Shared Variable Benchmarking Utility

Users Guide

Version 13.1

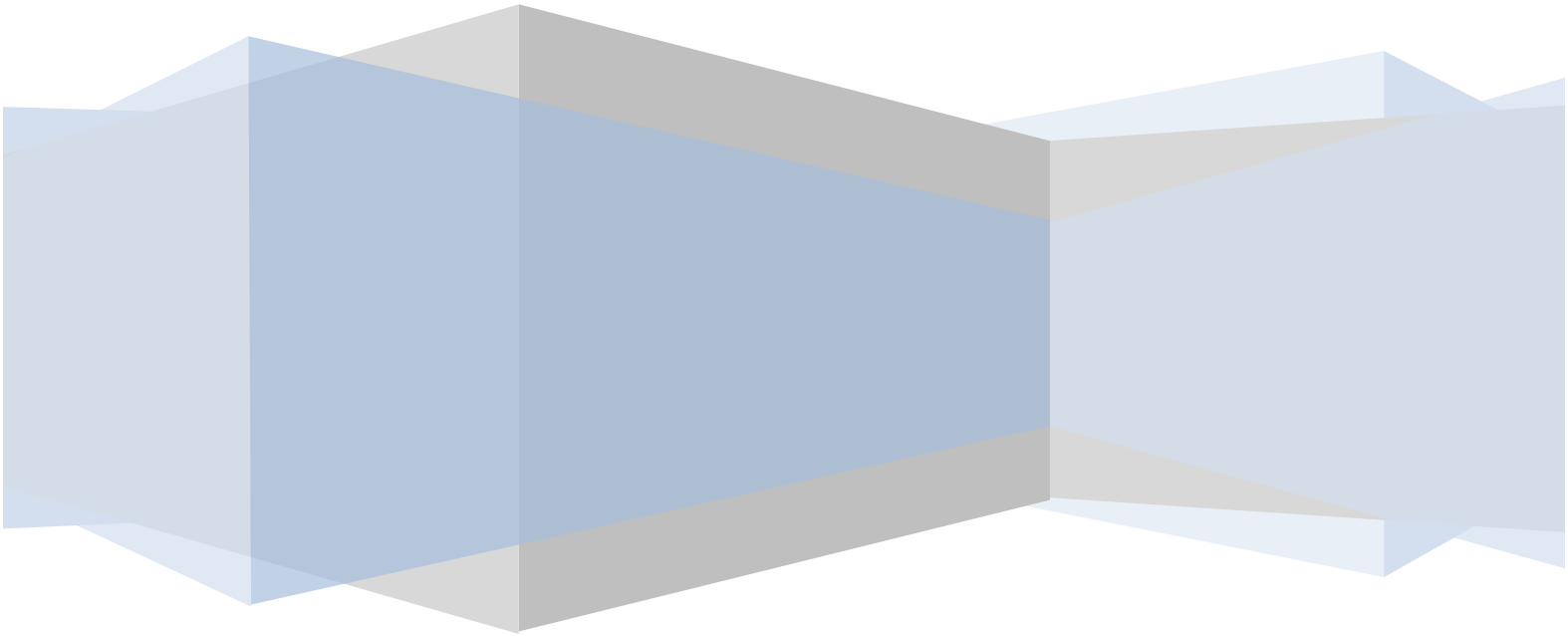


Table of Contents

Background 3
What Happens During a Benchmark? 4
RT Target Configuration 6
Benchmark Utility 7
Data Viewer 8

Background

If you've found this utility, you're more than likely looking for some way to do performance benchmarking on LabVIEW Shared Variables. Benchmarking network communication can be challenging in itself, but benchmarking network communication with additional layers of complexity (i.e. LabVIEW Shared Variables) requires additional consideration.

