NI TestStand™

Using TestStand

May 2008
373436C-01

NATIONAL INSTRUMENTS™

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 43 662 457990-0, Belgium 32 (0) 2 757 0020, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 5050 9800, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 358 (0) 9 725 72511, France 01 57 66 24 24, Germany 49 89 7413130, India 91 80 41190000, Israel 972 3 6393737, Italy 39 02 41309277, Japan 0120-527196, Korea 82 02 3451 3400, Lebanon 961 (0) 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 (0) 348 433 466, New Zealand 0800 553 322, Norway 47 (0) 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210, Russia 7 495 783 6851, Singapore 1800 226 5886, Slovenia 386 3 425 42 00, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 (0) 8 587 895 00, Switzerland 41 56 2005151, Taiwan 886 02 2377 2222, Thailand 662 278 6777, Turkey 90 212 279 3031, United Kingdom 44 (0) 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the info code feedback.

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

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/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 a note, which alerts you to important information.
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. Italic text 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.
`monospace italic`	Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Contents

Chapter 4
Debugging Sequences

Chapter 5
Using Variables and Properties

Chapter 6
Using Callbacks

Chapter 7
Adding Users and Setting Privileges

Chapter 8
Interactive Executions

Chapter 9
Calling Sequences Dynamically

Chapter 10
Customizing Reports

Appendix A
Technical Support and Professional Services

Index

1

Introduction to TestStand

NI TestStand is a flexible and open test management framework for building, customizing, and deploying a full-featured test management system.

If you are using TestStand for the first time, National Instruments recommends that you begin by completing the tutorials in this manual. The *NI TestStand Reference Manual* assumes you are familiar with the concepts and tutorials in this manual.

Refer to the *NI TestStand Reference Manual* and to the *NI TestStand Help* for more information about TestStand features. Select **Start»All Programs»National Instruments»TestStand *x.x*»Documentation» NI TestStand Help** or select **Help»NI TestStand Help** in the TestStand Sequence Editor to access the *NI TestStand Help*.

The tutorials in this manual begin with a general introduction to the TestStand Sequence Editor and continue by teaching you how to build sequences. Because the steps of the tutorials depend on elements from previous tutorials, National Instruments recommends following the chapter outline in order.

TestStand Overview

The *NI TestStand System and Architecture Overview Card* includes a complete architectural diagram of TestStand, descriptions of the various system components, and diagrams that illustrate sequence file execution flow, `Execution` object structure, and sequence file structure. Use this card to familiarize yourself with TestStand and refer to it while you review this manual.

Major Software Components of TestStand

The major software components of TestStand include the TestStand Engine, the TestStand Sequence Editor, user interfaces, module adapters, and process models.

TestStand Engine

The TestStand Engine is a set of Dynamic Link Libraries (DLLs) that export an extensive ActiveX Automation Application Programming Interface (API) for creating, editing, running, and debugging sequences. The TestStand Sequence Editor and User Interface (UI) Controls use the TestStand API. You can access the TestStand API from any programming environment that supports access to ActiveX Automation servers, which allows you to call the TestStand API from user interfaces and code modules. The *NI TestStand Help* includes the documentation for the TestStand API.

TestStand Sequence Editor

The TestStand Sequence Editor is a development environment you use to create, modify, execute, and debug sequences. You can also use the sequence editor to modify step types and process models. The sequence editor provides debugging tools similar to those you find in LabVIEW, LabWindows™/CVI™, and Microsoft Visual Studio, including breakpoints, single-stepping, stepping into or over function calls, tracing, a variables view, a Watch Expression view, and an Output view.

In the sequence editor, you can start multiple concurrent executions, execute multiple instances of the same sequence, and execute different sequences at the same time. Separate Execution windows show each execution. In trace mode, the Execution window shows the steps in the currently executing sequence. When you suspend an execution, the Execution window shows the next step to execute and provides debugging options.

You can configure and save the sequence editor layout of dockable panes and tabbed windows, and you can reset the layout to the default configuration by selecting **View»Reset UI Configuration**. Refer to the *NI TestStand Help* for more information about working with panes in the sequence editor.

User Interfaces

A TestStand User Interface is an application that provides a graphical user interface (GUI) for executing sequences on a production station. User interfaces are intended for use with deployed custom sequence editors or test systems and are designed to protect the integrity of test sequences. Refer to the *Features Comparison of Sequence Editor and User Interfaces* section of Chapter 1, *NI TestStand Architecture*, of the *NI TestStand Reference Manual* for more information about custom sequence editors, which are user interfaces in Editor Mode.

You can run user interfaces in Operator Mode or in Editor Mode. With the user interfaces in Operator Mode, you can start multiple concurrent executions, set breakpoints, and single-step. In Editor Mode, you can perform the same tasks as in Operator Mode and you can also modify sequences, create and modify sequence variables, sequence parameters, step properties, and so on.

TestStand includes separate user interface applications developed in LabVIEW, LabWindows/CVI, Microsoft Visual Basic .NET, C#, and C++ (MFC). Because TestStand includes the source code for each user interface, you can fully customize the user interfaces. You can also create a user interface using any programming language that can host ActiveX controls or control ActiveX Automation servers.

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

Module Adapters

Steps in a TestStand sequence can invoke code in another sequence or in a code module. When invoking code in a code module, TestStand uses a module adapter to determine the type of the code module, the calling conventions for the code module, the list of parameters the code module requires, and how to pass parameters to the code module.

The module adapters can open specific application development environments (ADEs), create source code for a new code module in an ADE, and show the source for an existing code module in an ADE.

You can use the following types of code modules:

- LabVIEW Virtual Instruments (VIs)
- C functions in source code, object files, or library modules created in LabWindows/CVI
- C/C++ functions in DLLs
- Objects in .NET assemblies
- Objects in ActiveX Automation servers
- Subroutines in HTBasic

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 located 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 or COM server.
- **HTBasic Adapter**—Calls HTBasic subroutines.
- **Sequence Adapter**—Calls other TestStand sequences with parameters.

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

Refer to the *Using LabVIEW with TestStand* manual for more information about using LabVIEW and the LabVIEW Adapter with TestStand. Refer to the *Using LabWindows/CVI with TestStand* manual for more information about using LabWindows/CVI and the LabWindows/CVI Adapter with TestStand.

Process Models

Testing a unit under test (UUT) requires more than just executing a set of tests. The test management system must also perform a series of operations before and after the test sequence executes to handle common test system tasks, such as identifying the UUT, notifying the operator of pass/fail status, generating a test report, and logging results. These operations define the testing process, and the set of operations and the flow of execution is called a process model.

With process models, you can write different test sequences without repeating standard testing operations in each sequence. You can modify the process model to suit the unique testing needs of a production line, production site, or company systems and practices. With TestStand, you can define a process model in the form of a sequence file, and you can edit a process model just as you edit other sequence files.

TestStand includes a Sequential process model, a Parallel process model, and a Batch process model. Use the Sequential model to run a test sequence on one UUT at a time. Use the Parallel and Batch models to run the same test sequence on multiple UUTs at the same time.

Each process model defines a set of Execution entry points. Each entry point is a sequence in the process model file. Multiple entry points in a process model allow different options for invoking test sequences. For example, the Sequential model, which is the default TestStand process model, includes the Test UUTs and Single Pass Execution entry points to invoke test sequences. The Test UUTs entry point initiates a loop that repeatedly identifies and tests UUTs. The Single Pass entry point tests a single UUT without identifying it. You can log results and produce reports with both entry points.

Refer to Appendix A, *Process Model Architecture*, of the *NI TestStand Reference Manual* for more information about process models.

2

Loading and Running Sequences

The TestStand Sequence Editor includes different windows where you load and run sequences.

Starting TestStand

Complete the following steps to launch the TestStand Sequence Editor.

1. Select **Start»All Programs»National Instruments»TestStand *x.x*» Sequence Editor**. The sequence editor launches the main window and the Login dialog box.
2. Use the default user name, `administrator`, in the User Name ring control. Leave the Password field empty.
3. Click **OK**.

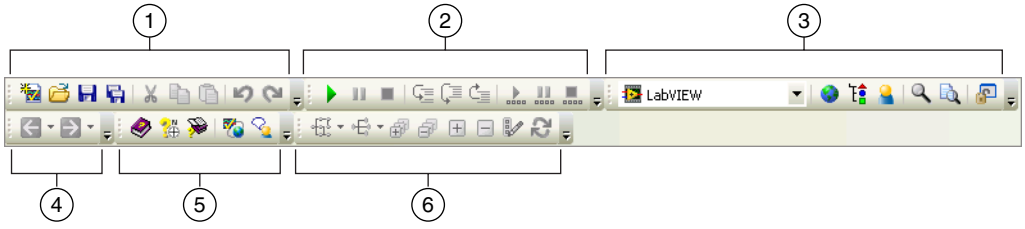
The sequence editor has four main parts—the menu bar, the toolbar area, the development workspace, and the status bar. Refer to the *NI TestStand Reference Manual* and to the *NI TestStand Help* for more information about the sequence editor.

Menu Bar

By default, the menu bar contains the File, Edit, View, Execute, Debug, Configure, Source Control, Tools, Window, and Help menus. The menus are fully customizable. You can create menus with any TestStand Sequence Editor commands you want. Browse the menus in the sequence editor to familiarize yourself with the contents of each menu. Refer to the *NI TestStand Help* for more information about each menu item and how to customize the menus.

Toolbar Area

By default, the toolbar area contains six toolbars with shortcuts to commonly used selections from the menu bar, as shown in Figure 2-1.



1	Standard Toolbar	4	Navigation Toolbar
2	Debug Toolbar	5	Help Toolbar
3	Environment Toolbar	6	Sequence Hierarchy Toolbar

Figure 2-1. Sequence Editor Toolbars

- **Standard Toolbar**—Contains buttons for creating, loading, and saving sequence files, as well as for cutting, copying, and pasting. This toolbar also includes the Undo and Redo buttons.
- **Debug Toolbar**—Contains buttons for executing a sequence, stepping into, stepping over, stepping out of, terminating, and pausing executions.
- **Environment Toolbar**—Contains the Adapter ring control; buttons for opening station globals, type palettes, and the User Manager; buttons for performing search and replace operations; and a button for unlocking and locking the UI to enable or disable customizing various aspects of the UI in the development environment.
- **Navigation Toolbar**—Contains the back and forward buttons, which you use to show the previous or next view in the history list of the sequence file or sequence hierarchy.
- **Help Toolbar**—Contains buttons to launch the *NI TestStand Help*, the NI TestStand support page of `ni.com`, and the NI TestStand Discussion Forum on `ni.com`.
- **Sequence Hierarchy Toolbar**—Contains buttons for the Sequence Hierarchy window, which graphically shows the relationship between sequences and subsequences. Use the Sequence Hierarchy window to inspect and navigate a complex group of sequences so you can more easily modify, debug, and maintain test sequences.

The toolbar area is fully customizable. You can create toolbars with buttons for any environment commands you want. Refer to the *NI TestStand Help* for more information about customizing toolbars.

Development Workspace

The development workspace is the main area of the sequence editor and contains tabbed windows and panes you can float, hide, dock, and resize. The sequence editor uses windows to show sequence files, sequence file differences, executions, sequence hierarchies, station globals, types, and the user manager. The sequence editor uses panes to show all other information.

