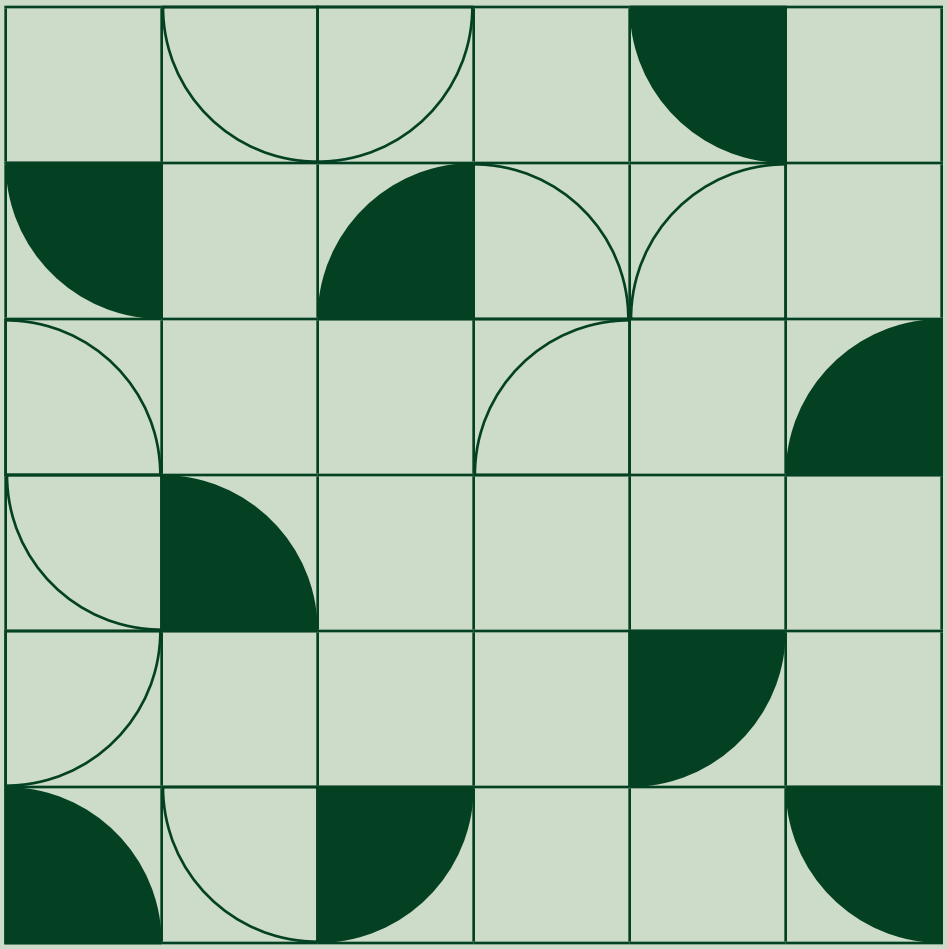
NI Hardware-in-the-Loop (HIL) Collection: Test Early and Often to Maximize Innovation

HIL Testing: A Methodology That Spans Industries



HIL Defined

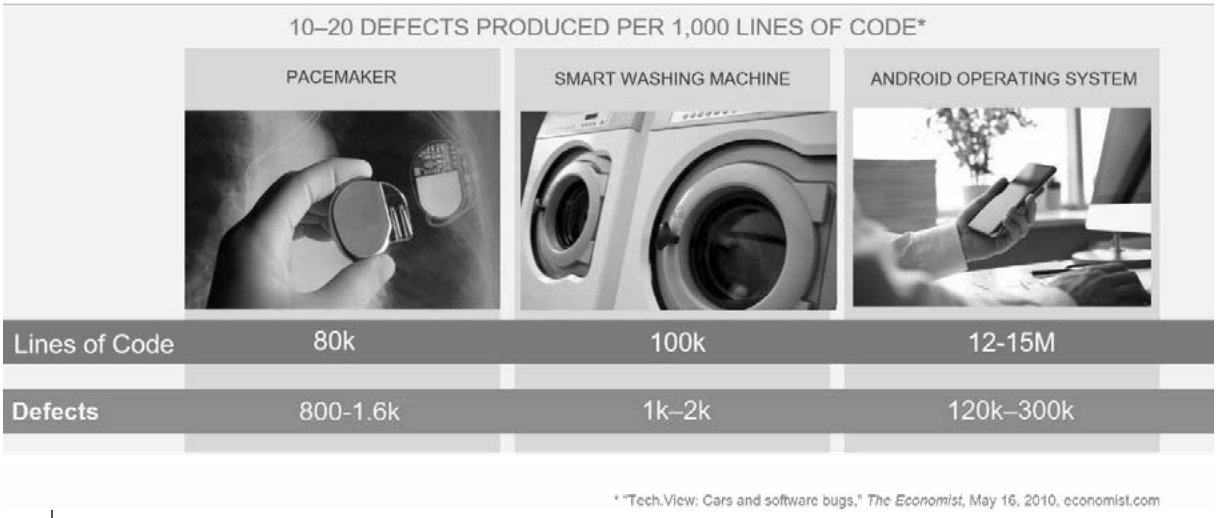
HIL is an embedded software test technique during which real signals from a controller are connected to a test system that simulates reality using a software model. This tricks the controller into thinking it is installed in the assembled product. Test and design iteration take place as though the real-world system is being used. This way, engineers can easily run through thousands of possible scenarios to properly exercise a controller without the cost and time associated with physical tests.

The Case for HIL

Companies use HIL to test embedded software, helping avoid production failures such as a loss of $1M a day from a broken downhole tool on an oil well, the recall of thousands of smart washing machines, or a defective pacemaker that has already been implanted in patients. These are disastrous situations for both users and the engineering teams that create these products. Financial penalties, brand reputation, ethical concerns, and more are at stake for the companies associated with these potential product failures. Hardware-in-the-loop (HIL) testing is a preventive test methodology that allows software engineers and test engineers to assess corner cases that are not practical in the field. Companies can also save money and time by testing earlier in the design cycle and iterating quickly instead of waiting for production test.

Why HIL outside of Aerospace and Automotive

HIL testing was first conducted in the aerospace industry during the Apollo missions by cutting-edge thinkers seeking to send humans into the unknown of space. The only way to test this scenario was with simulation. In the 50 years since, the benefit of testing embedded software early and often before deployment to costly production systems has appealed to a number of industries including aerospace, automotive, oil and gas, medical devices, white goods, and more. As devices become “smarter” with more onboard computing, the opportunity—and payoff—for iterative testing grows. Because of this, HIL is seeing increased adoption in all industries releasing products that rely on embedded software.



* "Tech.View: Cars and software bugs," *The Economist*, May 16, 2010, economist.com

FIG 1 The rise of onboard software heightens the need for more sophisticated testing earlier in the design cycle. Leading design and test organizations are using HIL to answer this call.

Benefit to Users: More Than Failure Detection, A Means to Innovation

The word "test" often defines the final steps needed to move a design to production and ultimately to market. Depending on the industry, test might be a valued part of an organization or it might be a painful afterthought that engineers must begrudgingly conduct to mark a project as complete. At face value, testing provides a final check to ensure everything is working as expected and generates reasonable confidence that a product will be successful in the field.

HIL elevates testing to more than a checkbox on a project plan. It becomes an integral part of the innovation that makes a design and company successful.

Forwa rd-thinking companies are use HIL outside of the traditional road-to-market testing. Though the long-term goal of HIL is to prevent a costly mistake in an expensive program, it's also a design tool that software engineers can use to iteratively test and tweak their software designs. This improves product quality before formal testing even begins. Additionally, software engineers can conceive of and test new ideas quickly, which helps them maximize innovation through timely feedback.

Elements of an HIL System: Software and Hardware Combination

The core elements of an HIL system are the device under test (DUT), data acquisition, and the model that receives processes and sends signals that mimic real-world scenarios. Additional elements may include test case automation, data management, custom communication protocols, fault insertion, and loads.

