## **OVERVIEW CARD**

# **TestStand**<sup>®</sup> **System and Architecture**

National Instruments TestStand is flexible test management software that offers the following major features:

- Out-of-the-box configuration and components that give you a ready-to-run, full-featured test management environment.
- Numerous methods for modifying, configuring and adding new components. These methods provide extensibility and enable you to create a test executive that meets your particular requirements without altering the core engine. You can also upgrade to newer versions of TestStand without losing your customizations.
- Sophisticated sequencing, execution, and debugging capabilities, and a powerful sequence editor that is separate from the operator interfaces.
- User interface controls for creating custom operator interfaces.
- Example operator interfaces with source code for National Instruments LabVIEW. LabWindows™/CVI™, Microsoft Visual Basic .NET C#, and C++ (MFC).

- Open language interface that provides support for many application development environments (ADEs). You can create code modules in a wide variety of ADEs and call preexisting modules or executables. You can also create your own operator interface in any programming language that can host ActiveX controls or control ActiveX automation servers.
- Comprehensive application programming interface (API) for building multithreaded test systems and other sophisticated test applications.
- Integration with third-party source code control (SCC) packages.
- Deployment tools to aid in transferring a test system from development to production.

3234574-01 Jul03

CVI™, LabVIEW™, National Instruments™, NI™, ni.com™, and TestStand™ are trademarks or trade names of National Instruments Corporation. Product and company names mentioned herein are trademarks or trade names of their respective companies. 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.

© 2003 National Instruments Corporation. All rights reserved. Printed in Ireland.



## **Architecture Overview**

## Sequence Editor

The main application program in TestStand. Use the sequence editor to create, modify, execute, and debug test sequences.

### Operator Interfaces

stomizable applications that execute and debug test sequences on a test station. Operator interfaces are available in several different programming languages and include full source code, allowing you to modify them to meet your specific needs.

### Process Models

Define the operations that occur for all test sequences such as identifying the Unit Under Test (UUT), notifying the operator of pass/fail status, generating a test report, and logging results. TestStand includes three fully customizable process models: Sequential, Parallel, and Batch.

## User Interface (UI) Controls

A powerful set of ActiveX controls and support APIs for creating custom TestStand Operator Interfaces.

## TestStand Engine

• A set of DLLs that provide an extensive ActiveX Automation API for controlling and interacting with TestStand. The TestStand Sequence Editor. User Interface (UI) Controls, and Operator Interfaces use this API. You can also write your own applications using the TestStand API.

## Sequence File Executions

Created by the TestStand Engine when you execute a test sequence using the sequence editor or an operator interface.

## Adapters

Allow TestStand to call code modules in a variety of different formats and languages. Adapters also allow TestStand to integrate with various ADEs to streamline test code generation and debugging.

## **Code Modules**

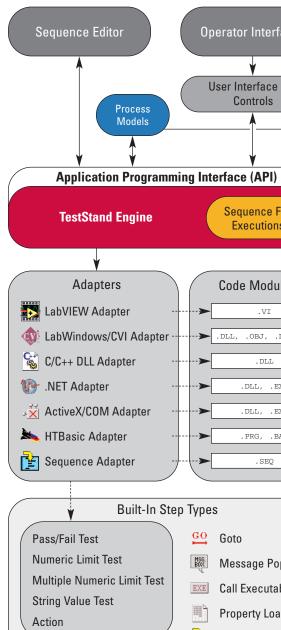
Program modules, such as LabVIEW VIs (.vi) or Windows Dynamic Link Libraries (.dll), which contain one or more functions that perform a specific test or action. TestStand adapters call code modules.

## **Built-In Step Types**

Define the standard behaviors for common testing operations. Some step types use adapters to call code modules that return data to TestStand for further analysis. Other step types perform standard operations such as calling an executable or displaying a dialog hox.

## **User-Defined Step Types**

Define a set of custom step properties and default behaviors for each step of that type. TestStand allows you to define your own step types. You can also define your own custom data types.



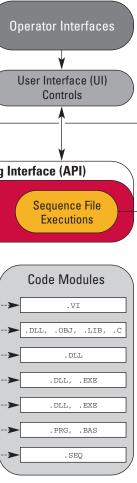
Synchronization Steps

🔥 Database Steps

IVI Steps

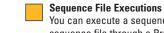
User-Defined Step Types

## **Sequence File Execution Flow**





- EXE Call Executable
- Property Loader
- Sequence Call
- f(x) Statement
- Label Label



### You can execute a sequence directly, or you can execute a sequence file through a Process Model entry point, such as Test UUTs.

