Getting Started with the MapleSim Connector for B&R Automation Studio

Getting Started with the MapleSim Connector for B&R Automation Studio

This document was produced using Maple and DocBook.

Contents

Introduction

The MapleSim™ Connector for B&R Automation Studio™ package provides all of the tools you need to prepare and export your dynamic systems models into B&R Automation Studio ANSI-C source code from a MapleSim model. You can create a model in MapleSim, simplify it in Maple™ by using an extensive range of analytical tools, and then generate the source code that you can incorporate into your toolchain.

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 B&R Connector. In general, the MapleSim B&R Connector supports systems of any complexity, including systems of DAEs of any index, in any mix of domains.

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 B&R ANSI-C Code Generation Steps

This chapter describes how to use the MapleSim Connector for B&R Automation Studio template and in the *Example: RLC Circuit Model (page 6)* section of this chapter, a step by step example shows you how to generate the C code. The MapleSim Connector for B&R Automation Studio template consists of the following steps in generating C code and is described in *Using the B&R Generation Template (page 1)*:

1. Subsystem selection
2. Inputs/Outputs and parameter management
3. C code generation options
4. Generate C code
5. View generated C code

MapleSim Connector for B&R Automation Studio Package

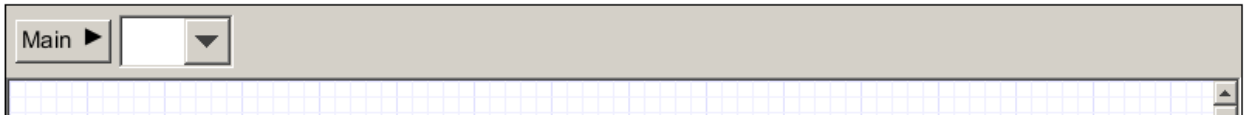
The BRConnector package is a collection of procedures for manually generating and compiling Automation Studio's ANSI-C code from MapleSim models, based on the model's algebraic equations and DynamicSystems objects.

For information about the MapleSim Connector for B&R Automation Studio package, in Maple, enter `?BRConnector` at a prompt in a worksheet.

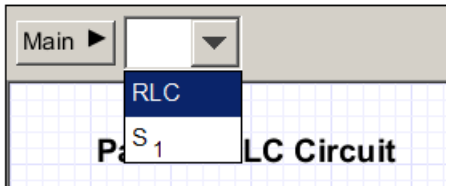
1.2 Opening the B&R Template

To open the B&R template

1. Click **Templates** ( ) in the **Main Toolbar** and select the **B&R Program Object Creation** template.
2. In the **Attachment** field, provide a worksheet name and click **Create Attachment**. Your MapleSim model opens in a Maple worksheet in the Subsystem Selection window. The toolbar shows all of the subsystems.



3. From the drop-down list select a subsystem. The subsystem and its contents appears in the Subsystem Selection window.



1.3 Using the B&R Generation Template

The MapleSim Connector for B&R Automation Studio provides a **B&R Generation** template in the form of a Maple worksheet for manipulating and exporting MapleSim subsystems. This template contain pre-built embedded components that allow you to generate C code from a MapleSim subsystem.

With this template, you can define inputs and outputs for the system, set the level of code optimization, generate the source code, and choose the format of the resulting C code and library code. You can use any Maple commands to

perform task analysis, assign model equations to a variable, group inputs and outputs and define additional input and output ports for variables.

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

Example models are available in the **B&R Connector Examples** palette in MapleSim.

Step 1: Subsystem Selection

This part of the template identifies the subsystem modeling components that you want to generate C code for. Since the B&R Automation Studio 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 to your model. 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 C code files. By creating a subsystem you not only improve the visual layout of a system in model workspace but you also prepare the model for export. The example in Chapter 2 shows you how to group all of the components into a subsystem.

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

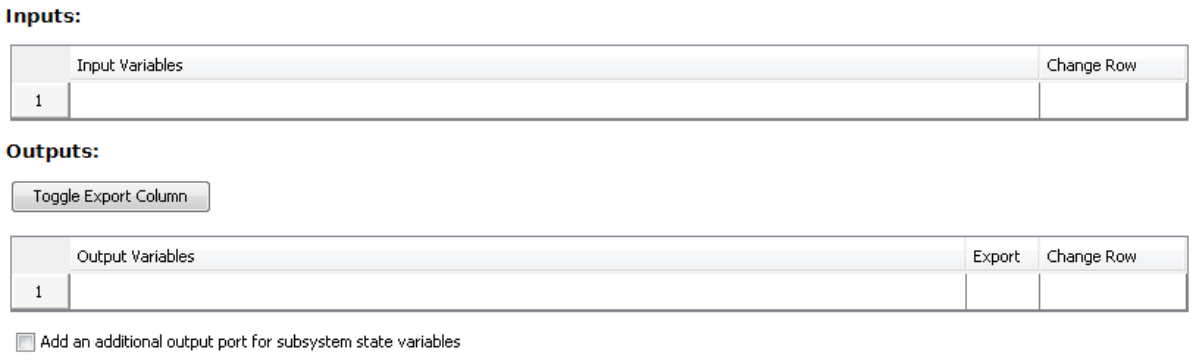
You can select which subsystems from your model you want to create C code for.

Load Selected Subsystem

After selecting a subsystem, click **Load Selected Subsystem**. All defined input and output ports are loaded.

Step 2: Inputs/Outputs and Parameter Management

The Port and Parameter Management interface 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.



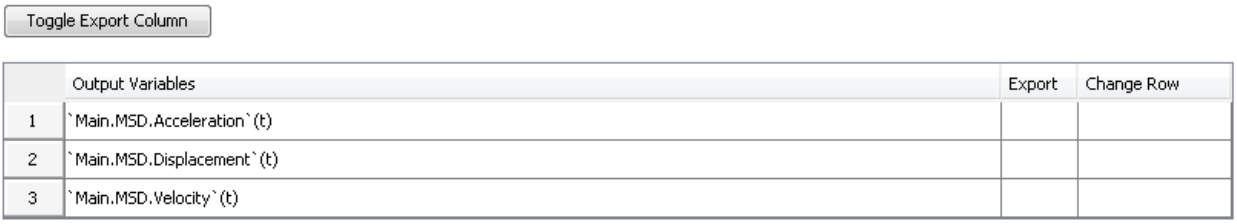
Select **Add an additional output port for subsystem state variables** to add extra output ports for the state variables.

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.

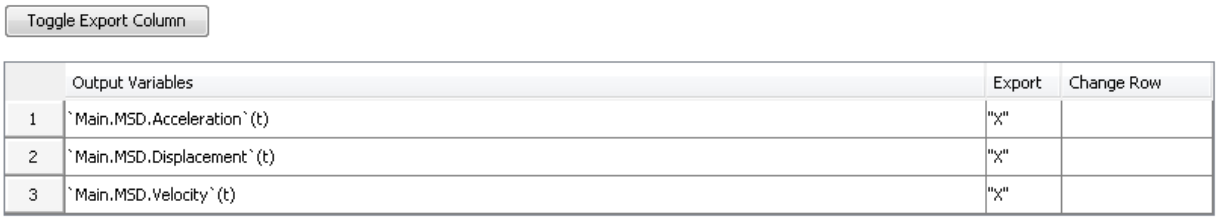
Note: If the parameters are not marked for export they will be numerically substituted.

Toggle Export Column

Use the **Toggle Button** to select or deselect all of the **Output Variables** or **Parameters** for export. In the following illustration, by default, no **Output Variables** are selected.



Click **Toggle Export Column** to select all of the **Output Variables** for export.



Note: Click individual **Output Variables** or **Parameters** to select specific items for export.

Step 3: C Code Generation Options

The C code Generation Options settings specify the advanced options for the code generation process.

Solver Options

Select the fixed step solver by specifying the numerical solution method for the model equations during the code generation process.

Select one of the following options:

Euler: *forward Euler* method

RK2: *second-order Runge-Kutta* method

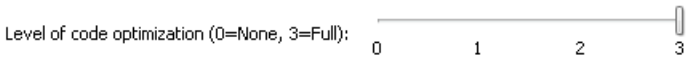
RK3: *third-order Runge-Kutta* method

RK4: *fourth-order Runge-Kutta* method

Implicit Euler: *implicit Euler* method

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:

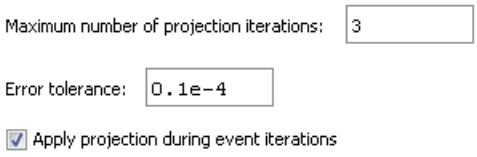
None (0): no optimization is performed; the default equations will be used in the generated code.

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 C code. 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.

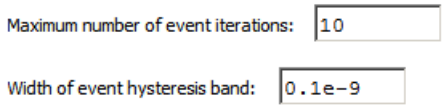
Set the **Error tolerance to** specify the desirable error tolerance to achieve after the projection.

