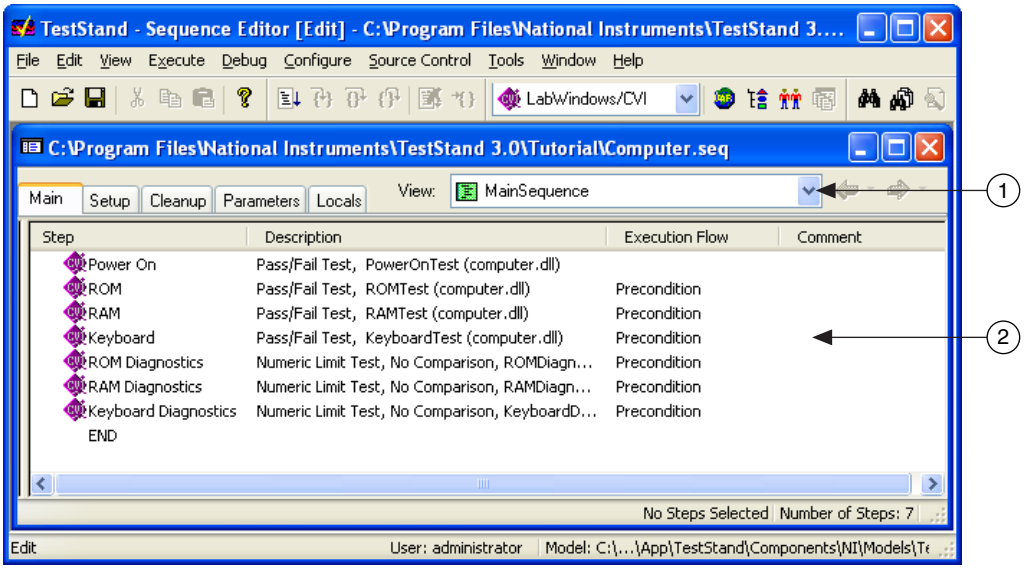
A sequence is made up of a series of steps. These steps can initialize an instrument, perform a complex test, or change the flow of an execution. Steps can also jump to another step, call a subsequence, call an external code module, or change the value of a variable or property.

You can complete this activity in 5 minutes.

1. Select **File»Open** to launch the Open dialog box, and then navigate to the `<TestStand>\Tutorial` subdirectory.
2. Select `Computer.seq` from the `Tutorial` subdirectory and click **Open**.

The `Computer.seq` sequence file is a simulated test of your computer in which you can designate various hardware components to "fail" the test. This sequence runs tests that are functions in a dynamic link library (`.dll`) written with LabWindows/CVI.

The sequence file opens in a separate window within the sequence editor. This window is called a *Sequence File window*, and is shown in Figure 4-3. You can load multiple sequence files into the sequence editor, which displays each file in its own Sequence File window.



1 View Ring Control	2 Sequence File Window

Figure 4-3. Sequence File Window for Computer.seq

3. Use the **View** ring control at the top right of the Sequence File window to select `MainSequence`. Figure 4-3 illustrates the View ring control.

   You can use the View ring control to view an individual sequence, a list of all sequences in the file, the sequence file global variables in the file, or the data and step types that you use in the file.

4. Browse the contents of each tab in the Sequence File window. Return to the Main tab when you finish.

Running a Sequence

When you run a sequence in the sequence editor, you are initiating an execution. This section discusses running a sequence directly. TestStand also allows you to run 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 **Execute»Run MainSequence**. The sequence editor launches the Execution window, as shown in Figure 4-4.

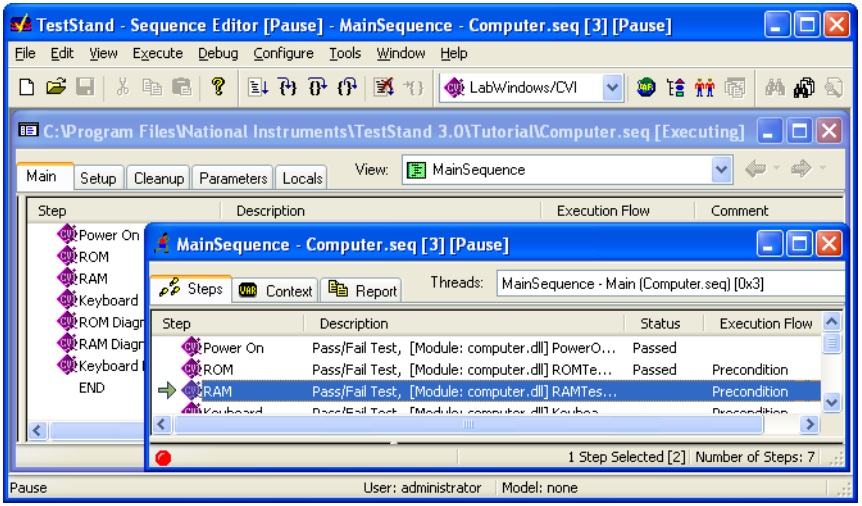


Figure 4-4. Execution Window

After the execution starts, a Test Simulator dialog box opens in front of the Execution window. Figure 4-5 illustrates the Test Simulator dialog box for this tutorial.