The sequence editor can show multiple files as tabs or as separate windows. You can use the mouse to resize the boundary between two panes. You can also rearrange panes using drag and drop, and you can configure the display state of panes to dock, hide, float, or auto-hide the pane.

You can configure and save the layout of the development workspace, menus, and toolbars. Select **View»Reset UI Configuration** to reset the layout to the default configuration state.

The sequence editor also includes an Insertion Palette you can use to insert steps into sequences. The Insertion Palette contains a Step Types list and a Templates list. The Step Types list shows all available step types, and the Templates list acts as a clipboard of preconfigured steps, variables, and sequences you commonly use. You can reuse the steps, variables, and sequences as you develop sequence files.

Refer to the *NI TestStand Help* for more information about configuring the panes within the development workspace and using keyboard navigation in the panes, windows, and controls in the sequence editor. Click the **Help Topic** (**F1**) button, shown at left, in the toolbar to launch the *NI TestStand Help* for specific information about the active pane, tab, or window in the sequence editor.

Status Bar

The status bar shows the following information in the sequence editor:

- User name of the current user
- Name of the current process model file; double-click in this area to open the current process model
- Number of steps you have selected
- Total number of steps in the active sequence
- Location of the report after execution

Understanding Sequences and Sequence Files

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

A sequence file can contain any number of sequences along with sequence file global variables, which you can use to share data among multiple sequences within a sequence file.

The Steps pane for an individual sequence includes the Setup, Main, and Cleanup step groups. The steps in the Setup step group execute before the steps in the Main step group. These steps typically perform operations such as initializing and configuring instruments, fixtures, and UUTs. The steps in the Main step group typically perform UUT test operations. The steps in the Cleanup group execute after the steps in the Main step group or when a sequence terminates. These steps typically perform operations such as powering down or releasing handles to instruments, fixtures, and UUTs.

A sequence can have parameters and any number of local variables. Use parameters to pass and receive values between sequences. Use local variables to hold values, maintain counts, hold intermediate values, and perform any other type of local data storage within a sequence.

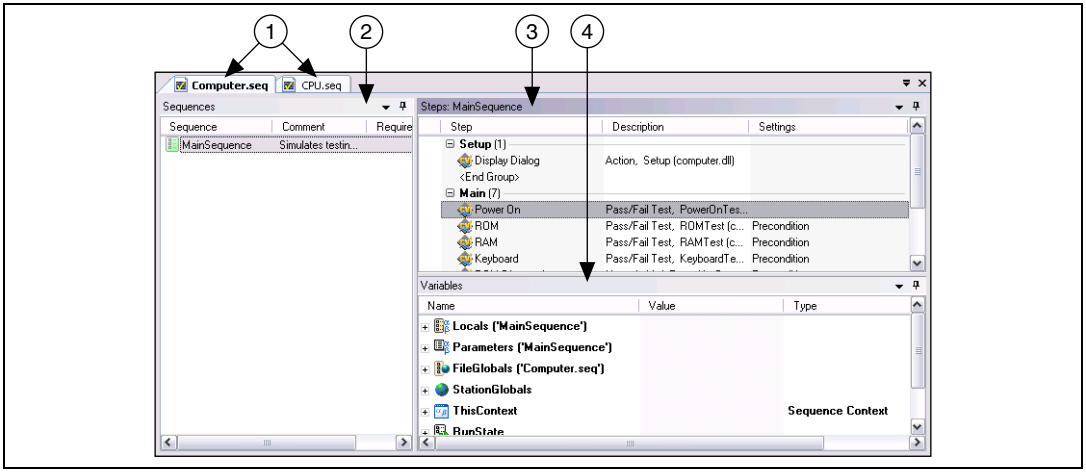
Use the Variables pane in the sequence editor to show and modify local variables, parameters, sequence file global variables, and station global variables. You can also use the Variables pane to show sequence context and run state information when executing sequences. Refer to the *Sequence Context* section of Chapter 3, *Executions*, of the *NI TestStand Reference Manual* for more information about the sequence context.

Loading a Sequence File

Complete the following steps to load and view a sequence file.

1. Select **File»Open Sequence File** and navigate to the `<TestStand Public>\Tutorial` directory. `<TestStand Public>` is located by default at `C:\Documents and Settings\All Users\Documents\National Instruments\TestStand x.x` on Windows 2000/XP and at `C:\Users\Public\Documents\National Instruments\TestStand x.x` on Windows Vista.
2. Select `Computer.seq` and click **Open**. The `Computer.seq` sequence file is a simulated test of a computer in which you can designate various hardware components to fail. The sequence runs tests that are functions in a dynamic link library (DLL) written with LabWindows/CVI.

   The sequence file opens in the Sequence File window, which includes the Steps pane, the Sequences pane, and the Variables pane, as shown in Figure 2-2. Use the Sequences pane to view the list of all sequences in the file and to select the sequence to show in the Steps pane. You can load multiple sequence files into the sequence editor, which shows each file in a separate Sequence File window.



1 Sequence File Windows	2 Sequences Pane	3 Steps Pane	4 Variables Pane

Figure 2-2. Sequence File Window

3. Select `MainSequence` in the Sequences pane.
4. Browse the contents of each pane in the Sequence File window. Ensure that the Sequences pane and the Steps pane are visible when you finish.

Running a Sequence

When you run a sequence in the sequence editor, you initiate an execution. You can run a sequence directly or 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 execution.

When you initiate an execution, the sequence editor opens an Execution window. Use the Execution window to view, or trace, steps as they execute, to monitor the values of variables and properties, and to examine the test report when the execution completes.

In trace mode, which is configurable and enabled by default, the Execution window shows the steps in the currently executing sequence. When you suspend the execution, the Execution window shows the next step to execute and provides debugging options.

Running a Sequence Directly

Complete the following steps to run `MainSequence` in the `Computer.seq` sequence file directly.

1. Select **Execute»Run MainSequence**. The sequence editor opens an Execution window to show the sequence as it executes. The first step in the Setup step group of `MainSequence` launches the Test Simulator dialog box, as shown in Figure 2-3. The Test Simulator dialog box prompts you to designate the computer component(s), if any, you want to fail during the execution.

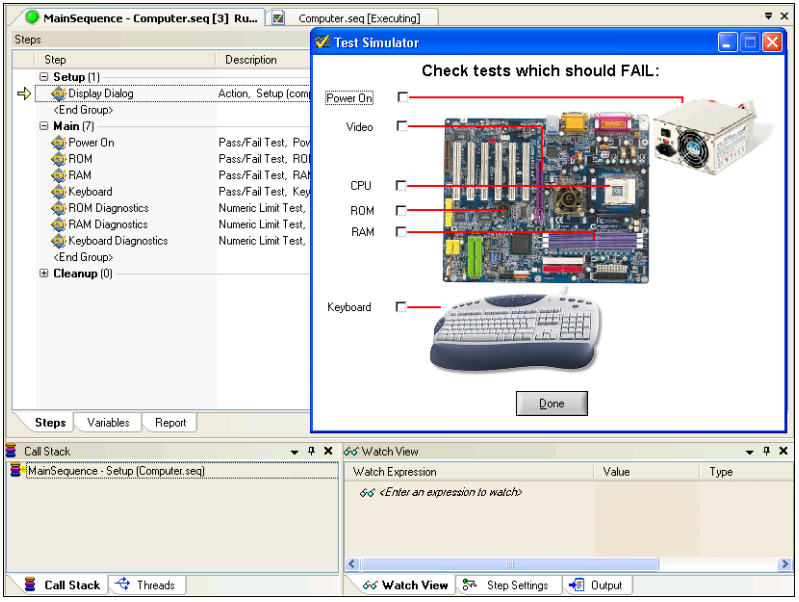


Figure 2-3. MainSequence Execution Window

2. Place a checkmark in the **RAM** test checkbox.
3. Click **Done**.
4. Observe the Execution window as it traces through the steps TestStand runs. The sequence editor uses a yellow pointer icon, called the execution pointer, to the left of the currently executing step in the Steps pane to indicate the progress of an execution. After the execution completes, the Execution window dims and the Status column of the `RAM` test contains the value `Failed`.

Note If you did not see the execution pointer, select **Configure»Station Options** to enable the tracing option. On the Execution tab, enable the **Enable Tracing** and **Allow Tracing into Setup/Cleanup** options and use the **Speed** slider control to adjust the tracing speed to the level you want. Click **OK** to close the Station Options dialog box.

5. Right-click the Execution window tab and select **Close** from the context menu or select **Window»Close Completed Executions** to close the Execution window. Repeat steps 1 through 3 to rerun the sequence.

Running a Sequence Using the Sequential Process Model

In addition to executing sequences directly, you can execute sequences using an Execution entry point, which is a sequence in a process model sequence file. Executing an Execution entry point performs a series of operations before and after calling the Main sequence of the sequence file. Common operations of the process model include identifying the UUT, notifying the operator of pass/fail status, generating a test report, and logging results.

The Sequential process model includes the Test UUTs and the Single Pass Execution entry points.

Test UUTs Execution Entry Point

The Test UUTs Execution entry point executes the sequence in a loop, prompting for a serial number at the beginning of each loop iteration and appending the results to the test report at the end of each iteration. After the loop completes, TestStand shows the full report.

Single Pass Execution Entry Point

The Single Pass Execution entry point executes the sequence once without requiring a serial number and generates a report. Use the Single Pass Execution entry point when you want to debug tests or determine if the sequence execution proceeds as you intended.

Complete the following steps to run `MainSequence` in the `Computer.seq` sequence file using the Test UUTs Execution entry point of the Sequential model.

1. Select **Configure»Station Options** to launch the Station Options dialog box.
2. Click the **Model** tab, select `SequentialModel.seq` from the Station Model ring control to select the Sequential model as the default process model, and click **OK**.
3. Select **Execute»Test UUTs**. Before executing the steps in `MainSequence`, the process model sequence launches a UUT Information dialog box that requests a serial number. Enter any number and click **OK**.
4. Use the checkboxes in the Test Simulator dialog box to select any test other than the Video or CPU tests to fail. You can also allow all the tests of the UUT to pass.

5. Click **Done**. Observe the Execution window as the sequence executes. After completing the steps in `MainSequence`, the process model shows a banner that indicates the result of the UUT.
6. Click **OK** to close the UUT Result banner. The process model generates a report, which TestStand shows after testing all the UUTs, and launches the UUT Information dialog box again.
7. Repeat steps 3 through 6 using several different serial numbers.
8. Click **Stop** in the UUT Information dialog box to stop the loop and complete the execution. After the execution completes, TestStand shows a full report in the Report pane of the Execution window for all the tested UUTs.
9. Examine the test report that includes the results for each UUT and close the Execution window.

Running a Sequence Using the Batch Process Model

The Batch process model executes the same test sequence on multiple UUTs at the same time and provides batch synchronization features, which you use to start and finish testing all UUTs in unison. For example, you can specify that because a step applies to a batch as a whole, the step should run only once per batch instead of once for each UUT.

Use the Batch model to specify that certain steps or groups of steps cannot run on more than one UUT at a time or that certain steps must run on all UUTs at the same time. The Batch model generates batch reports that summarize the test results for all the UUTs in a batch.