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 C code. 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.



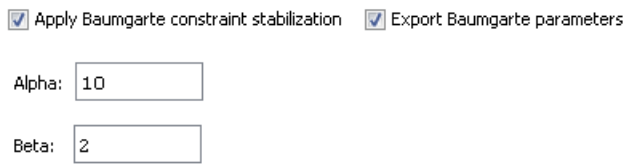
Set the **Maximum number of event iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

Set the **Width of event hysteresis 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.

Baumgarte Constraint Stabilization

The Baumgarte constraint stabilization method stabilizes the position constraint equations, by combining the position, velocity, and acceleration constraints into a single expression. By integrating the linear equation in terms of the acceleration, the Baumgarte parameters, alpha and beta, act to stabilize the constraints at the position level.



Baumgarte: Apply the Baumgarte constraint stabilization.

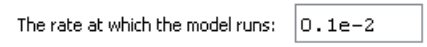
Export Baumgarte parameters: Add **Alpha** and **Beta** as parameters in the generated code.

Alpha: Set the derivative gain for Baumgarte constraint stabilization.

Beta: Set the proportional gain for Baumgarte constraint stabilization.

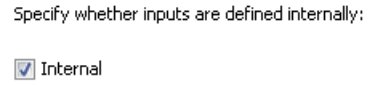
Baserate

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. Default is *[0.001]*.



Inputs

Select **Internal** to specify whether the inputs are internally defined.



Step 4: Generate B&R Program Object

Generating C code creates temporary files for viewing purposes in a user defined directory.

To generate C code

1. Provide the following information for the location and name of the generated code.

Target Directory: Browse to or create the location for the generated C code files.

Program Name: Provide a name for the generated C code folder. Within this folder three files are generated; B&R Automation Studio Program Object, ANSIC.prg and Local Variables .var.

2. To generate the C code click **Generate B&R Program**. The C code is saved to the target directory.

Step 5: View C Code

After the C code is generated, specific portions of the C code can be viewed:

B&R Automation Studio Program Object: Displays the code for implementation of the MapleSim Connector for B&R Automation Studio program

ANSIC.prg: Displays the MapleSim Connector for B&R Automation Studio program definition file

Local Variables .var: Displays the local variables

1.4 Viewing Examples

Within MapleSim there are many examples for you to view.

To view an example

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

Note: 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.

1.5 Example: RLC Circuit Model

In this example, you will generate C code for an RLC circuit model created in MapleSim.

To generate C code

1. Under the **Libraries** tab, open the **B&R Connector Examples** palette, and select the **RLC Parallel Circuit** example.
2. Click **Templates** ( ) in the **Main Toolbar**. The **Create Attachment for RLCcircuit** window appears.
3. From the list, select the **B&R Program Object Generation** template.
4. In the **Attachment** field, enter **RLC Circuit** as the worksheet name.
5. Click **Create Attachment**. Your MapleSim model opens in Maple, using the selected template.
6. In the **Subsystem Selection** section, select the **RLC** subsystem 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. 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 **C Code Generation Options** section, set the **Optimization Options** to **Full (3)**. 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.
9. In the **Generate B&R Program Object** section of the template, specify the target directory and name.
10. Click **Generate B&R Program**. The files are created and saved in the target directory.

Note: Generating a block may require a few minutes.

2 Exporting a Model and Generating C Code

This chapter describes how to prepare a model for generating C code using subsystems.

2.1 Preparing the 2-D Rigid Slider Crank Model for Generating C Code Using Subsystems

Preparing a Model

In this example, you will perform the steps required to prepare a slider-crank mechanism model for generating C code.

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

2. Define subsystem inputs and outputs.

3. Define and assign subsystem parameters.

4. Generate C code.

To open the slider-crank mechanism example

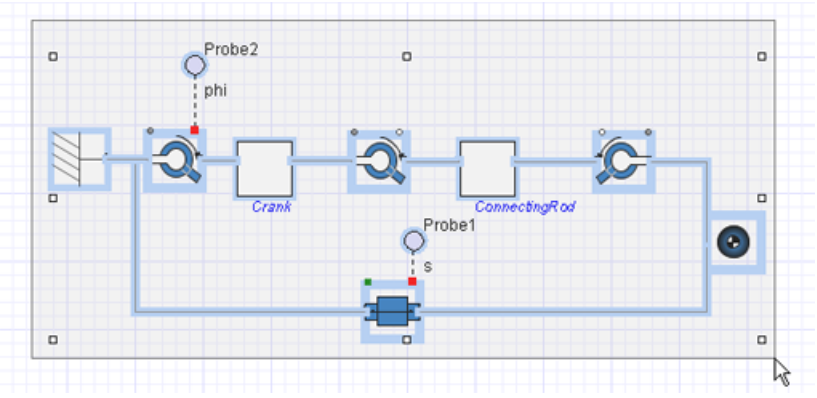
1. In MapleSim, under the **Libraries** tab, browse to the **Examples** > **User's Guide Examples** menu.
2. Open the **Planar Slider-Crank Mechanism** example in Chapter 6. The example appears in the **Model Workspace**.

Converting the Model to a Subsystem

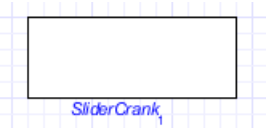
By converting your entire model or part of your model into a subsystem, you identify which parts of the model that you want to generate C code for. In this example, you will group all of the components into a subsystem.

To create a subsystem

1. Using the selection tool ( ) located in the **Model Workspace Toolbar**, 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**.



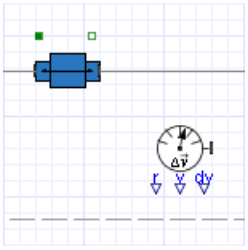
Defining Subsystem Inputs and Outputs

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 the MapleSim Connector for B&R Automation Studio 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 generated C code.

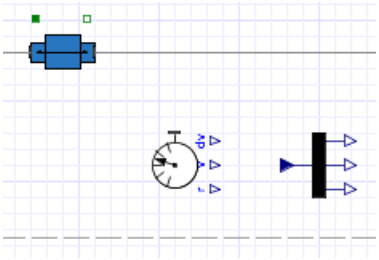
In this example, you will convert the displacements of the slider and the joint between the crank and connecting rod to output signals. The input signal needs to be converted to a torque that is applied to the revolute joint that represents the crank shaft.

To create a subsystem output port

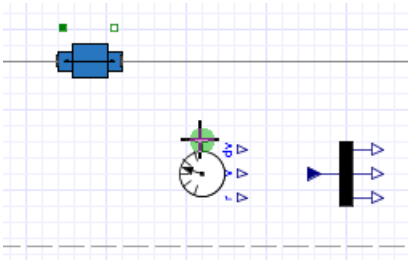
1. Double-click the subsystem block to view its contents. The broken line surrounding the components indicates the subsystem boundary, which can be resized by clicking and dragging its sizing handles.
2. Delete the probes that are attached to the model.
3. Under the **Libraries** tab on the left side 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.



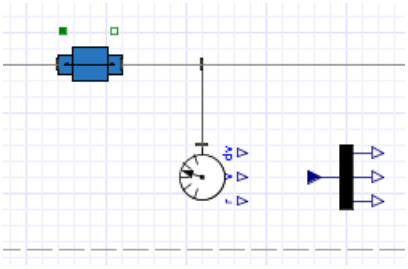
5. Right-click (**Control**-click for Macintosh®) the **Absolute Translation** component and select **Rotate Counterclockwise**.
6. From the **Signal Blocks > Routing > Demultiplexers** menu, drag a **Real Demultiplexer** component to the **Model Workspace** and place it to the right of the **Absolute Translation** component.



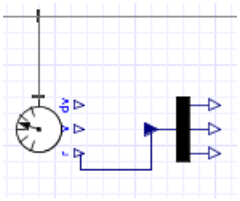
7. To connect the **Absolute Translation** component to the model, click the frame_b connector. The frame is highlighted in green when you hover your pointer over it.



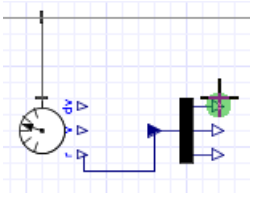
8. Draw a vertical line and click the connection line directly above the component. The sensor is connected to the rest of the diagram.



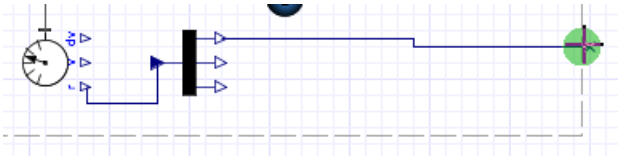
9. In the same way, connect the **r** output port (*TMOutputP*) of the **Absolute Translation** component to the navy blue input port of the demultiplexer. This is the displacement signal from the sensor in x, y, and z coordinates. Since the slider only moves along the x axis, the first coordinate must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



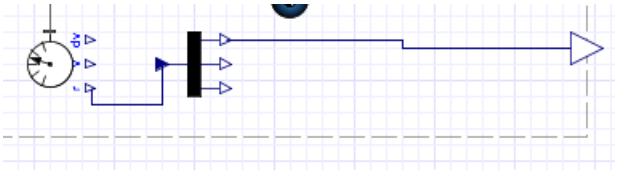
10. Hover your pointer over the first demultiplexer port and click your mouse button once.