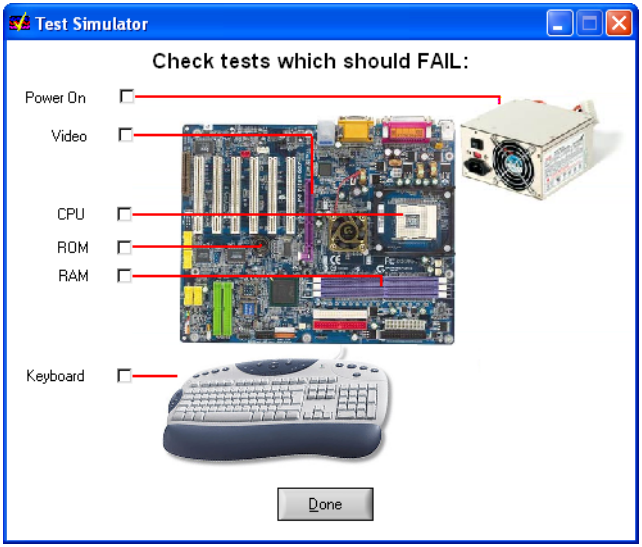


Figure 4-5. Test Simulator Dialog Box

This dialog box prompts you to designate the computer component(s), if any, that you want to return a value of "Fail" during the execution.

2. Select the RAM test by clicking its checkbox.
3. Click **Done**.
4. Observe the Execution window as it traces through the steps that TestStand runs.

   The sequence editor displays the progress of an execution by placing a yellow pointer icon to the left of the currently executing step in the Steps tab. The pointer icon is called the *execution pointer*. Notice that the status column for the RAM test contains the value `Failed`. When the execution completes, the status section of the window title changes from `[Running]` to `[Completed - <Status of Test>]`, and the Execution window dims.
5. After the execution completes, close the Execution window. You can rerun the sequence by repeating steps 1 through 4.

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. Enable the **Enable Tracing** and **Allow Tracing into Setup/Cleanup** options. Click **OK** to close the Station Options dialog box and select **Execute»Run MainSequence** to rerun your execution.

Tip To reduce the tracing speed, adjust the **Speed** slider control value on the **Execution** tab of the Station Options dialog box.

Editing Steps in a Sequence

TestStand contains a set of five predefined step types which define a list of standard properties and behaviors and give you the ability to call code modules using module adapters. You can use the following step types with any module adapter: Pass/Fail Test, Numeric Limit Test, Multiple Numeric Limit Test, String Value Test, and Action.

Inserting a Step

You can complete this activity in 5 minutes.

1. Select **File»Open**. Open `Computer.seq` from the `<TestStand>\Tutorial` directory.

   To insert a step that calls a code module, you must specify which module adapter the step uses.

2. Select **LabWindows/CVI** in the Adapter ring control.

   The LabWindows/CVI Adapter allows you to call C functions with a variety of prototypes. The function can be in a dynamic link library (`.dll`), source file (`.c`), object file (`.obj`), or static library file (`.lib`).

   The selected adapter only applies to the step types that can use the module adapter. The icon for the selected adapter appears as the icon for the step.

Note Once you add a step, you can change the adapter associated with that step in the Step Properties dialog box for the step.

3. Right-click the `RAM` step in the Sequence File window and select **Insert Step»Tests»Pass/Fail Test** from the *context menu*. When you make this selection, a Pass/Fail Test step is inserted after the `RAM` step.

   When you insert a step in a sequence, you must select a module adapter from the Adapter ring control, which is located on the sequence editor toolbar. TestStand then assigns the adapter you selected to that step. If you choose <None> as the selected adapter and then insert a step, the step you insert does not call a code module.

Normally, you use a Pass/Fail Test step to call a code module that makes its own pass/fail determination. After the code module executes, the Pass/Fail Test step evaluates a Boolean expression to determine whether the step passes or fails.

4. By default, the new test 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.

Tip To rename the step at a later time, select the step and press **<F2>** or right-click the step name and select **Rename** from the context menu.

Specifying the Code Module

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

You can complete this activity in 5 minutes.

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

   When you make this selection, the sequence editor launches an Edit Call dialog box specific to the module adapter associated with the step. For the LabWindows/CVI Adapter, the sequence editor displays the Edit LabWindows/CVI Module Call dialog box, which is shown in Figure 4-6.

2. Select **Dynamic Link Library (`*.dll`)** from the Module Type ring control. This selection specifies that the code module for the test calls a function within a DLL.
3. Click the **File Browse** button, which is located to the right of the module pathname control, and select `<TestStand>\Tutorial\computer.dll`. Click **OK**.

Note If prompted, select the **Use a relative path for the file you selected** option.

When you select a DLL, TestStand attempts to read the type library of the DLL and lists all the exported functions in the Function Name ring control.

4. Select **VideoTest** in the Function Name ring control. TestStand will use the prototype information stored in the type library of the DLL to populate the Parameters Table control.

5. In the Value Expression column of the **result** parameter, enter the following expression: `Step.Result.PassFail`.

   When TestStand returns from calling the `VideoTest` function, it will insert the value from the **result** parameter into the `Result.PassFail` property of the step.

   Figure 4-6 shows the completed Edit LabWindows/CVI Module Call dialog box.