Complete the following steps to run the `BatchUUT.seq` sequence file using the Test UUTs Execution entry point of the Batch model.

1. Open `<TestStand Public>\Tutorial\BatchUUT.seq` and examine the steps and comments in the sequence file to determine what the sequence does.

Note You do not need to change the default process model in the sequence editor for this tutorial. The `BatchUUT.seq` sequence file always uses the Batch model, regardless of the default process model of the sequence editor. Use the Sequence File Properties dialog box to specify if a sequence file always uses a particular process model or uses the sequence editor or user interface default process model. Refer to the *NI TestStand Help* for more information about the Sequence File Properties dialog box.

2. Select **Configure»Model Options** to launch the Model Options dialog box. In the Multiple UUT Settings section, change **Number of Test Sockets** to `2` and enable the **Tile Execution Windows** option. The Number of Test Sockets control specifies the number of UUTs to test in the batch.
3. In the Batch Settings section in the Model Options dialog box, select **Don't Synchronize** from the Default Batch Synchronization ring control and click **OK**.
4. Select **Execute»Test UUTs**. Before executing the steps in `MainSequence`, the process model sequence launches the UUT Information dialog box for the Batch model, which requests a batch serial number and UUT serial numbers for each test socket. You can also disable test sockets in the UUT Information dialog box.
5. Enter any batch serial number and UUT serial numbers. Click **Go**. Click **OK** in the Batch Model Example dialog box. TestStand launches several different front panels to indicate the progress of the executions. Click **OK** to continue.

   After completing the steps in `MainSequence`, the process model shows a banner that indicates the result of the UUTs. You can view the batch report and individual UUT reports.
6. Click **View Batch Report**. You can select Entire File to show all tested batches or Current Only to show the most recently tested batch. Select **Current Only**. TestStand launches an external viewer to show the reports. By default, XML and HTML reports use Microsoft Internet Explorer, and text reports use Microsoft Notepad. Close Internet Explorer and click **Next Batch** in the Batch Results dialog box.

Note To use another application as an external viewer, select **Configure»External Viewers** and follow the instructions in the Configure External Viewers dialog box. Refer to the *Using External Report Viewers* section of Chapter 10, *Customizing Reports*, for more information about using external viewers. Refer to the *NI TestStand Help* for more information about the Configure External Viewers dialog box.

7. Repeat steps 5 and 6 for several different batches.
8. Click **Stop** in the UUT Information dialog box to complete the execution. After the execution completes, TestStand shows test reports in the Report pane of the Execution window for all batches and UUTs.
9. Examine the reports that include results for each batch and UUT and select **Window»Close All Windows**.

3

Editing Steps in a Sequence

You can add a step to a sequence, specify a code module for a step, configure the properties of a step, and call a subsequence from a sequence.

The Insertion Palette contains a set of predefined step types you use to add steps to sequences. Step types define a set of 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.

When you build sequence files, you can also use the Templates list in the Insertion Palette. The Templates list acts as a clipboard of preconfigured steps, variables, and sequences you commonly use. You can reuse the steps, variables, and sequences as you develop sequence files.

Refer to Chapter 4, *Built-In Step Types*, of the *NI TestStand Reference Manual* for more information about TestStand step types.

Adding a New Step

Complete the following steps to add a Pass/Fail step to a sequence and configure the step to call a function in a LabWindows/CVI DLL code module by specifying the module adapter to use. TestStand does not need the source code to invoke a DLL in a code module. Instead, TestStand uses module adapters to determine the type of code module, how to call the code module, the list of parameters the code module requires, and how to pass parameters to the code module.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.

2. Click the **LabWindows/CVI** icon, shown at left, at the top of the Insertion Palette to specify the module adapter the step uses. You can also select adapters from the Adapter ring control on the Environment toolbar. The adapter you select applies only to the step types that can use the module adapter.

   When you insert a step in a sequence, TestStand assigns the adapter you selected from the Insertion Palette to the step. The icon for the step reflects the adapter you selected. If you select <None> as the adapter and then insert a step, the step you insert does not call a code module.

Use the **General** panel on Properties tab of the Step Settings pane to change the associated adapter after you insert the step.

3. On the Insertion Palette, select **Tests»Pass/Fail Test** and drag the step below the `RAM` step to add a Pass/Fail Test step. You can also select the `RAM` step and double-click the step on the Insertion Palette to add it, or you can right-click the `RAM` step in the Steps pane and select **Insert Step** from the context menu to add a step. By default, the name of the new test is `Pass/Fail Test`.

   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 if the step passed or failed.

4. Rename the step `Video Test`. Select the name in the Steps pane and type the new name, select the step in the Steps pane and press the **<F2>** key, or change the Name field on the Properties tab of the Step Settings pane of the `Pass/Fail Test` step to rename the step.

Specifying the Code Module

A code module is 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.

Complete the following steps to specify the code module the step executes and to specify the parameters for a function in the code module.

1. Select the `Video Test` step and click the **Module** tab of the Step Settings pane.

2. Click the **Browse** button, shown at left, and select `<TestStand Public>\Tutorial\computer.dll` and click **Open**. When you select a DLL, TestStand reads the type library and exported information of the DLL and lists the functions TestStand can call in the Function ring control.
3. Select **VideoTest** from the Function ring control. TestStand uses the prototype information stored in the type library or the exported information of the DLL to populate the Parameters Table control.
4. In the Value Expression column of the **result** parameter, type `Step.Result.PassFail` or click the **Expression Browser** button to select the variable in the Expression Browser dialog box. TestStand returns from calling the `VideoTest` function and inserts the value from the **result** parameter into the `Result.PassFail` property of the step.

Refer to the *NI TestStand Help* for more information about the Module tab. Refer to the *Using LabWindows/CVI with TestStand* manual for more information about calling LabWindows/CVI code modules from TestStand.

Configuring Step Properties

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

- When to load the code module for the step
- When to execute the step
- What information TestStand examines to determine the status of the step
- If TestStand executes the step in a loop
- What conditional actions occur upon step completion

Use the Properties tab of the Step Settings pane to configure step properties. You can select multiple steps and configure the properties each step has in common at the same time. Refer to the *NI TestStand Help* for more information about the Properties tab of the Step Settings pane.

Complete the following steps to examine and modify step properties.

1. Select the `Video Test` step and click the **Properties** tab of the Step Settings pane. Click **Preconditions** on the Properties tab of the Step Settings pane to show the Preconditions panel. A precondition specifies the conditions that must evaluate to `True` for TestStand to execute a step during the normal flow of execution in a sequence, such as running a step only if a previous step passes.
2. Complete the following steps to define a precondition so the `Video Test` step executes only if the `Power On` step passes.

   a. Click the **Preconditions Builder** button, shown at left, to launch the Preconditions Builder dialog box.

   b. In the Insert Step Status section, select the `Power On` step from the list of step names for the Main step group and click the **Insert Step Pass** button. The Conditions 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 Builder dialog box and confirm that the Preconditions panel matches the settings shown in Figure 3-1.

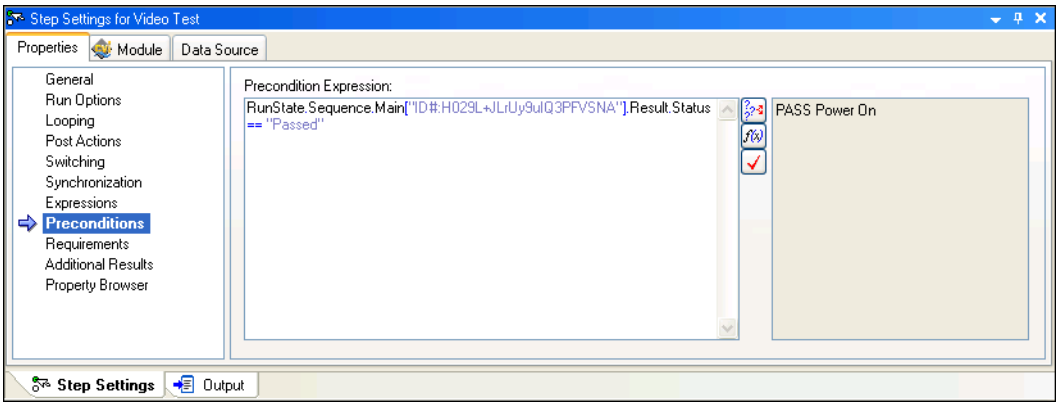


Figure 3-1. Preconditions Pane

3. Click **Post Actions** on the Properties tab of the Step Settings pane to specify 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.

   For custom conditions, you can enable the Use Custom Condition option, enter a custom condition expression that evaluates to `True` or `False`, and select the appropriate actions from the On Condition True and On Condition False ring controls.

4. Select **Terminate execution** from the On Fail ring control.

5. Click **Looping** on the Properties tab of the Step Settings pane to configure an individual step to run repeatedly in a loop when the step executes. Use the Loop Type ring control in the Looping panel to select the type of looping for the step. TestStand determines the final status of the step based on the number of passes, failures, or loop iterations that occur.

6. In the Looping panel, enter the following values into the corresponding controls. When you change the value of a property, TestStand shows the new value in bold to indicate that it differs from the default value.

   - **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 10 times and sets the overall status for the step to `Failed` if less than 8 of the 10 iterations pass.

7. Confirm that the Settings column on the Steps pane of the Sequence File window shows that the `Video Test` step contains Loop, Precondition, and Post Action settings. Use the tooltip in the Settings column to view the values for each setting.
8. Select **File»Save As** and save the sequence as `Computer2.seq` in the `<TestStand Public>\Tutorial` directory.

Note When you must save a file, this manual specifies the suggested name. If other users will use the tutorial files on the same computer, save the files with unique filenames.

9. Select **Execute»Single Pass** to run the sequence. If you select the Video test to fail, the sequence immediately terminates after calling the `Video Test` step 10 times in a loop. TestStand records the result of each loop iteration in the report. Close the Execution window.

Calling a Subsequence from a Sequence

Use a Sequence Call step to call another sequence within the calling sequence file or from a separate sequence file.

Complete the following steps to add a Sequence Call step to the `Computer2.seq` sequence file you created in the previous tutorial.

1. Insert a Sequence Call step after the `Power On` step and rename the step `CPU Test`.
2. On the Module tab of the Step Settings pane for the `CPU Test` step, click the **Browse** button located to the right of the File Pathname control and select `<TestStand Public>\Tutorial\CPU.seq` to specify the sequence the step invokes. You can also right-click the `CPU Test` step and select **Specify Module** from the context menu to show the Module tab of the Step Settings pane.
3. Select **MainSequence** from the Sequence ring control on the Module tab of the Step Settings pane.
4. Select **File»Save** to save the changes to the sequence file.
5. Double-click the `CPU Test` step or select **Open Sequence** from the context menu to open the sequence and show the `MainSequence` of `CPU.seq` in the Sequence File window.
6. Select **Execute»Run MainSequence** and examine the execution of the `CPU.seq` sequence file so you can recognize the `CPU.seq` sequence file later when you execute the `Computer2.seq` sequence file that calls `CPU.seq`.