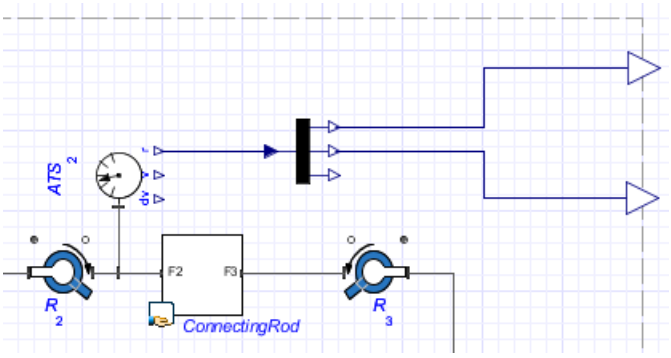
11. Drag your pointer to the subsystem boundary.



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



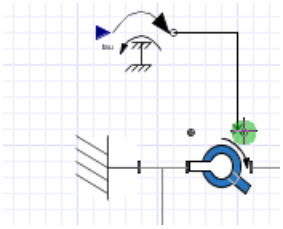
13. Add another **Absolute Translation** component above the **Connecting Rod** subsystem.
14. Right-click (**Control**-click for Macintosh) the **Absolute Translation** component and select **Flip Vertical.** Right-click the **Absolute Translation** component again and select **Rotate Clockwise**.
15. Add a **Real Demultiplexer** component to the right of the sensor and connect the components as shown below.



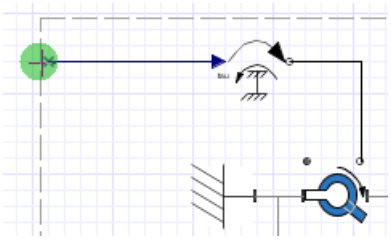
Note: Since the crank is moving in the x-y plane, you only need to output the first two signals.

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

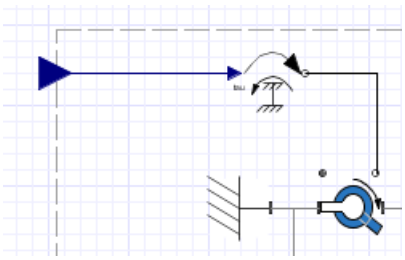
16. From the **1-D Mechanical > Rotational > Torque Drivers** menu, add a **Torque** component to the **Model Workspace** and place it above the **Fixed Frame** component.
17. Connect the white flange of the **Torque** component to the white flange of the leftmost **Revolute Joint**.



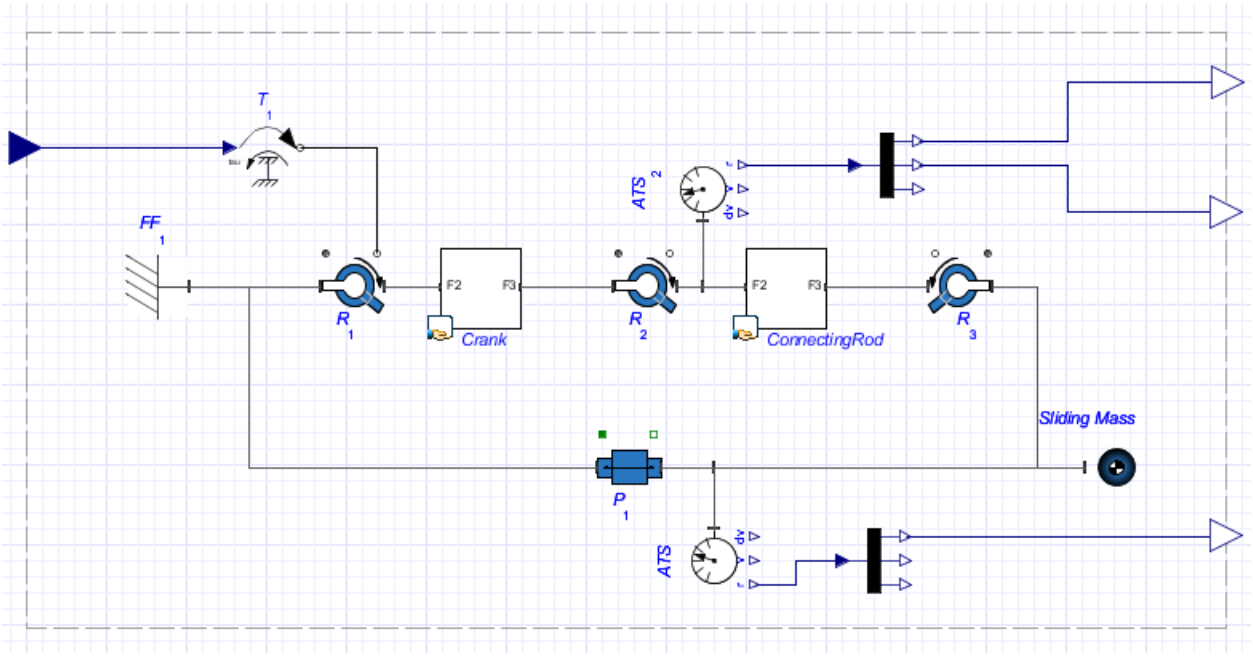
18. Click the input port of the **Torque** component and drag your pointer to the subsystem boundary.



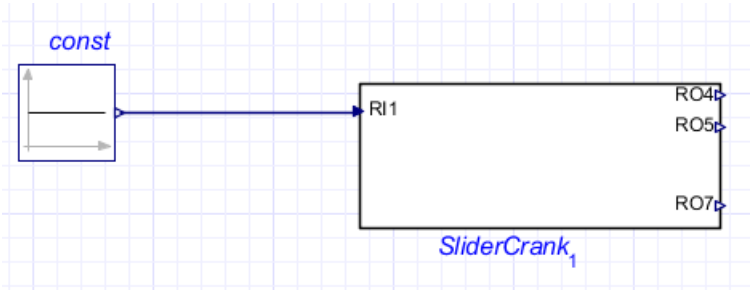
19. Click the boundary once. A real input port is added to your subsystem.



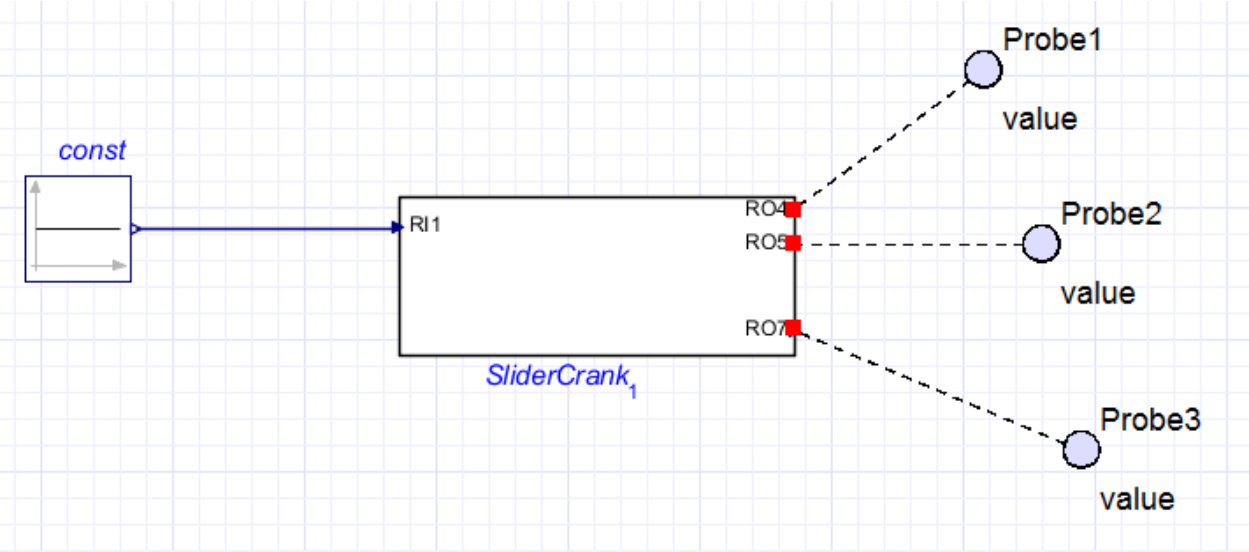
The complete subsystem appears below.