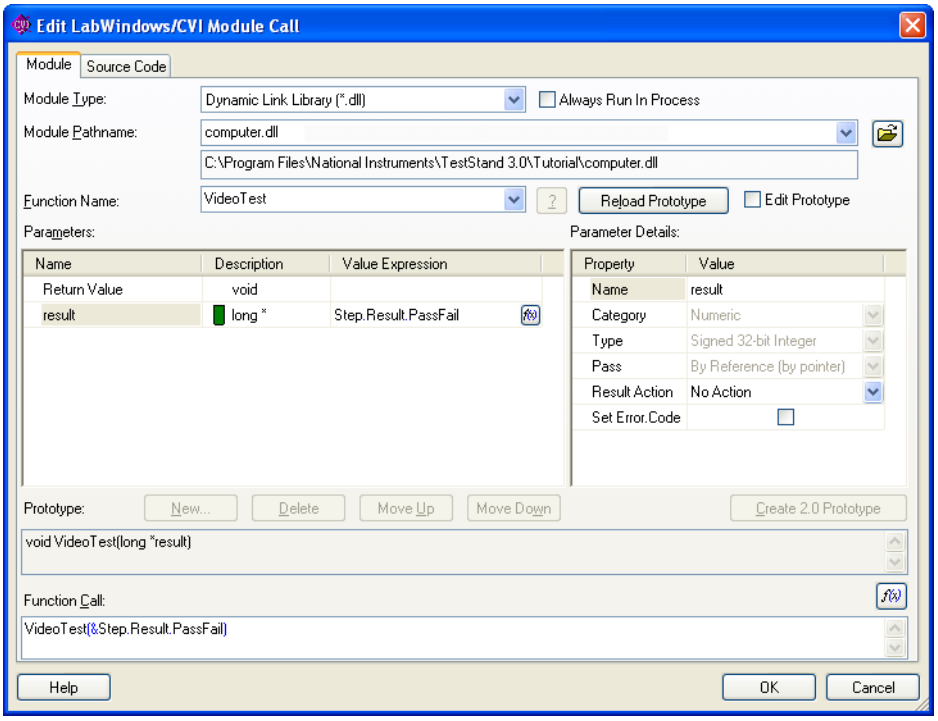


Figure 4-6. Edit LabWindows/CVI Module Call Dialog Box

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

Refer to the *TestStand Help* for more information about the Edit LabWindows/CVI Module Call dialog box. Refer to *Using LabWindows/CVI with TestStand* for more information about calling LabWindows/CVI code modules from TestStand.

Changing Step Properties

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

- When to load the step
- When to execute the step
- What information TestStand examines to determine the status of the step
- 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 in the Step Properties dialog box. In this exercise, you will examine common properties found in the Post Actions and Loop Options tabs, and you will use *preconditions* to modify steps.

Note For detailed information about the Step Properties dialog box, refer to the *TestStand Help*.

You can complete this activity in 10 minutes.

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

   When you make this selection, TestStand launches the Step Properties dialog box. The title of the Step Properties dialog box is specific to the step that you select. For the Video Test step, the Step Properties dialog box is named Video Test Properties, as shown in Figure 4-7.

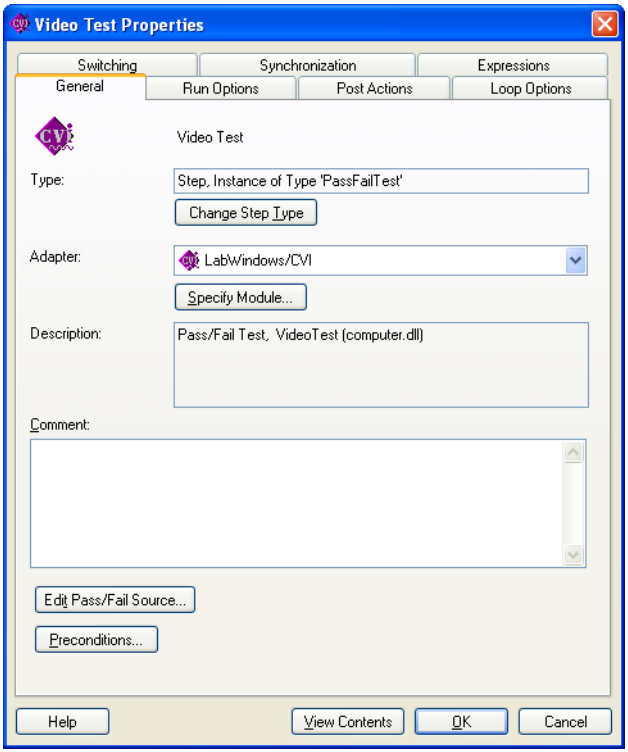


Figure 4-7. Step Properties Dialog Box

2. Click **Preconditions** in the Step Properties dialog box to launch the Preconditions dialog box, which is shown in Figure 4-8.

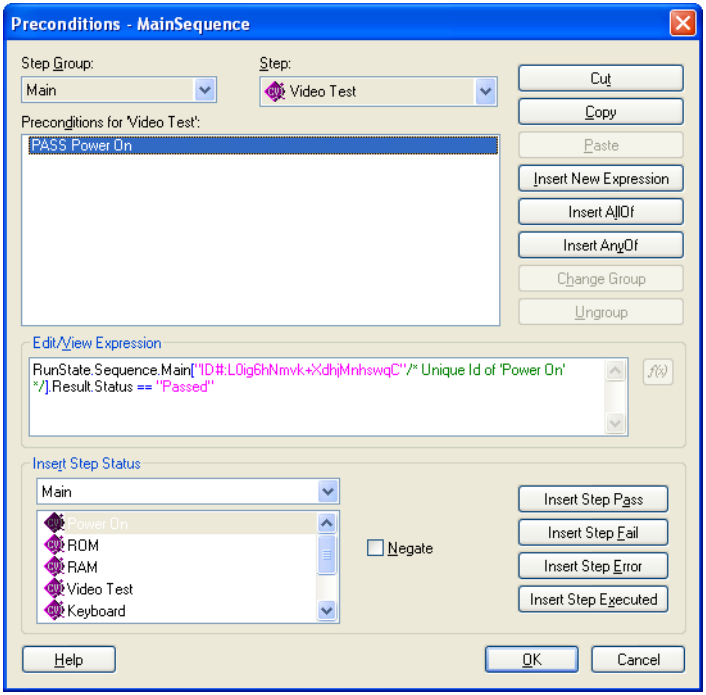


Figure 4-8. Preconditions Dialog Box

A precondition specifies the conditions that must be evaluated as `True` for TestStand to execute a step during the normal flow of execution in a sequence, such as only running a step if a previous step passes.