The Goal

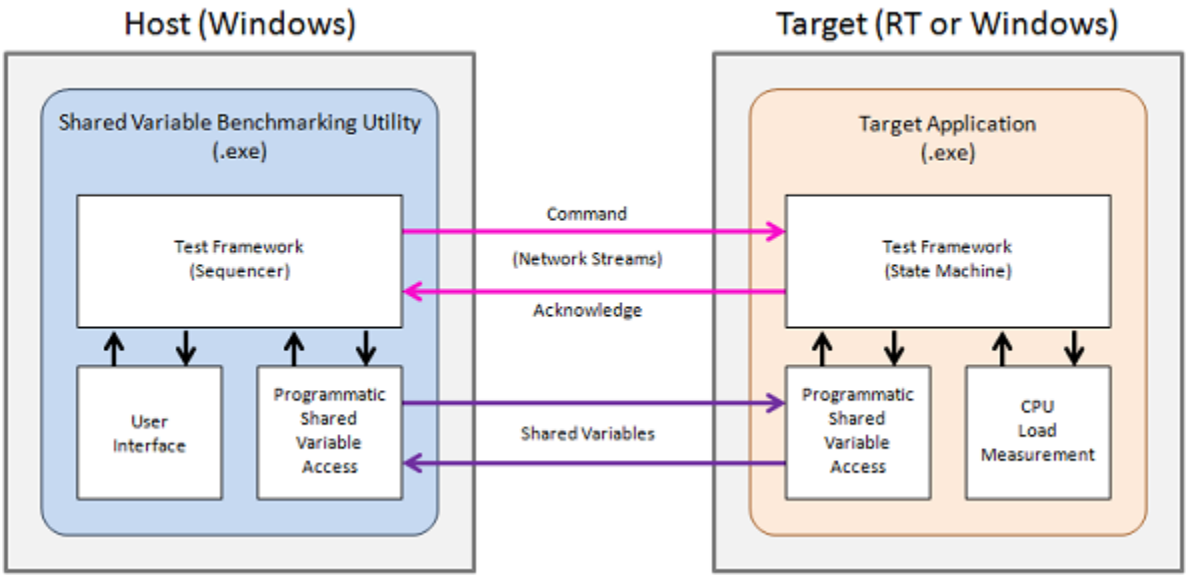
This utility was designed to provide a stable and unified platform on which to capture CPU load profiles for specific targets (real-time & windows) whilst performing network communication via shared variables. This type of information can be extremely helpful when comparing the performance and capabilities of various hardware targets.

Users of the LabVIEW Shared Variable have long struggled with their embedded applications outgrowing the hardware they are built upon. In these situations it's often the addition of more shared variables that takes a working application and degrades performance or functionality. This type of problem can be avoided by having a better understanding of hardware / software capabilities prior to designing an application.

The goal of this utility is to help users of Shared Variables make more informed decisions about the limitations of the hardware / LabVIEW software combination prior to coding an application.

The Architecture

In order for data to be analyzed properly, it's important to have an understanding of how the tool goes about collecting that data. The diagram below shows the basic architecture of how the Shared Variable Benchmarking Utility interfaces with a target.



What Happens During a Benchmark?

Thirteen steps (phases) are executed during a benchmark sequence. This section will give a brief description of what happens during each phase. Before each phase is discussed it is important to differentiate between a sequence and a test. One set of benchmarking parameters defines a sequence. A sequence can contain multiple tests where each tests executes one benchmark in a sweep across a range of shared variable quantities

Phase 1: Initialize Sequence

1. A unique directory for files used during the sequence is created at:

   `C:\Users\Public\Documents\National Instruments\Shared Variable Benchmarking Utility`

2. The sequence parameters are saved to an XML configuration file in the sequence directory.
3. Preliminary calculations (# of tests and # of variables per test) are made.

Phase 2: Connect to Target

1. A "Command" network stream from the host to target is opened.
2. An "Acknowledge" network stream from the target to host is opened.

Phase 3: Initialize Test

1. Active test index field is updated.
2. Test in Progress front panel indicator is updated

Phase 4: Prepare Shared Variable Libraries

1. Calculations are made to determine how many libraries of what size are needed for this test.
2. The necessary libraries of the desired data type are copied from a set of master libraries and are temporarily placed in the sequence directory.

Phase 5: Deploy Shared Variable Libraries

1. All of the libraries duplicated and placed in phase 4 are deployed either to the shared variable engine of the host or of the target (dependent on sequence parameters).

Phase 6: Open Shared Variable Connections

1. A command is sent to the target (via Command network stream) to open connections to all of the shared variables deployed in phase 5.
2. The host opens connections to of all the shared variables.
3. An acknowledgement is received from the target (via Acknowledge network stream) indicating success or failure.

Phase 7: CPU Load Benchmark

1. A command is sent to the target (via Command network stream) to begin the CPU load benchmark.
2. Simultaneously, the host and target begin writing to and reading from all the shared variables opened in phase 6.

3. After 5 seconds of stabilization time, the target takes 30 seconds of CPU load measurements (1 Hz).
4. The target averages the data collected to a single point representing this test.
5. An acknowledgement is received from the target (via acknowledge network stream) indicating success or failure and containing the CPU load data point collected.
6. The data point is added to the CPU load data filed in this test sequence class object.