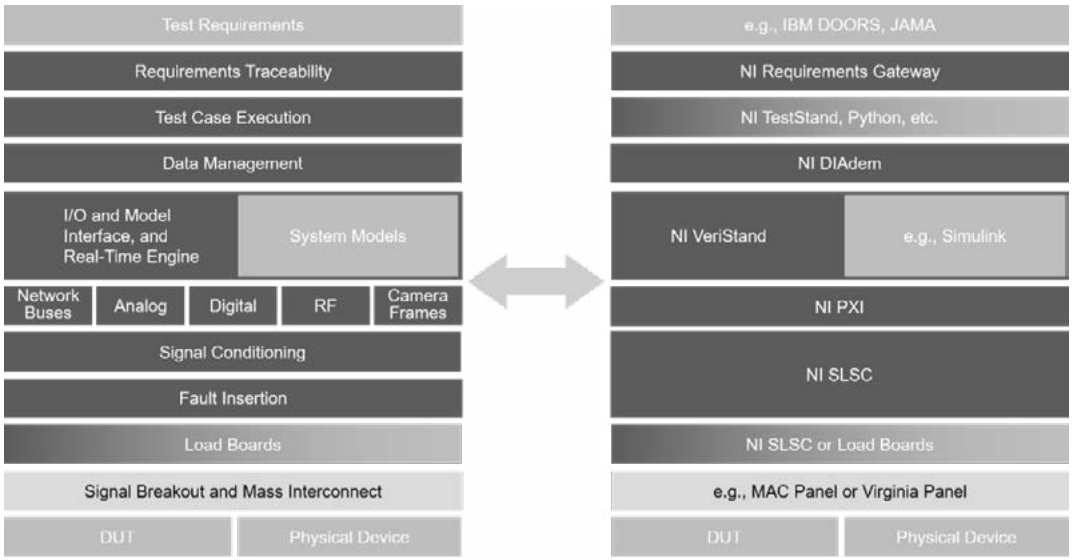


FIG 2 | Typical HIL system requirements mapped to the NI platform

Building an HIL Test Rig: NI's Open Approach to HIL

NI's software-defined platform maximizes the potential of already powerful hardware by allowing user customization through FPGA programming for custom signals and faster processing speeds, model integration, and seamless driver integration. With VeriStand, NI's configuration-based real-time test software, test engineers can incorporate models from more than 20 different environments including MathWorks Simulink®. SLSC hardware acts as a modular interface between the DUT and the measurement hardware (PXI or CompactRIO) and provides signal conditioning, fault insertion, and loading. Supported by NI's community of domain-expert partners, SLSC boasts a growing portfolio of modules and allows users to create their own as needed. This approach greatly reduces cabling issues, eases troubleshooting, and maximizes reuse from system to system and test to test.

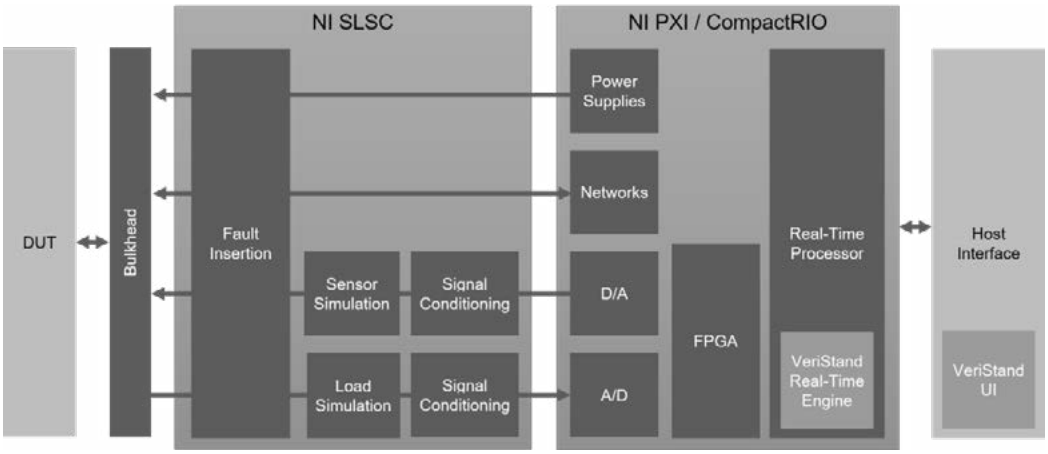


FIG 3 | Typical HIL system built on NI's open platform

Unlike black box, closed solutions from other vendors, NI's HIL offering is open to customization as needed. It is built on the latest industry-proven, commercial off-the-shelf (COTS) components that have published life cycles, so users can plan accordingly for test rigs that need to last decades or longer. With the openness of NI's platform, engineers can incorporate test hardware and software that they already use, which reduces the burden of migrating to a new solution.

Launching a new test system can be overwhelming, especially when test system changes can impact in-flight projects. NI's worldwide community of partners provides the level of customization users need, from completely turnkey solutions to specific points of integration assistance. Additionally, NI's support engineers worldwide can troubleshoot and offer guidance in the language and time zone customers need to make them successful.

Conclusion

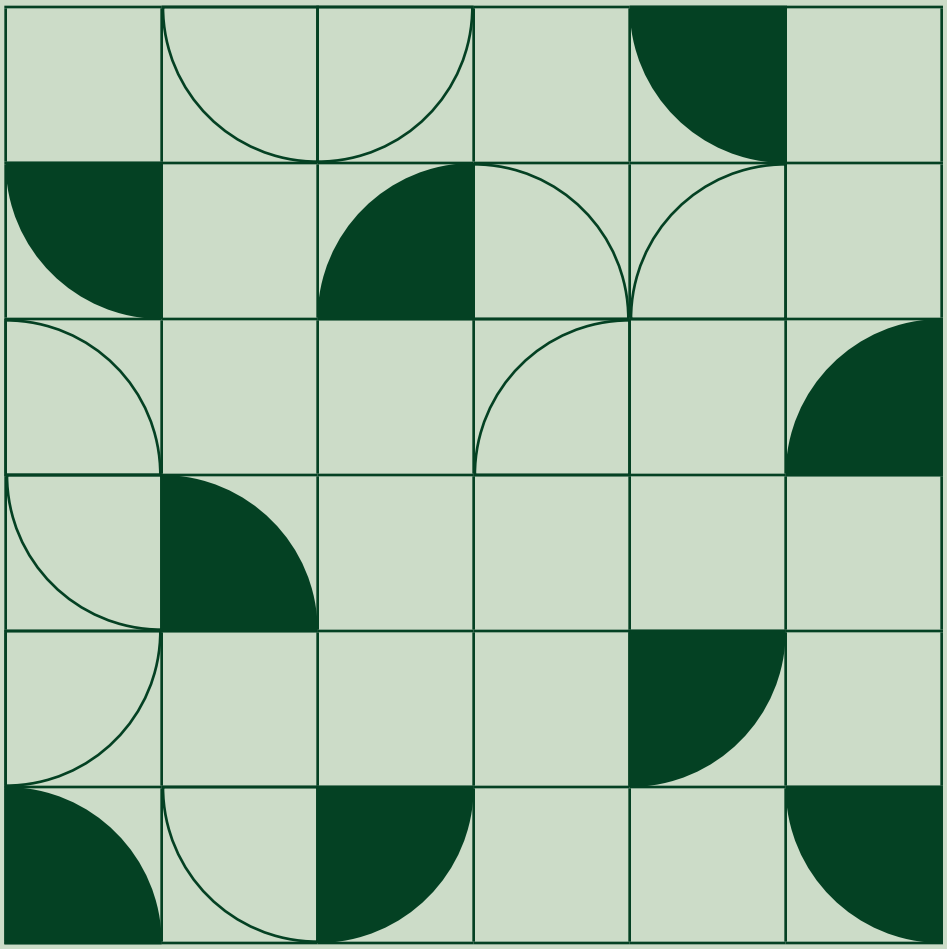
NI's 40-year history as an automated test and automated measurement technology provider can help HIL test engineers who need comprehensive I/O, high-end instrumentation, and an open platform to modify and reuse test rig parts as their test requirements change over time.