3. For the `Video Test` step, define a precondition so that the step only executes if the `Power On` step passes. Complete the following steps to modify the Preconditions dialog box, as shown in Figure 4-8:
   a. Under the Insert Step Status section, select the `Power On` step from the list of step names for the Main step group.
   b. Click **Insert Step Pass** to add a condition to the precondition list. The Preconditions for 'Video Test' text box now contains the string `PASS Power On`, which indicates that the step executes only if the `Power On` step passes.
   c. Click **OK** to close the Preconditions dialog box and return to the Step Properties dialog box.
4. Click the **Post Actions** tab, which is shown in Figure 4-9. This tab specifies what type of action occurs after the step executes. You can make the action conditional on the Pass/Fail status of the step or on any custom condition expression.

Note For steps that do not have a `Passed` or `Failed` status, you can use a custom condition. To do this, enable the **Specify Custom Condition** option and enter a custom condition expression that evaluates to `True` or `False`. Then, select the appropriate actions in the **On Condition True** and **On Condition False** controls.

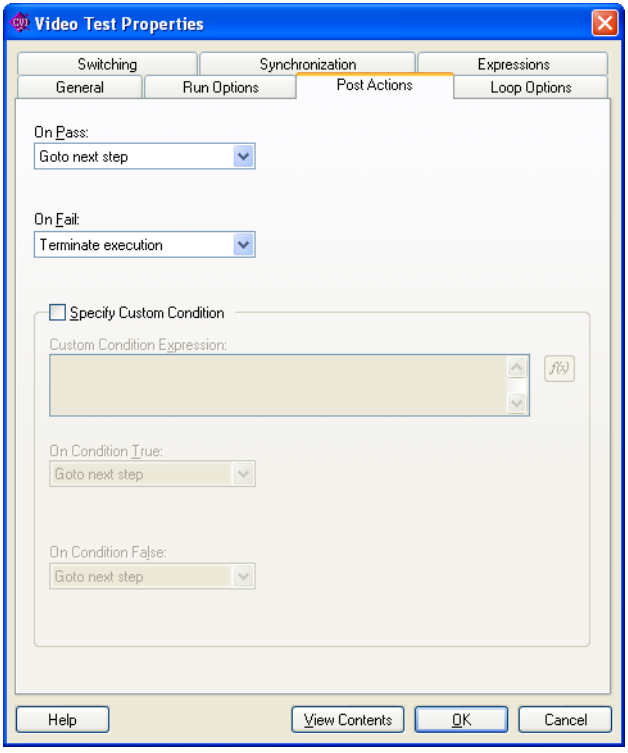


Figure 4-9. Post Actions Tab

5. Set the On Fail post action to **Terminate execution**.
6. Click the **Loop Options** tab, which is shown in Figure 4-10. Use this tab to configure an individual step to run repeatedly in a loop when it executes.

   Use the Loop Type ring control to select the type of looping for the step. TestStand determines the final status of the step based on whether a specific number of passes or failures occur, or the number of loop iterations reaches the maximum.

7. Change the following control values in the **Loop Options** tab:

   Loop Type `Fixed number of loops`

   Number of Loops `10`

   Loop result is Fail if `< 80 %`

   Using these settings, TestStand executes the Video Test step ten times and sets the overall status for the step to `Failed` if less than eight of the ten iterations pass. Figure 4-10 shows the completed Loop Options tab.

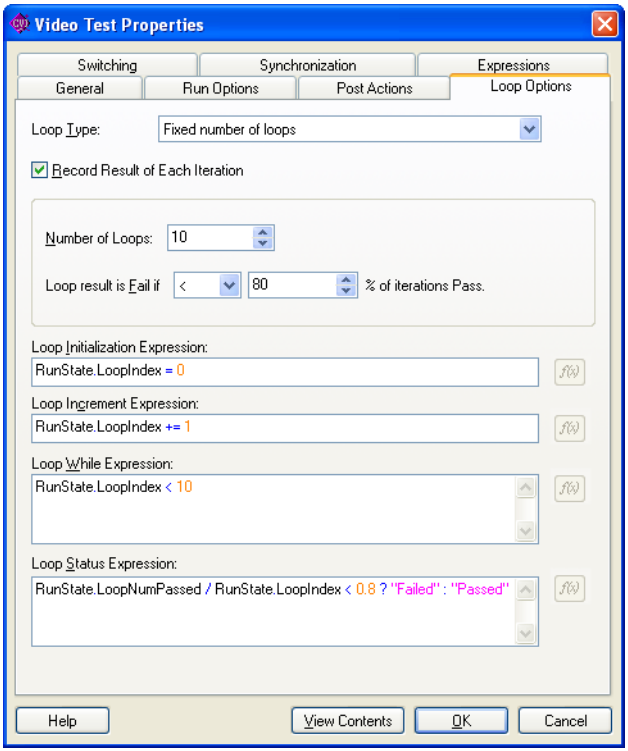


Figure 4-10. Loop Options Tab

8. Click **OK** to close the Step Properties dialog box.

   Notice that the Execution Flow column in the Sequence File window shows that the Video Test step contains Preconditions, Looping, and Post Actions.