Phase 8: Close Shared Variable Connections

1. A command is sent to the target (via Command network stream) to close connections to all of the shared variables.
2. The host closes connections to all of the shared variables.
3. An acknowledgement is received from the target (via Acknowledge network stream) indicating success or failure.

Phase 9: Undeploy Shared Variable Libraries

1. All of the libraries duplicated and placed in phase 4 are deployed either to the shared variable engine of the host or of the target (dependent on sequence parameters).

Phase 10: Delete Shared Variable Libraries

1. To keep temporary files from growing too large, all the libraries used during this test step are permanently deleted from disk.

Continue to Next Test or Complete Sequence

At this point, if there are additional tests remaining in the sequence, phases 3 thru 10 are repeated. If there are no additional tests remaining, the utility moves on to the sequence closing phases.

Phase 11: Disconnect from Target

1. Destroy the command network stream.
2. Destroy the acknowledge network stream.
3. The target returns to an idle state and awaits connection for another sequence

Phase 12: Compile & Log Data

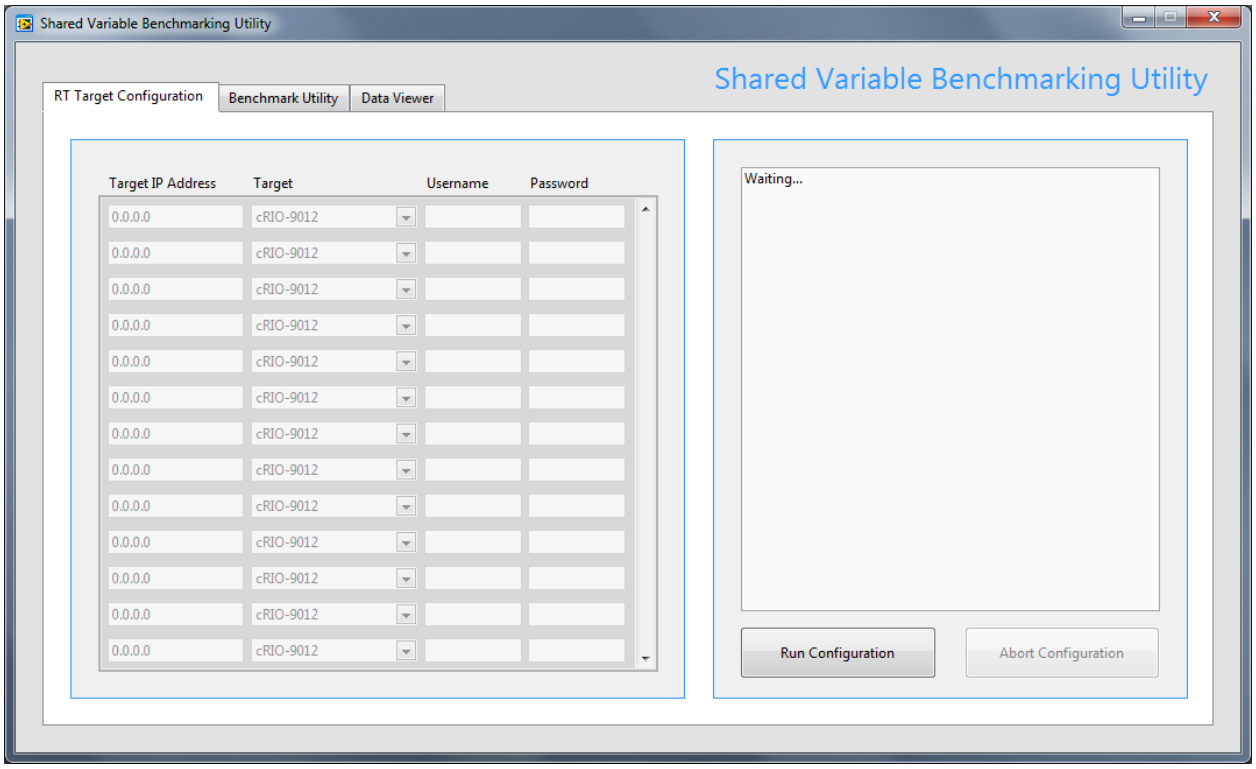
1. The array of CPU Load data is extracted from the benchmark sequence class object
2. The CPU load data is logged to a TDMS file titled "CPU Load Data.tdms" in the sequence directory.