7. Close the Execution window and the `CPU.seq` Sequence File window.
8. Select **Execute»Single Pass**. Select a test to fail and click **Done**. After the sequence executes, examine the test report. TestStand logs the results of the steps in the subsequence in addition to the steps from the parent sequence.
9. Close the Execution window.

Refer to the *NI TestStand Help* and to Chapter 5, *Module Adapters,* of the *NI TestStand Reference Manual* for more information about calling sequences.

Using Step Templates

The Insertion Palette contains the Step Types list and the Templates list. Use the Templates list to hold copies of steps, variables, and sequences you reuse during the development of sequence files. For example, you can add a step that calls a specific LabVIEW VI you typically use or a sequence that contains common setup steps, cleanup steps, and local variables.

Drag steps from the Steps pane, variables from the Variables pane, and sequences from the Sequences pane and drop them on the Templates list to add steps, variables, or sequences to the Templates list. Use the context menu to rename, copy, paste, delete, import, and export the items in the Templates list. Use drag and drop to rearrange the items in the list. Select **Insert Folder** from the context menu to add folders to the list.

You must drag a sequence, step, or variable from the Templates list to a sequence file, make changes, drag the item back to the Templates list, and delete the original item from the Templates list to edit the item.

Do not confuse step templates with custom step types, which have similar functionality but different use cases. Refer to Chapter 13, *Custom Step Types*, of the *NI TestStand Reference Manual* for more information about when to use step templates and custom step types.

Inserting a Template Step

Complete the following steps to import a set of template steps to the Insertion Palette, insert a step from the Templates list to a sequence, and add a new step to the Templates list.

1. Right-click the Steps folder in the Templates list of the Insertion Palette and select **Import** from the context menu to navigate to the `<TestStand Public>\Tutorial` directory.
2. Select `Tutorial Templates.ini` and click **Open**. The sequence editor adds a Tutorial folder under the Steps folder.
3. Expand the Steps folder and locate and expand the `Tutorial` folder to view the following imported template steps:
   - **Retry if Previous Step Fails**—A Message Popup step that prompts you to retry the previous step on failure.
   - **Open Notepad**—A Call Executable step configured to launch the Windows Notepad application.
   - **Output Message**—A statement step that logs an output message where the logged text is the step name.
   - **Waveform Popup**—A step that calls the WaveformGraphPopup function in the LabWindows/CVI user interface library.
4. In the Templates list, select **Retry if Previous Step Fails** and drag the step below the Power On step to add the Message Popup step to the sequence.
5. Rename the step to `Retry Power On`.
6. Save the changes you made and select **Execute»Single Pass**. When the sequence editor launches the Test Simulator dialog box, select the **Power On** test to fail and click **Done** to close the dialog box. TestStand executes the Retry Power On step and launches the Step ‘Power On’ Failed dialog box because the precondition for the step determined that the previous Power On step failed.
7. Click **Retry** in the Step ‘Power On’ Failed dialog box to instruct TestStand to execute the Power On step again. The Power On step fails again, and TestStand launches the Retry ‘Power On’ Failed dialog box again.
8. Click **Continue** in the Step ‘Power On’ Failed dialog box to instruct TestStand to continue the execution.
9. Review the report the execution generates. Notice that the Power On step executes twice, and the second call to Retry Power On continues the execution.

Creating a Template Step

Complete the following steps to change the Retry if Previous Step Fails step to automatically select the Continue button if you do not respond to the prompt.

1. In the Sequence File window, click the `Retry Power On` step in MainSequence.
2. Click the **Text and Buttons** tab of the Step Settings pane.
3. Select **Button 2** in the Timeout Button ring control.
4. Enter `20` in the Time to Wait control to instruct the step to wait for 20 seconds before selecting the Continue button. This technique is useful if an operator is not present to acknowledge a non-critical message during testing.
5. Save the changes you made and select **Execute»Single Pass**. When the sequence editor launches the Test Simulator dialog box, select the **Power On** test to fail and select **Done** to close the dialog box. TestStand executes the Retry Power On step and launches the Step 'Power On' Failed dialog box. When the timeout reaches zero, the execution continues as if you clicked Continue.
6. In the Sequence File window, drag the Retry Power On step into the Tutorial folder in the Templates list.
7. Rename the new template step `Timeout Retry`. You can use the new template step in subsequent development. The sequence editor automatically saves the templates list when you shut down the sequence editor.
8. Close the Execution window and the `computer2.seq` sequence file without saving the file.
9. Right-click the Tutorial folder in the Templates list and select **Delete** from the context menu to remove the folder.

4

Debugging Sequences

The TestStand sequence debugging features include tracing, breakpoints, conditional breakpoints, single-stepping, and watch expressions. Refer to the *NI TestStand Help* for more information about debugging tools. Refer to the *Using the Watch View Pane* section of Chapter 5, *Using Variables and Properties*, for more information about watch expressions.

Step Mode Execution

Note Before you begin this tutorial, select **Configure»Station Options** and confirm that the Enable Tracing and Allow Tracing into Setup/Cleanup options are enabled on the Execution tab of the Station Options dialog box.

Complete the following steps to set a breakpoint in a sequence file.

1. Open `<TestStand Public>\Tutorial\Computer2.seq`, which you created in the previous tutorial. You can also find this file in the `<TestStand Public>\Tutorial\Solution` directory.
2. Select **Execute»Break on First Step** to suspend an execution on the first step TestStand executes. A checkmark appears to the left of the menu item to indicate that you enabled this option.
3. Select the **Cleanup** step group on the Steps pane. 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. Insert a Message Popup step in the Cleanup step group, rename the step `Cleanup Message`, and complete the following steps to configure the step.
   a. On the Text and Buttons tab of the Step Settings pane, enter the text `"Cleanup Message"`, including the quotation marks, in the Title Expression expression control.

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

b. Enter the text `"I am now in the Cleanup Step Group."`, including the quotation marks, in the Message Expression expression control.

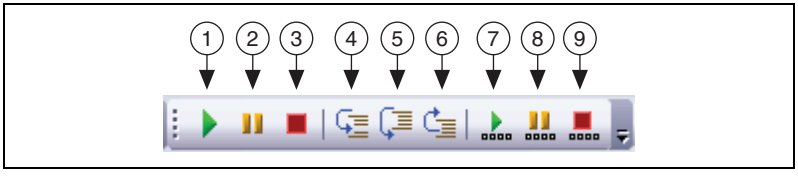
Note For this example, use the default values for all the settings on the Options tab and the Layout tab of the Step Settings pane. Refer to Chapter 4, *Built-In Step Types*, of the *NI TestStand Reference Manual* for more information about the Message Popup step type. Refer to the *NI TestStand Help* for more information about the Options and Layout tabs.

4. Select **File»Save As** and save the sequence as `Computer3.seq` in the `<TestStand Public>\Tutorial` directory.

Complete the following steps to single-step through a sequence during an execution.

1. Select **Execute»Run MainSequence**. After the execution starts, the sequence editor immediately pauses at the first step of the sequence because you enabled the Break on First Step option in step 2 of the previous tutorial. The Execution window tab includes a yellow paused icon to indicate the running state of the execution.

   When the execution pauses, you can single-step through the sequence using the Step Into, Step Over, and Step Out commands in the Debug toolbar, as shown in Figure 4-1.



1	Resume	4	Step Into	7	Resume All
2	Break	5	Step Over	8	Break All
3	Terminate Execution	6	Step Out	9	Terminate All

Figure 4-1. Single-Stepping Toolbar Buttons

   You can also find these and other debugging tools in the Debug menu of the sequence editor. Refer to the *NI TestStand Help* for more information about the single-stepping tools.

2. Click the **Step Over** button to execute the `Display Dialog` step, which launches the Test Simulator dialog box. 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 Group>`.

3. Insert a breakpoint at the `CPU Test` step. Click to the left of the step icon or right-click the step and select **Breakpoint»Toggle Breakpoint** from the context menu to insert a breakpoint. In the Execution window, a dark red stop sign icon appears to the left of the `CPU Test` step to indicate the breakpoint.

Note Right-click the stop sign icon and select **Breakpoint»Breakpoint Settings** to launch the Breakpoint Settings dialog box, in which you can configure a conditional breakpoint by specifying an expression that must evaluate to `True` for the breakpoint to be valid. Conditional breakpoints use a bright red stop sign icon in the Sequence File and Execution windows. Refer to Chapter 5, *Using Variables and Properties*, for more information about expressions.

4. Select **Debug»Resume** to continue the execution. After you make this selection, the sequence editor suspends the execution on the `CPU Test` step. Click the **Step Into** button to step into the subsequence and click the **Call Stack** pane. Figure 4-2 shows the Execution window Steps pane and Call Stack pane after you step into the subsequence.

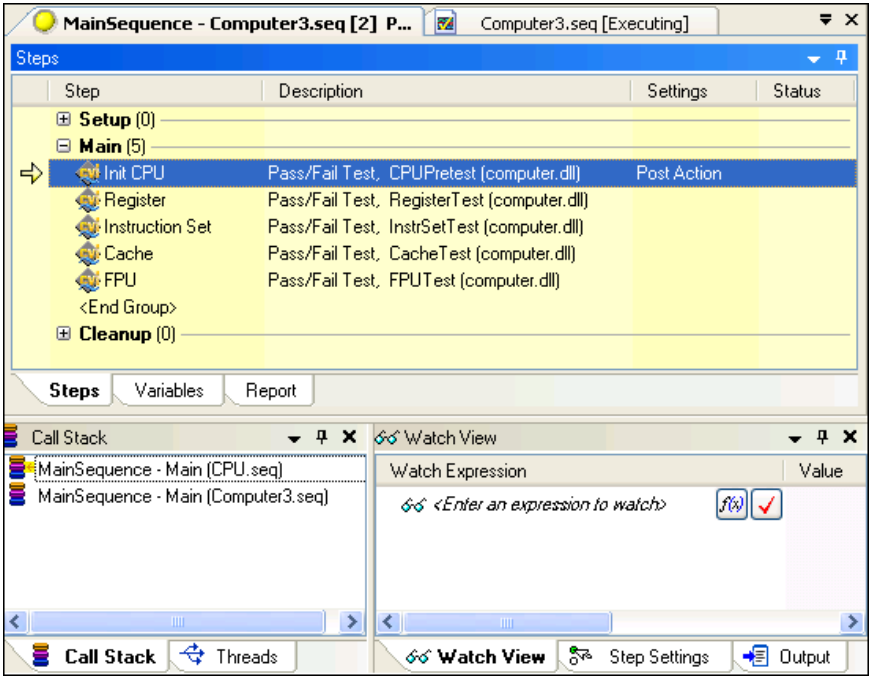


Figure 4-2. Steps Pane and Call Stack Pane while Suspended in Subsequence

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 call stack is the chain of active sequences that wait for nested subsequences to complete. The first item in the call stack is the most nested sequence invocation.

The Call Stack pane shows the call stack for the execution. A yellow pointer icon appears to the left of the most nested sequence invocation, as shown in Figure 4-2.

5. Click each sequence in the Call Stack pane to view each sequence invocation. Return to the most nested sequence invocation in the call stack.
6. Click the **Step Over** button to step through the subsequence one step at a time. Before you reach the end of the sequence, click 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.