9. Select **File»Save As**. Save the sequence in the `<TestStand>\Tutorial` directory as `Computer2.seq`.

Tip If other people will be using the tutorial files on your computer, use the **Save As** option to save your files with non-default file names. When you must save a file, this manual specifies the suggested name.

10. Run the sequence by selecting **Execute»Single Pass**. Notice that if you select the Video test step to fail, the sequence immediately terminates after calling the Video test step ten times in a loop.

    After TestStand generates the report, notice that when the Video Test step executes, the result of each loop iteration is recorded in the report.

11. Close the Execution window.

Debugging a Sequence

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

Step Mode Execution

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

You can complete this activity in 10 minutes.

Note Before beginning this tutorial, verify in the Station Options dialog box that the Enable Tracing and Tracing Into Setup/Cleanup options are enabled.

1. Select **File»Open** and select `<TestStand>\Tutorial\Computer2.seq`. You can also find this file in the `<TestStand>\Tutorial\Solution` directory.
2. Select **Execute»Break At First Step**. This command suspends 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 step in the Setup or Main step groups causes a *run-time error* to occur or if the operator terminates the execution, the flow of execution stops and jumps to the Cleanup step group.

4. Add a step that displays a message popup to the Cleanup step group to demonstrate this behavior, as follows:
   a. Right-click in the empty step list of the Cleanup step group and select **Insert Step»Message Popup** from the context menu.
   b. Rename the step `Cleanup Message`.
   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 Popup Step dialog box.
   d. Enter the text `"Cleanup Message"`, including the quotation marks, in the **Title Expression** expression control.
   e. Enter the text `"I am now in the Cleanup Step Group."`, including the quotation marks, in the **Message Expression** expression control.

Note You must enclose literal strings in double quotation marks (`"..."`) in any TestStand expression field.

   f. In the Button Options section, select **Button 1** in the **Timeout Button** control. The Timeout Button control specifies which message popup button activates automatically after a timeout period expires.
   g. Enter `10` in 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 ten seconds. This is useful if an operator is not present to acknowledge a non-critical message that displays during testing.

Figure 4-11 shows the Text And Buttons tab of the completed Configure Message Popup Step dialog box.

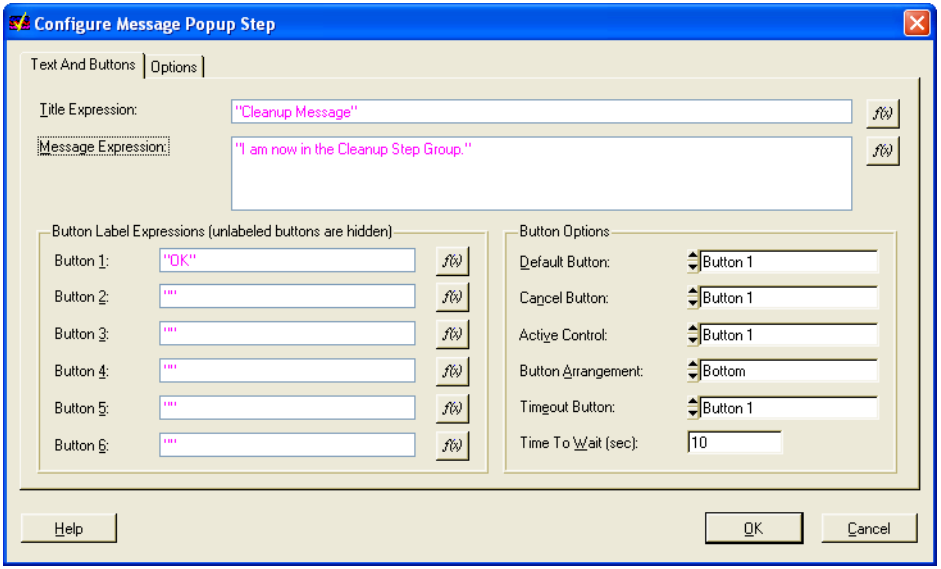


Figure 4-11. Configure Message Popup Step Dialog Box

Note The Options tab of the Configure Message Popup 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 a *modal dialog box* with respect to the application. For this example, leave all settings on the Options tab set to their default values. Refer to Chapter 4, *Built-In Step Types*, in the *TestStand Reference Manual* for more information about the Message Popup step type. Refer to the *TestStand Help* for more information about the Options tab.

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

5. Select **File»Save As**. Save the sequence as `Computer3.seq` in the `<TestStand>\Tutorial` directory.
6. Select **Execute»Run MainSequence**.

   After the execution starts, the sequence editor immediately pauses at the first step of the sequence. This behavior is caused by the **Break At First Step** option, which you enabled in Step 2 of this tutorial. Figure 4-12 illustrates the sequence editor at this point in the tutorial.

