Getting Started with the MapleSim Connector for LabVIEW and NI VeriStand Software

Getting Started with the MapleSim Connector for LabVIEW and NI VeriStand Software

This document was produced using a special version of Maple and DocBook.

Printed in Canada

Contents

Introduction

The MapleSim™ Connector for LabVIEW® and NI VeriStand™ Software provides all of the tools you need to prepare and export your dynamic systems models to National Instruments™ (NI) LabVIEW as External Model Interface (EMI) or Simulation Interface Toolkit (SIT) blocks, or as models for NI VeriStand™. You can create a model in MapleSim, simplify it in Maple™ by using an extensive range of analytical tools, and then generate virtual instruments (VIs) that you can incorporate into your LabVIEW or NI VeriStand toolchain.

You can also use these tools for exporting mathematical models that you have created from first principles in Maple as VIs.

Furthermore, various options allow you to use the C code generation feature in Maple to create code libraries of your MapleSim models for implementation in other applications.

Features include:

- Maple templates, which provide an intuitive user interface for optimizing your MapleSim model, and then generate a dynamic-link library (.dll) file for LabVIEW or NI VeriStand.
- A range of examples illustrating how to prepare and export your models.
- Commands for developing VIs of mathematical models from first principles in the Maple environment and examples to illustrate how to do it.
- Access to commands in the **LabVIEWConnector** package in Maple for developing dynamic-link library (.dll) files for LabVIEW or NI VeriStand.

Scope of Model Support

MapleSim is a comprehensive modeling tool where it is possible to create models that could go beyond the scope of this MapleSim Connector for LabVIEW and NI VeriStand Software release. In general, the MapleSim Connector for LabVIEW and NI VeriStand Software supports systems of any complexity, including systems of DAEs of any index, in any mix of domains, as long as they exhibit continuous behavior. Systems that contain any type of discontinuity, including discrete transforms, switches, logic gates, relational and Boolean operations are not supported by the current release of this product.

Apart from all of the engineering and signal components that are continuous, this product also supports lookup tables, and custom components that do not use discontinuous operations such as piecewise functions.

System Requirements

For installation instructions and a complete list of system requirements, see the **Install.html** file on the product disc.

1 Getting Started

1.1 Getting Help

In Maple, enter **?LabVIEWConnector** at a prompt in a worksheet.

1.2 Using the LabVIEW Component Block Generation Template

The MapleSim Connector provides **LabVIEW Component Block Generation** templates in the form of Maple worksheets for manipulating and exporting MapleSim subsystems. These templates contain pre-built embedded components that allow you to generate EMI Components, SIT Components, or C code from a MapleSim subsystem, export the subsystem as a LabVIEW block, and save the source code.

Using these templates, you can define inputs and outputs for the system, set the level of code optimization, chose the format of the resulting EMI Component, and generate the source code, library code, block script, or LabVIEW block. You can use any Maple commands to perform task analysis, assign model equations to a variable, group inputs and outputs to a single vector and define additional input and output ports for variables.

Note: Code generation now handles all systems modeled in MapleSim, including hybrid systems with defined signal input (RealInput) and signal output (RealOutput) ports.

Block generation for EMI or SIT consists of the following steps:

- Subsystem preparation
- Subsystem selection
- Port and parameter management
- EMI or SIT component options
- Generate EMI or SIT component code
- View generated EMI or SIT component code

Subsystem Preparation

Convert your model or part of your model into a subsystem. This identifies the set of modeling components that you want to export as a block component. Since LabVIEW only supports data signals, properties on acausal connectors such as mechanical flanges and electrical pins, must be converted to signals using the appropriate ports.

To connect a subsystem to modeling components outside of its boundary, you add subsystem ports. A subsystem port is an extension of a component port in your subsystem. The resulting signals can then be directed as inputs and outputs for the LabVIEW Component Block Generation templates.

Note: For connectors you must use signal components since acausal connectors can not be converted to a signal.

By creating a subsystem you not only improve the visual layout of a system in model workspace and but also prepare the model for export. The example in Chapter 2 shows you how to group all of the components into a subsystem.

Subsystem Selection

You can select which subsystems from your model you want to export to a LabVIEW block. After a subsystem is selected, click Load Selected Subsystem. All defined input and output ports are loaded.

Input Ports:

	Input Variables	Port Grouping Name	Change Row
1	`Main.'Differentiator::InputSignal'`(t)	"InputSignal"	

Output Ports:

	Output Variables	Port Grouping Name	Change Row
1	`Main.'Differentiator::VInput'`(t)	"VInput"	
2	`Main.'Differentiator::VOutput'`(t)	"VOutput"	

Port and Parameter Management

Port and Parameter Management lets you customize, define and assign parameter values to specific ports. Subsystem components to which you assign the parameter inherit a parameter value defined at the subsystem level. After the subsystem is loaded you can group individual input and output variable elements into a vector array, and add additional input and output ports for customized parameter values. Input ports can include variable derivatives, and output ports can include subsystem state variables.

Note: If the parameters are not marked for export they will be numerically substituted.

The following selections specify the input ports, output ports, and states for generating LabVIEW blocks.

EMI or SIT Input Ports:

☐ Add additional inputs for required input variable derivatives

Select **Add additional inputs for required input variable derivatives** to specify calculated derivative values instead of numerical approximations.

EMI or SIT Output Ports:

☐ Add an additional output port for subsystem state variables

Select **Add an additional output port for subsystem state variables** to add extra output ports for the state variables.

EMI Component Options

The EMI Component Options settings specify the advanced options for the code generation process.

Optimization Options

Set the level of code optimization to specify whether equations are left in their implicit form or converted to an ordinary differential equation (ODE) system during the code generation process. This option specifies the degree of simplification applied to the model equations during the code generation process and eliminates redundant variables and equations in the system.

Level of code optimization (0=None, 3=Full): 0 1 2 3

Select one of the following options:

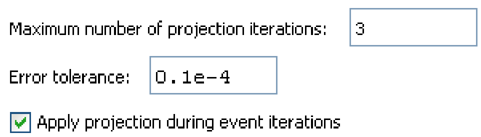
None (0): no optimization is performed; the default equations will be used in the generated code.

Partial (1, 2): removes redundant equations from the system.

Full (3): performs index reduction to reduce the system to an ODE system or a differential algebraic equation (DAE) system of index 1, and removes redundant equations.

Constraint Handling Options

The **Constraint Handling Options** specifies whether the constraints are satisfied in a DAE system by using constraint projection in the generated LabVIEW block. Use this option to improve the accuracy of a DAE system that has constraints. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate.



Set the **Maximum number of projection iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

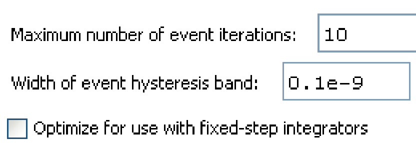
Set the **Error tolerance to** specify the desirable error tolerance to achieve after the projection.

Select **Apply projection during event iterations** to interpolate iterations to obtain a more accurate solution.

Constraint projection is performed using the **constraint projection** routine in the External Model Interface as described on The MathWorks™ web site to control the drift in the result of the DAE system.

Event Handling Options

The **Event Handling Options** specifies whether the events are satisfied in a DAE system by using event projection in the generated LabVIEW block. Use this option to improve the accuracy of a DAE system with events. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate.



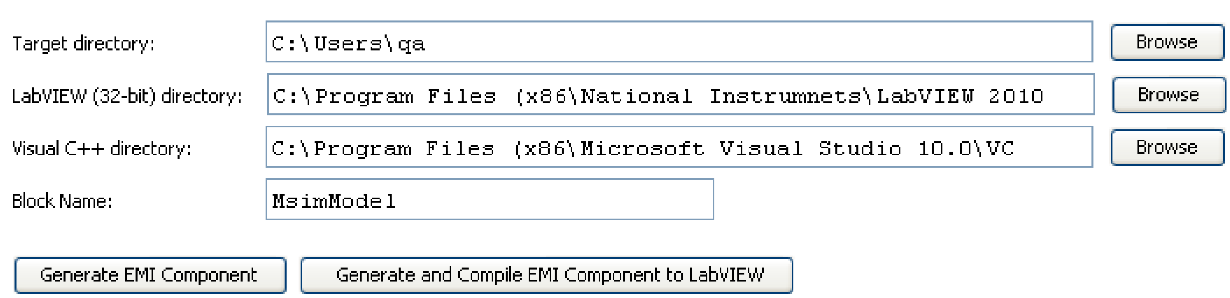
Set the **Maximum number of event iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

Set the **Width of event hysterias band** to specify the desirable error tolerance to achieve after the projection.

Select **Optimize for use with fixed-step integrators** to optimize the event iterations as a function of hysterias bandwidth.

Event projection is performed using the **event projection** routine in the External Model Interface as described on The MathWorks™ web site to control the drift in the result of the DAE system.

Generate EMI Component Code



Provide a block name, LabVIEW and Visual C++ directories and specify the location for the generated EMI file.

To generate EMI Component code without a LabVIEW connection, click **Generate EMI Component**.

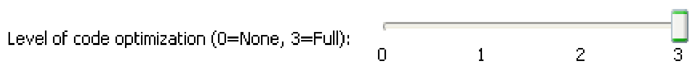
To generate EMI Component code, click **Generate and Compile EMI Component to LabVIEW**.

SIT Component Options

These settings specify the advanced options for the code generation process.

Optimization Options

Set the level of code optimization to specify whether equations are left in their implicit form or converted to an ordinary differential equation (ODE) system during the code generation process. This option specifies the degree of simplification applied to the model equations during the code generation process and eliminates redundant variables and equations in the system.



Select one of the following options:

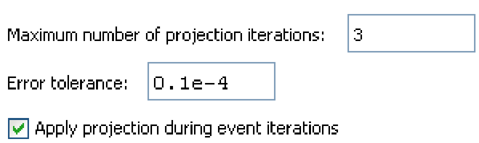
None (0): no optimization is performed; the default equations will be used in the generated code.

Partial (1, 2): removes redundant equations from the system.

Full (3): performs index reduction to reduce the system to an ODE system or a differential algebraic equation (DAE) system of index 1, and removes redundant equations.

Constraint Handling Options

The **Constraint Handling Options** specifies whether the constraints are satisfied in a DAE system by using constraint projection in the generated LabVIEW block. Use this option to improve the accuracy of a DAE system that has constraints. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate.