## Process Model Sequence File Execution

When you start an execution through a Process Model entry point, the process model defines how UUTs are tested. The Sequential model tests one UUT at a time. The Parallel model tests multiple independent test sockets at the same time. The Batch model tests a batch of UUTs using dependent test sockets.

## Process Model Result Processing

The TestStand Engine collects the results of each step that executes into a result list. Process models use the result list to generate reports and log data to databases.

## Unit Under Test (UUT)

Device or component that you are testing.

## Test Socket Execution

For each test socket, or fixture, in your system, the Parallel and Batch models launch a separate test socket execution that controls the testing of UUTs in that test socket.

## Report Generator

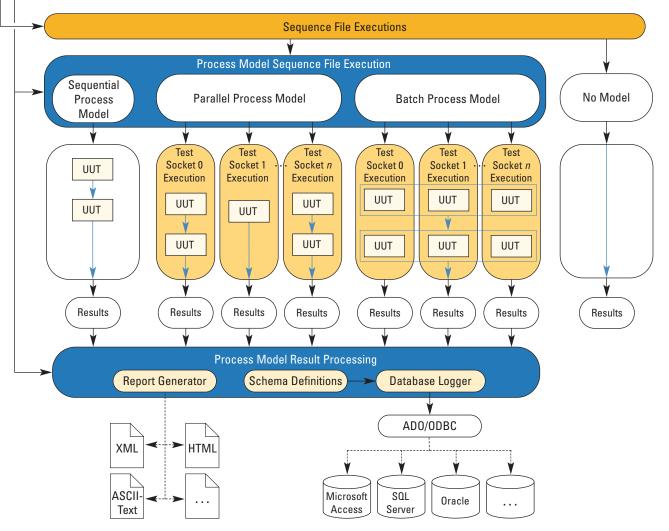
The report generator traverses test results to create reports in XML, HTML, and ASCII-text formats. Reports are fully customizable.

## Database Logger

The database logger traverses test results and exports data into database tables using schema definitions.

## Schema Definitions

Schema definitions define SQL statements, table definitions, and TestStand expressions that define how to log results to a database. Schemas are fully customizable.



## **Execution Object Structure**

## Execution Object

Contains the information TestStand needs to run a sequence, its steps, and any of the subsequences it calls. You can suspend, interactively debug, resume, and terminate or abort executions.

Thread Object Represents an independent path of control flow.

## Report Object

Contains the report text. The process model updates the Report object and the sequence editor or operator interface displays it.

## Call Stack

Chain of active sequences waiting for nested subsequences to complete. The last item in the call stack is the most-nested sequence invocation.

## Root SequenceContext Object

Represents the execution of the least-nested sequence invocation that contains a list of steps and calls to other sequences.

## SequenceContext Object

Represents the execution of a sequence called by another seauence.

## Current Step

Executing step of the currently executing sequence in the call stack.

## **Sequence File Structure**

Sequence File Contains any number of sequences, a set of types used in the sequence file, and any global variables shared by its sequences.

## Sequences

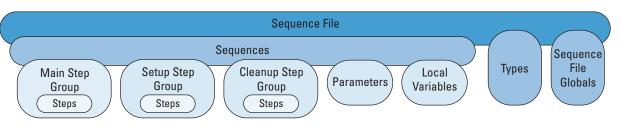
Contain groups of steps, local variables, and parameters used for passing data between subsequences.

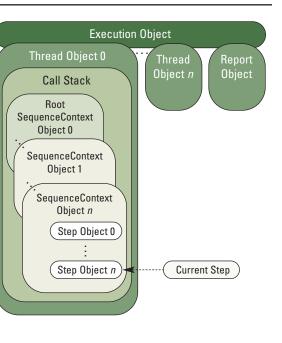
**Types** Sequence files contain definitions of all data types and step types that are used in its sequences. Variables and properties in a sequence are instances of data types. Steps in a sequence are instances of step types.

## Sequence File Globals

Variables that are accessible from any sequence or step within the sequence file in which they are defined.







### Main, Setup, Cleanup Step Groups

TestStand executes the steps in the Setup step group first, the Main step group second, and the Cleanup step group last. By default, a sequence branches to the Cleanup step group when a step generates an error in the Setup or Main step group.

