# LabVIEW™ Simulation Interface Toolkit

The LabVIEW Simulation Interface Toolkit provides two features for using LabVIEW with the software of The MathWorks, Inc. known as Simulink® and Real-Time Workshop®. One feature is the **NI Sinks and Source** palette, which you can use to graphically configure "Sinks" and "Sources" blocks in a simulation model in Simulink. With the second feature, you can use Real-Time Workshop to build files to run simulation models within the LabVIEW environment.

Simulink, an add-on toolkit for The MathWorks MATLAB® software, provides a graphical environment for the design and interactive execution of dynamic system models. You can create models in Simulink using customizable blocks. Using graphical wires to connect the blocks, you can specify data flow.

Real-Time Workshop is an add-on package for Simulink that generates C code from Simulink models. Real-Time Workshop then compiles the code into dynamic link libraries (DLLs). You can use these DLLs for simulation or use them on various hardware targets in real-time control applications.

LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. LabVIEW applications often are called virtual instruments, or VIs, because their appearance and operation often imitate physical instruments, such as oscilloscopes and multimeters.

The LabVIEW Simulation Interface Toolkit integrates Simulink and Real-Time Workshop with LabVIEW in a way that allows you to develop and test control systems first developed in the Simulink simulation environment. Refer to the *LabVIEW Bookshelf* for information about LabVIEW and to access the LabVIEW help resources. Access the *LabVIEW Bookshelf* by selecting **Start»Programs»National Instruments»LabVIEW»Search the LabVIEW Bookshelf**.



*October 2002*
*370420A-01*





NATIONAL INSTRUMENTS

# Contents



# Installation

To use the LabVIEW Simulation Interface Toolkit, you must be a properly licensed user of and have the following software installed on the development system.

- The MathWorks MATLAB version 6.0 or later
- The MathWorks Simulink 4.0 or later
- The MathWorks Real-Time Workshop 4.0 or later
- Microsoft Visual C++ 5.0 or later
- National Instruments LabVIEW 6.1 or later

Complete the following steps to install the LabVIEW Simulation Interface Toolkit.

1. Insert the LabVIEW Simulation Interface Toolkit CD.
2. Run the `setup.exe` program.
3. Follow the instructions that appear on the screen.

# Configuring NI Sinks and Sources Blocks in Simulink

The blocks located in the **NI Sinks and Sources** palette in the Simulink Library Browser provide the Simulink user two types of functionality. The display-oriented blocks—Scope, XY Graph, and Display—are linked to a LabVIEW user interface that displays values and signals generated in Simulink. These blocks use a LabVIEW waveform chart to display the values and signals. The following figure is an example of the LabVIEW user interface linked to a Scope block.


**Figure 1.** Scope Block

The waveform chart used in this Scope block has the following features:

Use the **Scale Lock** button, shown at left, to toggle autoscaling for each scale, the visibility of scales, scale labels, and plots, and to format scale labels, grids, grid lines, and grid colors.

Use the **Autoscale** button, shown at left, to automatically adjust the horizontal and vertical scales to reflect the data you wire to them.

Use the Operating tool to click the **Scale Format** button, shown at left, to configure the format, precision, and mapping mode.

Use the Panning tool, shown at left, to pick up the plot and move it around on the display.

Use the **Zoom** button, shown at left, to zoom in and out of the display.

Use the Cursor Movement tool, shown at left, to move the cursor on the graph.

Refer to Chapter 11, *Graphs and Charts*, of the *LabVIEW User Manual* for information about the waveform chart.

The other blocks on the **NI Sinks and Sources** palette are built-in Simulink blocks whose parameter dialog boxes are provided by a LabVIEW VI. The parameters in these blocks are used to set the parameters in the Simulink built-in blocks.

Refer to the Simulink documentation for information about the blocks.

# Using a Model DLL for Simulation in LabVIEW

Real-Time Workshop creates a DLL, called a model DLL, based on a Simulink model. The LabVIEW Simulation Interface Toolkit places a set of Simulation Interface VIs in the `labview\vi.lib\addons\Simulation Interface` folder. These VIs help you call the model DLL from LabVIEW.