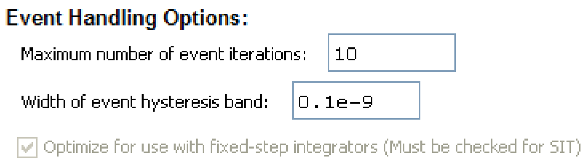
Set the **Maximum number of projection iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

Set the **Error tolerance to** specify the desirable error tolerance to achieve after the projection.

Select **Apply projection during event iterations** to interpolate iterations to obtain a more accurate solution. Constraint projection is performed using the **constraint projection** routine in the External Model Interface as described on The MathWorks™ web site to control the drift in the result of the DAE system.

Event Handling Options

The **Event Handling Options** specifies whether the events are satisfied in a DAE system by using event projection in the generated LabVIEW block. Use this option to improve the accuracy of a DAE system with events. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate.



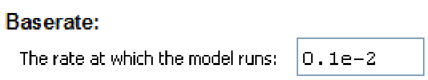
Set the **Maximum number of event iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

Set the **Width of event hysterias band** to specify the desirable error tolerance to achieve after the projection.

Event projection is performed using the **event projection** routine in the External Model Interface as described on The MathWorks™ web site to control the drift in the result of the DAE system.

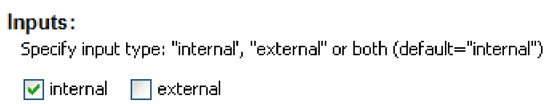
Baserate

The **Baserate** specifies the rate at which the model runs. Use this option to improve the accuracy of a DAE system with events. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate.

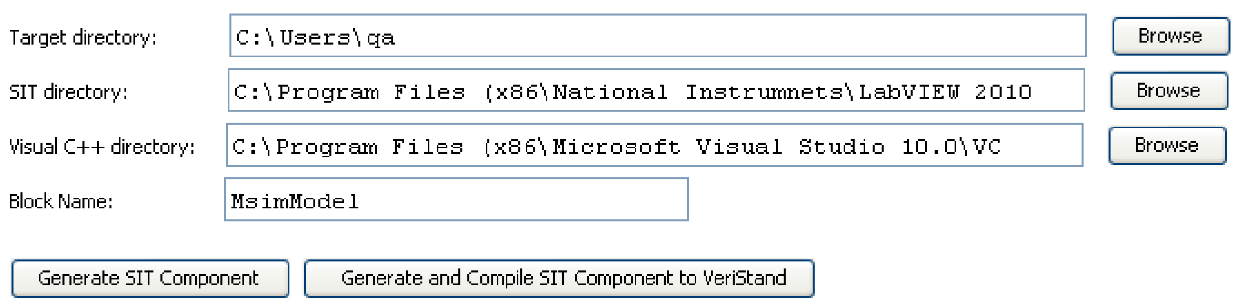


Inputs

Specify the input type; internal, external or both.



Generate SIT Component Code



Provide a block name, SIT and Visual C++ directories and specify the location for the generated SIT file.

To generate SIT Component code without a VeriStand connection, click **Generate SIT Component**.

To generate and compile SIT Component code into VeriStand, click **Generate and Compile SIT Component to VeriStand**.

View EMI or SIT Component Code

After you generate the EMI Component code and create the block, a LabView command window opens and the block with any of the following specified parameters is generated in LabVIEW:

- Header File

- C Code

1.3 Using the LabVIEW Block Generation Templates

The MapleSim Connector for LabVIEW and NI VeriStand Software provides an **NI LabVIEW EMI Block Generation** template and an **NI VeriStand and LabVIEW SIT Component Model Generation** template in the form of Maple worksheets for manipulating and exporting MapleSim subsystems. These templates contain pre-built embedded components that allow you to generate LabVIEW blocks from a MapleSim subsystem, export the subsystem as a LabVIEW block and Microsoft® Visual Studio® project, and save the source code.

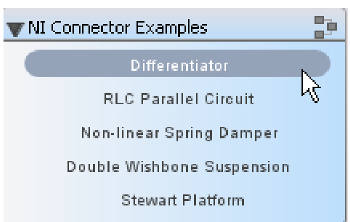
Using either of these templates, you can define inputs and outputs for the system, generate the source code and library code.

Example models are available in the **NI Connector Examples** palette in MapleSim.

Viewing Examples

To view an example:

1. In the **Libraries** tab on the left side of the MapleSim window, expand the **NI Connector Examples** palette, and click the entry for the model that you want to view.



Some models include additional documents, such as templates that display model equations or define custom components.

2. In the **Project** tab, expand the **Attachments** palette and then expand **Documents**. You can open any of these documents by right-clicking its entry in the list and clicking **View**.

After you add a template to a model, it will be available from this list.

1.4 Example: RLC Circuit Model

In this example, you will generate a LabVIEW EMI or SIT block, or a block for NI VeriStand using an RLC circuit model that was created in MapleSim.

To generate a LabVIEW block:

1. From the **NI Connector Examples** palette, open the **RLC Parallel Circuit** example.

2. Click the templates button () in the main toolbar.

3. From the list, select **NI LabVIEW EMI Component Block Generation** to generate a LabVIEW EMI block or **NI VeriStand and LabVIEW SIT Component Model Generation** to generate a block for the LabVIEW Simulation Interface Toolkit or NI VeriStand.

4. In the **Attachment** field, enter **RLC Circuit** as the worksheet name.

5. Click **Create Attachment**. Your MapleSim model is opened in Maple, in the template that you selected.

6. Browse to the **RLC Parallel Circuit 1** subsystem by selecting the subsystem name from the drop-down menu in the toolbar above the model diagram. This menu displays all of the subsystems and components in your MapleSim model.

7. In the **Model Input from MapleSim** section of the template, click **Load Selected Subsystem**. All of the template fields are populated with information specific to the subsystem displayed in the model diagram. You can now specify which subsystem parameters will be kept as configurable parameters in the generated block.

8. In the **Setting Parameters** section, select the **Main.RLC Parallel Circuit1::R** parameter entry in the **Parameter Name** list.

Note: The **Keep as Block Parameter** box is checked by default. Also by default, all parameters in the model are kept as configurable parameters.

9. To change the default value of the resistance parameter, select **Main.RLC Parallel Circuit1::R** from the **Parameter Name** list, enter **5** in the **Value** field, and click the **Update** button.

10. In the **Advanced Code Generation Settings** section, set the **Code Optimization** option to **Full**. This option specifies the degree of simplification applied to the model equations during the code generation process. This option eliminates redundant variables and equations in the system.

If you plan to generate a LabVIEW EMI block, follow the steps in the **Generating a LabVIEW EMI Block** section below. If you plan to generate a block for NI VeriStand or the LabVIEW Simulation Interface Toolkit, follow the steps in the **Generating a LabVIEW Block for NI VeriStand or SIT** section below.

Generating a LabVIEW EMI Block

1. In the **Generate the Shared Library Code** section of the template, specify the LabVIEW and Visual C++ directories.

2. Click **Generate to LabVIEW** to generate the Visual Studio project and dynamic-link library (.dll) file for the EMI block.

3. In LabVIEW, open a new VI and open the block diagram window by selecting **Windows>Show Block Diagram**.

4. Right-click the canvas and select **Control Design & Simulation>Simulations>Control and Simulation Loop**. Click the canvas and draw a simulation loop box.

5. Right-click the simulation loop box and select **Control Design & Simulation>Simulations>Utilities>External Model.** Click a point in the box to position the model.

6. In the **Select an External Model Library** window, browse to the **Release** subfolder located in the default directory that you specified in the **LabVIEW EMI Block Generation** template and open the .dll file that you generated.

7. Click **OK**.

Note: Generating a block may require a few minutes.

You can now use the RLC circuit block in a LabVIEW EMI diagram. To view a complete example that describes how to prepare and export a slider-crank model as a LabVIEW EMI block, see *Example: Exporting a Model as a LabVIEW EMI Block (page 9)*.

Generating a LabVIEW Block for NI VeriStand or the LabVIEW SIT

1. In the **Generate the Shared Library Code** section of the template, specify the LabVIEW SIT and Visual C++ directories.

2. In the **SIT version** field, specify a value of **2009** or **2010**.

3. Click the **Generate Dll** button

Note: Generating a block may require a few minutes.

For more information about preparing your block for either the NI VeriStand or LabVIEW SIT environment, see *Working with Your Block in NI VeriStand or LabVIEW SIT (page 17)*.

2 Example: Exporting a Model as a LabVIEW EMI Block

2.1 Preparing a Model for Export

In this example, you will perform the steps required to prepare a slider-crank mechanism model and export it as a LabVIEW EMI block.

1. Convert the slider-crank mechanism model to a subsystem.

2. Define subsystem inputs and outputs.

3. Define and assign subsystem parameters.

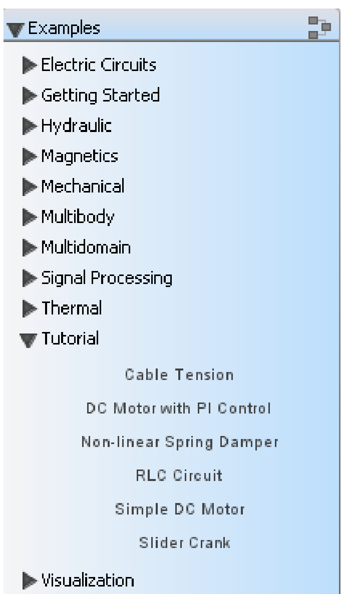
4. Export the model using the LabVIEW EMI Block Generation template.

5. Implement the EMI block in LabVIEW.

To open the slider-crank mechanism example:

1. In MapleSim, expand the **Examples** palette and then expand the **Tutorial** submenu.

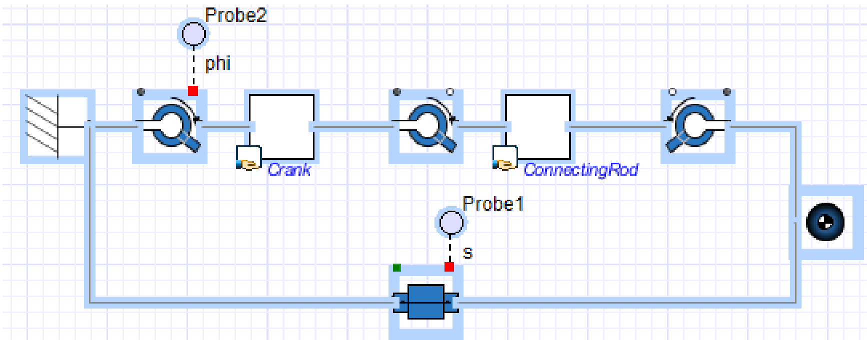
2. Open the **Slider Crank** example.