> The powerful combination of the NI VeriStand platform, LabVIEW FPGA, the real-time PXI module, and years of fast prototype development and experience with NI products helped us quickly and easily design and develop the whole HIL system.
>
> **G. Paviglianiti**
> Whirlpool Fabric Care, Advanced Development

Next Steps

- Read how Whirlpool is making washing machines more reliable with HIL
- See how Siemens uses HIL for wind turbine testing
- Learn how NI tools are used to control a heart simulator

Save Time and Maximize Reuse in HIL Testing with the SLSC Extension for PXI and CompactRIO



Overview

NI SLSC (Switch, Load, and Signal Conditioning) is an add-on for NI data acquisition products like PXI and CompactRIO. SLSC standardizes connectivity and provides a modular approach to signal conditioning, fault insertion, and other test needs. This white paper discusses the details of SLSC and showcases the growing ecosystem of SLSC modules and accessories created by NI and NI partners.

What is SLSC?

As technology advances, embedded software is becoming increasingly prevalent in complex, safety-critical systems such as automobiles and aircraft. Hardware-in-the-loop (HIL) testing is a methodology adopted by many industries to test embedded software while simulating the real-world inputs to that system. This approach allows testers to test early and often without risking damage to costly hardware and without incurring safety risks associated with unvetted product.

Because successful HIL testing hinges on accurately simulating real-world signals, custom signal conditioning and in-house development of simulated loads to guarantee signal integrity has been the norm. While it's true that no one is as knowledgeable about specific test requirements as the test engineer working on the system, NI has observed that much of this custom engineering is common across companies and industries. This observation led to the development of SLSC, an add-on to NI's powerful measurement and control platform that streamlines signal conditioning and test stimulus needs.

SLSC extends PXI and CompactRIO and consists of a chassis with modules. Each module has a Rear Transition Interface (RTI) which provides flexible standardized connectivity from SLSC to PXI or CompactRIO. Additionally, an SLSC system has a standard pinout that allows for the use of standard cables and eliminates point to point wiring.

SLSC plug-in modules can operate in the chassis in three different modes: stand alone, pass through, or cascaded. Cascaded mode can be used to cascade the signal path through multiple SLSC modules and implement functionality like signal fault insertion. You can choose from a variety of third-party modules or create your own modules based on a detailed hardware and software module development kit (MDK) from NI.

SLSC hardware is designed to simplify overall system integration by reducing system point-to-point wiring through signal accumulation and standard cable use. Each SLSC chassis consists of an SLSC digital bus, which you can use to discover, configure, and set parameters on the individual modules. Signals pass through SLSC modules either from the front connector or the rear transition interface (RTI) connector. You have the flexibility to design your own secondary backplane RTIs to reduce system wiring.

FIG 1 | SLSC extends the functionality of measurement and control hardware such as PXI

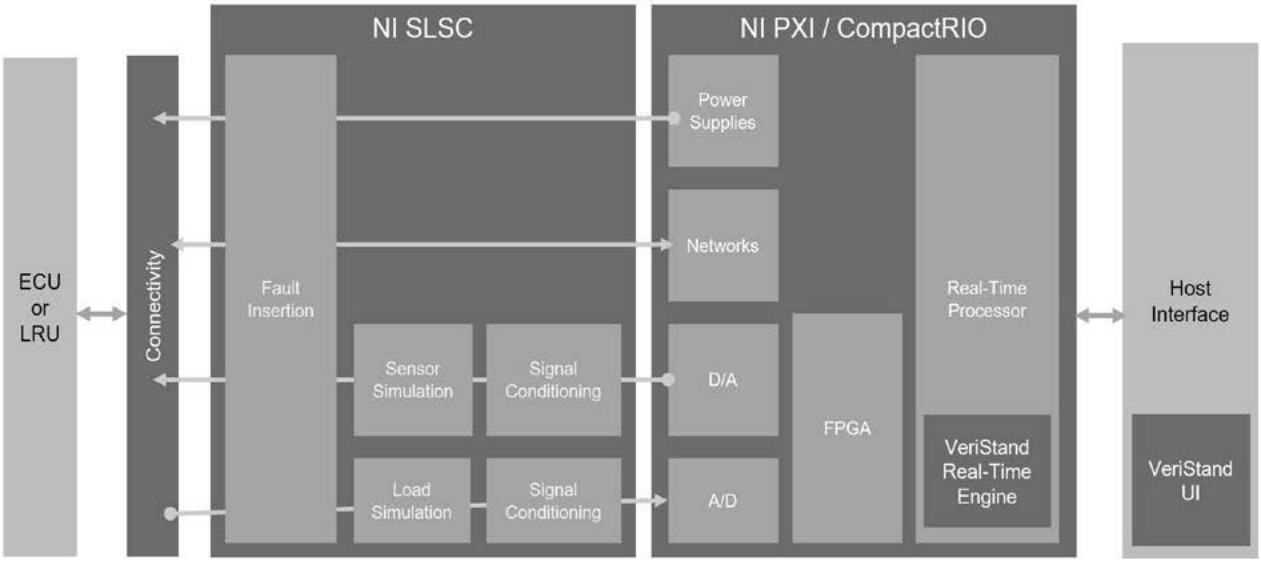


FIG 2 | SLSC extends the functionality of measurement and control hardware such as PXI

HIL System Diagram with SLSC

An SLSC system is comprised of four different component types: chassis, modules, rear transition interfaces (RTIs), and cables. The chassis and architecture have been designed and are produced by National Instruments with both NI and partners producing components that complete the ecosystem.

- SLSC Chassis—The SLSC chassis in conjunction with PXI is designed to handle more power and provide more board footprint than PXI alone, which makes it ideal for high power loads, switching, and signal conditioning. The chassis handles communication to the modules as well as mechanical fixturing and cooling.
- SLSC Modules—SLSC modules provide the switching, load simulation, or signal conditioning for the signal paths. Modules can communicate and be powered through the SLSC chassis or through auxiliary lines on the modules themselves.
- Rear Transition Interface (RTI)—The RTIs provide standard connectivity from SLSC modules to commercial off the shelf (COTS) cable options and back to DAQ modules. RTIs are generally SLSC module agnostic and are chosen based on the connectivity and cabling desired to bring signals back to the PXI or cRIO system.
- Cabling—One of SLSC's greatest strengths is reducing time consuming and expensive point to point wiring by allowing signals to be passed around through banks within standard available cables. A variety of cables exist within the SLSC ecosystem that allow you to appropriately split out and combine signal paths.

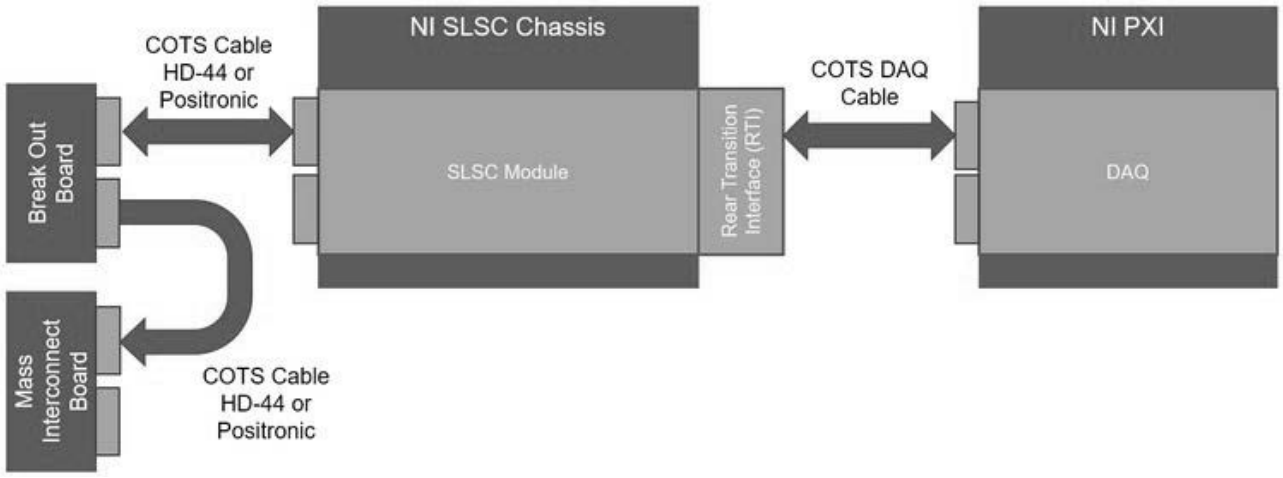


FIG 3 An example system diagram of SLSC components and their interface with a PXI DAQ card

The Growing SLSC Ecosystem

NI brings automated test and automated measurement expertise from an array of industries. This broad expertise allows us to create technology like SLSC from our observations of cross-industry needs. However, you need products that are targeted to your specific application. That's why SLSC, designed to be an open platform, is enhanced by modules created by NI, our partners, and you through our module development kit (MDK).