Refer to the *LabVIEW Simulation Interface Toolkit Help* for information about the Simulation Interface VIs. Access the *LabVIEW Simulation Interface Toolkit Help* by selecting **Help»LabVIEW Simulation Interface Toolkit Help** in LabVIEW.

**Note** The model DLL contains all aspects of the Simulink model but is independent of the Simulink model.

## Building a Model DLL Using Simulink and Real-Time Workshop

Complete the following steps to build a model DLL that LabVIEW can use to run a simulation model.

**Note** If you are using the LabVIEW Real-Time (RT) Module, select a local system as the target platform in LabVIEW before building a model DLL. Real-Time Workshop cannot build a model DLL if you select a remote device.

1. Open the Simulink model for which you want to build a model DLL.
2. Select **View»Show Library Browser** to access the **NI Sinks and Sources** library.
3. Place generic inport and outport blocks from the **NI Sinks and Sources** library on the Simulink model. The inport and outport blocks correspond to the inputs and the outputs of the model DLL.

   LabVIEW uses these blocks to send data to and receive data from the model DLL. Refer to the *Exchanging Data with the Model DLL* section for information about sending and receiving data.

**Note** If the Simulink model already contains inport and outport blocks, continue to the next step.

4. Select **Simulation»Simulation Parameters** to display the **Simulation Parameters** dialog box.
5. Click the **Solver** tab. Set the **Type** option to **Fixed-step** in the **Solver Options** section.

6. Click the **Real-Time Workshop** tab. Enter `nidll.tlc` in the **System target file** box.

   You also can click **Browse** to open the **System Target File Browser** dialog box. Select **nidll.tlc—LabVIEW DLL Target** from the listbox and click the **OK** button to continue.

7. Click the **Build** button in the **Category** section to begin building the model DLL.

   The Command Window on the MATLAB desktop displays the status of the Real-Time Workshop as it builds the model DLL.

8. Select a target IP address from the **Select target IP address (for DLL download)** dialog box that appears.

**Note** After the LabVIEW Simulation Interface Toolkit builds a model DLL, it places the model DLL into a project folder on the computer. You have the option of downloading the new model DLL to a remote target.

Complete one of the following three steps to select a target IP address:

- Select the IP address from the list of previously used targets in the **Target IP Address** pull-down menu. Click the **DOWNLOAD** button to continue.
- Select **other remote target** from the **Target IP Address** pull-down menu to enter a new IP address. Click the **DOWNLOAD** button to continue.
- Click the **SKIP DOWNLOAD** button if you do not want to transfer the model DLL to a remote target. Click the **SKIP DOWNLOAD** button again to continue.

**Note** If you choose not to transfer the model DLL file to the remote target when prompted, you can use the `ftpToTarget` library located in the model project folder to transfer the file later. You also can use FTP to directly transfer the file to the `c:\ni-rt\system` folder on the remote target.

The following message in the MATLAB desktop Command Window indicates that Real-Time Workshop has completed building the model DLL.

**`### Successful completion of Real-Time Workshop build procedure for model: ModelName.`**

## Understanding the Model DLL

To create a model DLL, Real-Time Workshop converts the Simulink model and any of its submodels into C code and compiles them into a model DLL named *ModelName*.dll, where *ModelName* is the name of the Simulink model. Real-Time Workshop then places the model DLL into a new model project folder, *ModelName*_nidll_rtw, in the current working directory of MATLAB.

**Note** The name of the model DLL must conform to the 8.3 file naming convention so the model DLL can run on a remote LabVIEW RT target.

To call a model DLL from LabVIEW, use the Simulation Interface VIs that LabVIEW Simulation Interface Toolkit provides. The Simulation Interface VIs, located in the labview\vi.lib\addons\Simulation Interface folder, allow initialization, finalization, single-step execution, and manipulation of the internal parameters of the model DLL.