20. Click **Main** in the **Model Workspace Toolbar** to browse to the top level of the model.
21. From the **Signal Blocks > Sources > Real** menu, drag a **Constant** source into the Nonlinear Spring Damper and connect its output port to the input port of the **SliderCrank** subsystem as shown below.



22. Click **Attach Probe** ( ) in the **Model Workspace Toolbar**.
23. Click the top output port of the **SliderCrank** subsystem.
24. Drag the probe to an empty location on the **Model Workspace**, and then click the workspace to position the probe.
25. In the same way, add probes to the other **SliderCrank** output ports as shown below.



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.

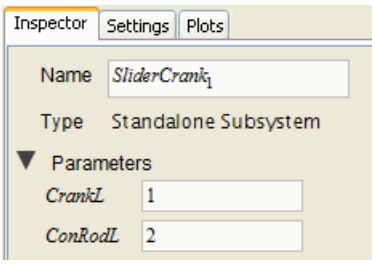
To edit parameters

1. While in the detailed view of the **SliderCrank** subsystem, click **Parameters** () in the **Navigation Toolbar**. 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

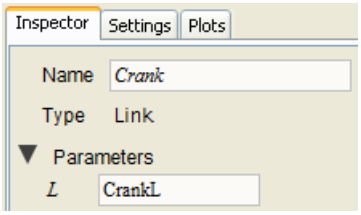
Name	Type	Default Value	Default Units	Description
CrankL	Real	1		Length of the crank
ConRodL	Real	2		Length of the connecting rod

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

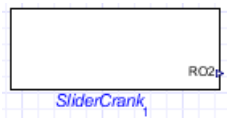


7. In the **Model Workspace**, select the **Crank** subsystem.

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. Click SliderCrank1 in the **Navigation Toolbar** above the workspace to switch back to the SliderCrank view, and then select the **ConnectingRod** subsystem.
10. Change its length value to **ConRodL**.
11. Click **Main** in the **Navigation Toolbar** 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 C code.

Generating C Code for the Template

After preparing the model, you can use the MapleSim Connector for B&R Automation Studio file Generation template to set the C code generation options for generation.

To generate C code

1. Click **Templates** ( ) in the **Main Toolbar**.
2. The **Create Attachment** window appears. From the list, select **B&R Program Object Generation**.
3. In the **Attachment** name field, enter **Slider Crank B&R** as the worksheet name and click **Create Attachment**. The slider-crank model opens in the **Subsystem Selection** window in the **B&R Connector Template**.
4. In the **Subsystem Selection** section, select the **SliderCrank$_1$** subsystem from the drop-down menu and click **Load Selected Subsystem**. All of the template fields are populated with information specific to the subsystem.
5. Specify the location for the **Target Directory** and provide a name for the generated files.
6. Click **Generate B&R Program**. The files are created and saved in the target directory.

2.2 Generating C Code for the Nonlinear Spring Damper Model

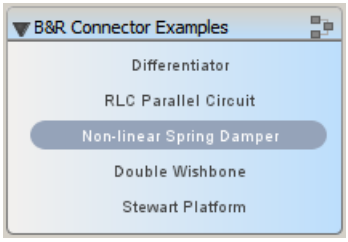
Preparing a Model

In this example, you will perform the steps required to generate C code using the Nonlinear Spring Damper model.

1. Open the Nonlinear Spring Damper example.
2. Generate the B&R template.
3. Define template settings.
4. Generate C code and view files.

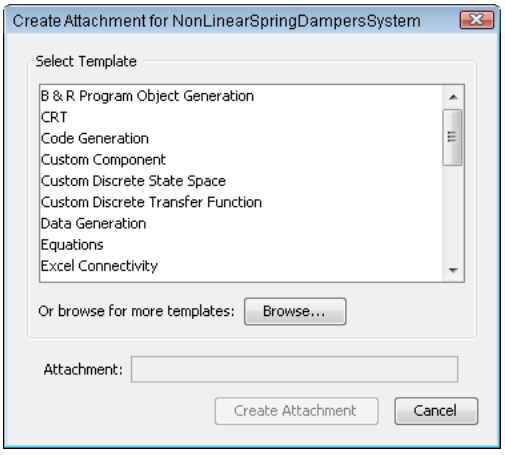
To open the Nonlinear Spring Damper example

1. Under the **Libraries** tab, expand the **B&R Connector Examples** palette.
2. Open the **Nonlinear Spring Damper** example.



To generate the MapleSim Connector for B&R Automation Studio template

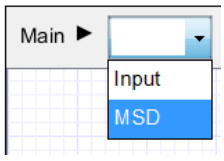
1. From the **B&R Connector** Examples palette, open the **Nonlinear Spring Damper** example.
2. Click **Templates** ( ) in the **Main Toolbar**. The **Create Attachment** window appears.



3. From the list, select the **B&R Program Object Generation** template.
4. In the **Attachment** field, enter **Nonlinear Spring Damper** as the worksheet name.
5. Click **Create Attachment**. Your MapleSim model opens in Maple, using the selected template.

To define the template settings

1. Browse to the Nonlinear Spring Damper subsystem by selecting the subsystem name from the drop-down menu in the **Navigation Toolbar** above the model diagram. This menu displays all of the subsystems and components in your MapleSim model.



2. 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 to keep as configurable parameters in the generated block.

3. In the **C Code Generation Options** section, set the following options:

C Code Generation Options	Setting
Solver Options • Fixed step solver	**Euler**
Optimization Options • Level of code optimization (0=None, 3=Full):	**3**
Constraint Handling Options • Maximum number of projection iterations • Error tolerance • Apply projection during event iterations	**3** **$0.1e^{-4}$** □ Apply projection during event iterations (*clear check box*)
Event Handling Options • Maximum number of event iterations • Width of event hysteresis band	**10** **$0.1e^{-9}$**
Baserate • The rate at which the model runs	**$0.1e^{-1}$**
Inputs • Specify whether inputs are defined internally	□ Internal (clear check box)

4. In the **Generate B&R Program** section of the template, specify the target directory and name.

To generate C code

1. Click **Generate B&R Program**. The files are created and saved in the target directory. You can view the three generated files (B & R Automation Studio Program Object, ANSIC.prg, and Local Variables .var) in the **View C Code** section of the template.

Note: Generating a block may require a few minutes.

3 Working With Your Code in B&R Automation Studio

This chapter describes how to work with your C code in the B&R Automation Studio application using the MSD model that you generated in *Generating C Code for the Nonlinear Spring Damper Model (page 14)*

Note: For a complete description of the B&R Automation Studio see the Automation Studio **B&R Help Explorer**.

3.1 Preparing a MapleSim Model to Run as a New B&R Project

The preparation procedure consists of the following steps:

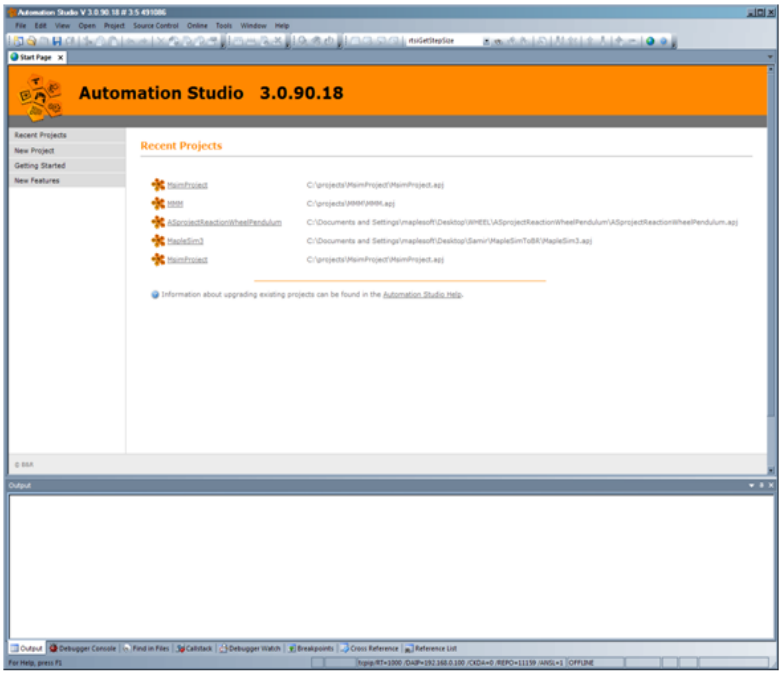
1. Creating a new project
2. Adding the program object to the CPU
3. Configuring the MSD program object on the active CPU
4. Adding library objects to the project
5. Building the configuration and transferring the project to the target
6. Adding a Watch
7. Adding a Trace
8. Simulating the project
9. Viewing the Simulation Results

Creating a New Project

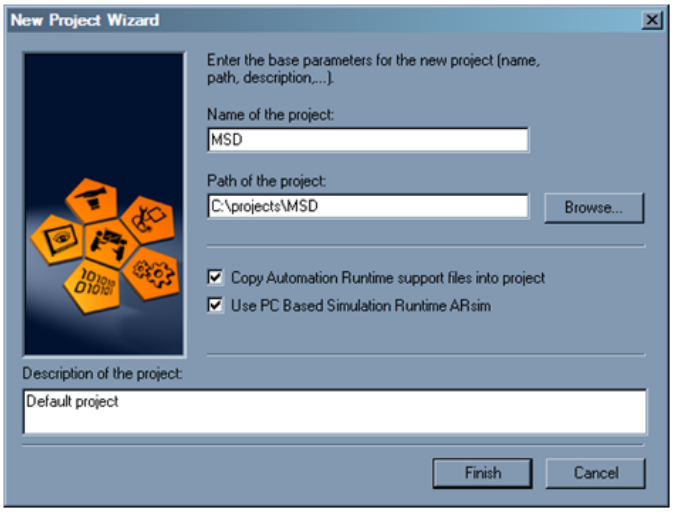
A project contains the application source data (programs, libraries and data objects) and configurations (hardware description, software and I/O mapping data, libraries, and data objects). You must first create a project in order to enter a controller application.