Our partners, like Bloomy who has created a MIL-STD-1553 module for SLSC, bring domain expertise that yield products that are specific to your industry and unique test needs.

Below is a list of products currently available for SLSC from NI and NI partners. Due to the complex nature of HIL systems, please contact us for assistance when reviewing the technology listed below. NI and our Alliance Partners are dedicated to ensuring your test requirements are met.

CORE COMPONENTS	NI PART NUMBER	VENDOR	DESCRIPTION
NI SLSC-12001	784532-01	NI	Chassis, 24 VDC
Module Dev Kit	785205-01	NI	Module Development Kit

MODULES	NI PART NUMBER	VENDOR	DESCRIPTION
SLSC-12101	785204-01	NI	Prototyping Module
SLSC-12201	785356-01	NI	33 V DIO Module with Thresholding
SLSC-12251	784444-01	NI	16 Ch 8 A Fault Insertion Module
SLSC-12252	786445-01	NI	8 Ch 30 A Fault Insertion Module
SET-1320	785671-01	SET	32 Ch Optical Isolated Output
SET-2010	786433-01	SET	Routing Base Module
SET-2210	786434-01	SET	Routing Instrument Connect Daughter Module
SET-2310	786435-01	SET	Routing Line Fault Daughter Module
SET-1310	Available through partner	SET	32 Ch Isolated Input
SET-1210	Available through partner	SET	Resistor Simulator Card
SET-141x	Available through partner	SET	Modular Card
SET-1623	Available through partner	SET	32 Ch ARINC TX/RX
SET-2315	Available through partner	SET	Real-time fault module
SET-1215	Available through partner	SET	High Precision Resistance Thermometer Simulator (PT100/PT1000)
SET-1220	Available through partner	SET	Capacitive Fuel Sensor Emulation
SET-1240	Available through partner	SET	xVDT Emulator

MODULES	NI PART NUMBER	VENDOR	DESCRIPTION
SET-1250	Available through partner	SET	32 Channel 4–20mA Current Sink
SET-2090	Available through partner	SET	32 Channel Pass Through Card
SET-1640	Available through partner	SET	PS15 Sensor Interface Card
Bloomy SLSC VDT/Resolver Simulation Module	Available through partner	Bloomy	VDT Conditioning
Bloomy Thermocouple Simulator	Available through partner	Bloomy	Thermocouple Simulation
Bloomy Load Module	Available through partner	Bloomy	Load Measurements
Bloomy Multipurpose Module	Available through partner	Bloomy	Multipurpose Module
AL-1010	Available through partner	Aliaro	Multifunction Module for Automotive
AL-2010	Available through partner	Aliaro	CAN/LIN/Ethernet Bus Switch Board
AL-3010	Available through partner	Aliaro	Resistor Emulation Board
ALMA-10201	786924-01	ALMA	2 Channel Oxygen Sensor Simulation
ALMA-10401	786925-01	ALMA	4 Channel Oxygen Sensor Simulation
OP8901	Available through partner	OPAL-RT	32-Channel High-Speed Digital I/O Conditioning Board with FIU
OP8920	Available through partner	OPAL-RT	32-Channel Isolated Digital I/O Conditioning Board with FIU
OP8930	Available through partner	OPAL-RT	16-Channel Analog I/O Conditioning Board with FIU
OP8940	Available through partner	OPAL-RT	32-Channel Pass-Through with FIU Board

REAR TRANSITION INTERFACES*	NI PART NUMBER	VENDOR	DESCRIPTION
RTI-12301B	786330-01	NI	RTI DIO 32 for SHC68-C68-RDIO2
RTI-12302	785377-01	NI	64 Ch for RDIO2
RTI-12303	785375-01	NI	DIO/AO/AI X 4 Bank Nanofit
RTI-12304	785374-01	NI	DIO 37DSub
RTI-12344	785376-01	NI	AO 37DSub
RTI-12305	785896-01	NI	HD44 Connector

*SEE MODULE USER MANUALS FOR RTI RECOMMENDATIONS.

CABLES*	NI PART NUMBER	VENDOR	DESCRIPTION
SHC68-C68-RDIO2	156166-01	NI	1 meter
SHC68-C68-RDIO2	156166-02	NI	2 meter
SH37F-37M-1	778621-01	NI	37-pin Female to Male Shielded I/O Cable, 1 meter
SH37F-37M-2	778621-02	NI	37-pin Female to Male Shielded I/O Cable, 2 meter

*SEE MODULE USER MANUALS FOR CABLE RECOMMENDATIONS.

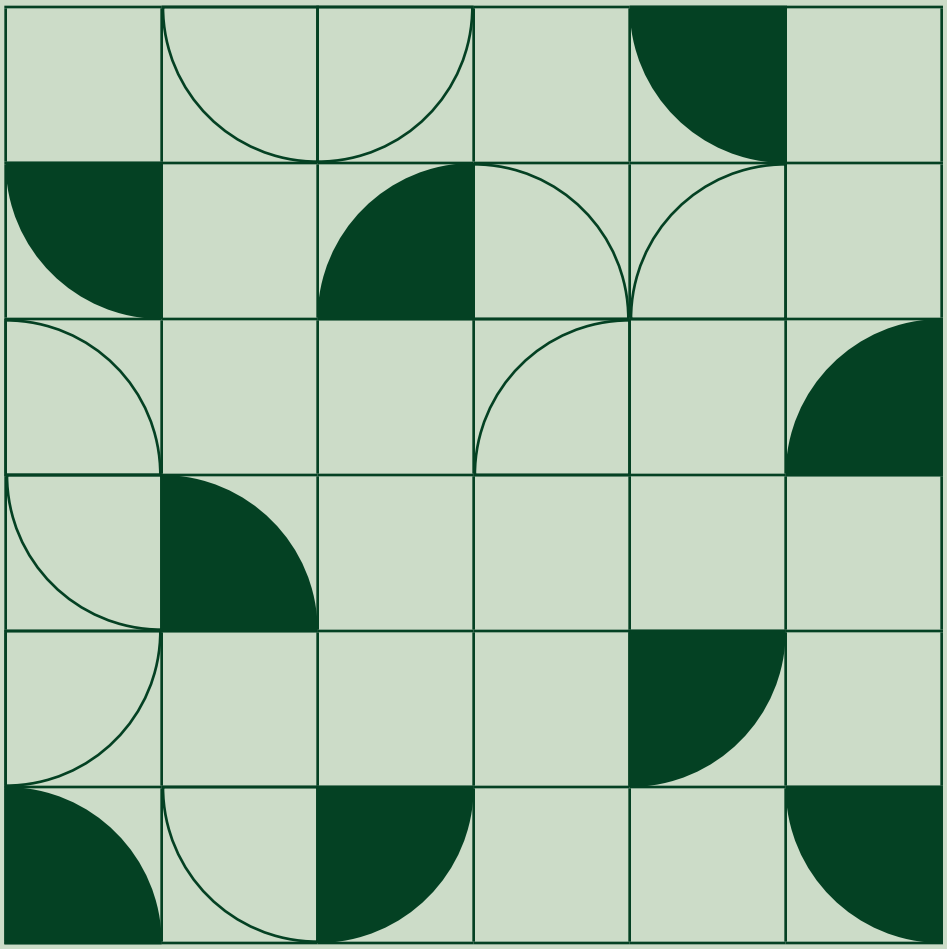
ACCESSORY	NI PART NUMBER	VENDOR	DESCRIPTION
Front Panel Filler Kit	785206-01	NI	4 count
Rear Panel Filler Kit	785207-01	NI	4 count
Filter Replacement Kit	785208-01	NI	N/A
Fan Replacement Kit	785218-01	NI	N/A
Power Connector	785219-01	NI	5 count
RTI Strain Relief	785999-01	NI	5 count