### Parameters

Each sequence has its own list of parameters. Use these parameters to pass data to a subsequence.

### Local Variables

Store data relevant to the execution of the sequence. You can access local variables from within steps and code modules defined in a sequence.

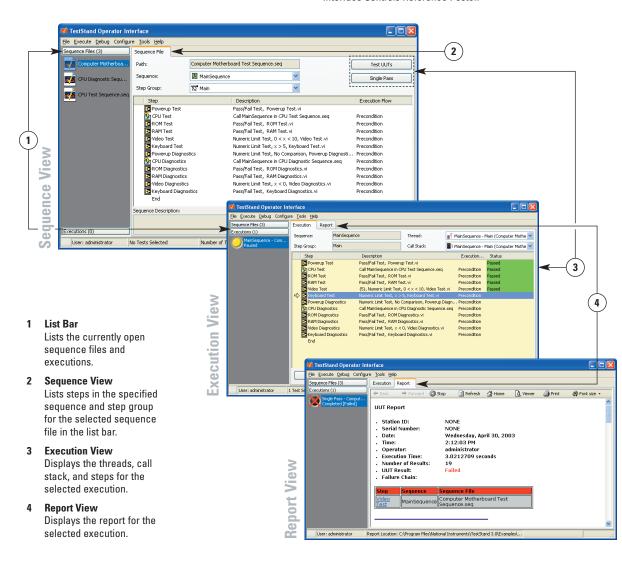
Actions that perform built-in operations or call code modules. A step is an instance of a step type, which defines a set of step properties and default behaviors for each step.

## **OVERVIEW CARD TestStand**<sup>™</sup> **System and Architecture**

## **Operator Interface Overview**

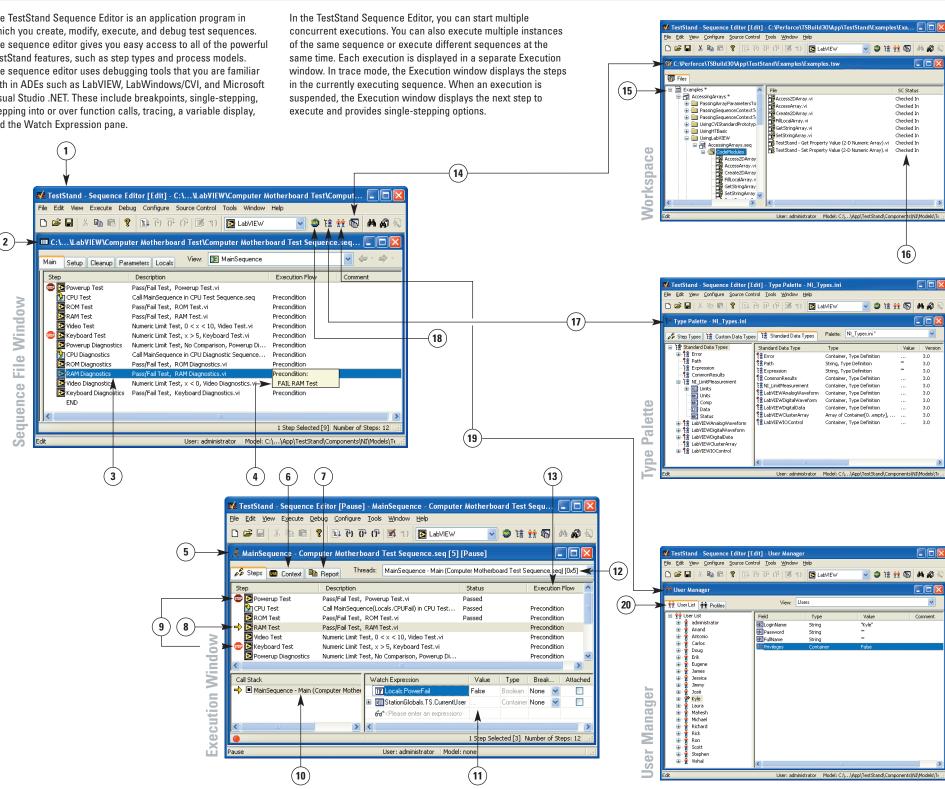
TestStand includes multiple operator interface examples for out-of-the-box use. Each operator interface is a separate full-featured application program. These interfaces-developed in LabVIEW, LabWindows/CVI, Microsoft Visual Basic .NET, C#, and C++ (MFC)-are available in both source and executable formats. You can also create your own operator interface using any programming language that can host ActiveX controls or control ActiveX automation servers.