To create a new Automation Studio project file

1. Open **B&R Automation Studio**. The **Main Window** appears.



2. From the **File** menu, select **New Project**. The wizard for creating a new project is started.

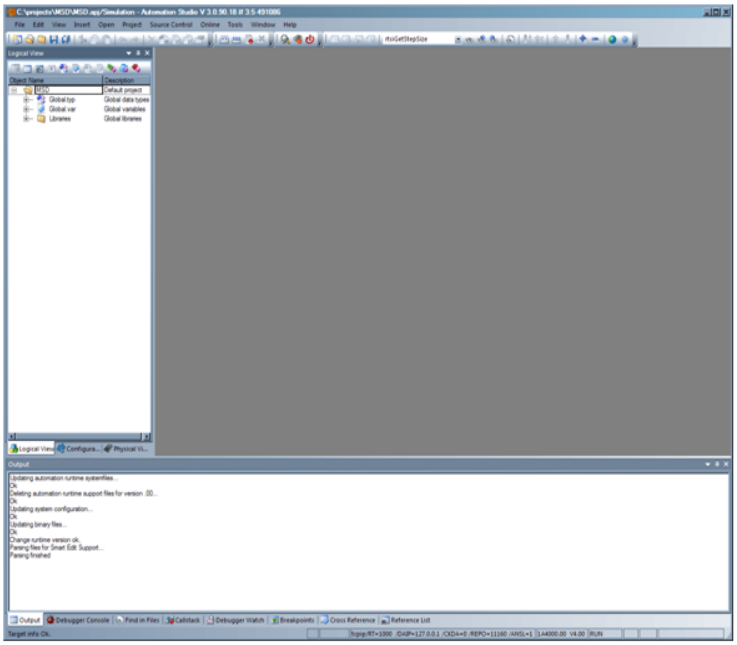


3. Provide the following basic project data definitions:

Parameter	**Description**
Project name	Project name
Path for the project	Specify the target directory for the generated files. Use the **Browse** button to select a specific directory.
Copy Automation Runtime support files into project	Select this option to copy all of the Automation Runtime files to the current project. You can continue to work even if this option is not selected. This setting is used for the next project and must be deselected if the project uses specific target hardware.
Use PC Based Simulation Runtime ARsim	Select this option to set the PC based Simulation Runtime ARsim as the target. This setting is used for the next project and must be deselected if the project uses specific target hardware.

Description of the project	Provide a brief description of the project

4. Click **Finish**. The project files are added to the project directory and the **Project** window appears displaying the target directory and the global data types, global variables and the four default global libraries.

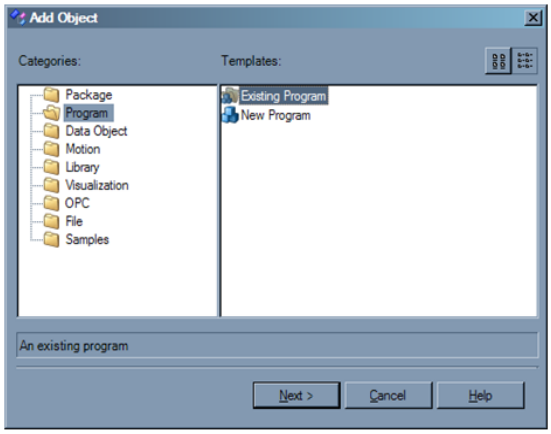


Adding the Program Object to the CPU

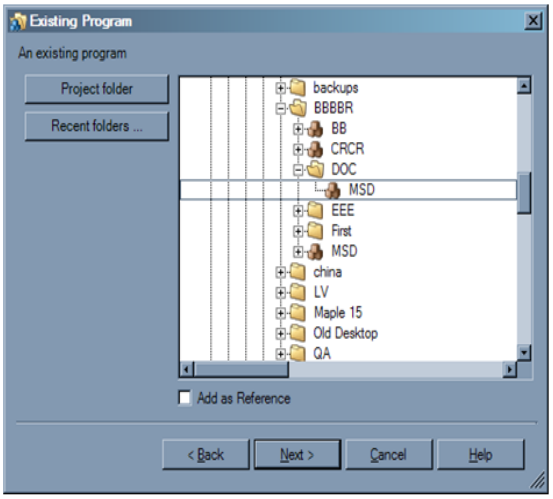
In order to utilize the MapleSim model, the CPU requires the generated C code to be added as a B&R program object.

To add objects to the CPU

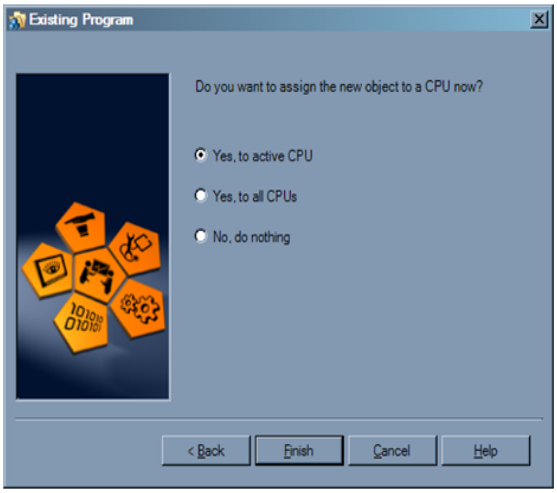
1. Switch to the **Logical View** and from the **Main Tool** bar, select **Insert > Add Object...** (alternatively select the **Logical View**, right-click and select **Add Objects**). The **Add Object** window appears.



2. Select the **Program, Existing Program** folder and click **Next**. The **Existing Program** window appears showing the program list.



3. Select **MSD** and click **Next**. The **Existing Program** window appears showing the object assignment selection.



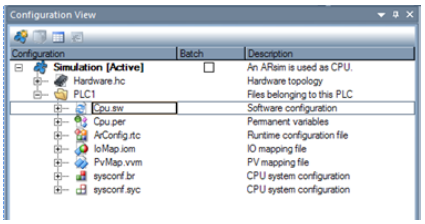
4. Select **Yes, to active CPU** and click **Finish**. The **Project** window appears displaying added objects to the CPU; implementation code (**MSD.c**), local variables (**MSD.var**) and MapleSim model implementation (**cMsimModel.h**).

Configuring the MSD Program Object on the Active CPU

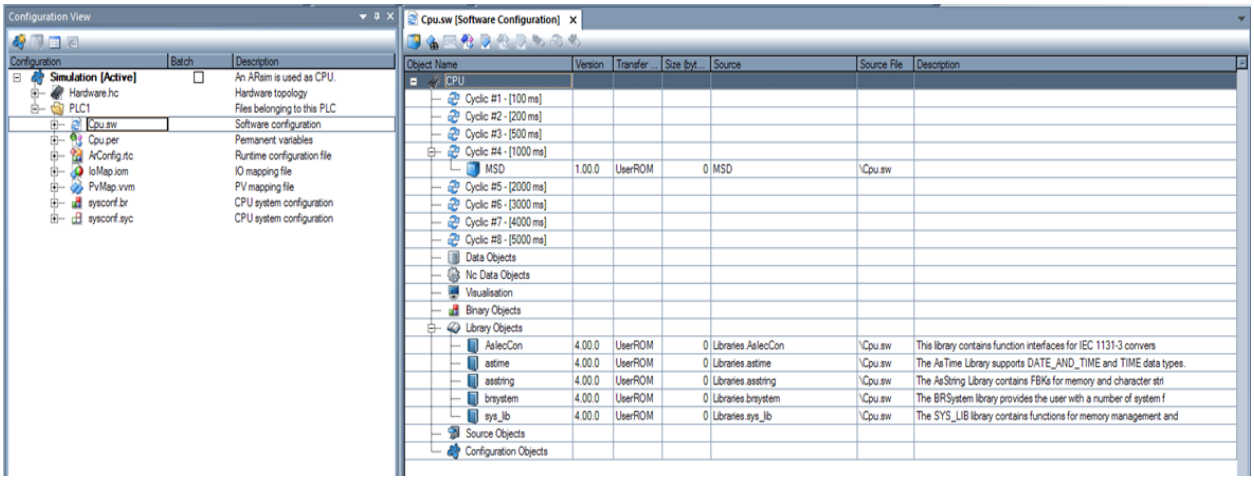
The CPU requires the program object duration runs at the same rate as the baserate specified in the **B&R Template** (see *Baserate (page 5)*).