While NI and our partners are developing and releasing modules we feel will be widely used, we know that there are specific requirements that are unique to your test system. To address these needs, our partners can design and deliver a test system based on NI's platform that is tailored to your unique test demands. 786924-01

Next Steps

- See How Saab Saves Time and Money with SLSC
- Learn More About SLSC

Getting Started with VeriStand Hardware-in-the-Loop Software



Overview

VeriStand is a software tool that provides a framework for real-time testing applications such as embedded software validation and real-time control and monitoring of mechanical test cell applications. It contains a wide variety of features to help you get up and running more quickly. Review this article to get started with VeriStand and learn about some of its built-in functionality.

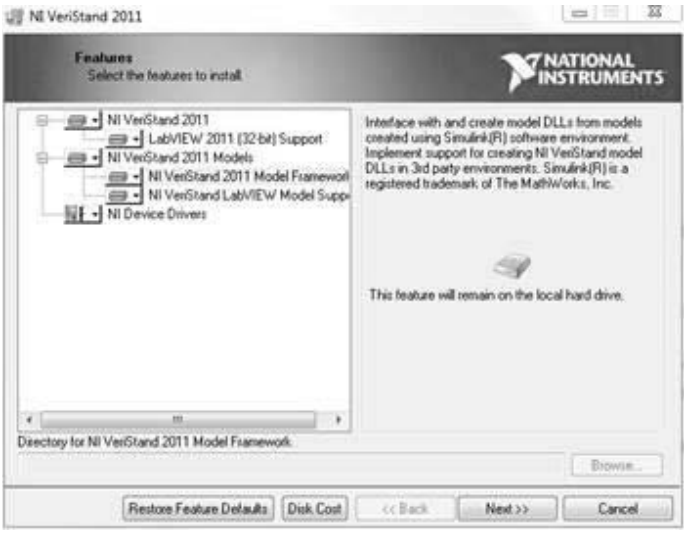
Software Installation

The first step is to install VeriStand software and associated driver software on your Windows computer. Then install VeriStand Engine software on all real-time targets to which you are deploying.

Host PC Software Installation

Install VeriStand by using the VeriStand DVD or downloading it at ni.com/veristand/download. When you run the installer, you have the option to select the components you need for your application.

- LabVIEW support adds a VeriStand palette to LabVIEW for automated control of VeriStand from LabVIEW.
- The VeriStand Model Framework adds support for building simulation models in third-party environments such as The MathWorks, Inc. Simulink® simulation software and ANSI C.
- With VeriStand LabVIEW model support, you can convert a LabVIEW VI into a simulation model that can be used in VeriStand.



Note: The VeriStand Model Framework can be installed independently and licensed free of charge. For machines that do not need the VeriStand environment, select only the VeriStand Model Framework for installation. After you have installed the VeriStand Model Framework, you can license it for free by using the VeriStand Model Generation Activation Utility at http://joule.ni.com/nidu/cds/view/p/id/2466/lang/en.

After you install the necessary VeriStand software components, install **NI Device Drivers** from either the Device Drivers DVD or from ni.com/drivers. Install the following drivers:

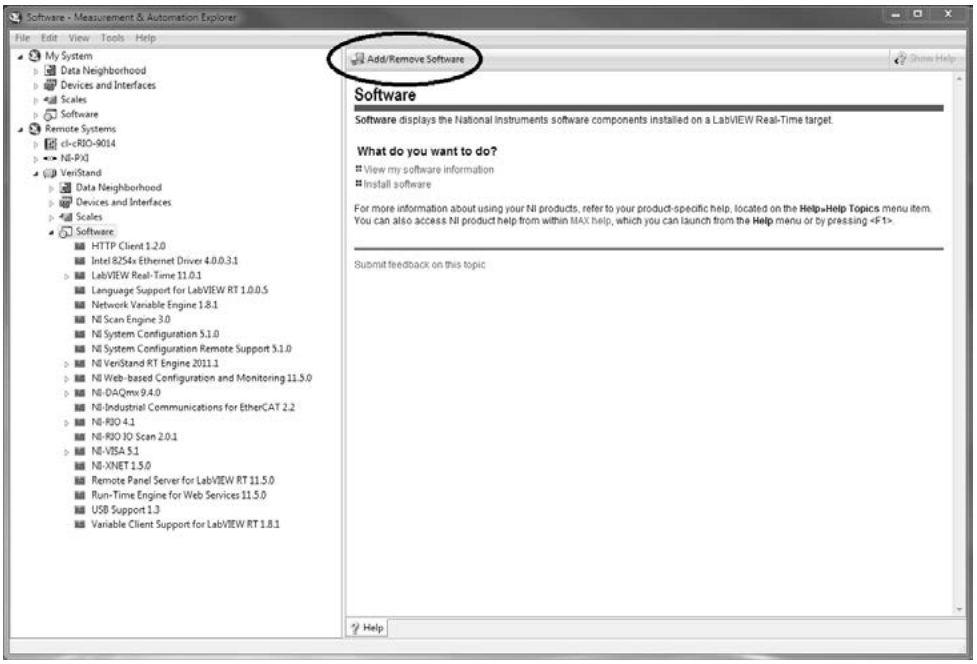
- NI-DAQmx
- NI-VISA
- **Optional**: NI-RIO (for CompactRIO or FPGA functionality)
- **Optional**: NI-XNET (for CAN, LIN, or FlexRay functionality)
- **Optional**: NI-Industrial Communications for EtherCAT (for CompactRIO Scan Engine and EtherCAT functionality)

After all installations are complete on the host PC, open **Measurement & Automation Explorer (MAX)** to confirm the installed software by expanding the **My System»Software** item within the tree structure as shown below.

Software Installation on the Real-Time Target

After you have all the necessary software on your host computer, follow these steps to install software on the real-time target computer.

In MAX, select your VeriStand real-time target under the Remote Systems tree item. Select the software item for your real-time target and choose **Add/Remove Software**.



For CompactRIO targets only: Select the **Custom Software Installation** option.

Choose to install the VeriStand Engine. This installs the VeriStand Engine as well as the software components required to run it.

Optional: To implement a distributed real-time test system that uses GE reflective memory interfaces, select the GE reflective memory software on the target for installation.

Optional: If you are using a CompactRIO device and want to install the Scan Engine Custom device, select the NI-Industrial Communications for EtherCAT and I/O Variable Remote Configuration Web Service for installation as well.