Converting the Model to a Subsystem

By converting your entire model or part of your model into a subsystem, you identify which parts of the model that you want to export. In this example, you will group all of the components into a subsystem.

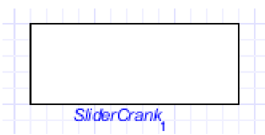
1. Using the selection tool () located above the model workspace, draw a box around all of the components in the model.



2. From the **Edit** menu, select **Create Subsystem**.

3. In the **Create Subsystem** dialog box, enter **SliderCrank** as the subsystem name.

4. Click **OK**. A **SliderCrank** subsystem block appears in the model workspace.



Defining Subsystem Inputs and Outputs

MapleSim uses a topological representation to connect interrelated components without having to consider how signals flow between them, whereas traditional signal-flow modeling tools require explicitly defined system inputs and outputs. Since LabVIEW only supports data signals, properties on acausal ports, such as mechanical flanges and electrical pins, must be converted to signals using the appropriate components. The resulting signals are directed as inputs and outputs for the subsystem in MapleSim and for the EMI block.

Note: Currently, code generation is limited to subsystems with defined signal input (*RealInput*) and signal output (*RealOutput*) ports.

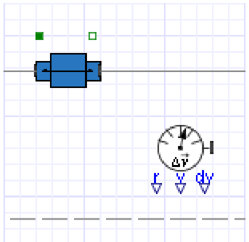
In this example, you will convert the displacements of the slider and the joint between the crank and connecting rod to output signals. The input signal needs to be converted to a torque that is applied to the revolute joint that represents the crank shaft.

1. Double-click the subsystem block to view its contents. The broken line surrounding the components indicates the subsystem boundary, which can be resized by clicking and dragging its sizing handles.

2. Delete the probes that are attached to the model.

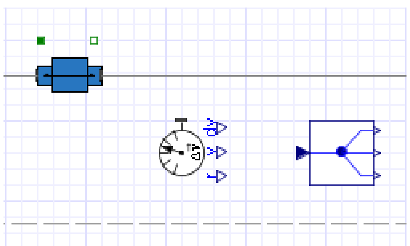
3. In the **Libraries** tab on the left side of the MapleSim window, expand the **Multibody** palette and then expand the **Sensors** submenu.

4. Drag the **Absolute Translation** component to the model workspace and place it below the **Prismatic Joint** component.

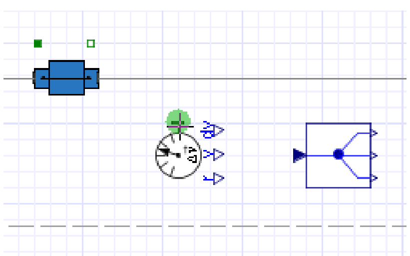


5. Right-click (**Control**-click for Macintosh®) the **Absolute Translation** component and select **Rotate Counterclockwise**.

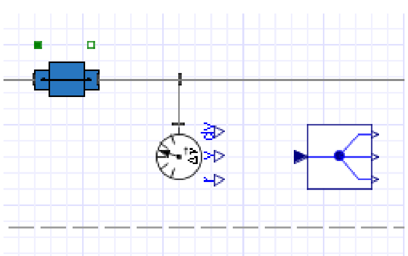
6. From the **Signal Blocks** → **Routing** → **Demultiplexers** menu, drag a **3-port Demultiplexer** component to the model workspace and place it to the right of the **Absolute Translation** component.



7. To connect the **Absolute Translation** component to the model, click the frame_b connector. The frame is highlighted in green when you hover your pointer over it.

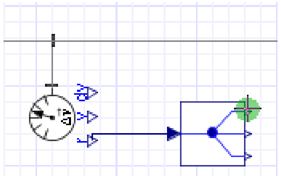


8. Draw a vertical line and click the connection line directly above the component. The sensor is connected to the rest of the diagram.

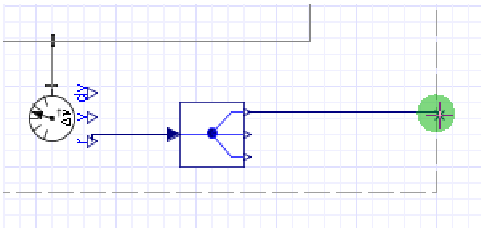


9. In the same way, connect the **r** output port (*TMOutputP*) of the **Absolute Translation** component to the navy blue input port of the demultiplexer. This is the displacement signal from the sensor in x, y, and z coordinates. Since the slider only moves along the x axis, the first coordinate must be an output signal.

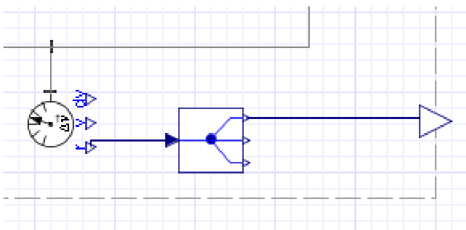
10. Hover your pointer over the first demultiplexer port and click your mouse button once.



11. Drag your pointer to the subsystem boundary.



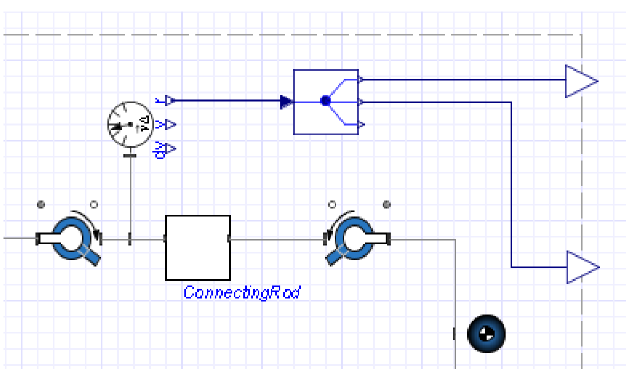
12. Click the boundary once. A real output port is added to your subsystem.



13. Add another **Absolute Translation** component above the **Connecting Rod** subsystem.

14. Right-click (**Control**-click for Macintosh) the **Absolute Translation** component and select **Flip Vertical.** Right-click the **Absolute Translation** component again and select **Rotate Clockwise**.

15. Add a **3-port Demultiplexer** component to the right of the sensor and connect the components as shown below.

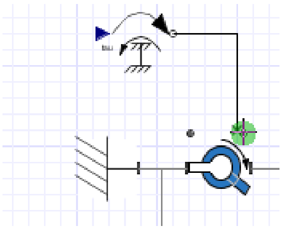


Since the crank is moving in the x, y plane, you only need to output the first two signals.

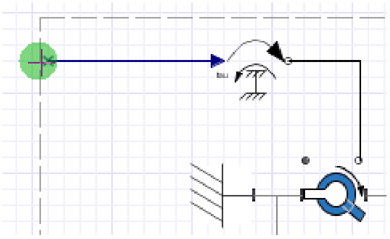
You will now add a real input port to your subsystem to control the torque on the crank shaft.

16. From the **1-D Mechanical → Rotational → Torque Drivers** menu, add a **Torque** component to the model workspace and place it above the **Fixed Frame** component.

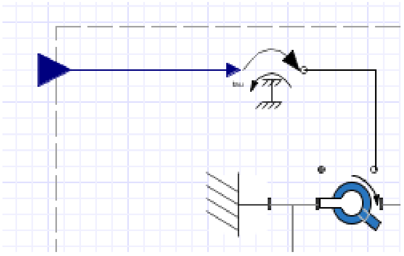
17. Connect the white flange of the **Torque** component to the white flange of the leftmost **Revolute Joint**.



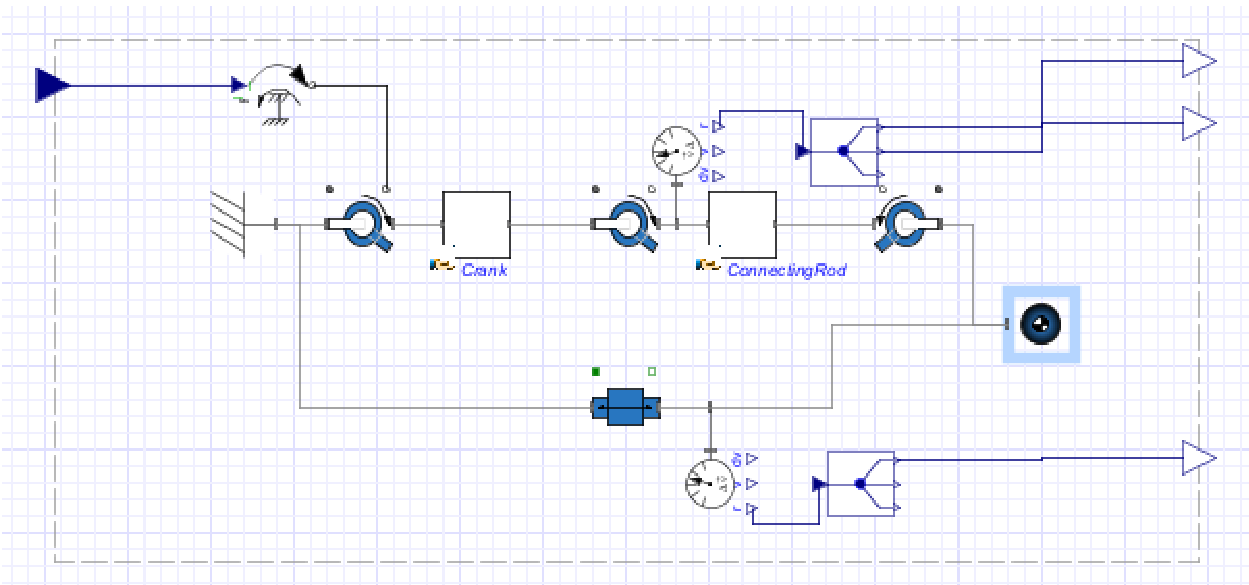
18. Click the input port of the **Torque** component and drag your pointer to the subsystem boundary.