In addition, LabVIEW creates two example VIs and places them in the model project folder. The first example VI, *ModelName*_daq_driver.vi, is a data acquisition (DAQ) application that reads input values from a DAQ board, sends the values as inputs to the SIT Step Model VI, retrieves the outputs from the SIT Step Model VI, and writes the outputs to the DAQ board. The second example VI, *ModelName*_driver.vi, is a version of the same VI, with the DAQ calls replaced by controls and indicators.

Refer to the *Creating a LabVIEW VI to Call the Model DLL* and the *Creating a Real-Time Model-Based Control* sections for examples using the Simulation Interface VIs in a LabVIEW VI. Refer to the *LabVIEW Simulation Interface Toolkit Help* for information about the Simulation Interface VIs.

The model project folder also contains a LabVIEW custom control, *ModelName* modelParamEnum.ctl, that is an enumeration of all tunable parameters of the model DLL. Refer to the example VIs in the model project folder to see this control used with the SIT Set Model Parameters VI to modify the model DLL.

## Downloading the Model DLL to a Remote Target

Statically linked DLLs are downloaded automatically to an RT execution target. However, to facilitate simultaneous handling of multiple model DLLs, the SIT Initialize Model VI loads the model DLL dynamically. Then the SIT Initialize Model VI sends a 32-bit signed integer reference to the other Simulation Interface VIs to identify the model DLL.

Because the model DLL loads dynamically, LabVIEW RT does not automatically download the model DLL to an RT execution target along with other LabVIEW VIs. So while building the model DLL, the LabVIEW Simulation Interface Toolkit prompts you to transfer the model DLL to a remote target.

## Exchanging Data with the Model DLL

After you create a model DLL that LabVIEW can call, you can exchange data with the model DLL from a LabVIEW VI. Every call to the SIT Step Model VI takes one time step in the simulation model. To continuously exchange data with the model DLL, use the Simulation Interface VIs with a loop structure.

Any LabVIEW VI that you create can exchange data with the simulation model through the inputs and outputs of the SIT Step Model VI. The data inputs are values you want to send to the model DLL with each loop iteration, and the data outputs are values you want to receive from the model DLL after each loop iteration.

In Simulink, the inputs and outputs correspond to the generic inport and outport blocks that you added to the block diagram. In LabVIEW, the inputs and outputs of the model DLL correspond to arrays of double-precision values transferred to and from the SIT Step Model VI. These inputs and outputs of the model DLL accept scalar or array data.

**Note** When you connect an inport or outport block in Simulink to an array of data, the elements of that array appear as consecutive elements in the corresponding LabVIEW array. During the build process, the LabVIEW Simulation Interface Toolkit creates a `portsreadme.txt` file in the model project folder. This file describes the mapping between the LabVIEW arrays and Simulink inport and outport blocks.

You also can exchange data with the model DLL through the inputs and outputs of the SIT Set Model Parameters. The SIT Set Model Parameters VI updates the model DLL with data from controls on the front panel. It also sends data to the indicators on the front panel when the parameters change.

# Creating a LabVIEW VI to Call the Model DLL

The example VI, *ModelName*_driver VI, in the following figure communicates with the model DLL using a While Loop. Each loop iteration in the VI corresponds to one time step of the Simulink model.