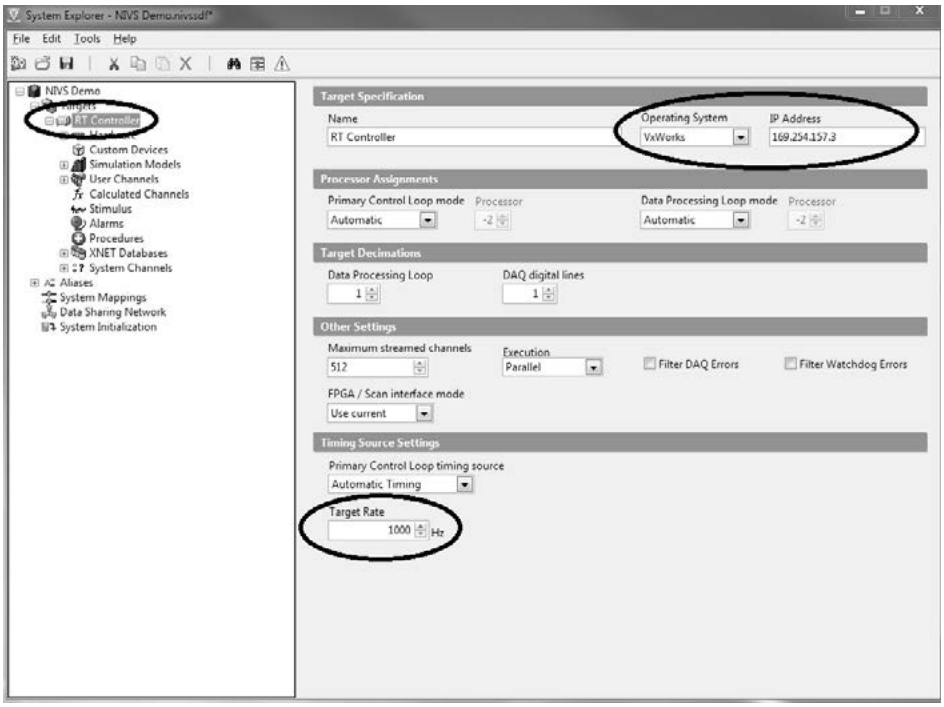
Complete the software installation process and reboot your real-time target. You can see a list of currently installed software on your target by looking under the **Remote Systems»VeriStand Target»Software** item within the MAX tree structure as previously depicted.

VeriStand Project Setup

Open VeriStand **(Start»Programs»National Instruments»NI VeriStand)** and create a New NI VeriStand Project. A window prompts you to choose your project name and directory path.

Note: NI recommends keeping all project dependencies relative to the project file location and placing them in the same folder or in a subfolder. This includes items such as Workspace files (.nivsscren), System Definition files (.nivssdf), Stimulus Profile files (.nivsstimprof), Real-Time sequence files (.nivsseq), models, and FPGA bitfiles or configuration files.

In the NI VeriStand Project Explorer window, expand the **System Definition File** tree item and open the *.nivssdf file found there by right-clicking on the file and selecting **Launch System Explorer**.



Configuring the VeriStand Engine Using the System Explorer

While in the Controller section, the setting for Target Rate in the Timing Source Settings section sets the Primary Control Loop rate on your target. The Primary Control Loop controls the timing for the VeriStand Engine and keeps updated channel values. Find more information on the Primary Control Loop and other individual loops running on the VeriStand Engine in NI VeriStand Engine Architecture.

Expand **RT Controller** in the tree and note the various items you can add to your system definition.

Selecting Deployment Targets

PXI Real-Time and cRIO-908x Targets

Configure the PXI target by highlighting **Controller** in the tree, selecting **PharLap** for the OS, and using the same IP address displayed for the PXI system in MAX. Rename the controller name to a unique name of your choosing.

All Other CompactRIO Real-Time Targets

Configure the CompactRIO target by highlighting **Controller** in the tree, selecting **VxWorks** for the OS, and using the same IP address displayed for the CompactRIO system in MAX. Rename the controller name to a unique name of your choosing.

Running the VeriStand Engine on the Localhost Windows Computer

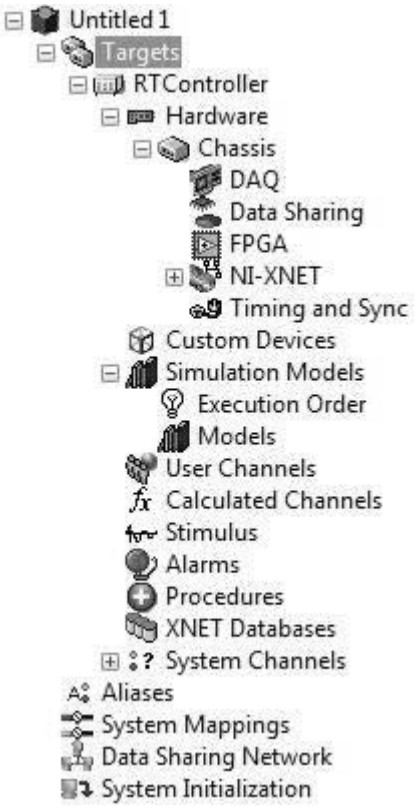
Configure the Windows target by highlighting **Controller** in the tree and selecting Windows for the OS. Note localhost is the automatic selection for IP address, which indicates that the system definition runs on the host PC. Rename the controller name to a unique name of your choosing.

01
Hardware

Expand **Hardware** and then **Chassis**. This is where you identify your NI-DAQ, Data Sharing (Reflective Memory), NI-FPGA, NI-XNET, or Timing and Sync devices. You can also add multiple chassis.

02
Custom Devices

Customize and extend the out-of-box functionality of VeriStand into a device that you can add to a system definition file and deploy to a real-time target. VeriStand includes three custom devices that you can add here as well as any custom devices you have created yourself. Check the NI VeriStand Add-Ons Community to view other existing custom devices and the NI VeriStand Custom Device Developer's Guide when considering building your own custom device.NI-XNET, or Timing and Sync devices.



03
Simulation Models

Expand **Simulation Models**. Add your compiled models from one of the supported modeling environments listed in the document Using Simulation Models With NI VeriStand. If you have multiple models, you also can set the order in which models execute on the VeriStand Engine.

04
User Channels

User channels store a single value and can function as variables to be used in other areas of your system definition.

05
Calculated Channels

Calculated channels are created to perform calculations on other channels in the system. You can create your own formula or perform built-in operations such as lowpass filter, average, or peak and valley.

06
Stimulus

View and configure the stimulus generators in the Legacy Stimulus Profile Editor, which simulate real-world signals to perform tests on a system. If you're starting a new project without any legacy VeriStand dependencies, ignore this section.

07
Alarms

Configure an alarm to warn the user that the value of a channel has gone outside a specified range of values. Alarms can also trigger a procedure to execute.

08
Procedure

Configure a procedure to execute a set of actions on the VeriStand Engine. A procedure can be signaled to begin at startup or trigger off an alarm or another procedure.

09
NI-XNET Databases

Add any NI-XNET databases to your system. Databases can be CANdb (.dbc), NI-CAN (.ncd), LDF (.ldf), or FIBEX (.xml) files.

10
System Channels

Expand **System Channels** to view the channels that monitor the state or condition of various system items. These are often used for troubleshooting system behavior.

11
System Mappings

This section displays all defined system mappings, which are connections between source and destination channels. These are configured in the System Configuration Mappings window, which is examined in the next section.

12
Data Sharing Network

Add and configure a reflective memory network. To learn more about using reflective memory with VeriStand, go to Creating a Distributed System With NI VeriStand.

13
System Initialization

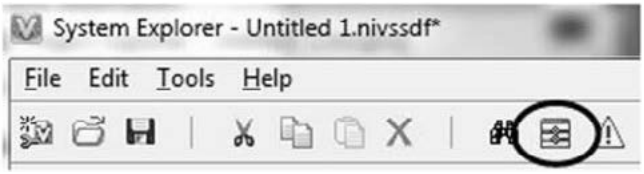
If you have multiple targets, you can use this section to set the order that targets deploy relative to each other and determine target reboot action.