Phase 13: Log Errors

1. If an error occurred at any point during the test sequence, this phase will be reached immediately (as all other phases will have propogated the error without execution).
2. The error code is analyzed to see if the code exists in the tools custom error code library.
3. An error log is built which contains a variety of information helpful in determining the cause of the error. The error log is titled "Error Log.txt" and is placed in the sequence directory.

RT Target Configuration

The interface on this tab is used to prepare networked real-time targets for benchmarking. When the application is first launched, the RT Target Configuration tab will appear as shown in the image below.



The pane on the left houses is used to specify which targets to configure, while the pane on the right displays feedback during the configuration process. After configuration (success / failure) an option is given to save the contents of the feedback pane to file. It is helpful to analyze this log in the event a target is failing configuration.

Multiple targets can be configured in a single run. If an error occurs during configuration the error is printed and subsequent targets in the sequence will be unaffected. Entries may be added or removed by right clicking on the control and selecting the desired option.

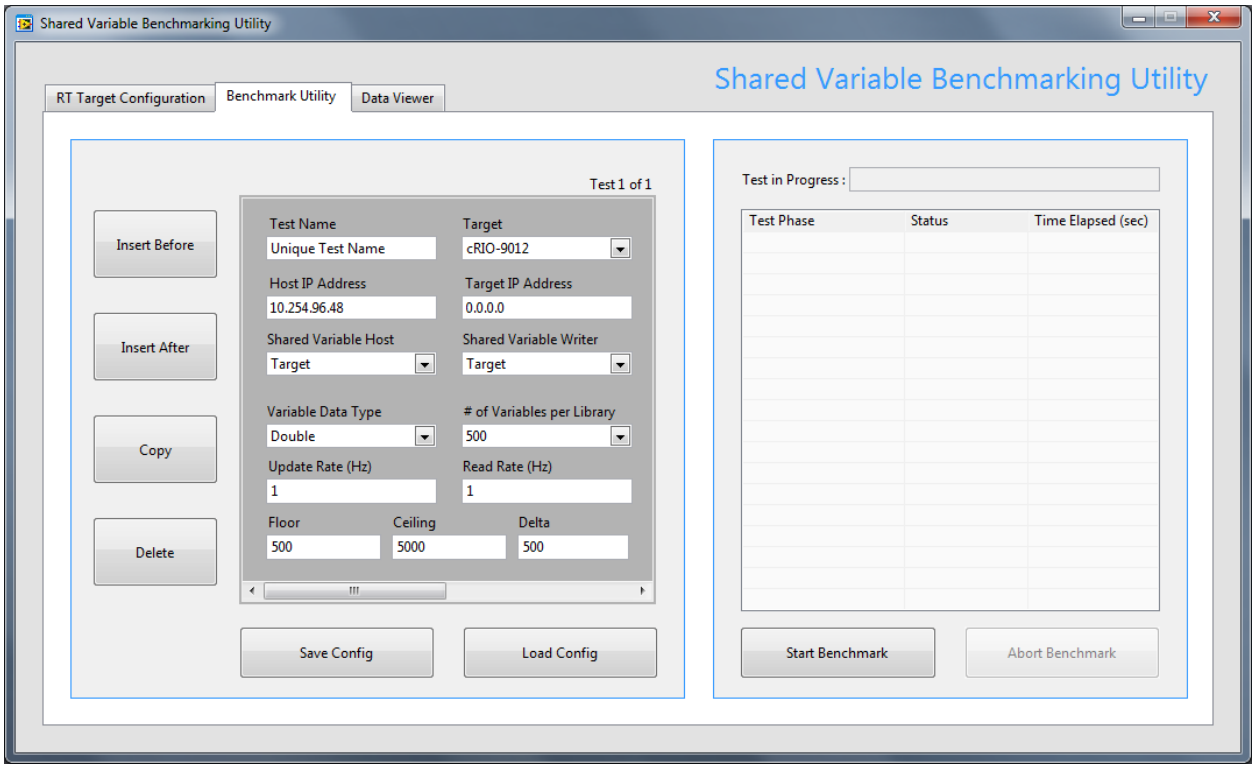
Target IP Address - The IP address of the target to configure specified in dot notation

Target - The type of the target to configure (if the specific type is not listed, select a general option)

Username & Password - If the target is password protected, a username and password must be provided (if no password is required, leave these fields blank)

Benchmark Utility

The interface on this tab is used to specify a sequence of benchmarks to execute. When the application is first launched, the Benchmark Utility tab will appear as shown in the image below.



The pane on the left is used to specify parameters for a sequence of benchmarks, while the pane on the right displays feedback during the benchmarking process.