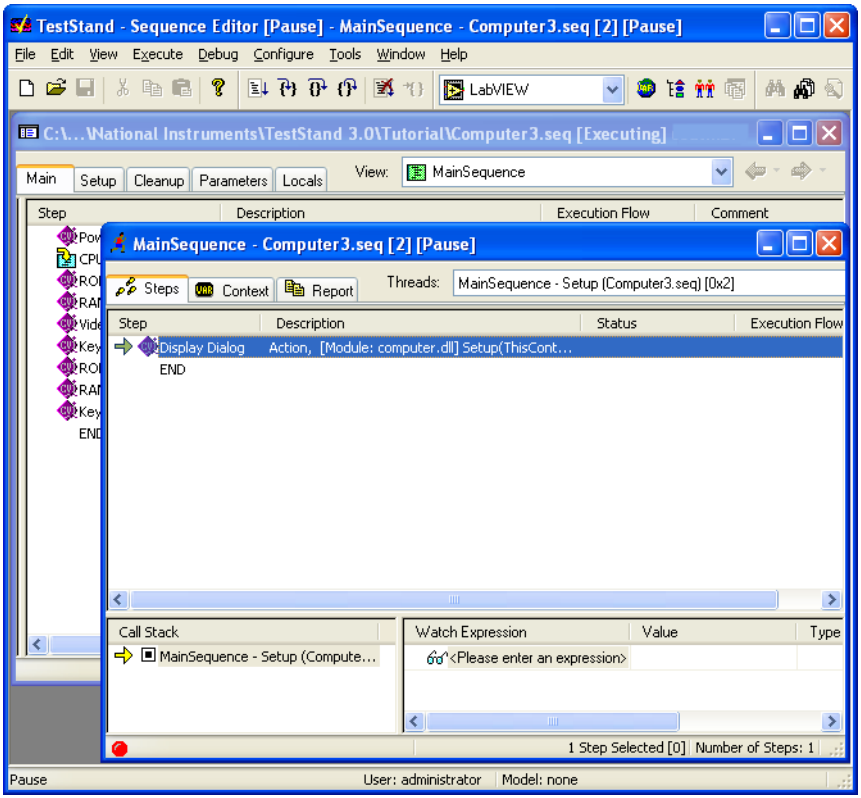
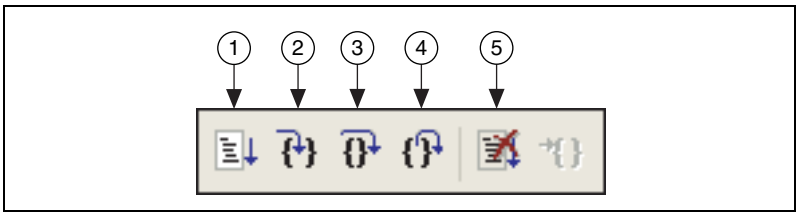


Figure 4-12. Paused Execution of Computer3.seq

Notice the left corner of the status bar in the Execution window, which contains an LED indicator that displays the running state of the execution. The running state should now be paused, as indicated by the red LED.

Debugging Tools

When the 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 4-13. These tools can also be found in the Debug menu of the sequence editor. For a detailed discussion of the single-stepping tools, refer to Chapter 3, *Executions*, in the *TestStand Reference Manual*.



1 Run
2 Step Into
3 Step Over
4 Step Out
5 Terminate Execution

Figure 4-13. Single-Stepping Toolbar Buttons

You can complete this activity in 10 minutes.

1. Click **Step Over** 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 you close the Test Simulator dialog box, the sequence editor suspends the sequence execution at the end of the Setup step group on END.
3. Activate the Sequence File window for `Computer3.seq` by selecting it from the **Window** menu.
4. Right-click the `CPU Test` step in the Main step group, and select **Breakpoint»Toggle Breakpoint** from the context menu. Notice that a dark red stop sign icon appears to the left of the step name.

Note You can right-click the stop sign icon to bring up the Breakpoint Settings dialog box. Use this dialog box to specify an expression that must be `True` in order for the breakpoint to be valid. These breakpoints are conditional breakpoints and are identified with a bright red stop sign icon. For more information about expressions, refer to Chapter 5, *Using Variables and Properties*, in *Using TestStand.*

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 on the CPU Test step again.
7. Click **Step Into** to step into the subsequence.

Tip You may have to adjust the size of the Execution window to make all elements visible.

Figure 4-14 shows the Steps view for the Execution window after you step into the subsequence.

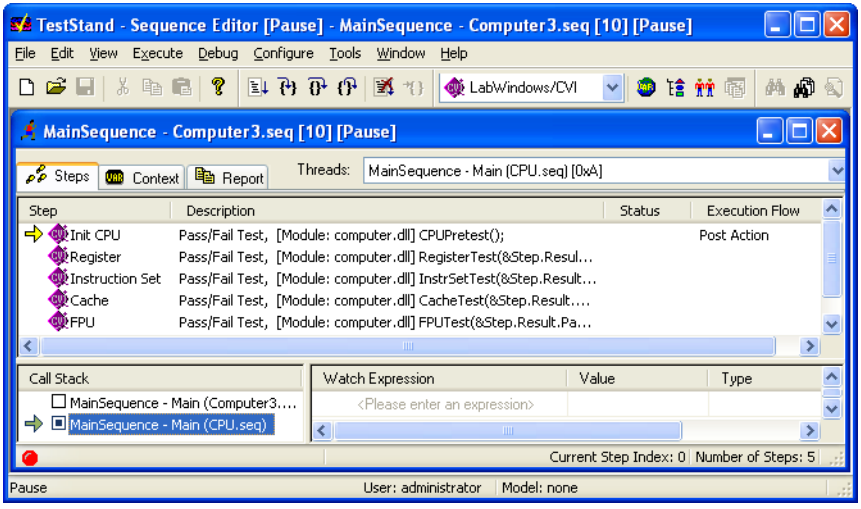


Figure 4-14. Steps View while Suspended in Subsequence

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 in Figure 4-14 shows that `MainSequence` in `Computer3.seq` is calling `MainSequence` in `CPU.seq`.

When the 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 **Step Over** to start stepping through the subsequence.

10. Before reaching the end of the sequence, click **Step Out**. 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 by clicking **Step Over** 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 completing the execution. The Execution window dims when the execution is complete.

Note 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**.

Note TestStand displays the Cleanup Message dialog box even though you terminated the sequence execution. When an operator or run-time error terminates the execution, TestStand 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 the sequence by selecting **File»Save**.
20. Close the `Computer3.seq` file.