14
Aliases

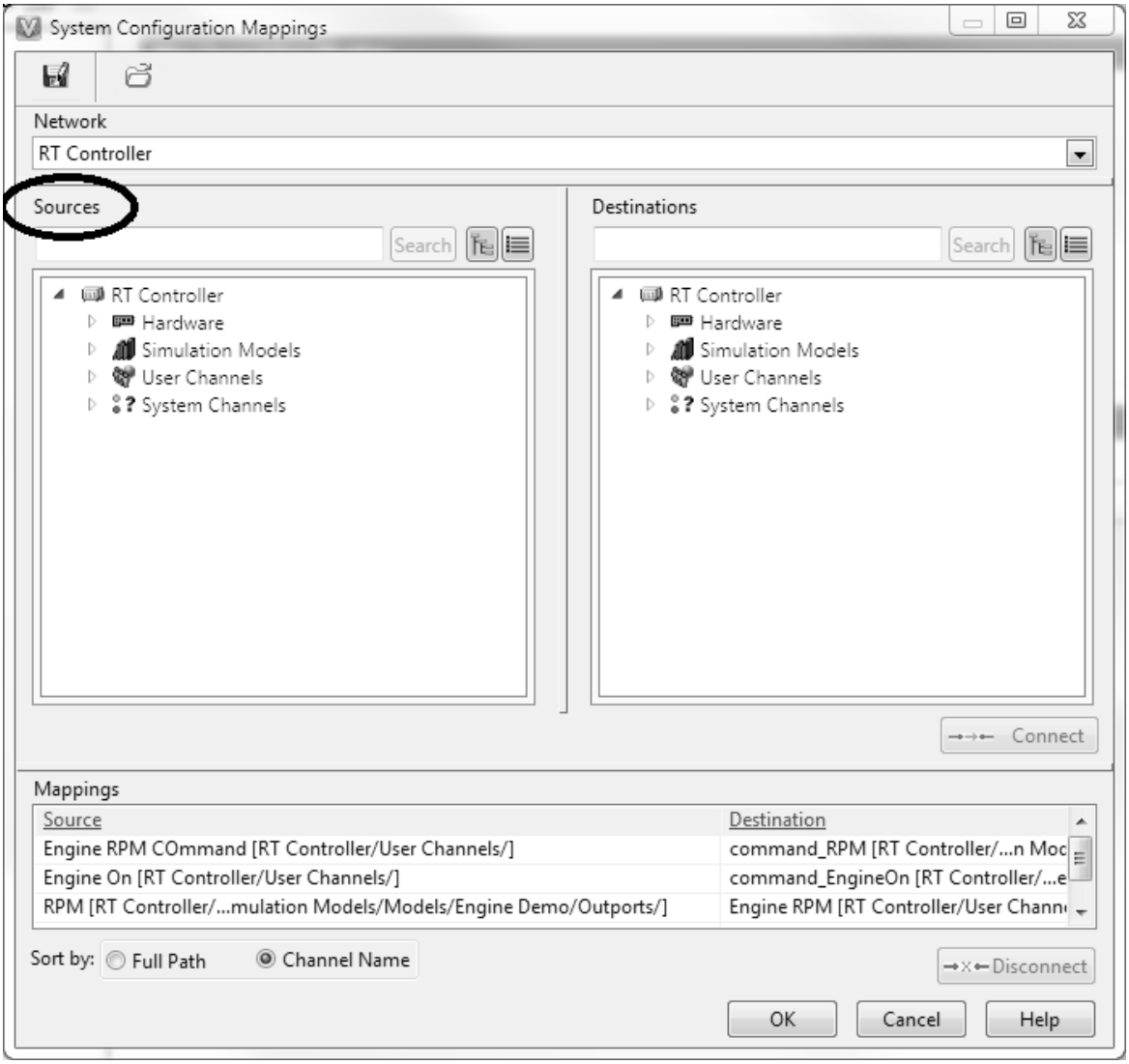
Configure an alias to give a channel or group of channels a unique name in your system definition. Right-click on **Alias** and select **Add Alias**. Type in the alias name and description and then click the **Browse** button next to the channel to select the channel to rename. Aliases are useful for many reasons, including sharing one workspace with multiple system definitions and mapping workspace objects to those aliases. Because of this, you can rename system definition channels within an alias without the workspace losing its mapping.

Mapping System Channels

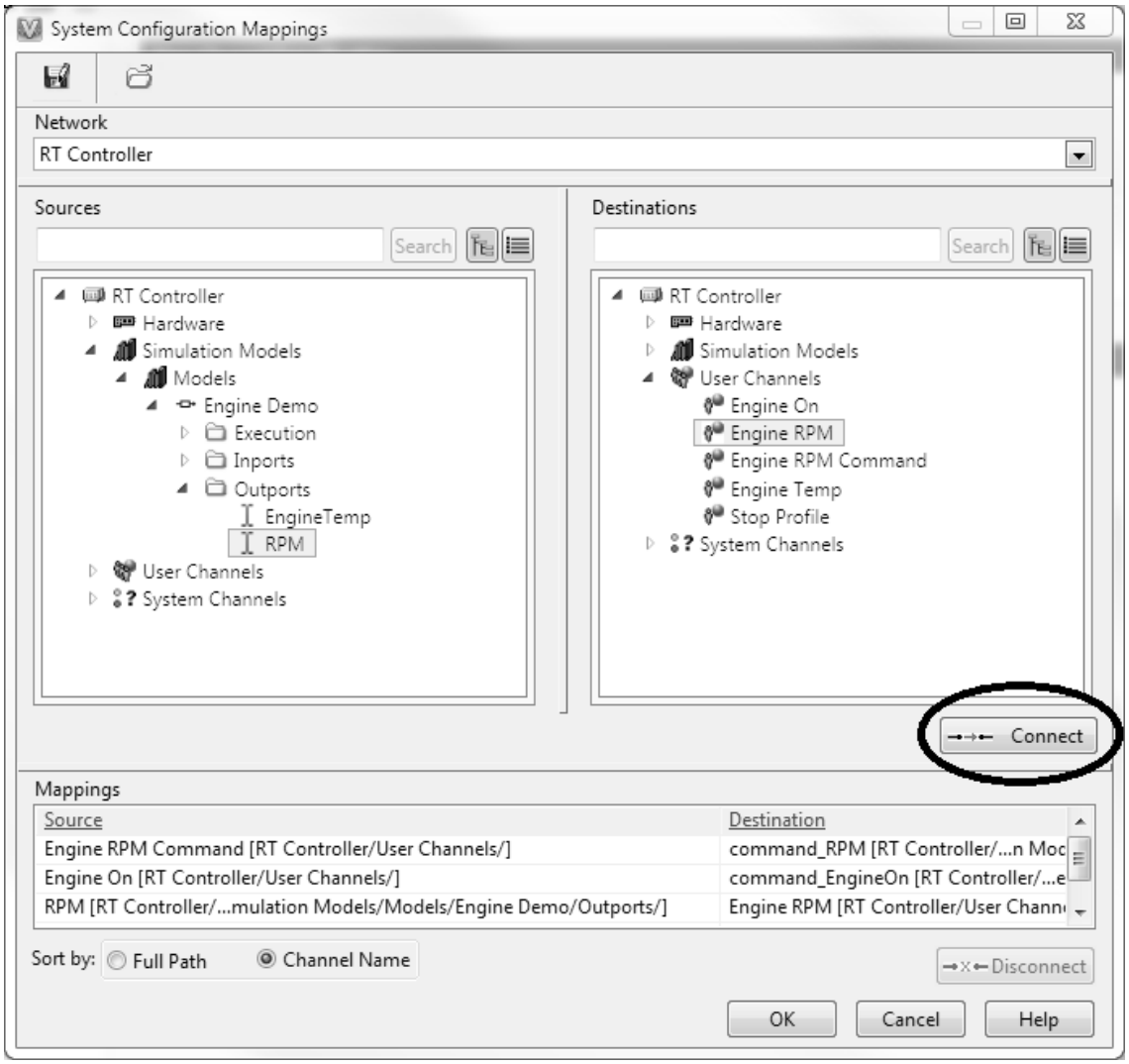
You can easily connect channels to each other in VeriStand using a mapping tool. This tool helps you quickly connect simulation models to physical I/O as well as any other channel in your system. To configure mappings for your VeriStand system, select **Tools»Edit Mappings** or click on the **Configure Mappings** button, shown below, to connect channels to each other such as a model output to a physical channel or a calculated channel to an alias.