19. Click the boundary once. A real input port is added to your subsystem.

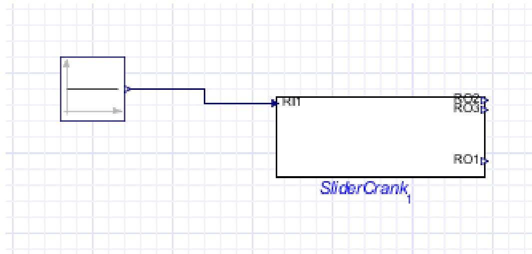


The complete subsystem appears below.



20. Click **Main** above the model workspace to browse to the top level of the model.

21. From the **Signal Blocks** → **Sources** → **Real** menu, drag a **Constant** source into the model workspace and connect its output port to the input port of the **SliderCrank** subsystem as shown below.

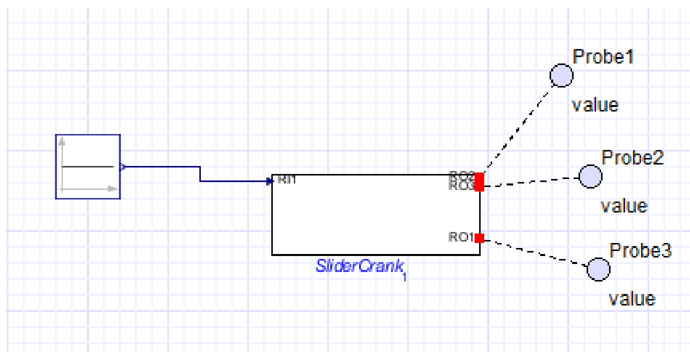


22. Click the probe button ( ) above the model workspace.

23. Click the top output port of the **SliderCrank** subsystem.

24. In the model workspace, click the probe once to position it.

25. In the same way, add probes to the other **SliderCrank** output ports as shown below.



2.2 Defining and Assigning Subsystem Parameters

You can define custom parameters that can be used in expressions in your model to edit values more easily. To do so, you define a parameter with a numeric value in the parameter editor. You can then assign that parameter as a variable to the parameters of other components; those individual components will then inherit the numeric value of the parameter defined in the parameter editor. By using this approach, you only need to change the value in the parameter editor to change the parameter values for multiple components.

1. While in the detailed view of the **SliderCrank** subsystem, click the parameter editor button ( ) above the model workspace. The parameter editor appears.

2. In the **New Parameter** field, define a parameter called **CrankL** and press **Enter**.

3. Specify a default value of **1** and enter **Length of the crank** as the description.

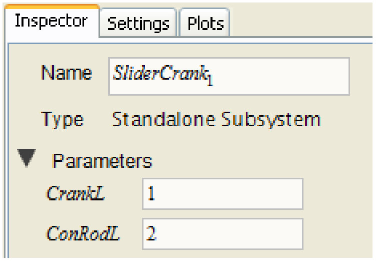
4. In the second row of the table, define a parameter called **ConRodL** and press **Enter**.

5. Specify a default value of **2** and enter **Length of the connecting rod** as the description.

SliderCrank subsystem default settings

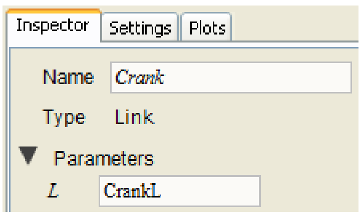
Name	Type	Default Value	Default Units	Description
CrankL	Real	1		Length of the crank
ConRodL	Real	2		Length of the connecting rod

6. Click to switch to the diagram view. The parameters are defined in the **Parameters** pane.



7. In the model workspace, select the **Crank** subsystem.

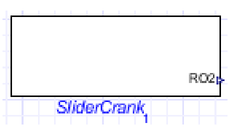
8. In the **Parameters** pane, change the length value (**L**) to **CrankL**.



The **Crank** subsystem now inherits the numeric value of **CrankL** that you defined.

9. Select the **ConnectingRod** subsystem and change its length value to **ConRodL**.

10. Click the **Main** button above the model workspace to navigate to the top level of the model.



You will include these parameter values in the model that you export. You are now ready to convert your model to an EMI block.

2.3 Exporting Your Model Using the LabVIEW EMI Block Generation Template

After preparing the model, you can use the LabVIEW EMI Block Generation template to set export options and convert the model to a LabVIEW EMI block.

1. Click the templates button ( ) in the main toolbar.

2. From the list, select **NI LabVIEW EMI Component Block Generation**.

3. In the **Attachment** field, enter **Slider Crank EMI** as the worksheet name and click **Create Attachment**. The slider-crank subsystem is opened in the **LabVIEW EMI Block Generation Template** in Maple.

4. Use the navigation controls above the model diagram to select the **SliderCrank** subsystem and click **Retrieve System**. All of the template fields are populated with information specific to the subsystem.

5. Click **Generate to LabVIEW** to generate the block.

6. Set the LabVIEW and Visual C++ directory paths.

7. At the bottom of the template, click **Generate to LabVIEW** to generate the Visual Studio project and dynamic-link library (.dll) file for the EMI block.

8. In LabVIEW, open a new VI and open the block diagram window by selecting **Windows>Show Block Diagram**.

9. Right-click the drawing canvas and select **Control Design & Simulation>Simulations>Control and Simulation Loop**. Click the canvas and draw a simulation loop box.

10. Right-click the simulation loop box and select **Control Design &Simulation>Simulations>Utilities>External Model.** Click the simulation loop box to position the model.

11. In the **Select an External Model Library** window, browse to the **Release** subfolder located in the default directory that you specified in the **LabVIEW EMI Block Generation** template and open the .dll file that you generated.

12. Click **OK**.

13. Connect the output of the block to a scope and the input to a sine wave.

3 Working with Your Block in NI VeriStand or LabVIEW SIT

Using the RLC circuit block that you generated in *Example: RLC Circuit Model (page 6)*, this chapter describes how to work with your block in NI VeriStand or the LabVIEW SIT environment.

- *Preparing Your MapleSim Model to Run in NI VeriStand (page 17)*
- *Importing a MapleSim Model to the LabVIEW SIT Environment (page 25)*

3.1 Preparing Your MapleSim Model to Run in NI VeriStand

Creating a New Project File

1. Open NI VeriStand.

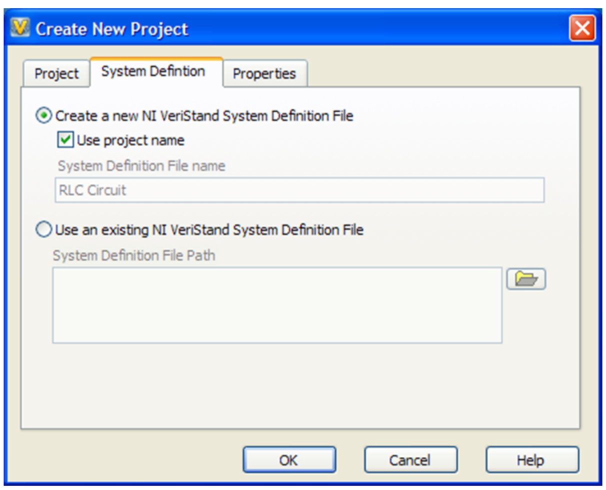
2. From the **File** menu, select **New Project**.

3. In the **New Project Name** field, enter **RLC Circuit**.

4. Click **OK**.

5 In the **Create New Project** window, select the **System Definition** tab.

6. Select the **Create a new NI VeriStand System Definition file** radio button.



7. Click **OK**. A new project file is created, along with a system definition file.

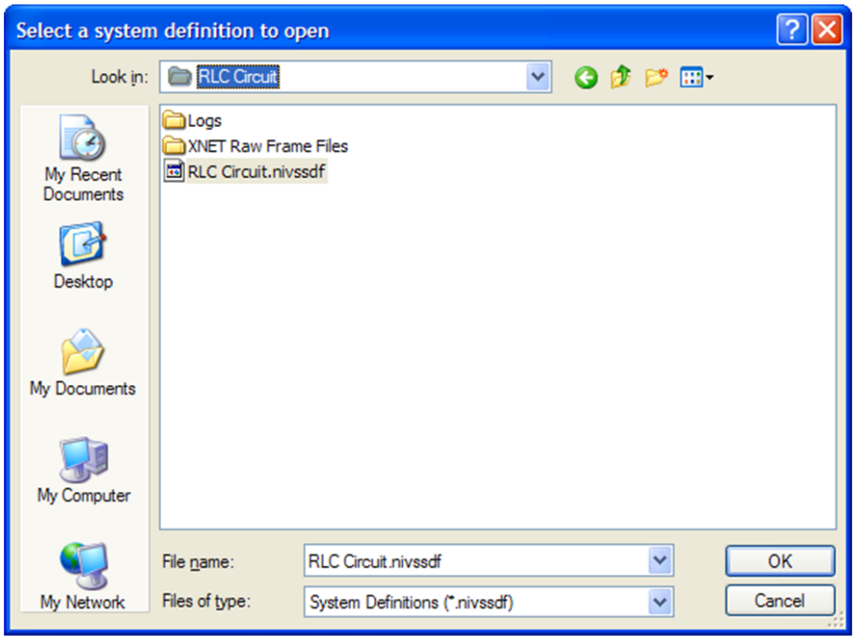
Adding the MapleSim Model to the System Definition File