You have completed the tutorials in this manual. Additional TestStand tutorials are available in the *Using TestStand*, *Using LabVIEW with TestStand*, and *Using LabWindows/CVI with TestStand* manuals.

5

Where to Go from Here

Visit the TestStand Web site at `ni.com/teststand` and NI Developer Zone at `ni.com/zone` for the most up-to-date information.

Related Documentation

TestStand 3.0 includes a documentation set featuring electronic and printed resources for your reference.

The following electronic resources are available:

- *TestStand Bookshelf*—Use the *TestStand Bookshelf* to access all of the documentation in electronic format. You can also search the *TestStand Bookshelf* by keyword and/or phrase to find information quickly. To access the *TestStand Bookshelf*, select **Start»Programs»National Instruments»TestStand 3.0»Online Help»TestStand Bookshelf**.

  To search the entire *TestStand Bookshelf* by keywords and/or phrases, you must have Adobe Acrobat Reader version 5.05 or later with Search and Accessibility installed. Select **Help»About Adobe Acrobat Plug-Ins** to check the version number and verify that **Search** and **MSAA** are listed as plug-ins.

  If you do not have Adobe Acrobat Reader version 5.05 or later or these plug-ins, refer to the Adobe Systems Incorporated Web site to download the correct version of Acrobat Reader.

- *TestStand Help*—Contains information about the TestStand environment and the TestStand UI Controls and Engine APIs. The *TestStand Help* also includes basic information about using an ActiveX automation server. To access the *TestStand Help*, select **Start»Programs»National Instruments»TestStand 3.0»Online Help»TestStand Help**.
  - *TestStand Environment Reference Help*—Contains information about the menus, windows, and dialog boxes found in the TestStand environment.
  - *TestStand UI Controls API Reference Help*—Contains detailed information about the properties, methods, events, constants, and enumerations of the TestStand UI Controls API. It also contains

basic information regarding the property pages available for the TestStand UI Controls.

- *TestStand API Reference Help*—Contains information about the TestStand ActiveX automation server, API concepts, and writing applications with TestStand. It also contains information regarding the properties, methods, events, constants, and enumerations of the TestStand API.
- *TestStand Supplemental Reference Help*—Contains specific information about the sequence context and its properties, as well as information about database connectivity in TestStand. It also contains information about the examples included in the TestStand 3.0 software package.

Note If you are opening the *TestStand Help* from the `<TestStand>/Doc/Help` directory, National Instruments recommends that you open the `TSHelp.chm` file. The `TSHelp.chm` file is a collection of all of the help files available in TestStand and provides a complete table of contents and index.

The following printed resources are also available:

- *TestStand Quick Start Guide*—Provides system requirements as well as installation and licensing information.
- *TestStand System and Architecture Overview Card*—Provides an illustrated overview of TestStand components and architecture, as well as a documentation map. You can access a PDF of this reference card from the *TestStand Bookshelf*.
- *Using TestStand*—Contains information about the TestStand environment and the basic features you use to build and run sequences. For information about using TestStand with specific application development environments, refer to the following manuals:
  - *Using LabVIEW with TestStand*—Provides information about using LabVIEW with TestStand.
  - *Using LabWindows/CVI with TestStand*—Provides information about using LabWindows/CVI with TestStand.
- *TestStand Reference Manual*—Contains information about TestStand concepts and features.

- *TestStand User Interface Controls Reference Poster*—Provides an illustrated overview of the TestStand User Interface (UI) Controls. You can access a PDF of this reference poster from the *TestStand Bookshelf*.
- *TestStand API Reference Poster*—Provides an illustrated overview of the TestStand API. You can access a PDF of this reference poster from the *TestStand Bookshelf*.

TestStand Examples

TestStand installs example programs in the `<TestStand>\Examples` directory that will assist you in the development of your own test sequences. You can also visit NI Developer Zone at `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 both 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.

Visit `ni.com/training` to learn more about National Instruments training classes for TestStand.

Additional Support Information

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

- **Support**—Visit `ni.com/support` for access to the following support tools:
  - KnowledgeBase—Search a database of tips, common questions, and more
  - Troubleshooting Wizards
  - Application notes and white papers
- **Developer Resources**—Visit `ni.com/zone` for access to the following developer resources:
  - 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 about measurement principles, standards organizations, and a wide range of technology and measurement terms.

A

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Online technical support resources include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of technical support resources available in English, Japanese, and Spanish at ni.com/support. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, hardware schematics and conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.
  - **Assisted Support Options**—Contact NI engineers and other measurement and automation professionals by visiting ni.com/support. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.
- **Training**—Visit ni.com/training for self-paced tutorials, videos, and interactive CDs. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Glossary

A

abort	To stop an execution without running any of the steps in the Cleanup step groups in the sequences on the call stack. When you abort an execution, no report generation occurs.
active window	The window that user input affects at a given moment. The title of an active window is highlighted.
ActiveX server	Any executable code that makes itself available to other applications according to the ActiveX standard. ActiveX implies a client/server relationship in which the client requests objects from the server and asks the server to perform actions on the objects.
adapter	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.
Application Development Environment (ADE)	A programming environment such as LabVIEW, LabWindows/CVI, or Microsoft Visual Basic, in which you can create code modules and operator interfaces.
Application Programming Interface (API)	A set of classes, methods, and properties that you use to control a specific service, such as the TestStand Engine.

B

breakpoint	An interruption in the execution of a program.
button	A dialog box item that, when selected, executes a command associated with the dialog box.