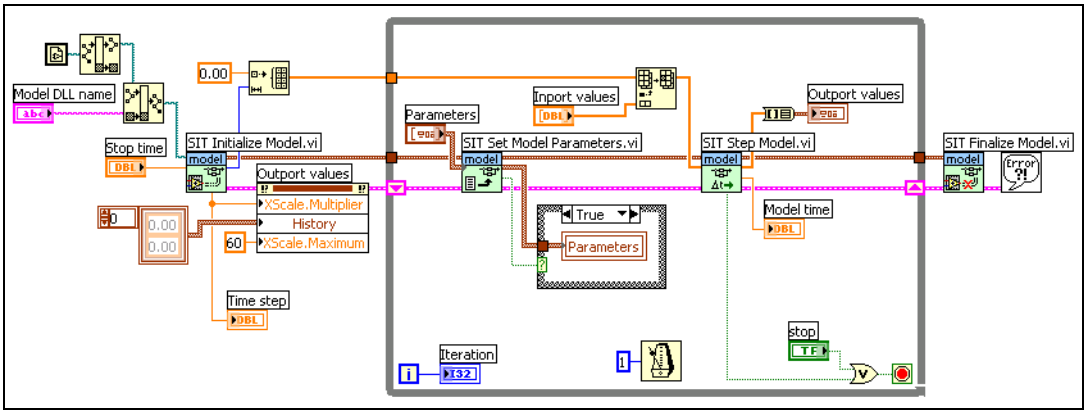


**Figure 2.** *ModelName*_driver VI

The SIT Initialize Model VI at the left of the block diagram prepares the model DLL for execution, sets the final time for the model DLL, and returns the following information about the model DLL: the time step, number of inports, and number of outports.

The SIT Step Model VI takes one time step of the simulation model with each iteration of the loop, accepting the input array and returning the output array. SIT Step Model VI also indicates if this is the final time step of the loop.

The loop terminates either when the user presses the **Stop** button on the front panel or after the SIT Step Model VI indicates that the model DLL has reached its specified stop time. The loop exits and the SIT Finalize Model VI executes.

*ModelName*_driver VI uses the Simulation Interface VIs to update parameters dynamically within the model DLL as it executes. The SIT Set Model Parameters VI receives parameters from the controls on the front panel and updates the model DLL when the parameters change.

# Creating a Real-Time Model-Based Control

By combining the Simulation Interface VIs with DAQ I/O, you can implement real-time control systems or hardware-in-the-loop simulation. By using the DAQ functions in LabVIEW RT applications, the VIs execute simulation models in real time with real I/O. The following example VI, *ModelName*_daq_driver VI, is an example block diagram of this type of VI.

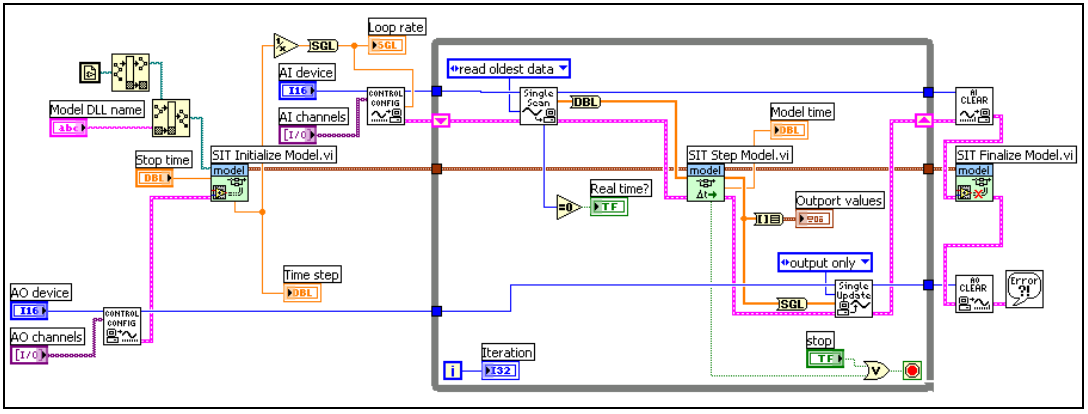


**Figure 3.** *ModelName*_daq_driver VI

The *ModelName*_daq_driver VI uses a continuous single-point analog input operation to control the timing of the VI. The analog output is a continuous single-point operation. The scan clock of the analog input controls the time interval between scans of the analog output. Notice that the number of analog input channels must equal the number of inports in the simulation model, and the number of analog output channels must equal the number of outports.

Refer to the DAQ examples located in `\labview\examples\daq\solution\control.llb` and the LabVIEW RT examples located in `\labview\examples\rt\RT Control.llb` for more examples of real-time control using DAQ functions.