1. From **Start >All Programs > National Instruments > NI VeriStand 2010**, select **System Explorer**.

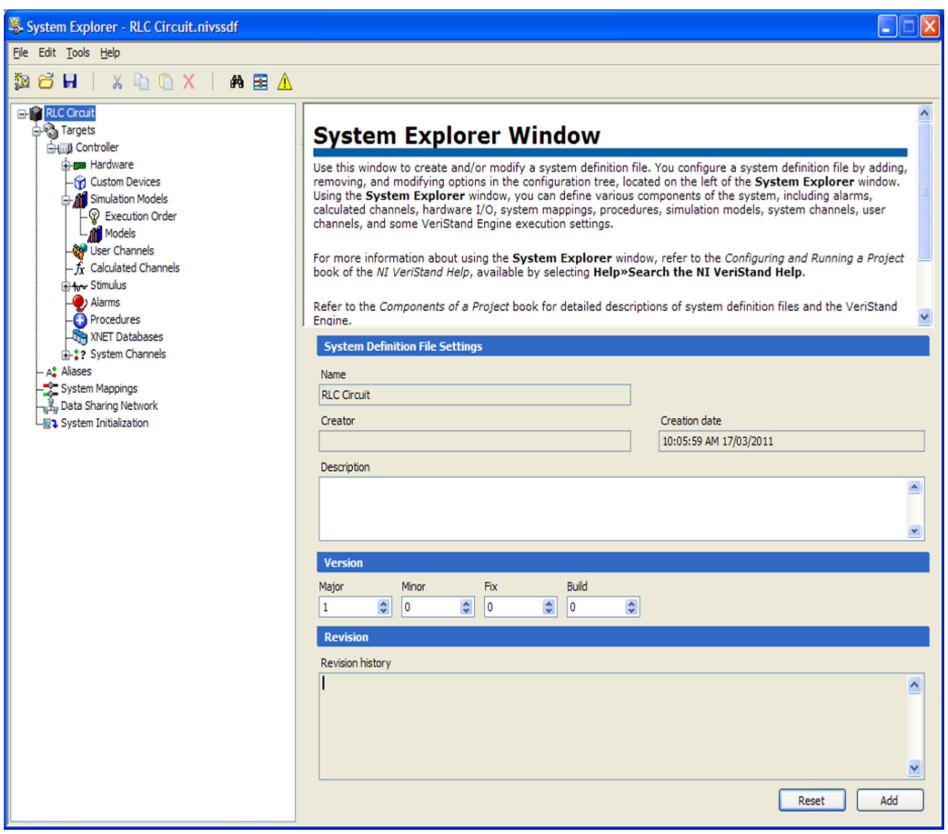
2. In the **System Explorer** window select **File**, then **Open**,

3. Navigate to the project you created in the previous section, *Preparing Your MapleSim Model to Run in NI VeriStand (page 17)*.

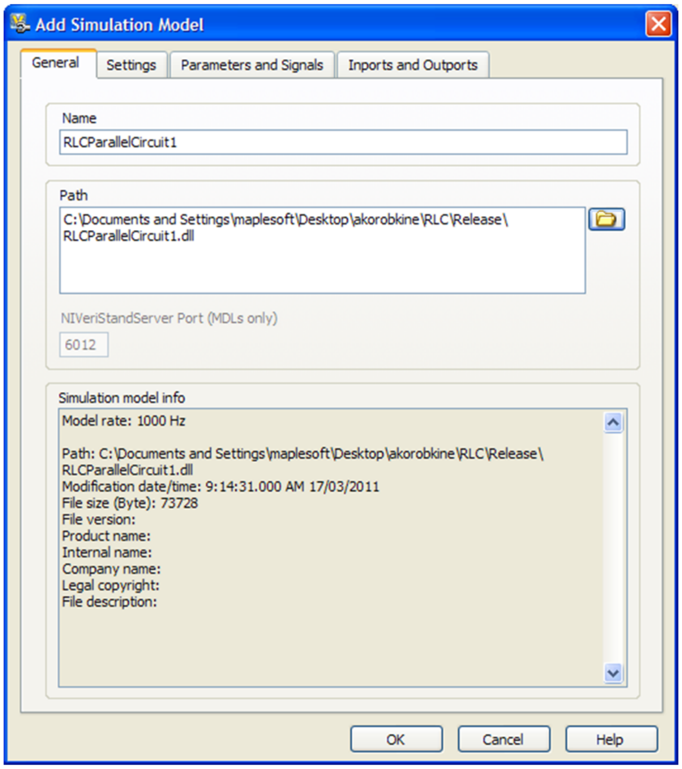
4. Open **RLC Circuit.nivssdf**.



5. In the left pane, expand **Controller,** click **Simulation Models** then click **Models**.



6. Click the **Add a Simulation Model** button located above the right pane. The **Add Simulation Model** window is displayed.



7. In the **General** tab, click the browse button () and open the .dll that you created in *Example: RLC Circuit Model (page 6)*.

8. In the **Parameters and Signals** tab, select **Import all Signals**.

9. In the **Inports and Outports** tab, select **Segment into scalar channels**.

10. Click **OK**.

11. Save your changes.

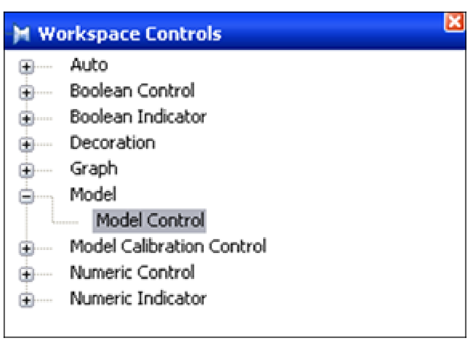
Running the Project

1. In the **NI VeriStand - Getting Started** window, click the **Run Project** button.

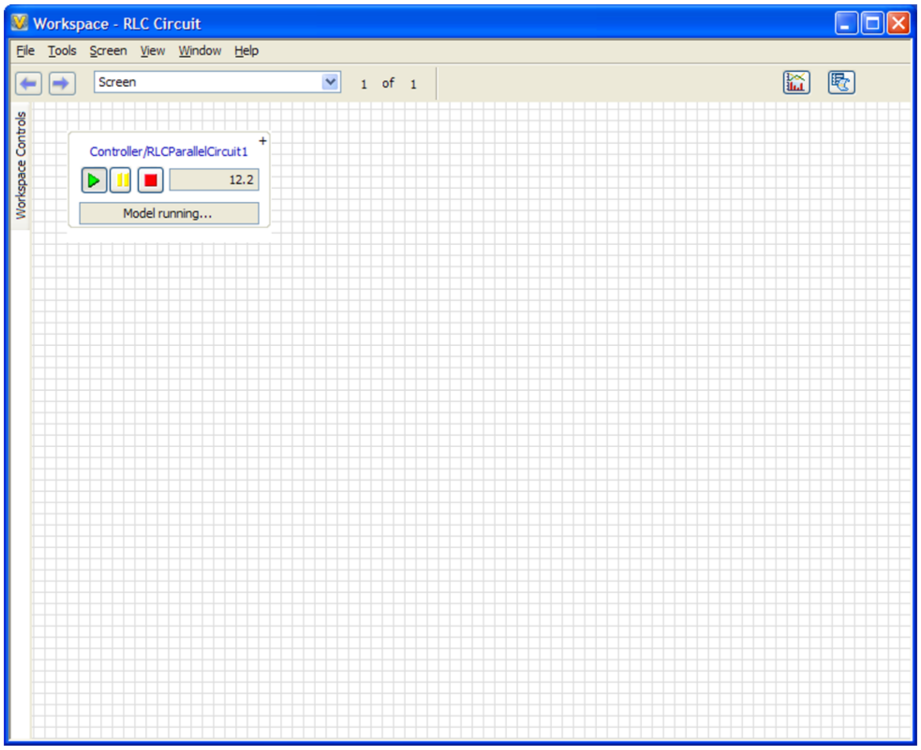
2. In the blank workspace that you opened, from the **Screen** menu, select **Edit Mode**.

3. Click the **Workspace Controller** tab on the left side of the workspace.

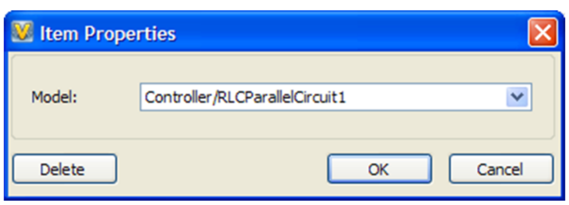
4. In the **Workspace Controls** menu, expand **Model**.



5. From the **Workspace Controls** pane, drag the **Model Control** label into the workspace to add a model control component.



6. From the **Item Properties** window, select **RLC**.

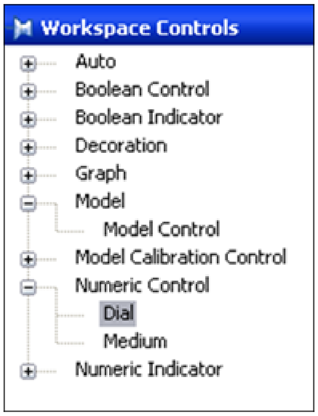


7. Click **OK**.

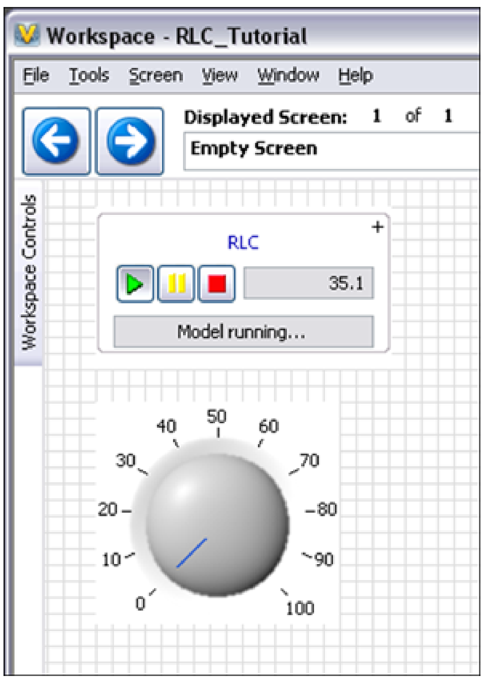
Adding a Dial to the Workspace

1. Click the **Workspace Controller** tab.

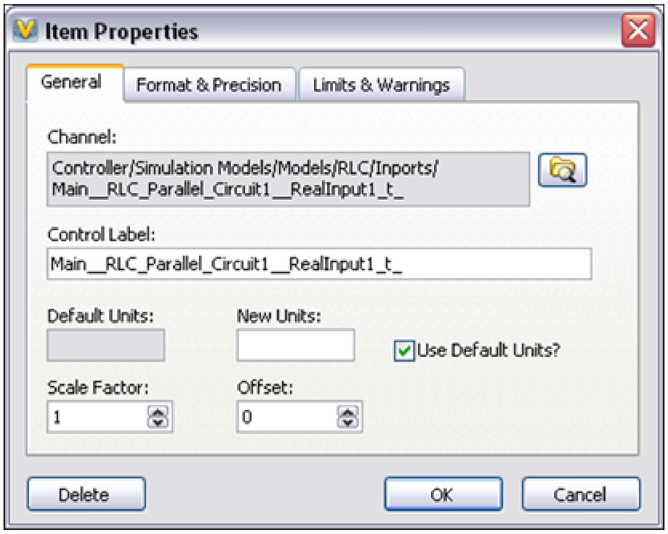
2. In the **Workspace Controls** menu, expand **Numeric Control.**



3. Drag the **Dial** component into the workspace.



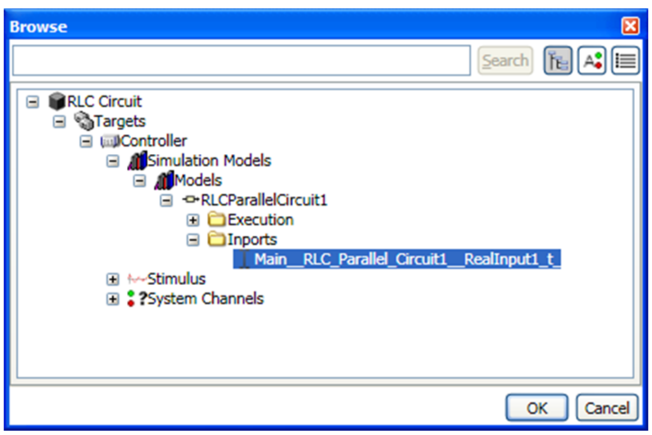
The **Item Properties** window is displayed.