To configure the MSD program object on the active CPU

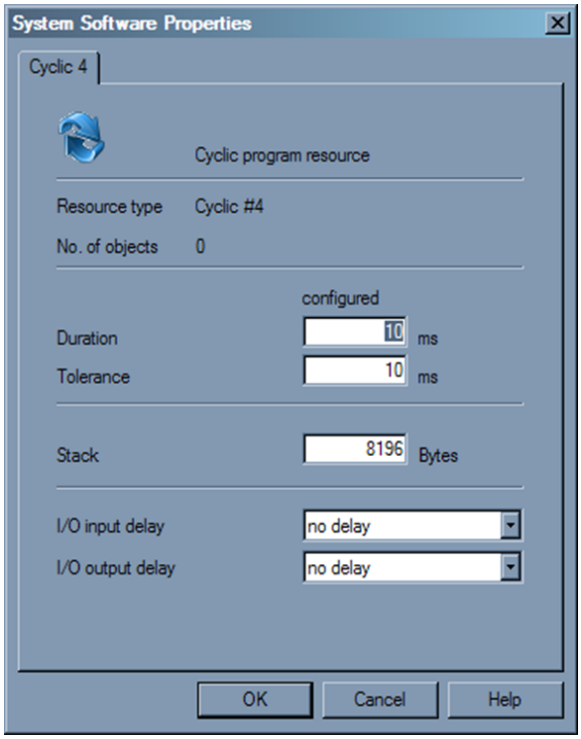
1. Switch to the **Configuration View** and expand **PLC1**.



2. Double-click **CPU.SW**. The **CPU.SW** window appears.



3. Right-click on the **Cyclic #4** object program. The **System Software Properties** window appears.



4. Set the **Cyclic #4** object program **Duration** to 10 ms.

Note: By default the B&R **Cyclic #4** object is 100 ms. Ensure that the **Cyclic #4** object program duration matches the baserate specified in the B&R Template (see *Baserate (page 5)*). For this example the cyclic duration is 10ms. To change the default duration value for the B&R **Cyclic #4** object, right-click on **Cyclic #4 > Properties** and change **Duration** to 10ms and **Tolerance** to 10ms. For addition information click **Help**.

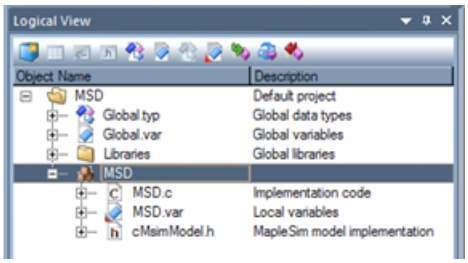
5. Click **OK**. The **Cyclic #4** object program duration value changes to 10 ms and the project is automatically saved.

Adding Library Objects to the Project

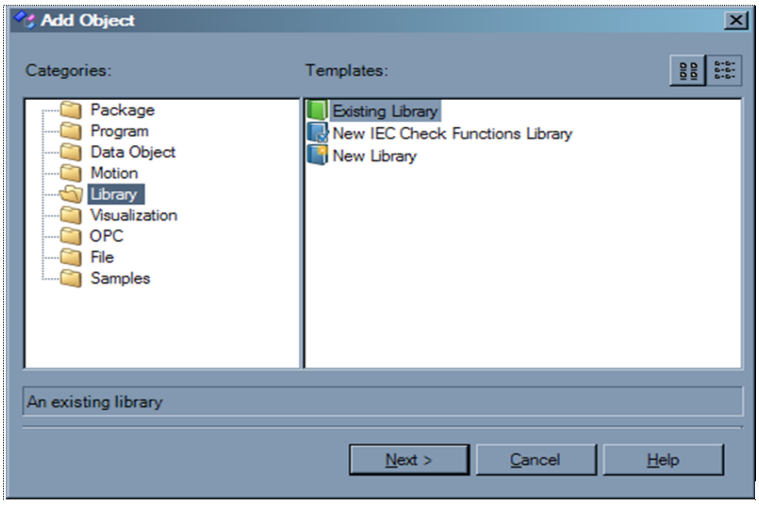
A library is a collection of project specific reusable functions grouped into functional categories.

To add a library object to a project

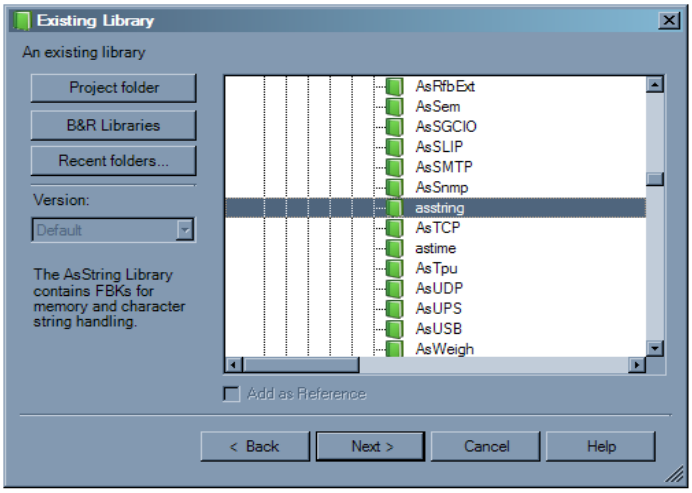
1. Switch to the **Logical View** and select **MSD**.



2. From the **Main Tool** bar, select **Insert > Add Object...** (alternatively select the **Logical View**, right-click and select **Add Objects**). The **Add Object** window appears.

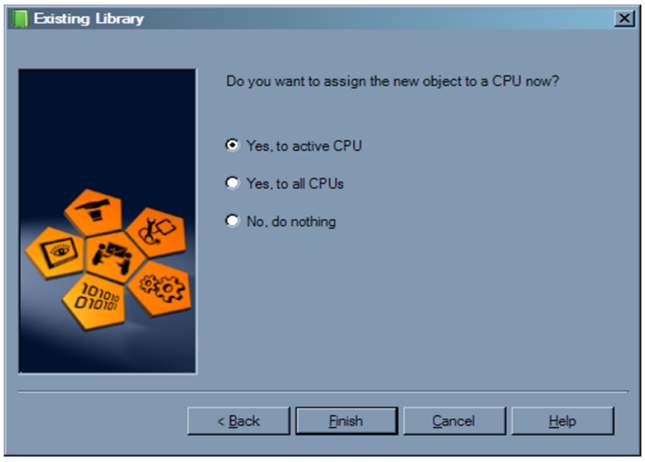


3. Select **Library > Existing Library**. Click **Next**. The **Existing Library** window appears.



4. Select the following files:
   - asstring
   - brsystem
   - syslib

5. Click **Next**. The **Existing Library** window appears showing the object assignment selection.



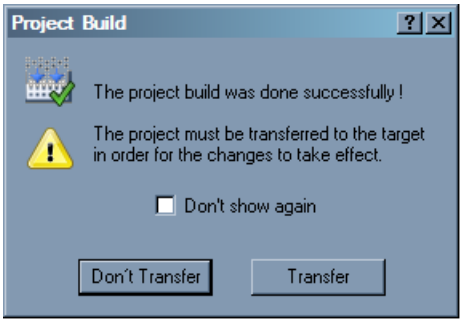
6. Select **Yes, to active CPU** and click **Finish**. The libraries are successfully added to the Project and assigned to the CPU.

Building the Configuration and Transferring the Project to the Target

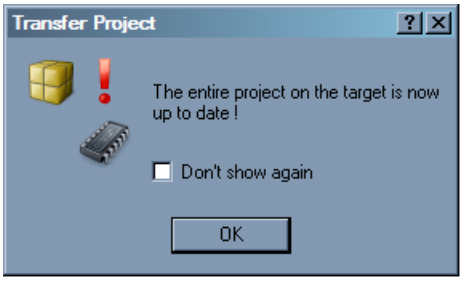
This procedure builds and transfers the project files to the target device.

To build the configuration and transfer the project to the target

1. Switch to the **Configuration View** and expand **PLC1**.
2. From the **Main Tool** bar, select the **Project > Build Configuration** (alternatively press **F7**). When the project is built and the **Project Build** window appears.