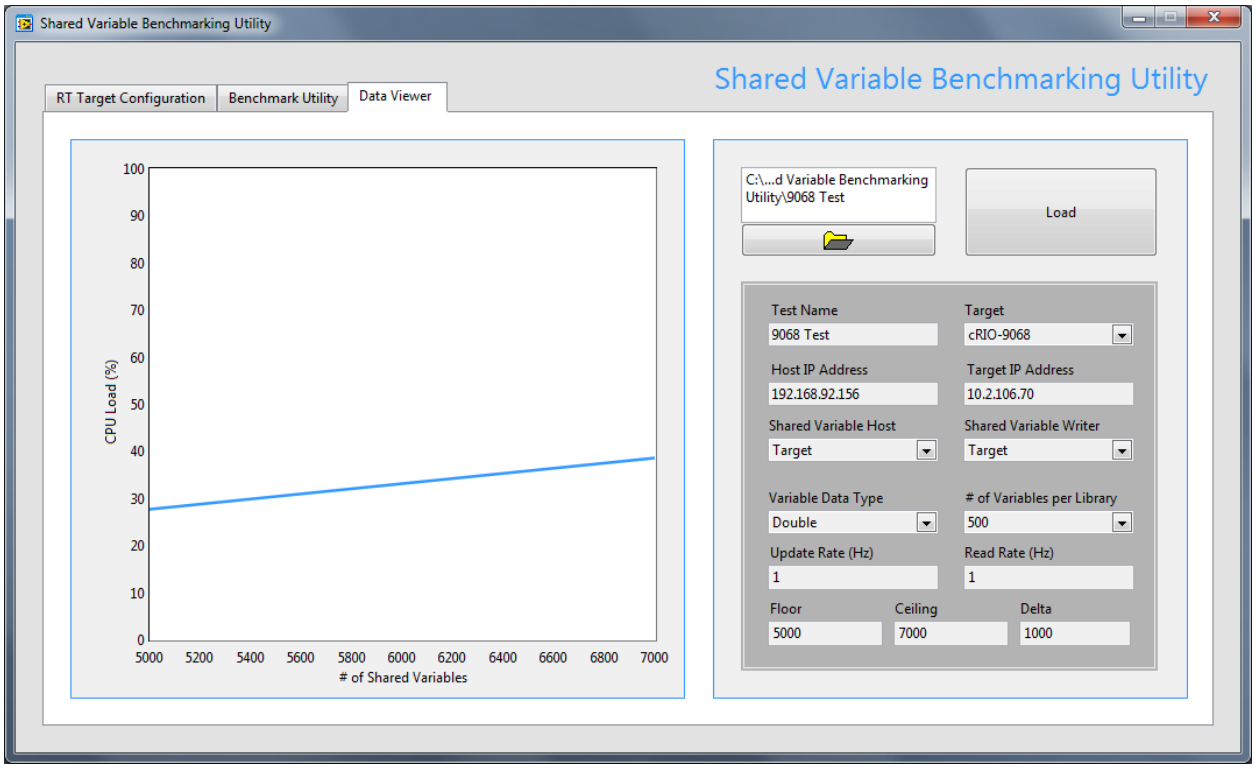
Each array element represents a single benchmark to execute. If desired, multiple benchmarks can be configured to execute in sequence. Use the insert, copy and delete buttons to configure additional benchmarks in the sequence. A sequence configuration file may also be saved to disk or loaded into the utility.

Each benchmark is comprised of multiple tests, which sweep across a range of variable quantities. This sweep is defined by the **Floor, Ceiling, & Delta** parameters. Floor represents the minimum number of variables in the sweep, ceiling represents the maximum number of variables and the delta defines how many tests are executed over that range.

The feedback pane will populate at run-time with all the test phases and will be updated with a status and elapsed time as phases complete. You can determine which benchmark and test within the benchmark is in progress by the string indicator in the feedback pane.

Data Viewer

The interface on this tab is used to quickly view the data collected during a benchmark. For more flexibility it is recommended that you open the TDMS files in Excel and generate graphs / charts there.



The quick data viewer only loads a single benchmark of data at a time. The configuration file for that benchmark is loaded in the indicator on the right and the TDMS data is loaded in the graph on the left.

In order to use the quick data viewer, browse to the directory for the benchmark you wish to view at:

`C:\Users\Public\Documents\National Instruments\Shared Variable Benchmarking Utility`

Load the files into the data viewer using the **Load** button.