4. In the **Item Properties** window, click the channel button ( ).

5. Expand **Controller>Simulation Models>Models>RLCParallelCircuit1>Inports**.

6. Select **Main_RLC_Parallel_Circuit1__RealInput1_t_**



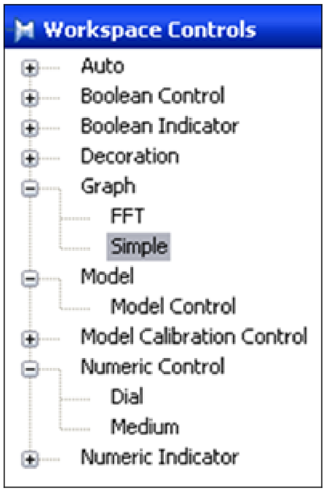
7. Click **OK**.

8. Click **OK**.

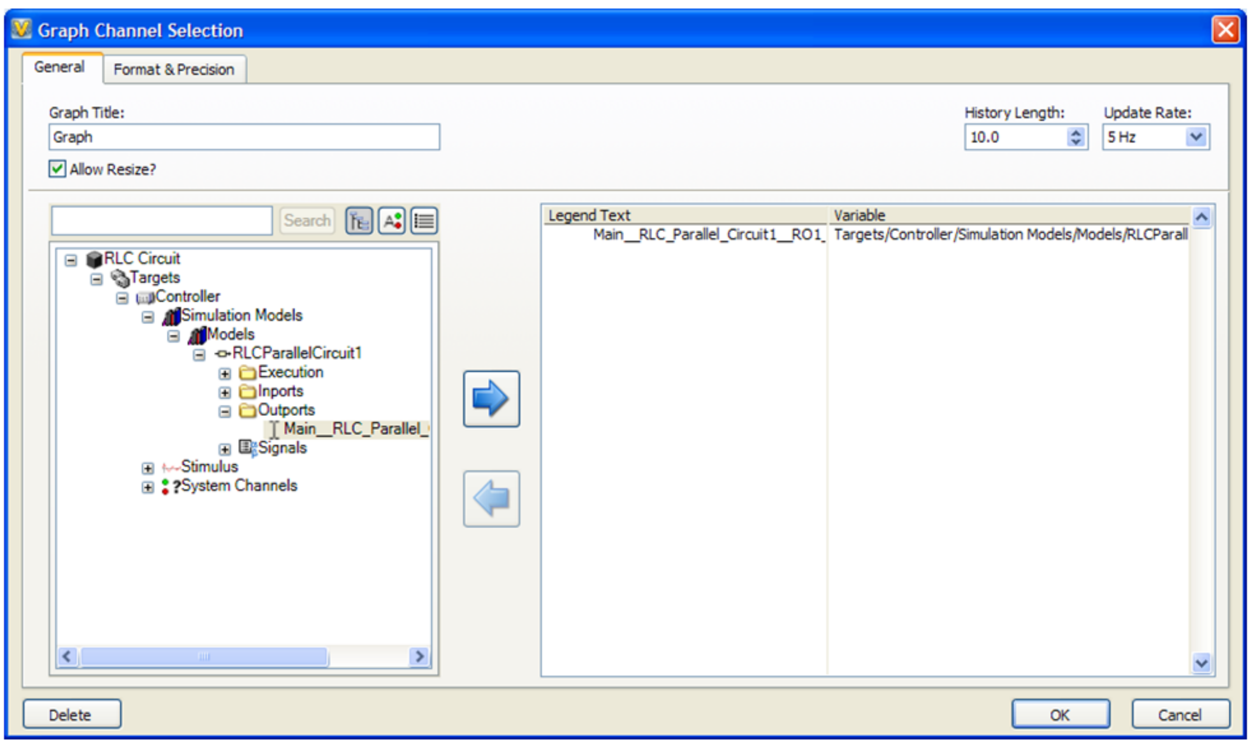
Adding a Graph to the Workspace

1. Click the **Workspace Controller** tab.

2. In the **Workspace Controls** menu, expand **Graph** and drag the **Simple** label into the workspace.



3. In the **Graph Channel Selection** window, in the left pane, expand **Controller > Simulation Models > Models > RLCParallelCircuit1 > Outports**.



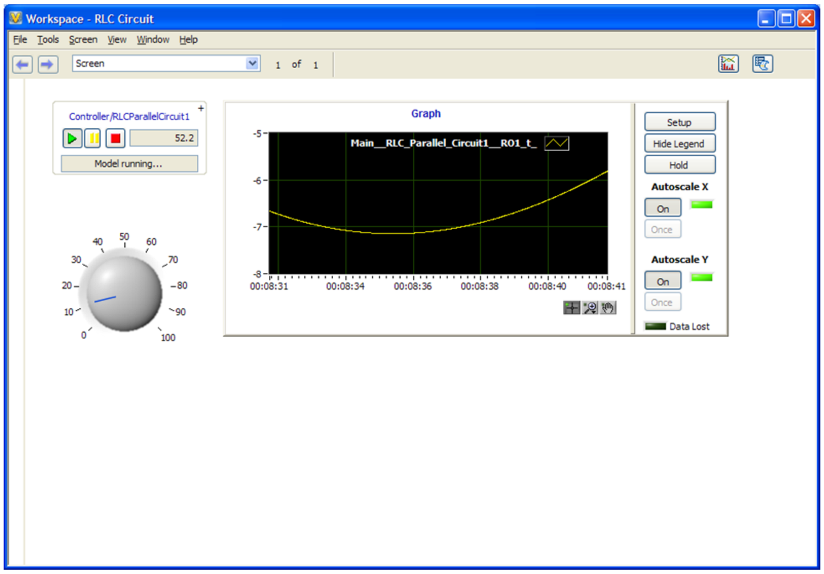
4. Select **Main_RLC_Parallel_Circuit1__RO1_t_**

5. Click the right-pointing arrow () to include the outport quantity in the graph.

6. Click **OK**.

7. From the **Screen** menu, clear the **Edit Mode** option.

8. In the workspace, rotate the dial to change the input behavior. The results are displayed in the graph.

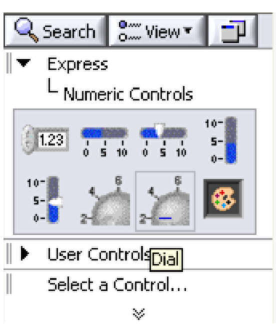


3.2 Importing a MapleSim Model to the LabVIEW SIT Environment

Creating a LabVIEW SIT Interface

1. Open a new VI file.

2. In the Front Panel, right-click to open the **Numeric Controls** panel.



3. Select **Numeric Controls** and then select **Dial**.

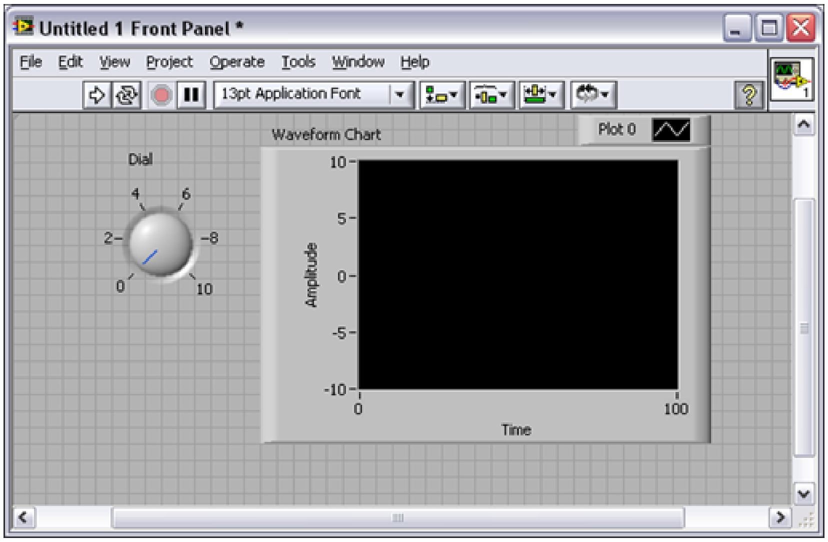
4. Drag the **Dial** component into the Front Panel