3. Click **Transfer**. The project files are transferred to the device and the **Transfer Project** window appears.



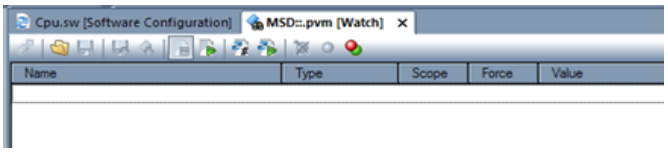
4. Click **OK**.

Adding a Watch

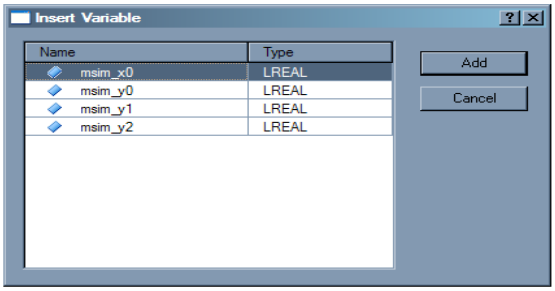
Adding a **Watch** lets you monitor the program variables during the simulation or on the PLC.

To add a watch

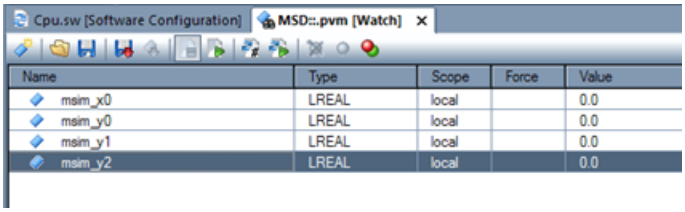
1. Switch to the **Logical View** and right-click on **MSD** program object. A drop-down menu appears.
2. Select **Open > Watch**. The **MSD::pvm (Watch)** window appears.



3. Right-click inside the window and select **Insert Variable...**. The **Insert Variable...** window appears.



4. Select the four variables and click **Add**. The four variables are loaded into the **MSD::pvm (Watch)** window.



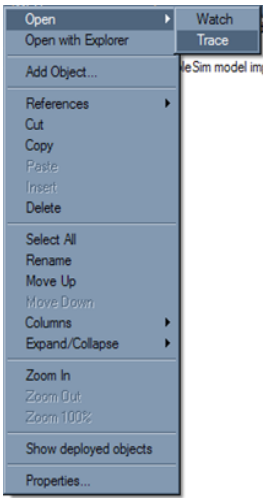
Note: msim_x0 is the input and requires a nonzero value in order to trace the simulation (see *Simulating the Project (page 26)*).

Adding a Trace

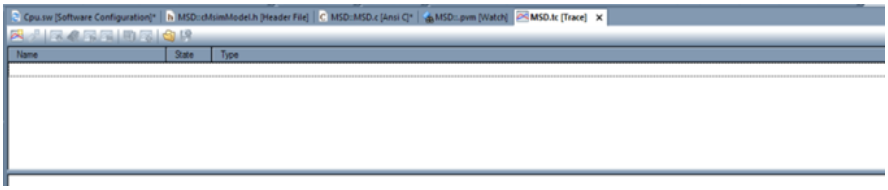
The **Trace** tool records and plots program values.

To add a Trace

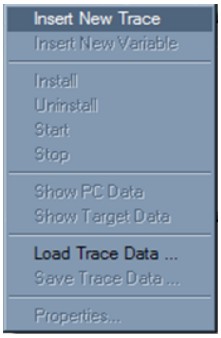
1. Switch to the **Logical View** and right-click on **MSD** program object. A drop-down menu appears.



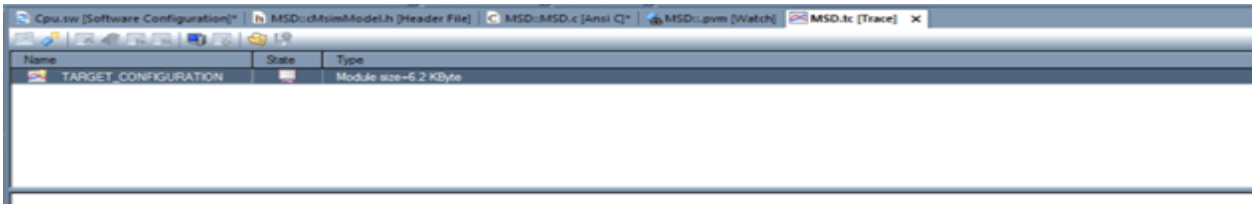
2. Select **Open > Trace**. The **MSD.tr (Trace)** window appears.



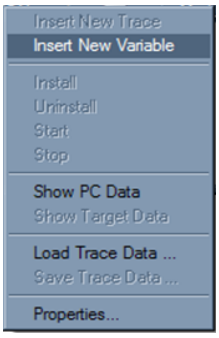
3. Right-click anywhere in the top frame of the **Trace** window. A drop-down menu appears.



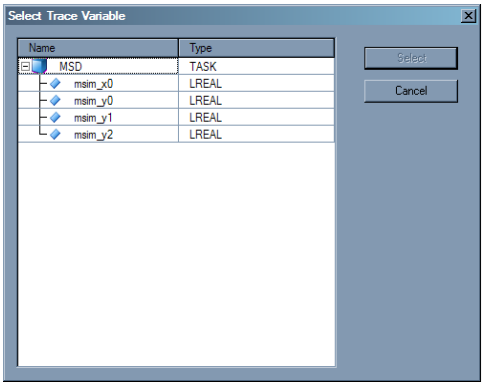
4. Select **Insert New Trace**. The **Trace** window appears with the **TARGET_CONFIGURATION**.



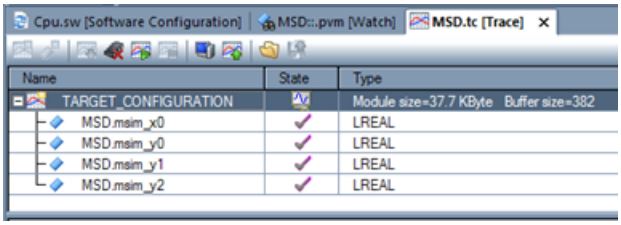
5. Right-click on **TARGET_CONFIGURATION**. A drop-down menu appears.



6. Select **Insert New Variable**. The **Select Trace Variable** window appears.



7. Select all the variables and click **Select**. The **Trace** window appears with the **TARGET_CONFIGURATION** variables.



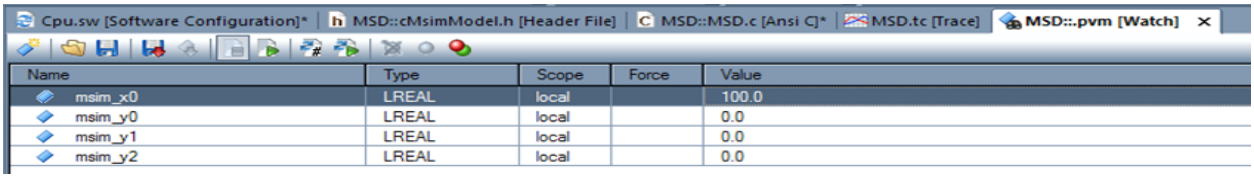
8. Right-click on **TARGET_CONFIGURATION > Install**. The **Trace** is correctly installed on the Target.

Simulating the Project

At this point the project is ready for simulation.

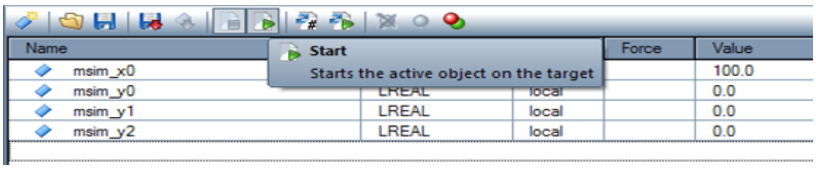
To simulate the project

1. Switch to the **MSD::pvm (Watch)** window and double click on the **msim_x0 Value** cell. Enter an input value of 100.

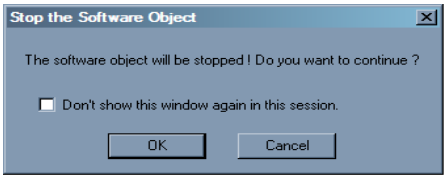


Note: In order for the simulation results to correspond with the MapleSim model use the same input signal values.

2. From the **Toolbar** click the **Start** ( ) button (alternatively right-click inside the **Watch** window and select **Start**). The simulation starts and the computed values appear.



3. Run the simulation until the values reach a steady state for approximately 30 seconds.
4. To stop the simulation click the **Stop** ( ) Button. The **Stop the Software Object** window appears.