   Continue single-stepping through the sequence by clicking the **Step Over** button until the `Cleanup Message` step you added to the Cleanup step group executes. Click **OK** to close the Cleanup Message dialog box and click the **Step Over** button to complete the execution. The Execution window dims when the execution completes. Do not close the Execution window.
7. Select **Execute»Restart** to rerun the execution. The Execution window must be the active window to restart the execution. After the sequence editor suspends the execution on the first step, select **Debug»Terminate Execution**. TestStand launches 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. Click **OK** to close the Cleanup Message dialog box.
8. Select **Execute»Restart** to rerun the execution. The Execution window must be the active window to restart the execution. After the sequence editor suspends the execution on the first step, select **Debug»Abort**. The execution of the sequence immediately stops, and TestStand does not execute any steps in the Cleanup step group. The Execution window dims when the execution completes.
9. Close the Execution window and save the sequence file before you close `Computer3.seq`.

5

Using Variables and Properties

You can create and use variables and properties and monitor the values of the variables and properties.

You can define variables in the following ways to share data among steps of a sequence or among several sequences:

- Use local variables to store data relevant to the execution of the sequence. Each step in the sequence can directly access local variables.
- Use sequence file global variables to store data relevant to the entire sequence file. Each sequence and step in the sequence file can directly access sequence file global variables.
- Use station global variables to maintain statistics or to represent the configuration of a test station.You can access station global variables from any sequence or step on the test station. Unlike with other variables, TestStand saves the values of station global variables from one TestStand session to the next.

Use the Variables pane in the sequence editor to show and modify local variables, parameters, sequence file global variables, and station global variables. You can also use the Variables pane to show sequence context and run state information when executing sequences. Refer to the *Sequence Context* section of Chapter 3, *Executions*, of the *NI TestStand Reference Manual* for more information about the sequence context.

Using Local Variables

Complete the following steps to create and use local variables. You can apply the concepts you learn in this tutorial to sequence file global and station global variables.

1. Open `<TestStand Public>\Tutorial\Computer2.seq`, which you created in Chapter 3, *Editing Steps in a Sequence*. You can also find this file in the `<TestStand Public>\Tutorial\Solution` directory.
2. Click the **Variables** pane in the Sequence File window. Expand the **Locals** item to view the local variables currently defined for `MainSequence` in `Computer2.seq`.

By default, TestStand defines only the ResultList local variable when you create a new sequence. The ResultList local variable is an array TestStand uses to store the results from the steps TestStand executes in the sequence. TestStand uses the array of step results for generating reports and for logging results to a database.

3. On the Variables pane, right-click **Locals**, select **Insert Local»Number** from the context menu to insert a new numeric local variable, and rename the local variable `LoopIndex`.
4. Complete the following steps to insert and configure a For step.
   a. Click the **Steps** pane in the Sequence File window and insert a **FlowControl»For** step below the `Power On` step. Notice that TestStand also adds an End step below the For step.
   b. Drag the `End` step below the `RAM` step. TestStand automatically indents the CPU, ROM, and RAM steps between the For and End steps.
   c. Click the `For` step and click the **For Loop** tab of the Step Settings pane.
   d. Enter `5` in the **Number of Loops** control.
   e. For the Loop Variable control, click the **Expression Browser** button, shown at left, or right-click in the Expression control and select **Browse** from the context menu to launch the Expression Browser dialog box, in which you can interactively build an expression and create variables and parameters. TestStand supports all applicable expression operators and syntax you use in C, C++, Java, and Microsoft Visual Basic. Refer to the *NI TestStand Help* for more information about the Expression Browser and the expression operators and syntax TestStand supports.

      You can also create variables directly from the Expression Browser dialog box by right-clicking inside the Variables/Properties tab and selecting the appropriate Insert item from the context menu.

Note All expression fields support type-ahead and auto-completion with drop-down lists and context-sensitive highlighting. At any point while editing an expression, press **<Ctrl-Space>** to show a drop-down list of valid expression elements.

   f. Expand the **Locals** item. Each item in the top section of the Variables/Properties tab is a property or variable of TestStand.

g. Select the `LoopIndex` variable under the `Locals` property and click the **Insert** button. The Expression Browser enters `Locals.LoopIndex` into the Expression control.

Note To refer to a subproperty, use a period to separate the name of the base property from the name of the subproperty. For example, reference the LoopIndex subproperty of the Locals property as `Locals.LoopIndex`.

h. Click the **Check Expression for Errors** button, shown at left, to verify that the expression does not contain any illegal syntax.

i. In the Expression Browser dialog box, click **OK** to return to the For Loop tab of the Step Settings pane. The Loop Variable now contains the `Locals.LoopIndex` expression. Notice that the Custom Loop section shows the expressions TestStand uses when executing the For step when using a fixed number of loops.

5. Select **File»Save As** and save the sequence as `Computer4.seq` in the `<TestStand Public>\Tutorial` directory.
6. Select **Execute»Break on First Step** to remove the checkmark that appears to the left of the menu item and disable this option.
7. Select **Execute»Single Pass** to run the sequence. Click **Done** in the Test Simulator dialog box.
8. After the sequence executes, examine the test report, which shows that TestStand executed the steps within the loop (CPU, ROM, and RAM) five times.
9. Close the Execution window.

Using the Execution Window Variables Pane

Complete the following steps to use the Variables pane of the Execution window to examine the value of the LoopIndex variable while TestStand executes the sequence.

1. Insert a breakpoint on the `End` step associated with the For loop you created in the previous tutorial.
2. Select **Execute»Single Pass** to run the sequence.
3. Click **Done** in the Test Simulator dialog box. The execution suspends on the `End` step.
4. Select the **Variables** pane of the Execution window and expand the `Locals` section.

5. Click the `LoopIndex` subproperty under the `Locals` property. The numeric value of `LoopIndex` is `0`.

   Before executing the steps in a sequence, TestStand creates a run-time copy of the sequence so parallel executions of the same sequence do not alter variable or property values in other executions. When an execution completes, TestStand discards the run-time sequence copy.

   For each active sequence, TestStand maintains a sequence context that represents the execution state of the sequence. The contents of the sequence context changes depending on the currently executing sequence and step. You can access the run-time copies and the original versions of properties and variables from a sequence context.

   The Variables pane shows the sequence context for the sequence invocation currently selected on the Call Stack pane. The sequence context contains all the variables and properties the steps in the selected sequence invocation can access. Use the Variables pane to examine and modify the values of these variables and properties.

   Table 5-1 lists the first-level properties in the sequence context. Refer to the *NI TestStand Help* and to Chapter 3, *Executions*, of the *NI TestStand Reference Manual* for more information about sequence contexts.

Table 5-1. First-Level Properties of the Sequence Context

Sequence Context Subproperty	Description
`Locals`	Contains the sequence local variables for the current sequence invocation.
`Parameters`	Contains the sequence parameters for the current sequence invocation.
`FileGlobals`	Contains the sequence file global variables for the current execution.
`StationGlobals`	Contains the station global variables for the engine invocation. TestStand maintains a single copy of the station global variables in memory.
`ThisContext`	Holds a reference to the current sequence context. Use this property to pass the entire sequence context as an argument to a subsequence or to a code module.
`RunState`	Contains properties that describe the state of execution in the sequence invocation, such as the current step, the current sequence, and the calling sequence.

6. Select **Debug»Resume**. The execution resumes and suspends at the `End` step again.
7. Click the **Variables** pane again. The value of `Locals.LoopIndex` is now `1`. Leave the execution in the paused state for the next tutorial.

Docking the Variables Pane

You can configure the sequence editor to float, hide, dock, and resize the various panes in the development workspace. Some panes, such as the Variables pane, must stay attached to the associated sequence file or execution window.

Note When you launch the TestStand Sequence Editor for the first time or when you reset the development workspace configuration, depending on the screen resolution, the layout of the development workspace defaults to the Small Screen Example configuration, in which the Variables pane is a tab next to the Steps pane, or the Large Screen Example configuration, in which the Variables pane is to the right of the Steps pane.

Select **Configure»Sequence Editor Options** to launch the Sequence Editor Options dialog box to select a configuration. On the UI Configuration tab, select an example configuration in the **Saved Configurations** control and click the **Load Selected** button. Click **OK** to close the Sequence Editor Options dialog box.

Complete the following steps to make the Steps and Variables panes visible at the same time.

1. Click the **Variables** pane tab and drag it to the right using the tab, not the titlebar. Notice that when you perform this operation, the sequence editor shows docking guides to show you where you can place the pane within the Execution window. As you drag the mouse over each of the docking guides, the sequence editor highlights where the pane would appear if dropped.

2. Drop the pane on the right-most docking guide, shown at left, so the pane appears to the right of the Steps and Report panes.
3. Click the **Steps** pane tab to hide the Reports pane so you can see the Steps pane and the Variables pane at the same time.
4. Select **Debug»Resume**. The execution resumes and suspends at the `End` step again. The value of `Locals.LoopIndex` in the Variables pane is now `2`.

5. Drag the titlebar of the Variables pane and drop it on the center docking guide, shown at left, in the middle of the Steps pane to return the Variables pane to its previous location. Notice that the Variables tab is to the left of the Steps tab instead of to the right.
6. Click the **Variables** pane tab and drag the tab to the right and drop it on the Steps pane tab to adjust the order of the tabs. Leave the execution in the paused state for the next tutorial.

Note Select **View»Reset UI Configuration** at any time to restore the panes to the original state.

Using the Watch View Pane

Use the Watch View pane to monitor values of complex expressions and of specific variables and properties without viewing the entire sequence context as you trace or single-step through a sequence. To easily specify the variables or properties you want to monitor, you can drag individual variables or properties from the Variables pane to the Watch View pane. TestStand updates the values on the Watch View pane when execution suspends at a breakpoint. If you enabled tracing, TestStand also updates the values after executing each step and highlights the values that change.

Complete the following steps to create a complex watch expression that uses the `LoopIndex` property.

1. Click the `LoopIndex` property on the Variables pane of the Execution window and drag the property to the Watch View pane. The value of the `LoopIndex` watch expression is `2`.
2. Edit the existing watch expression directly in the Watch Expression column on the Watch View pane to change the expression to the following:

```
Str (Locals.LoopIndex * 20) + "%"
```

3. Resume the execution. When the execution suspends on the `End` step again, the value of the watch expression changes from `40%` to `60%`.
4. Click the breakpoint icon to the left of the `End` step in the Execution window to remove the breakpoint.
5. Resume and complete the execution.
6. Close the Execution window.
7. Save the sequence file.

8. Select **Debug»Breakpoints/Watches** to launch the Edit Breakpoints/Watch Expressions dialog box, in which you can edit and delete breakpoints and watch expressions you previously created.
9. Review the contents of the Breakpoints and Watch Expressions tabs in the Edit Breakpoints/Watch Expressions dialog box, delete the expression you created for this tutorial if you want, and click **Done** to close the dialog box.

You can also right-click the Watch View pane and select **Insert Watch Expression** from the context menu or right-click the expression and select **Edit Watch Expression** from the context menu to launch the Watch Expression Settings dialog box, in which you can create and edit expressions.

Refer to the *NI TestStand Help* for more information about the Watch View pane and the Watch Expression Settings dialog box. Refer to Chapter 3, *Executions*, of the *NI TestStand Reference Manual* for more information about using variables and properties.