5. Right-click the Front Panel

6. Select **Graph Indicators**

7. Select **Chart.**

8. Drag the chart component into the Front Panel

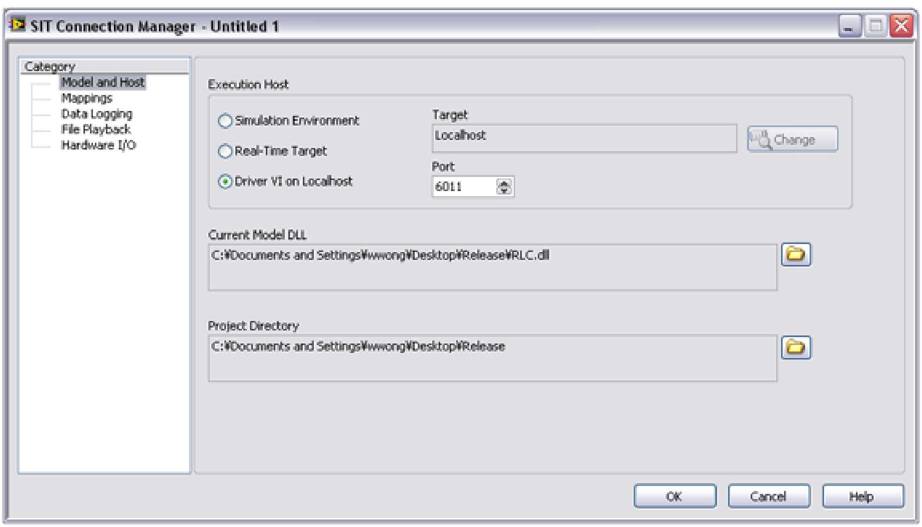


Connecting the MapleSim Model and the LabVIEW SIT User Interface

1. From the **Tools** menu, select **SIT Connection Manager**

2. In the **SIT Connection Manager** window, select **Driver VI on Localhost**.

3. In the **Current Model DLL** section, browse to and select the .dll that you created.



4. In the left pane of the **SIT Connection Manager**, select **Mappings**.

5. In the **Current Mappings** table, double-click the first row which corresponds to the dial that you inserted.

6. Expand **rlcparallelcircuit1>Input_param**

7. Select **Main_RLC_Parallel_Circuit1__RealInput1_t_**

8. Click **OK**.

9. Below **Current Mappings**, double-click **Waveform Chart**.

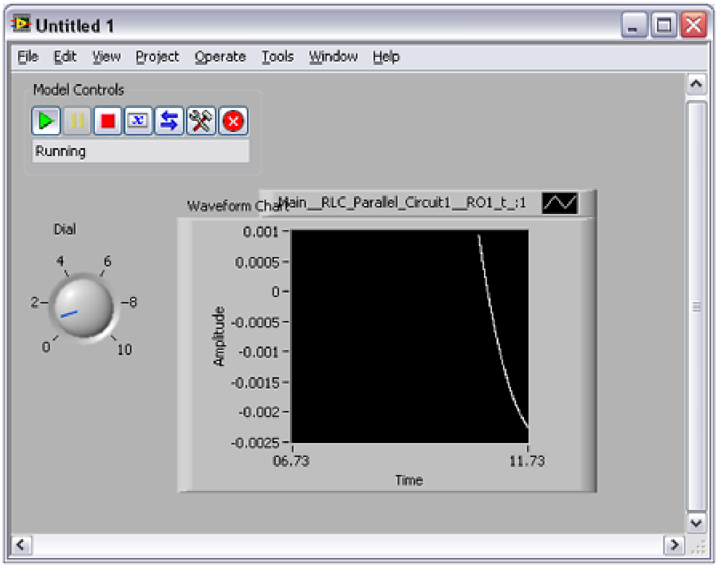
10. Expand **rlcparallelcircuit1>Output>Main_RLC_Parallel_Circuit1__RO1_t_**

11. Select **Port 1 - Main_RLC_Parallel_Circuit1__RO1_t_**

12. Click **OK**.

13. Click **OK**. The **SIT Connection Manager** will now build the model.

14. Click the run button () to run the simulation. When the simulation is complete, you can rotate the knob to change the output.



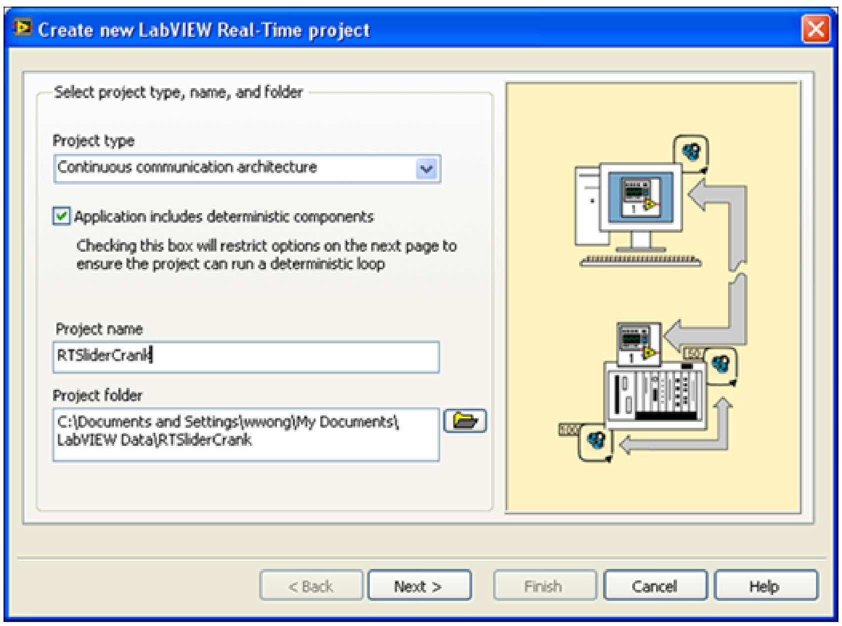
4 Running a Simulation on a LabVIEW Real-Time Target Machine

You can run a simulation on a LabVIEW real-time target machine by using any .dll file that you generate using the MapleSim Connector for LabVIEW and NI VeriStand Software. In this chapter, the steps for running a real-time simulation are demonstrated using the slider-crank .dll file that was generated in *Example: Exporting a Model as a LabVIEW EMI Block (page 9)* in Chapter 2 of this guide. These steps can also be applied to any .dll file for which you want to run a real-time simulation.

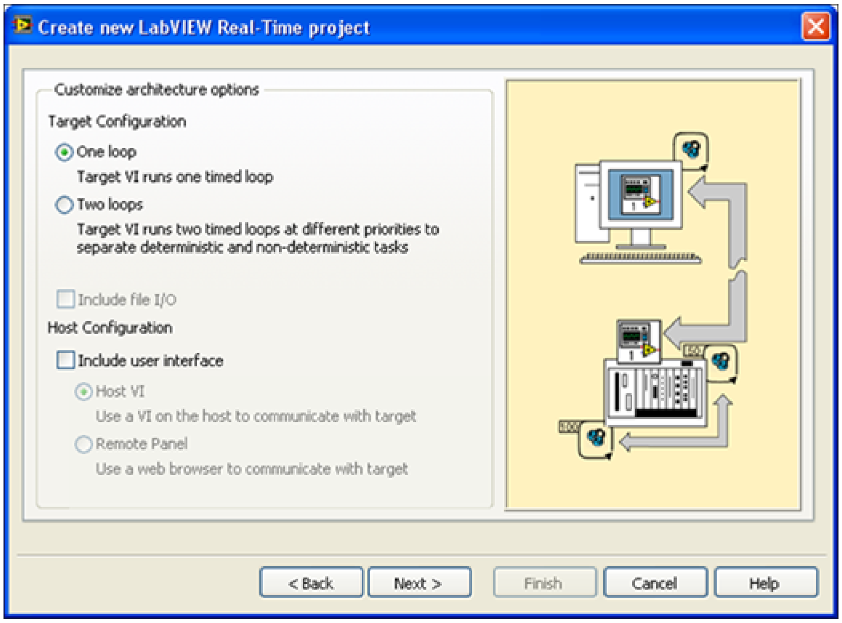
4.1 Preparing the LabVIEW Real-Time Project

1. From the LabVIEW **Getting Started** window, click **Real-Time Project**.

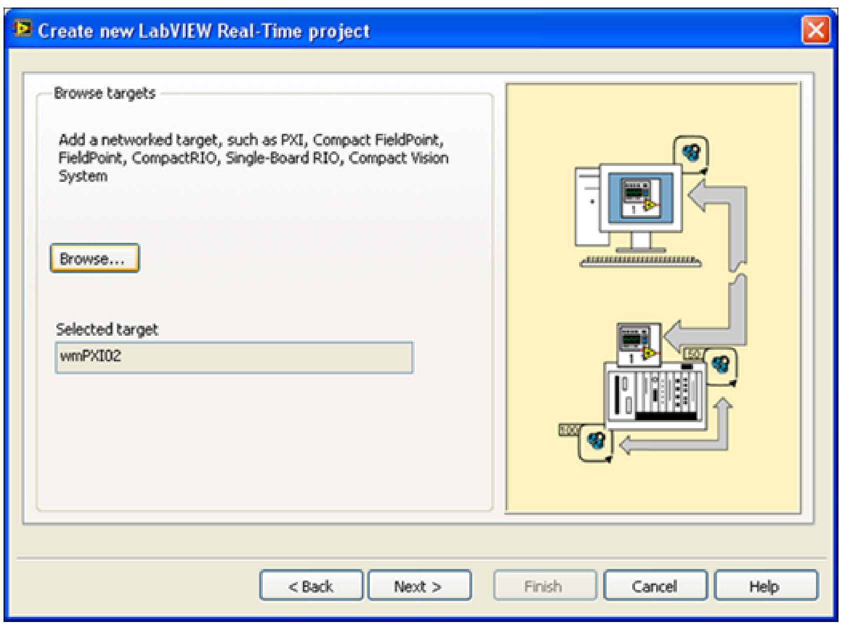
2. Keep the project type as **Continuous communication architecture,** change the project name to **RTSliderCrank**, and click **Next**.