The TestStand Operator Interfaces are fully customizable. Like the TestStand Sequence Editor, the operator interfaces allow you to start multiple concurrent executions, set breakpoints, and single-step through executions. While the operator interfaces do not allow you to modify sequences, display sequence variables, sequence parameters, step properties, and so on, you can add these capabilities to the operator interfaces using the TestStand UI Controls. For more information, refer to the TestStand User Interface Controls Reference Poster.



## **Sequence Editor Overview**

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



## **Documentation Map & Overviews**

### Sequence Editor

TestStand development environment for creating, modifying, executing, and debugging sequences.

- Sequence File Window Displays sequences and other items found in a sequence file.
- Step

Performs built-in operations or calls code modules.

Tool Tip

Displays details about the item under the mouse cursor.

- 5 Execution Window Displays the threads, call stack, and steps that an execution runs.
- 6 Context View Displays the run-time variables and properties that steps can access as well as their values.
- 7 Report View

Displays the report for the current execution.

8 Execution Pointer

Points to the currently executing step at the selected call stack level.

Breakpoints

Step settings that suspend an execution before executing the step.

10 Call Stack Pane

Displays the nested sequence invocations for the selected thread.

11 Watch Expression Pane

Monitors the values of specified variables and properties during an execution.

12 Thread List

Contains a list of threads in the current execution.

13 Execution Flow

Indicates the control flow settings for steps, such as preconditions, synchronization, and post actions.

14 Workspace

Manages projects for source code control (SCC) integration and deployment.

15 Project

Organizes sequence files and code module files in folders.

16 Source Code Control (SCC)

TestStand integrates with an SCC provider to add files, obtain the latest versions of files, and check files in and out.

17 Type Palette

Contains definitions of custom data types and step types that all sequence files can use.

18 Station Globals

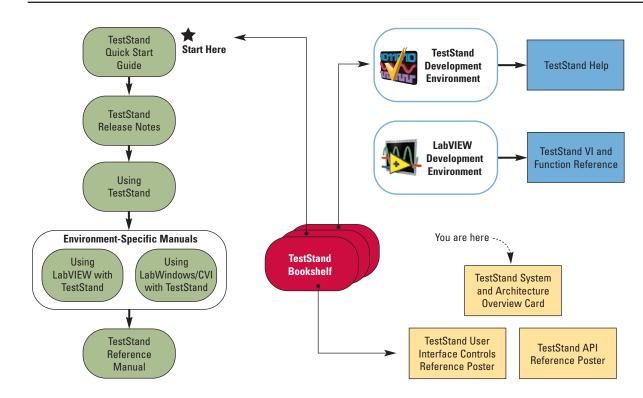
Displays the variables that can be accessed from anywhere within TestStand and maintained from one session to the next.

19 User Manager

Administers users, login names, passwords, and privileges.

20 User List

Displays users for the test station.



## TestStand Bookshelf

Use the TestStand Bookshelf to access all of the TestStand documentation in electronic format. You can also search the *TestStand Bookshelf* by keyword and/or phrase to locate information quickly.

## Online Help

### TestStand Help

Use this help file to learn more about the TestStand environment and the TestStand User Interface Controls and Engine APIs. The TestStand Help also includes basic information about using an ActiveX automation server.

### TestStand VI and Function Reference Help

Use this help file to learn more about TestStand-specific VIs and functions. This help file is only accessible from LabVIEW.

## **Cards and Posters**

TestStand User Interface Controls Reference Poster Use this poster to learn about the controls available for writing custom user interfaces for TestStand.

### TestStand API Reference Poster

Use this poster to get an overview of the TestStand API. This poster lists the properties, objects, and methods of the TestStand API, as well as API inheritance.

## Printed Documentation

## **TestStand Quick Start Guide**

Use this document as a step-by-step installation reference. This document also contains information about the different TestStand licensing options.

## TestStand Release Notes

Use this document to learn about new features and upgrade information as you begin using TestStand.

## Using TestStand

Use this manual to familiarize yourself with the TestStand environment and the basic features you use to build and run test sequences.

## Using LabVIEW with TestStand

Use this manual to learn how to use LabVIEW with TestStand.

## Using LabWindows/CVI with TestStand

Use this manual to learn how to use LabWindows/CVI with TestStand.

## TestStand Reference Manual

Use this reference manual to learn about TestStand concepts and features.