6

Using Callbacks

You can customize the execution of a sequence using callbacks, which are sequences for handling common tasks, such as prompting for serial numbers or logging to reports.

TestStand includes source code for the default callback sequences so you can modify them to meet specific needs.

Process Model Callbacks

TestStand process models contain sequences that define the operations TestStand performs before and after testing a UUT. You can use an execution entry point, such as Test UUTs or Single Pass, to invoke sequences within a process model to execute a sequence in a sequence file you create. The process models also contain Model callbacks, which are sequences you can override to customize the behavior of a process model without editing the process model directly.

The TestStand process models define a TestReport callback that generates the test report for each UUT. In most cases, the TestReport callback in the process model file is sufficient because the TestReport callback handles many types of test results. You can, however, override the default TestReport callback with a custom TestReport callback in a sequence file you create. When you use a process model execution entry point, such as Test UUTs or Single Pass, to run a sequence file, the process model calls the custom TestReport callback in the sequence file instead of the default TestReport callback in the process model.

Figure 6-1 shows the callbacks the TestStand default Sequential process model calls and the order in which TestStand executes the callbacks within the Test UUTs and Single Pass Execution entry points.

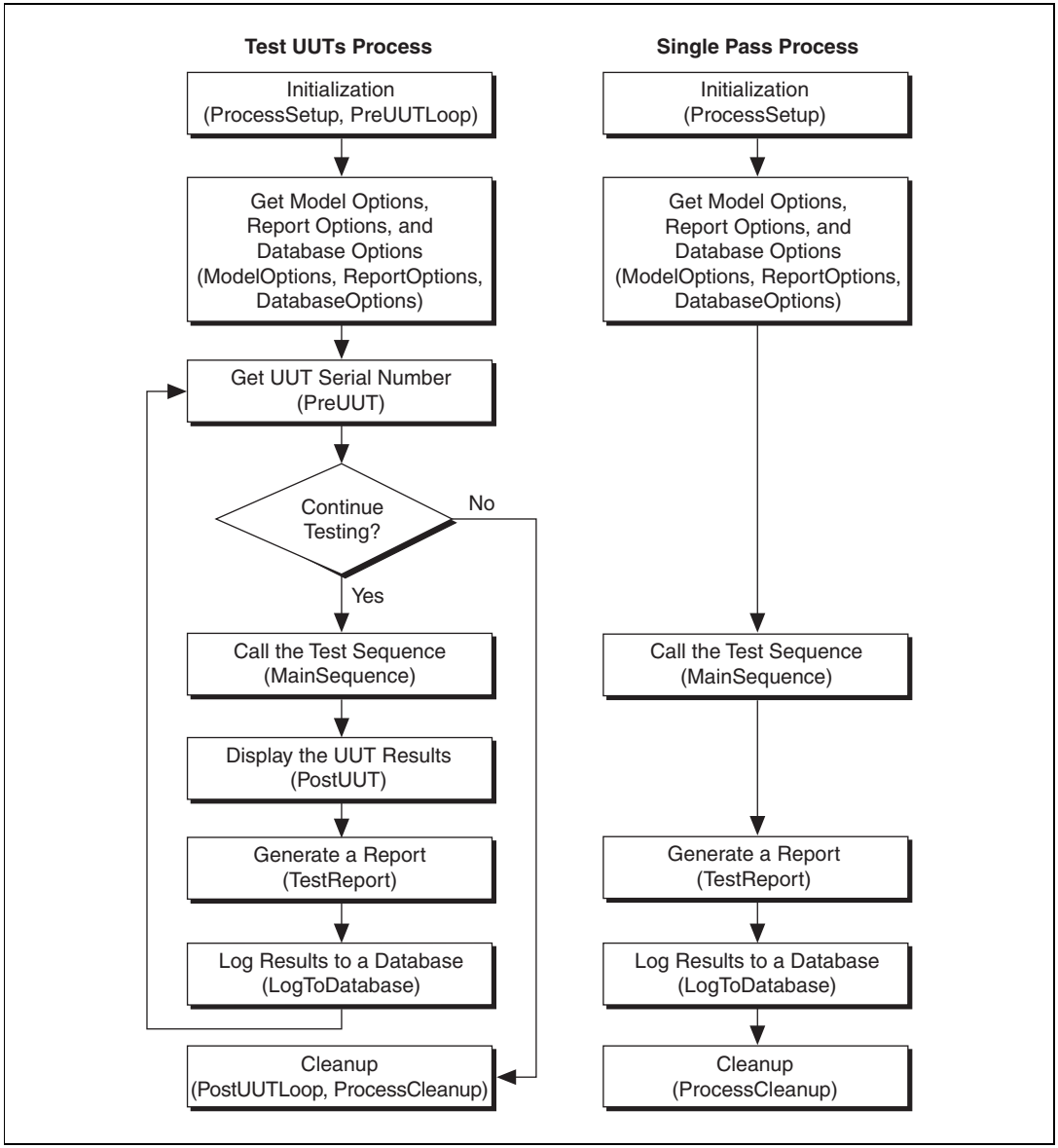


Figure 6-1. TestStand Sequential Model Callbacks

You can modify the process models or replace the models entirely to alter the behavior of the process models for all sequences.

Note To modify the TestStand process models directly, copy the files from the `<TestStand>\Components\Models\TestStandModels` directory to the `<TestStand Public>\Components\Models\TestStandModels` directory and make changes to the copies. `<TestStand>` is the location where you installed TestStand. When you copy installed files to modify, rename the files after you modify them if you want to create a separate custom component. You do not have to rename the files after you modify them if you only want to modify the behavior of an existing component. If you do not rename the files and you use the files in a future version of TestStand, changes National Instruments makes to the component might not be compatible with the modified version of the component. Storing new and customized files in the `<TestStand Public>` directory ensures that new installations of the same version of TestStand do not overwrite the customizations and ensures that uninstalling TestStand does not remove the files you customize.

Viewing Process Model Callbacks

Complete the following steps to examine the Model callbacks in the Sequential process model, which is the default process model TestStand uses to execute sequences.

1. Open `<TestStand>\Components\Models\TestStandModels\SequentialModel.seq`.
2. On the Sequences pane, select **Test UUTs**, which is the Test UUTs Execution entry point TestStand executes when you select Test UUTs from the Execute menu. The Main step group of the `Test UUTs` sequence calls several callback sequences, including the PreUUTLoop callback, the PreUUT callback, the MainSequence callback, the PostUUT callback, the TestReport callback, the LogToDatabase callback, and the PostUUTLoop callback.
3. Double-click the `PreUUT Callback` step, not the `PreUUTLoop Callback` step, to view the PreUUT callback sequence, which includes an Identify UUT step and a Set Serial Number step.
4. Right-click the `IdentifyUUT` step and select **Run Selected Steps** from the context menu to launch the UUT Information dialog box, which is similar to the dialog box you see when you execute a sequence using the Test UUTs Execution entry point. You can replace the PreUUT callback with a custom callback to change the way TestStand obtains a UUT serial number, such as reading it from a bar code instead.

5. Click **OK** in the UUT Information dialog box.
6. Close the Execution window.
7. Click the back button on the Navigation toolbar to go back to the Test UUTs sequence.
8. Double-click the `PreUUTLoop Callback` step to view the PreUUTLoop callback sequence, which is empty because it is a placeholder. If you want to add steps that execute before the UUT loop, create the steps in this callback.
9. Close `SequentialModel.seq` and decline any prompts to save the changes.

Overriding a Process Model Callback

Complete the following steps to override the default PreUUTLoop callback sequence in the Sequential model by creating a PreUUTLoop callback sequence in a client sequence file. Use this technique when you want to perform a task only once before operating on multiple UUTs, such as initializing hardware.

1. Open `<TestStand Public>\Tutorial\Computer4.seq`, which you created in Chapter 5, *Using Variables and Properties*. You can also find this file in the `<TestStand Public>\Tutorial\Solution` directory.
2. Select **Edit»Sequence File Callbacks** to launch the Sequence File Callbacks dialog box and select the `PreUUTLoop` callback.
3. Click **Add**. The value in the Present column changes from `no` to `yes`. When you click Add, the sequence editor creates a new empty callback sequence in the sequence file. Now, when you start an execution using an Execution entry point, TestStand calls the callback in the sequence file instead of the callback sequence in the Sequential model. Click **OK** to close the Sequence File Callbacks dialog box.
4. Insert a Message Popup step in the Main step group of the PreUUTLoop callback sequence and rename the new step `Pre UUT Loop Message`.
5. On the Text and Buttons tab of the Step Settings pane, enter the literal string `"Pre UUT Loop Callback Message"` in the Title Expression control. Enter the literal string `"Now in the Pre UUT Loop Callback"` in the Message Expression control.
6. Select **File»Save As** and save the sequence as `Computer5.seq` in the `<TestStand Public>\Tutorial` directory.

7. Select **Execute»Test UUTs** to execute the sequence. TestStand launches the Pre UUT Loop Callback Message dialog box. Click **OK** to close the Pre UUT Loop Callback Message dialog box. TestStand launches the UUT Information dialog box from the PreUUT Callback sequence in the Sequential model. Enter a serial number and click **OK**.
8. Run through several iterations of the sequence.
9. Click **Stop** in the UUT Information dialog box. TestStand launched the Pre UUT Loop Callback Message dialog box only once at the very beginning of the execution because the PreUUTLoop callback executes before the loop, and the PreUUT callback executes within the loop. Close all the windows in the sequence editor.

You can make similar modifications to the Parallel and Batch process models.

Refer to Chapter 10, *Customizing Process Models and Callbacks,* and Appendix A, *Process Model Architecture*, of the *NI TestStand Reference Manual* for more information about the process models, callbacks, and modifying callbacks.

7

Adding Users and Setting Privileges

The TestStand Sequence Editor includes a user manager for adding and removing users, for adding and removing groups, and for managing the privileges of each user.

Using the User Manager

The TestStand User Manager helps you implement policies and procedures for using test stations. The user manager is not a security system, and it does not inhibit or control the operating system or third-party applications. Use the system-level security features the operating system provides to secure test station computers against unauthorized use.

Adding a New User

Note This tutorial assumes you are currently logged in as the `administrator` user. If you are not logged in as administrator, select **File»Login** and select **administrator** from the User Name ring control. Leave the Password field empty and click **OK**.

Complete the following steps to add a new user.

1. Select **View»User Manager** or click the **User Manager** button, shown at left, on the toolbar to show the User Manager window, which shows all the users and groups configured on the test station.
2. Right-click **Users**, select **Insert User** from the context menu, and type your name to rename the new user.
3. Right-click the new user you just added, select **Properties** from the context menu to launch the User Properties dialog box, and complete the following steps.
   a. Confirm that the **User Name** control includes your name.
   b. Type your name in the **Full Name** control.
   c. Type a password in the **Password** and **Confirm Password** controls.

   d. Select **Operator** in the Group Privileges control. The default Operator, Technician, Developer, and Administrator groups define a set of privilege settings for the new user to inherit. By default, the Operator group grants a user the privilege to execute, terminate, and abort sequences but does not grant the privilege to create or debug sequences.
   e. Click **OK** to close the User Properties dialog box.
4. Select **File»Login** to launch the Login dialog box. The User Name ring control now includes the new user you just added.
5. Select the user you just created, enter the appropriate password, and click **OK**.
6. Open `<TestStand Public>\Tutorial\Computer.seq`.
7. Select the **Execute** menu and notice that the Run MainSequence and Single Pass options of the Execute menu are disabled because the user you just created does not have the privileges to execute them.
8. Right-click the Steps pane to insert a new step. The Insert Step menu command is also disabled because the user privileges have changed.
9. Select **File»Login**. Select **administrator** from the User Name ring control. Leave the Password field empty and click **OK**.