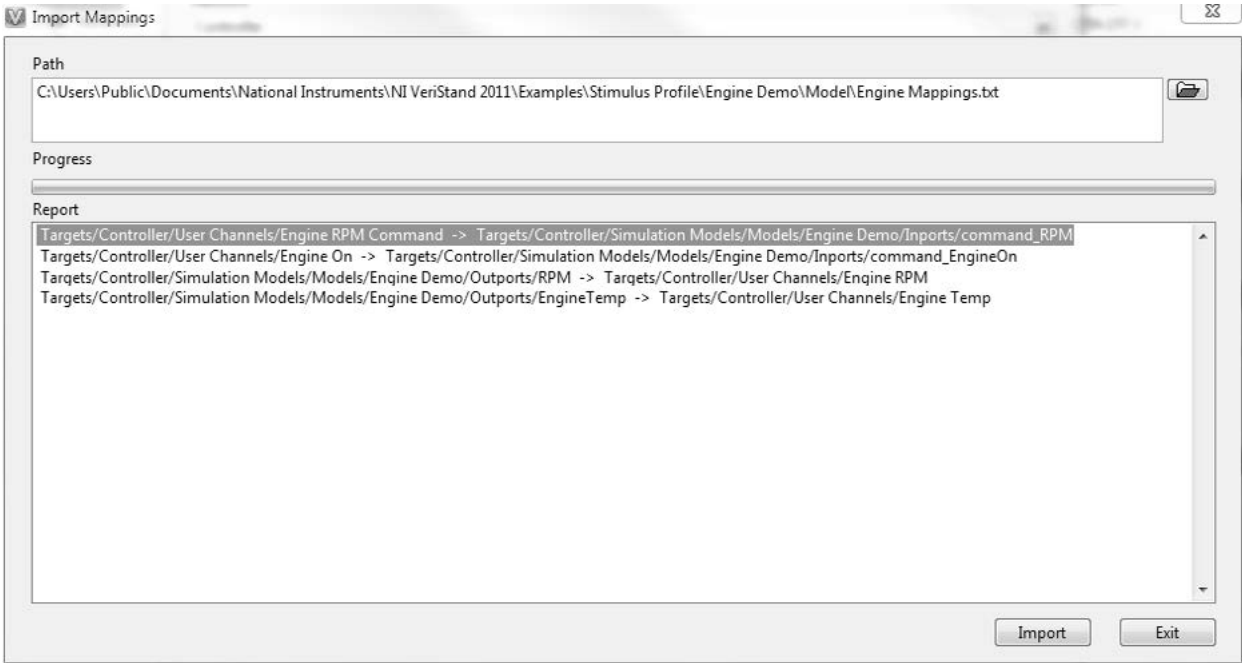
Select a channel in the tree under Sources in the System Configuration Mappings window that opens.



Select a channel under Destinations and then click Connect to map the channels. Note that the Source and Destination channels now appear under Mappings.



System channel mappings can be exported and saved to file. You can then import this file to automate the system mapping process at a later time.



Deploying VeriStand Projects

After you have configured your system definition, save and close the System Explorer. There are two options for running VeriStand projects.

Run: Launches the Workspace window. If you have configured the system to run on a PC, clicking the button begins running the project. If you have configured the system to run on an real-time target, clicking the Run button deploys the system definition file if one is not already running. If a system definition file already is running on the real-time target, clicking the Run button connects to the target and launches the Workspace window without redeploying the system definition file.

Deploy: Clicking the button deploys the system definition to the target that you specified in the System Explorer. However, it does not open the Workspace window. If a system definition file already is running on the real-time target, deploying a new system definition will replace the system definition that is currently running.

Building Simulation Models for use with VeriStand

You can use VeriStand with a wide variety of simulation modeling environments and programming languages. Model subsystems can be built independently and integrated within the VeriStand environment, so you can easily replace simulated components with real components as they become available.

Generating Models From The MathWorks, Inc. Simulink Simulation Software

If you are using an NI real-time target and you don't know which OS is running on the target, view What Operating System Is My Real-Time Controller Running and Why?

- For model deployment on Windows and PharLap-based systems, view Setting Up The MathWorks, Inc. MATLAB® Environment to Work in NI VeriStand.
- For deployment on VxWorks-based CompactRIO systems, go to Developing Shared Libraries for the cRIO-901x and Other VxWorks Targets for Windows XP development computers and Generating Models From The MathWorks, Inc. Simulink Simulation Software for Deployment on VxWorks Systems for Windows Vista and later.

Additional Modeling Environments

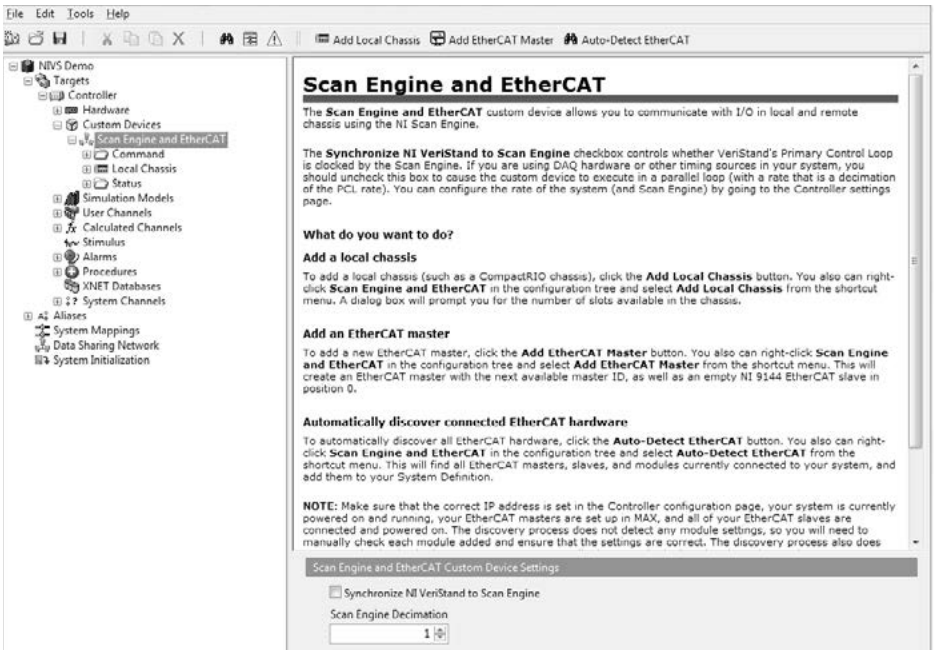
For more information on interacting with models from other modeling environments and programming languages in VeriStand, go to Using Simulation Models With NI VeriStand. This document has a complete list of supported modeling environments that have been tested and verified as being able to create compiled models that can be imported in VeriStand.

Using the EtherCAT and Scan Engine Add-On for VeriStand

With the Scan Engine and EtherCAT custom device, you can easily read scanned I/O from C Series modules located in a CompactRIO or NI 9144 EtherCAT chassis. The add-on also supports the use of custom FPGA personalities with an NI 9144 chassis.

Note: The EtherCAT and Scan Engine add-on for VeriStand is compatible with CompactRIO targets and the NI 9144 EtherCAT expansion chassis.

Download NI VeriStand Add-On: Scan Engine and EtherCAT custom device from the VeriStand developer community and follow the installation instructions.



To add the **Scan Engine and EtherCAT Custom Device** to your VeriStand system, follow these steps:

1. Open VeriStand and your System Definition.
2. Right-click on **Custom** Devices and select **Scan Engine and EtherCAT**.
3. Select **Auto-Detect Modules** or right-click and select **Add Local Chassis** to use CompactRIO chassis and manually choose the appropriate module for each slot and correct settings for each module.

Next Steps

- VeriStand has many features you can use out of the box without programming. To see video demonstrations of some of these features, visit ni.com/veristand/demos.
- For an in-depth walkthrough of the VeriStand environment, download the VeriStand tutorial.
- To learn more about creating add-ons for VeriStand, watch the VeriStand add-ons webcast.
- For instructor-led training on VeriStand features, take the NI VeriStand Fundamentals training course.
- Go to our contact page to connect with a sales representative.