3. Keep all of the default architecture options values and click **Next**.

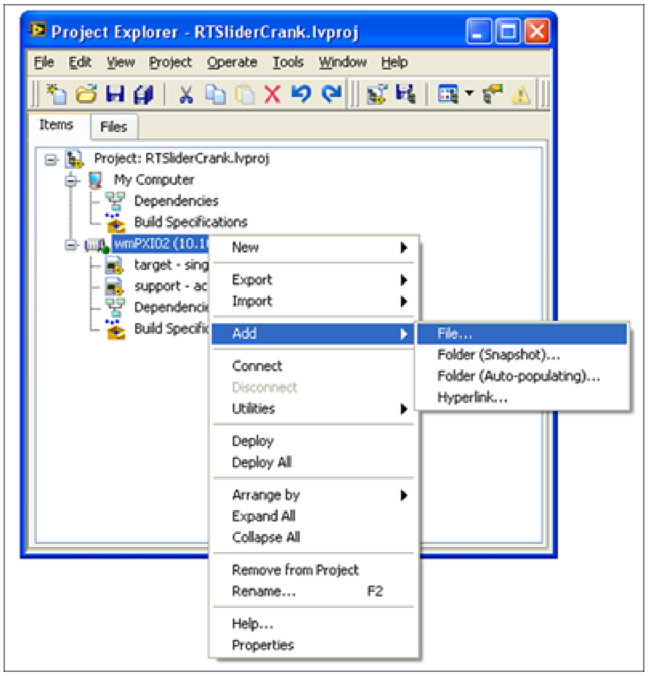


4. Click **Browse...** From the drop-down menu, browse to locate the real-time target platform. Click **Next**.



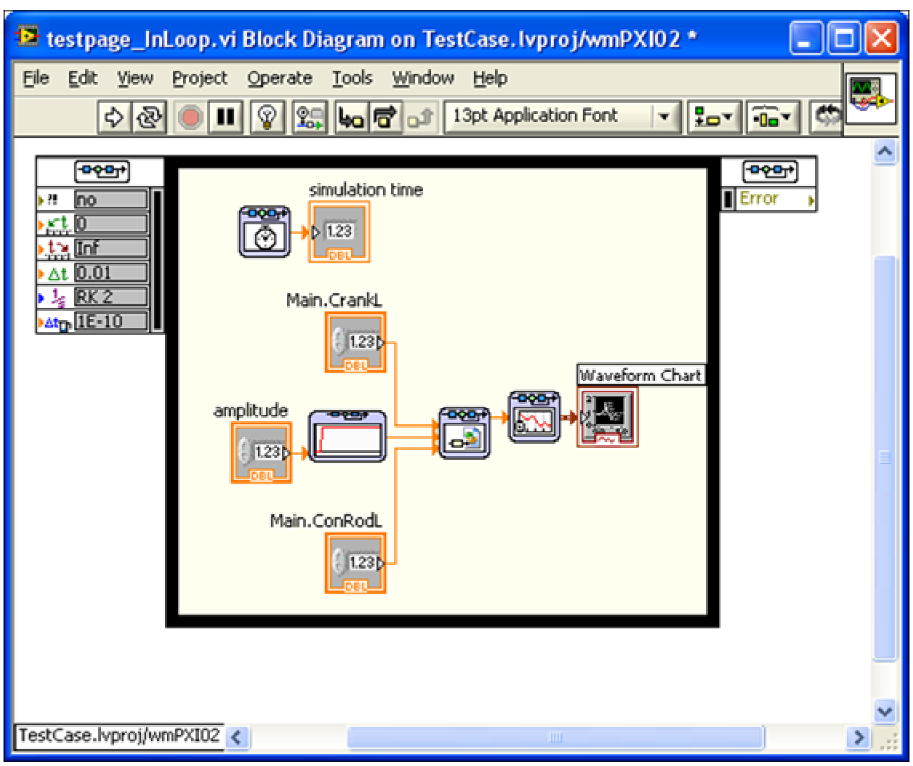
5. Click **Finish** to create the model. The model is displayed.

6. From the **Project Explorer**, right-click the entry of the target platform and select **Add >File..**.

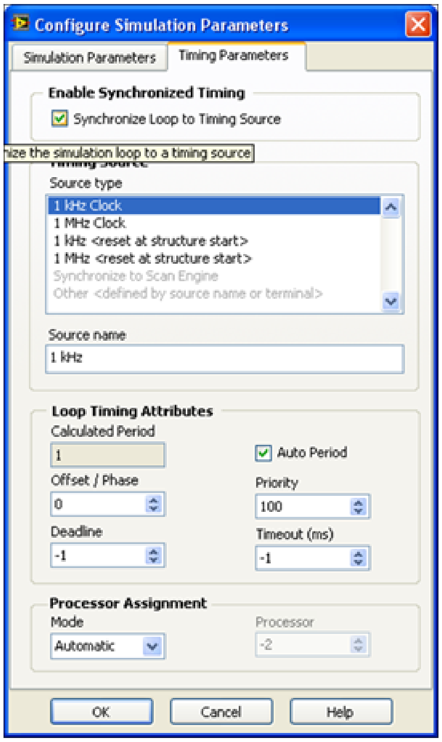


7. Browse to the **Release** subfolder located in the default directory that you specified in the **LabVIEW EMI Block Generation** template and open the .dll file that you generated. Click **OK**.

8. Navigate to the block diagram of the VI. Double-click the **Simulation Parameters** window to the left of the simulation loop. The **Configuration Simulation Parameters** window is displayed.



9. Click the **Time Parameters** tab and select **Synchronize loop to time source**. Click **OK**.



10. Save the file.

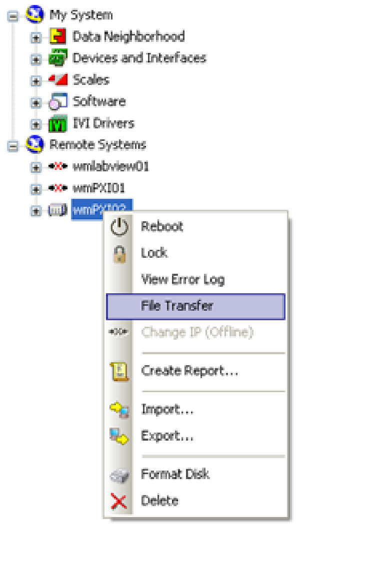
4.2 Moving the .dll File to the Target Real-Time Machine

1. From the start menu, select **Measurement and Automation Explorer**.

2. In the **Measurement and Automation Explorer** window, expand **Remote Systems**.

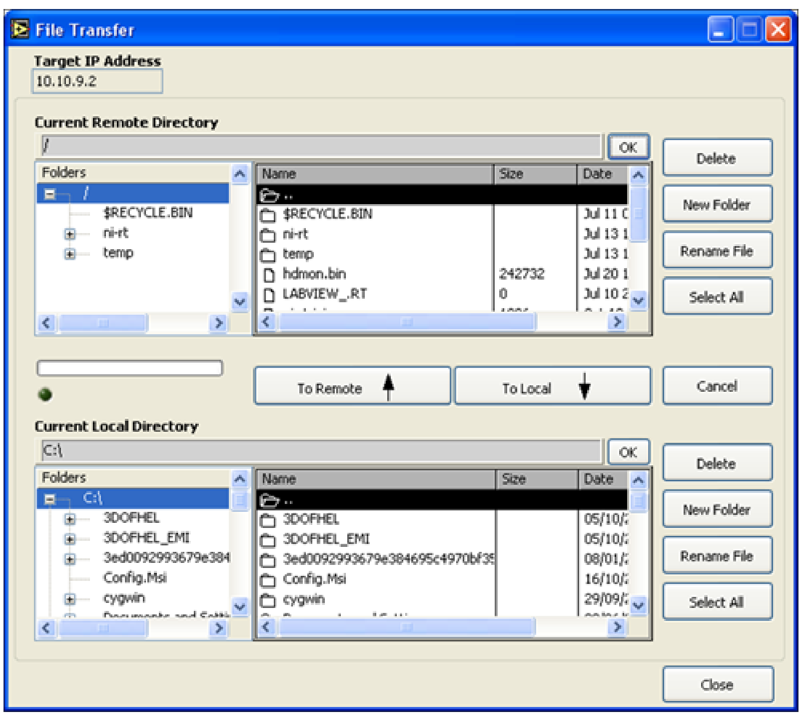
3. Right-click the entry for your target machine

4. Select **File Transfer**



5. Browse to the directory that contains the .dll file you created.

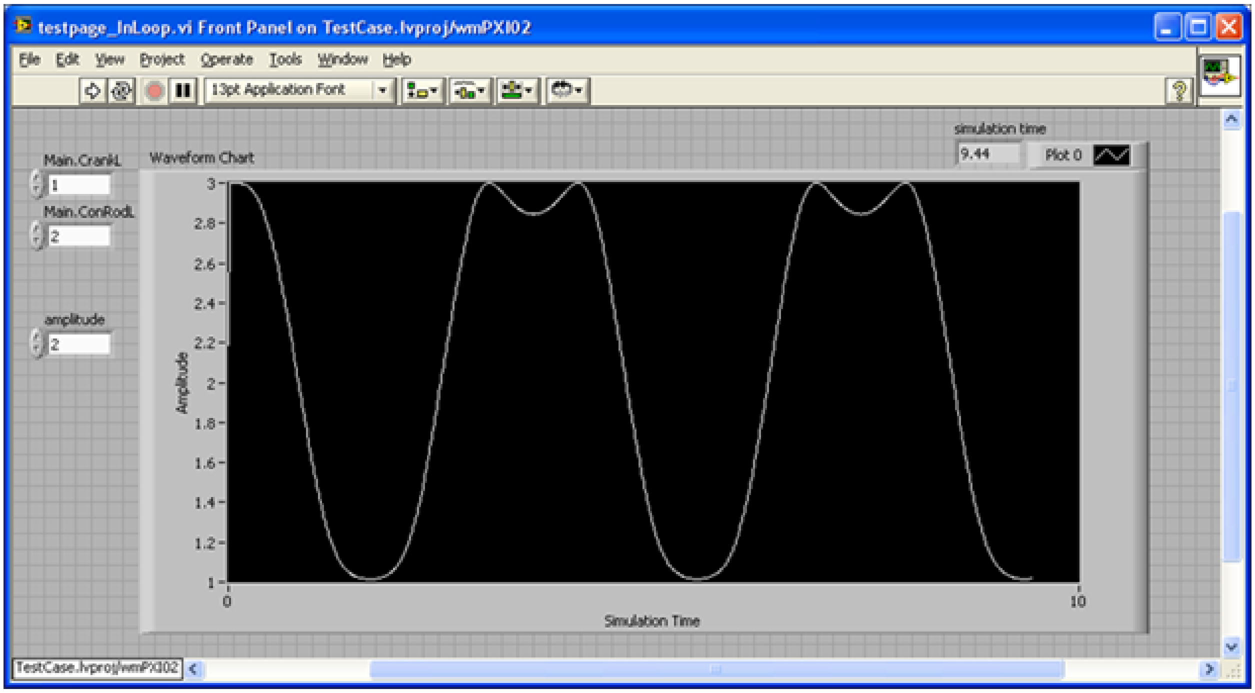
6. Select the .dll file.



7. Click **To Remote** to move the .dll file from your local machine to the target machine in the **ni-rt/system** directory

8. Click **Close**

9. Run the simulation by clicking the run button in the front panel of the VI.



Index