Creating a New Group

You can use the user manager to modify the default groups and to create new groups that define a combination of appropriate privileges.

Complete the following steps to create a new group.

1. Select **View»User Manager** or click the **User Manager** button on the toolbar to show the User Manager window if the window is not open.
2. Expand the **Groups** item to show the four default groups.
3. Right-click the **Operator** group and select **Copy** from the context menu.
4. Right-click the **Groups** item and select **Paste** from the context menu.
5. Rename the new group `Senior Operator`. The new group is identical to the Operator group except for the name.

Modifying Privileges

TestStand stores privileges as Boolean properties and organizes the privileges in the following categories:

- **Operate**—Contains privileges for executing sequences and terminating and aborting executions.
- **Debug**—Contains privileges for controlling execution flow, executing manual and interactive executions, and editing station globals and run-time variables.
- **Develop**—Contains privileges for editing and saving sequence files, editing workspace files, and using source code control.
- **Configure**—Contains privileges for editing process model files and configuring station options, users, adapters, application settings, and report, database logging, and model options.
- **Custom**—Contains custom privileges you define. Customize the `NI_UserCustomPrivileges` data type to add new privileges. Refer to the *Defining Custom Privileges* section of Chapter 7, *User Management*, of the *NI TestStand Reference Manual* for more information about custom privileges.

Each privilege category includes a Boolean `GrantAll` subproperty. A user or group has a privilege if you set the property value of the privilege to `True`. You can set the `GrantAll` property of a privilege category to specify if a user or group has all privileges within a privilege category, regardless of the property value of the individual privileges.

Note TestStand stores the privilege categories as subproperties of the `Privileges` property. The `Privileges` property also includes a Boolean `GrantAll` subproperty. The property `Privileges.GrantAll` applies to all privilege categories. If you set this property to `True`, the user or group has all privileges. You must set this property to `False` to honor privilege settings within each privilege category.

You can grant privileges in several ways. For example, a user or group has the privilege to terminate an execution if his or her user privileges or group privileges meet one of the following set of conditions:

- `Privileges.GrantAll` is `True`. This also grants rights to all other privileges.
- `Privileges.GrantAll` is `False` and `Privileges.Operate.GrantAll` is `True`. This also grants rights to other privileges of the `Operate` privilege category.
- `Privileges.GrantAll` and `Privileges.Operate.GrantAll` are `False` and `Privilege.Operate.Terminate` is `True`. This ensures that a user has the privilege only to terminate an execution.

Complete the following steps to modify the default privileges for the group you created in the previous section of this tutorial.

1. Expand the **Senior Operator** item and expand **Privileges**.
2. Expand the **Debug** item, which is a property that contains Boolean subproperties. Use the **Value** column ring control to change the `SinglePass` property under Debug to `True`.
3. Complete the following steps to add the user you created in the previous section of this tutorial to the Senior Operator group.
   a. Right-click the user you previously created under Users and select **Properties** from the context menu.
   b. Disable **Operator** in the Group Privileges control and enable **Senior Operator** instead.
   c. Click **OK** to close the User Properties dialog box.
4. Select **File»Login** to launch the Login dialog box, select the user you previously created, enter the appropriate password, and click **OK**.
5. Select the `Computer.seq` window.
6. Select the **Execute** menu and notice that the Single Pass option is available now but the Run MainSequence option is still disabled because the user you just created does not have the privilege to execute sequences without a Model entry point.
7. Close all the windows in the sequence editor.
8. Select **File»Login**.
9. Select **administrator** from the User Name ring control. Leave the Password field empty and click **OK**.

You can also use the TestStand API to add and remove users programmatically, as demonstrated in the `CreateDeleteUsers.seq` example located in the `<TestStand Public>\Examples\CreateDeleteUsers` directory.

8

Interactive Executions

Unlike when you run sequences directly, when you run steps in interactive mode, you can execute specific steps in a sequence.

Running Selected Steps as a Separate Execution

Complete the following steps to run selected steps from a Sequence File window as a separate execution.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.
2. Insert a breakpoint at the `Power On` step.
3. Hold down the **<Ctrl>** key and click the `Power On`, `ROM`, and `ROM Diagnostics` steps to select these steps.
4. Select **Execute»Run Selected Steps** to start a new execution, called a root interactive execution. By default, when you run selected steps from a Sequence File window, TestStand also executes the Setup and Cleanup step groups. You can use the Station Options dialog box to modify these settings to control if TestStand runs the Setup and Cleanup step groups as part of the root interactive execution. Refer to the *NI TestStand Help* for more information about the Station Options dialog box.
5. Click **Done** to close the Test Simulator dialog box. The execution stops at the `Power On` step breakpoint. The execution pointer for the interactive execution is a narrow yellow arrow. The sequence editor uses a full yellow arrow as an execution pointer for a normal execution.
6. Select **Debug»Step Over** twice to single-step through the execution until you reach the `ROM Diagnostics` step. Only the steps you selected execute. TestStand dims the other steps.
7. Resume and complete the execution.
8. Close the Execution window after the execution completes.

9. Ensure that the `Power On`, `ROM`, and `ROM Diagnostics` steps are selected and repeat steps 4 through 8 but select **Execute»Run Selected Steps Using»Single Pass** in step 4. TestStand executes the steps you selected using the Single Pass Execution entry point, which generates a UUT report.

Running Selected Steps During an Execution

Complete the following steps to interactively execute selected steps in a sequence while suspended at a breakpoint during an execution.

1. Ensure that the `Power On`, `ROM`, and `ROM Diagnostics` steps are selected and select **Execute»Single Pass** to start a new execution, select the **ROM** test to fail in the Test Simulator dialog box, and click **Done**. The execution stops at the breakpoint on the `Power On` step.
2. Step through the execution until you reach the `RAM Diagnostics` step. Notice that the `ROM` step failed.
3. Place a second breakpoint at the `ROM` step in the Execution window and select the `ROM` and `ROM Diagnostics` steps.
4. Right-click the `ROM Diagnostics` step and select **Loop on Selected Steps** from the context menu. In the Loop on Selected Steps dialog box, type `100` in the Loop Count control and click **OK** to close the Loop on Selected Steps dialog box.

   The sequence editor starts an interactive execution for the steps you selected and enters a paused state at the breakpoint for the ROM step. The execution pointer for the normal execution remains on the ROM Diagnostics step, and an execution pointer for the new interactive execution points to the ROM step. Refer to the *NI TestStand Help* for more information about the Loop on Selected Steps dialog box.
5. Single-step through the interactive execution. The interactive execution toggles only between the ROM step and the ROM Diagnostics step. The status of the ROM step does not pass. It continues to fail.
6. Select **Debug»Terminate Interactive Execution** rather than complete 100 loops of the interactive execution. TestStand returns the execution to a paused state on the ROM Diagnostics step.

7. Complete the following steps to force the execution to continue from a step other than the currently paused step.
   a. Click the `ROM` step so it is the only highlighted step.
   b. Right-click the `ROM` step and select **Set Next Step to Cursor** from the context menu. The execution pointer moves from the ROM Diagnostics step to the ROM step.
   c. Step over to single-step once. The ROM step executes instead of the ROM Diagnostics step.
8. Resume and complete the execution. The report contains entries for each step you executed interactively.
9. Remove the breakpoint from the `Power On` step in the Sequence File window. Close all the windows in the sequence editor and do not save any changes to the sequence file.

Use the Interactive Executions section on the Executions tab of the Station Options dialog box to control whether an interactive execution records results, runs steps in the setup and cleanup step groups, and evaluates preconditions, as well as how TestStand handles step failures, errors, and branching during interactive executions.

9

Calling Sequences Dynamically

You can add a step to a sequence that dynamically runs one of two sequences, and you can pass parameters to the sequence you call.

Dynamically Specifying a Sequence to Run

Complete the following steps to open an existing sequence, add steps to prompt the operator for a CPU type and the number of CPUs to test, and add a step to call one of two different sequences depending on the type of CPU the user specifies.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.
2. Select the **Variables** pane in the Sequence File window, right-click **Locals**, select **Insert Local»String** from the context menu, and rename the local variable `CPUType`.
3. Select the **Steps** pane in the Sequence File window to show the steps in the `MainSequence` sequence.
4. Insert a Message Popup step below the `Power On` step, rename the new step `Select CPU Type`, and complete the following steps to configure the `Select CPU Type` step.
   a. On the Text and Buttons tab of the Step Settings pane, enter the following values into the corresponding controls:
      - **Title Expression**—`"Select CPU"`
      - **Message Expression**—`"Please select the CPU type for the UUT."`
      - **Button 1**—`"INTEL CPU"`
      - **Button 2**—`"AMD CPU"`
      - **Cancel Button**—`None`
   b. Click the **Layout** tab and enable the **Make Modal** option to prevent the sequence editor from hiding the Select CPU dialog box and to prevent you from interacting with the sequence editor until you close the Select CPU dialog box.
   c. Click the **Properties** tab and click **Expressions** to show the Expressions panel.

d. Enter the following expression in the Post-Expression control to assign the string value `"AMD"` or `"INTEL"` to the local variable depending on the button users click:

```
Locals.CPUType = ((Step.Result.ButtonHit == 2) ?
"AMD" : "INTEL")
```

You can also use the Expression Browser dialog box to create this expression. Click the **Expression Browser** button or right-click in the **Post-Expression** control and select **Browse** from the context menu to launch the Expression Browser dialog box.

5. Insert a Message Popup step below the `Select CPU Type` step, rename the new step `Specify Number of CPUs`, and complete the following steps to configure the `Specify Number of CPUs` step.
   a. On the Text and Buttons tab of the Step Settings pane, enter the following values into the corresponding controls:
      - **Title Expression**—`"Number of CPUs"`
      - **Message Expression**—`"Please select the number of CPUs installed for the UUT."`
      - **Button 1**—`"1"`
      - **Button 2**—`"2"`
      - **Button 3**—`"3"`
      - **Button 4**—`"4"`
      - **Cancel Button**—`None`

   b. Click the **Layout** tab and enable the **Make Modal** option.
6. Insert a Sequence Call step below the `Specify Number of CPUs` step, rename the step `CPU Test`, and complete the following steps to configure the step.
   a. On the Module tab of the Step Settings pane, enable the **Specify By Expression** option.
   b. Enter `Locals.CPUType + "Processor.seq"` in the File Path or Reference control and `"MainSequence"` in the Sequence control.

   c. Click the **Load Prototype** button, shown at left, to select the prototype for the Sequence Call step.
   d. Click the **Browse** button in the Load Sequence Prototype dialog box and select `<TestStand Public>\Tutorial\AMDProcessor.seq`.

e. Click **OK** to close the Load Sequence Prototype dialog box. TestStand populates the Parameters section on the Module tab with the parameter list for the sequence.

f. Click in the Value column of the **CPUsInstalled** parameter and enter the following expression:

```
RunState.Sequence.Main["Specify Number of
CPUs"].Result.ButtonHit
```

You can also use the Expression Browser dialog box to locate the property.