C

call stack	The chain of active sequences that are waiting for nested subsequences to complete.
callback	A sequence that is used to handle common tasks, such as serial number inquiry or report logging.
checkbox	A dialog box input that allows you to toggle between two possible options.
code module	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.
code template	A source file that contains skeleton code. The skeleton code serves as a starting point for the development of code modules for steps that use the step type.
context menu	Menus accessed by clicking on an object. Menu options pertain to that object specifically.
control	An input and output device for entering data that appears on a panel or window.
CPU	Central processing unit.

D

dialog box	A prompt mechanism, in which you enter additional information needed to complete a command.
DLL	Dynamic Link Library.

E

engine	*See* test executive engine.
entry points	A sequence in the process model file that TestStand displays as a menu item, such as Test UUTs, Single Pass, and Report Options.
execution	An object that contains all the information TestStand needs to run a sequence, its steps, and any subsequences it calls. Typically, the TestStand Sequence Editor creates a new window for each execution.

Execution entry points — A sequence in a process model that runs tests against a UUT. Execution entry points call the MainSequence callback in the client sequence file. The default process model contains two execution entry points: Test UUTs and Single Pass. By default, Execution entry points are listed in the Execute menu. Execution entry points are only visible in the menu when the active window contains a sequence file that has a MainSequence callback.

execution pointer — A yellow pointer icon that shows the progress of execution by pointing to the currently executing step in the Steps tab.

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.

expression — A formula that calculates a new value from the values of multiple variable or properties. In expressions, you can access all variables and properties in the sequence context that is active when TestStand evaluates the expression. The following is an example of an expression:

```
Locals.MidBandFrequency = (Step.HighFrequency +
Step.LowFrequency) / 2
```

G

global variable — TestStand defines two types of globals: sequence file globals and station globals. Sequence file globals are accessible by any sequence or step in the sequence file. Station globals are accessible by any sequence file loaded on the station. The values of station global variables are persistent across different executions and even across different invocations of TestStand.

H

highlight — The way in which input focus appears on a TestStand screen; to move the input focus onto an item.

L

LabVIEW — Laboratory Virtual Instrument Engineering Workbench. A program development application based on the G programming language and used commonly for test and measurement purposes.

M

Main sequence	The sequence that initiates the tests on a UUT. The process model invokes the Main sequence as part of the overall testing process. The process model defines what is constant about your testing process, whereas the Main sequence defines the steps that are unique to the different types of tests you run.
menu bar	Horizontal bar that contains names of main menus.
method	Performs an operation or function on an object.
MFC	Microsoft Foundation Class Library
modal dialog box	A dialog box that you must dismiss before you can operate other application windows.
module adapter	A component that the TestStand Engine uses to invoke code in another sequence or in a code module, such as LabVIEW. When invoking code in a code module, the adapter knows how to call it, and how to pass parameters to it.

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.

O

object	A service that an ActiveX server makes available to clients.
operator interface	A program that provides a graphical user interface for executing sequences at a production station. Sometimes the sequence editor and operator interfaces are different aspects of the same program.

P

post actions — Actions that TestStand takes depending on the pass/fail status of the step or a custom condition the engine evaluates after executing a step. Post actions allow you to execute callbacks or jump to other steps after executing the step.

preconditions — A set of conditions for a step that must be true for TestStand to execute the step during the normal flow of execution in a sequence.

process model — A sequence file you designate that performs a standard 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 container of information, which stores and maintains a setting or attribute of an object. A property can contain a single value, an array of values of the same type, or no value at all. A property can also contain any number of subproperties. Each property has a name.

R

RAM — Random-access memory.

run-time error — An error condition that forces an execution to terminate. When the error occurs while running a sequence, TestStand jumps to the Cleanup step group, and the error propagates to any calling sequence up through to the top-level sequence.

S

sequence — A series of steps that you specify for execution in a particular order. Whether and when a step is executed can depend on the results of previous steps.

sequence context — A TestStand object that contains references to all global variables and all local variables and step properties in active sequences. The contents of the sequence context changes depending on the currently executing sequence and step.

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 global variables	Variables you can use to store data relevant to the entire sequence file. Each sequence and step in the sequence file can directly access these globals.
Sequence File window	A separate window within the sequence editor that a sequence file appears in.
step	Any action, such as calling a test module to perform a specific test, that you can include within a sequence of other actions.
step group	A set of steps in a sequence. A sequence contains the following groups of steps: Setup, Main, and Cleanup. When TestStand executes a sequence, the steps in the Setup step group execute first, the steps in the Main step group execute next, and the steps in the Cleanup step group last.
step result	A container property that contains a copy of the subproperties from the Result property of a step and additional execution information such as the name of the step and its position in the sequence. TestStand automatically creates a step result as each step executes and places the step result into a result list which TestStand uses to generate its reports.
step status	A string value that indicates the status of a step in an execution. Every step in TestStand has a Result.Status property. Although TestStand imposes no restrictions on the values to which the step or its code module can set the status property, TestStand and the built-in step types use and recognize a predefined set of values.
step type	A component that defines a set of custom step properties and standard behavior for each step of that type. All steps of the same 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 another sequence calls. You specify a subsequence call as a step in the calling sequence.

T

template	*See* code template.
terminate	To stop an execution by halting the normal execution flow, and running all the Cleanup step groups in the sequences on the call stack.
test executive engine	A module or set of modules that provide an API for creating, editing, executing, and debugging sequences. A sequence editor or operator interface uses the services of a test executive engine.

U

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

V

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

W

window	A working area that supports specific tasks related to developing and executing programs.

Index

L

M

N

O

P

R

S

T

V

W