7. Select **File»Save As** and save the sequence as `Computer6.seq` in the `<TestStand Public>\Tutorial` directory.

Running a Sequence Dynamically

Complete the following steps to run a sequence dynamically.

1. Place a breakpoint on the `CPU Test` step, select **Execute»Single Pass**, and click **Done** in the Test Simulator dialog box.
2. Click the **INTEL CPU** button in the Select CPU dialog box and click the **2** button in the Number of CPUs dialog box.
3. Step into the execution after it pauses at the breakpoint on the `CPU Test` step to single-step into the subsequence. Select the **Call Stack** pane. The call stack list shows `INTELProcessor.seq` at the top of the sequence call stack.
4. Select the **Variables** pane and expand the **Parameters** item. The value of the **CPUsInstalled** parameter equals the value on the button you clicked in the Number of CPUs dialog box. The `MainSequence` in the `INTELProcessor.seq` sequence file also requires a ModelName parameter. The Sequence Call step you created did not specify the **CPUsInstalled** parameter, so the TestStand Engine initializes the parameter to its default value.
5. Resume and complete the execution. When the execution completes, review the report but do not close the Execution window.
6. Select **Execute»Restart** to restart the execution and click **Done** in the Test Simulator dialog box.
7. Click the **AMD CPU** button in the Select CPU dialog box and click the **3** button in the Number of CPUs dialog box.

8. Step into after the execution pauses at the breakpoint on the `CPU Test` step to single-step into the subsequence. The Call Stack pane lists `AMDProcessor.seq` at the top of the call stack.
9. Resume and complete the execution and review the report. Close all the windows in the sequence editor.

10

Customizing Reports

You can customize report generation within TestStand. Using the callback structure discussed in Chapter 6, *Using Callbacks*, you can create a TestReport callback routine to develop reports in any format. TestStand also provides several options to configure the format of the test report without creating a callback.

Configuring Test Report Options

Complete the following steps to configure the test report options.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.
2. Select **Configure»Report Options** to launch the Report Options dialog box and complete the following steps.
   a. On the Contents tab, enable the **Include Step Results** option and configure the following step result settings:
      - Enable the **Include Test Limits** option.
      - Enable the **Include Measurements** option and select **Insert Graph** from the Include Arrays ring control.
   b. Enable the **Include Execution Times** option.
   c. Click the **Edit Format** button to launch the Numeric Format dialog box, in which you configure how TestStand stores numeric values in the test report. By default, TestStand configures the numeric format to report numbers with 13 digits of precision.
   d. Change **Number of Fractional Digits** to `2` and click **OK** to close the Numeric Format dialog box.
   e. Select **XML Document** from the Report Format ring control.
3. Click the **Report File Pathname** tab, which you use to configure the name and path for the test report file. For example, you can create a new file for each UUT or include the time and date in the name of the report file. Use the default values for the options on this tab and click **OK** to close the dialog box.

4. Select **Execute»Test UUTs** to execute the sequence. Run through several iterations of the sequence, selecting components other than the Video and CPU tests to fail.
5. Click **Stop** in the UUT Information dialog box to stop sequence execution. The test report contains failure chain information for UUTs that fail. The failure chain shows the step whose failure caused the UUT to fail and shows the Sequence Call steps through which the execution reached the failing step. Each step name in the failure chain links to the section of the report that shows the result for the step. Close the Execution window.
6. Select **Configure»Report Options**, select **ASCII Text File** from the Report Format ring control, and select **Exclude Passed/Done/Skipped** from the Result Filtering Expression ring control, which determines the conditions to meet before TestStand logs the results to the test report. In this example, you configure TestStand to record only the results of the steps that do not pass or steps that complete without any status. TestStand also changes the Include Arrays ring control to `Insert Table`.
7. Click **OK** to close the Report Options dialog box.
8. Repeat steps 4 and 5 and examine the text version of the test report. Close the Execution window.

Using External Report Viewers

You can view the test report in external applications more suited for showing and editing text, such as Microsoft Word or Microsoft Excel.

Complete the following steps to change the file extension of a report and use an external report viewer to view the test report.

1. Complete the following steps to change the file extension of the report.
   a. Select **Configure»Report Options**.
   b. Click the **Report File Pathname** tab.
   c. Disable the **Use Standard Extension for Report Format** option.
   d. Type `doc` in the Extension string control to create test reports with a `.doc` file extension.
   e. Click **OK** to close the Report Options dialog box.

2. Complete the following steps to configure TestStand to automatically launch the external viewer associated with the `.doc` file extension.
   a. Select **Configure»External Viewers**.
   b. Enable the **Automatically Launch Default External Viewers** option.
   c. Click **OK** to close the Configure External Viewers dialog box.
3. Select **Execute»Test UUTs** to execute the sequence and run through several iterations of the sequence, selecting components other than the Video and CPU tests to fail.
4. Click **Stop** in the UUT Information dialog box to stop sequence execution. TestStand generates the text report and launches WordPad or Word to show the test report.
5. Examine the test report and close the external report viewing application.
6. Complete the following steps to change the report settings back to the default settings.
   a. Select **Configure»Report Options**.
   b. On the Contents tab, select **All Results** from the Result Filtering Expression ring control.
   c. Select **XML Document** from the Report Format ring control.
   d. Click the **Edit Format** button to launch the Numeric Format dialog box, change **Number of Fractional Digits** to `13`, and click **OK** to close the Numeric Format dialog box.
   e. On the Report File Pathname tab, enable the **Use Standard Extension for Report Format** option.
   f. Click **OK** to close the Report Options dialog box.
7. Complete the following steps to change the external viewer settings back to the default settings.
   a. Select **Configure»External Viewers**.
   b. Disable the **Automatically Launch Default External Viewers** option.
   c. Click **OK** to close the Configure External Viewers dialog box.
8. Close all the windows in the sequence editor.

You can also use the Configure External Viewers dialog box to define a file association independent of the operating system. Refer to the *NI TestStand Help* for more information about the Configure External Viewers dialog box.

Adding Additional Results to Reports

Use the Additional Results panel on the Properties tab of the Step Settings pane to specify additional results TestStand collects for a step, such as module parameters, step properties, and variables. Enable the **Include in Report** option for additional results you want TestStand to include in reports.

Complete the following steps to create a step that calls a DLL code module to return a numeric array and add the numeric array and other values to the report.

1. Select **File»New Sequence File** to open a new sequence file. Save the sequence file as `CustomReport.seq` in the `<TestStand Public>\Tutorial` directory.
2. Complete the following steps to create a local variable.
   a. On the Variables pane, right-click **Locals** and select **Insert Local»Array of»Number** from the context menu.
   b. In the Array Bounds dialog box, type `49` in the Upper Bounds control and click **OK**.
   c. Rename the local variable `NumArray`.

3. On the Insertion Palette, select the **C/C++ DLL** adapter icon, shown at left.
4. Add an Action step to the Main step group and complete the following steps to configure the Action step.
   a. On the Module tab of the Step Settings pane, click the **Browse** button and navigate to `<TestStand Public>\Tutorial\NumericArray.dll` to specify the code module for the step.
   b. Select **GetNumericArray** from the Function ring control.
   c. In the Value Expression column of the **measurements** parameter, type the `Locals.NumArray` expression to copy the value from the **NumericArray** output parameter to the Locals.NumArray variable when TestStand returns from calling the GetNumericArray function.
   d. In the Value column of the `Dim 1 Size` property, type `-1` to specify that TestStand uses the actual size of the `Locals.NumArray` property.

5. Click the Properties tab of the Step Settings pane and complete the following steps to add the numeric array and other values to the report.
   a. Click **Additional Results** on the Properties tab of the Step Settings pane to show the Additional Results panel.
   b. Place a checkmark in the **measurements [Out]** parameter checkbox to log the output value of the parameter.
   c. Click the **Add Custom Result** button, shown at left.

   d. Type `"Time"` in the Name column and type `Time()` in the Value to Log column.
   e. Click the **Add Custom Result** button again.
   f. Type `"Date"` in the Name column and type `Date()` in the Value to Log column.
6. Save the changes you made and select **Execute»Single Pass** to execute the sequence and view the report that includes the array data, the time, and the date.
7. Close all the windows in the sequence editor.

Adding to Reports Using Callbacks

Complete the following steps to add a logo to the header of the HTML report using a Report callback in the process model.

1. Open `<TestStand Public>\Tutorial\Computer.seq`.
2. Select **Configure»Report Options**. On the Contents tab, select **HTML Document** from the Report Format ring control and click **OK**.
3. Select **Edit»Sequence File Callbacks** to launch the Sequence File Callbacks dialog box, select the `ModifyReportHeader` callback, and click **Add** to add the callback to the sequence file.
4. Click **Edit** to close the Sequence File Callbacks dialog box and edit the new `ModifyReportHeader` callback sequence in the Sequence File window, which TestStand automatically opens.
5. Select the **Variables** pane, right-click **Locals**, select **Insert Local»String** from the context menu, and rename the local variable `AddToHeader`. Click in the Value column of the `AddToHeader` variable and type the following text:

```
<IMG ALT='Logo Goes Here' SRC='Logo.jpg'><br></br>
<A HREF='http://www.ni.com'>Visit Our Web
Site</A><br></br>
```

6. Select the **Steps** pane, insert a Statement step in the Main step group, and rename the step `Add Custom Logo`. On the Expression tab of the Step Settings pane, enter the following expression in the Expression control:

   ```
   Parameters.ReportHeader = Locals.AddToHeader +
   Parameters.ReportHeader
   ```

7. Select **File»Save As** and save the sequence as `Computer7.seq` in the `<TestStand Public>\Tutorial` directory.
8. Select **Execute»Single Pass** and click **Done** in the Test Simulator dialog box. View the report after the execution completes and notice the new logo image at the top of the UUT report.
9. Complete the following steps to change the report settings back to the default settings.
   a. Select **Configure»Report Options**.
   b. On the Contents tab, select **XML Document** from the Report Format ring control.
   c. Click **OK** to close the Report Options dialog box.
   d. Close all the windows in the sequence editor.

Refer to Chapter 6, *Database Logging and Report Generation*, of the *NI TestStand Reference Manual* for more information about customizing reports.

A

Technical Support and Professional Services

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

- **Support**—Technical support resources at ni.com/support include the following:
  - **Self-Help Technical Resources**—For answers and solutions, visit ni.com/support for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on. Registered users also receive access to the NI Discussion Forums at ni.com/forums. NI Applications Engineers make sure every question submitted online receives an answer.
  - **Standard Service Program Membership**—This program entitles members to direct access to NI Applications Engineers via phone and email for one-to-one technical support as well as exclusive access to on demand training modules via the Services Resource Center. NI offers complementary membership for a full year after purchase, after which you may renew to continue your benefits.

    For information about other technical support options in your area, visit ni.com/services, or contact your local office at ni.com/contact.
- **Training and Certification**—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. 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, National Instruments Alliance Partner 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.

Index

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W