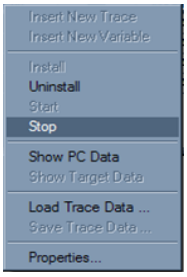
5. Click **OK**.

Viewing the Simulation Results

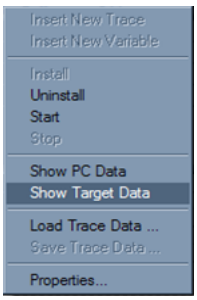
After simulation the results can be viewed.

To view the project

1. Switch to the **MSD.tc (Trace)** window and right-click on **TARGET_CONFIGURATION**. A drop-down menu appears.



2. Click **Stop** to stop the trace.
3. Right-click on **TARGET_CONFIGURATION**. A drop-down menu appears.



4. Select **Show Target Data**. The results for each of the target configuration variables appear.


Cpu.sw [Software Configuration]
MSD::pvm [Watch]
MSD.tc [Trace]
Name
State
Type
TARGET_CONFIGURATION
Module size=37.7 KByte Buffer size=382
MSD.msim_x0
MSD.msim_y0
MSD.msim_y1
MSD.msim_y2
LREAL
TARGET_DATA
UnitX
UnitY
x-pos. 0.000000
y-pos. 0.000000
x-diff. 0.000000
y-diff. 0.000000

4 Preparing a MapleSim Model to Run as a New B&R Project on an X-20 System

This chapter describes how to run a model on an X-20 System and how to run your C code using the MSD model that you generated in *Generating C Code for the Nonlinear Spring Damper Model (page 14)*

Note: For a complete detailed description of the MapleSim Connector for B&R Automation Studio see the Automation Studio B&R Help Explorer.

4.1 Preparing a MapleSim Model to Run as a New B&R Project

The preparation procedure consists of the following steps:

1. *Establishing a Connection with the X20 (page 29)*
2. *Creating a New Project (page 32)*
3. *Adding the Program Object to the CPU (page 36)*
4. *Configuring the MSD Program Object on the Active CPU (page 37)*
5. *Adding Library Objects to the Project (page 39)*
6. *Building the Configuration and Transferring the Project to the Target (page 40)*
7. *Adding a Watch (page 41)*
8. *Adding a Trace (page 42)*
9. *Simulating the Project (page 44)*
10. *Viewing the Simulation Results (page 45)*

4.2 Establishing a Connection with the X20

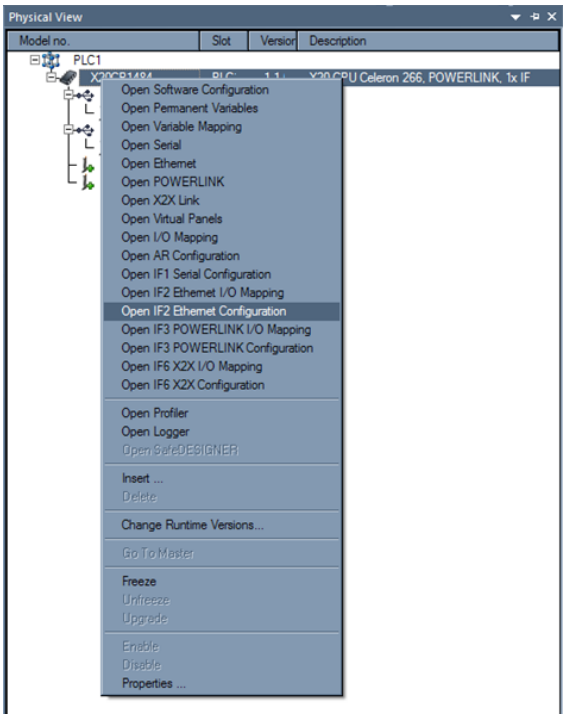
In order to run a real-time target machine using the generated C code you must first establish a connection as follows:

1. Configure the ethernet settings on the X20
2. Configure the PC's Ethernet interface
3. Configure the online connection

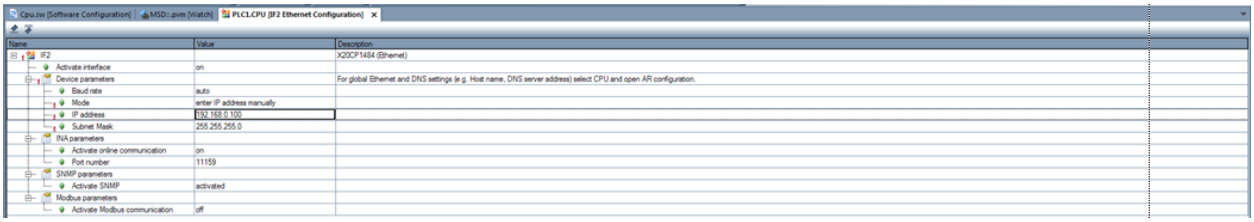
Note: Ensure that the online connection is configured properly so that B&R Automation Studio can verify the target system hardware configuration. Check the connection between the computer and controller, and ensure that an operating system is installed on the target system.

To configure the ethernet settings on the X20

1. From the **Physical View** right-click on the **X20** device name. A list appears.



2. Click **IF2 Ethernet Configuration**. The **IF2 Ethernet Configuration** window appears.

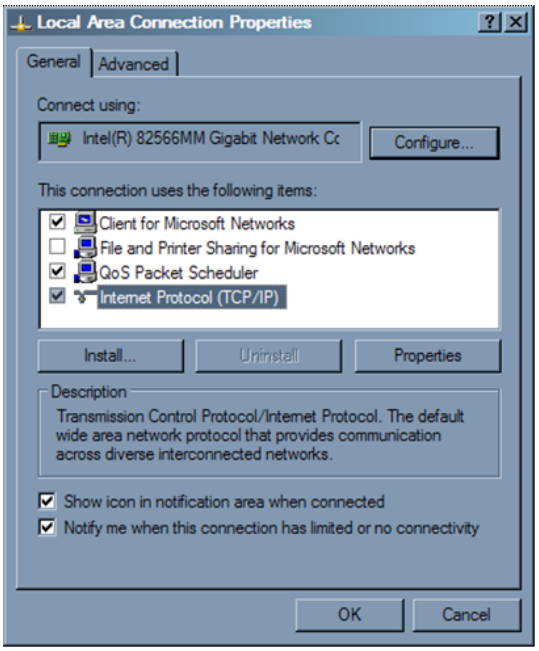


3. Change the **Mode Value** to **Enter IP address manually** so the IP address can be entered manually.

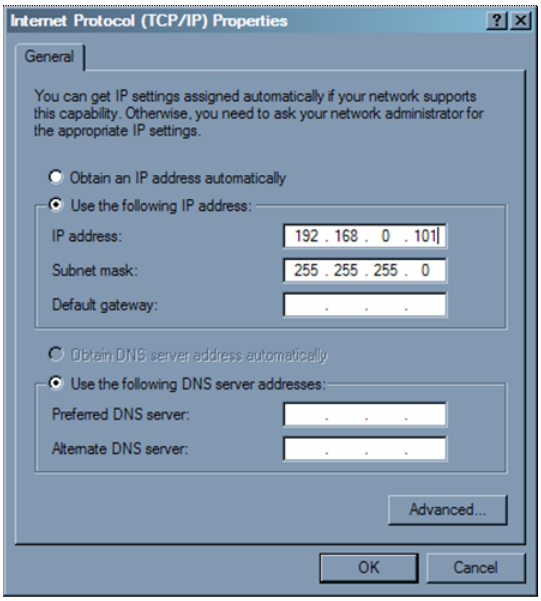
Note: By default the X20 CPU IP address is 10.0.0.2 and the subnet mask is 255.255.255.0.

To configure the PC's Ethernet interface

1. Click **Start > Settings > Network Connections**. The **Local Area Connection Properties** window appears.



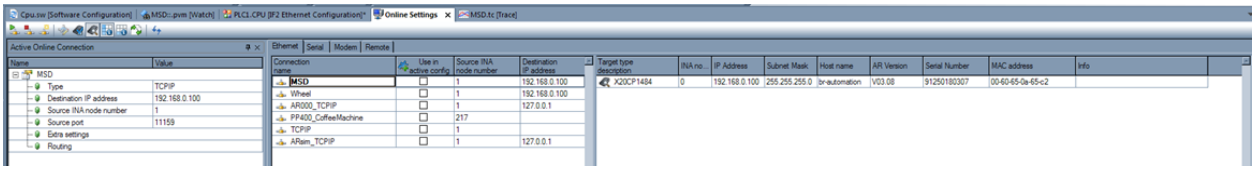
2. Double-click the desired LAN connection. The **Internet Protocol (TCP/IP) Properties** window opens showing the connection status.



3. Enter the **IP address** and **Subnet mask**.
4. Click **OK**.

To configure the online connection

1. From the Main Menu select **Online > Settings...**. The **Online Settings** window appears.



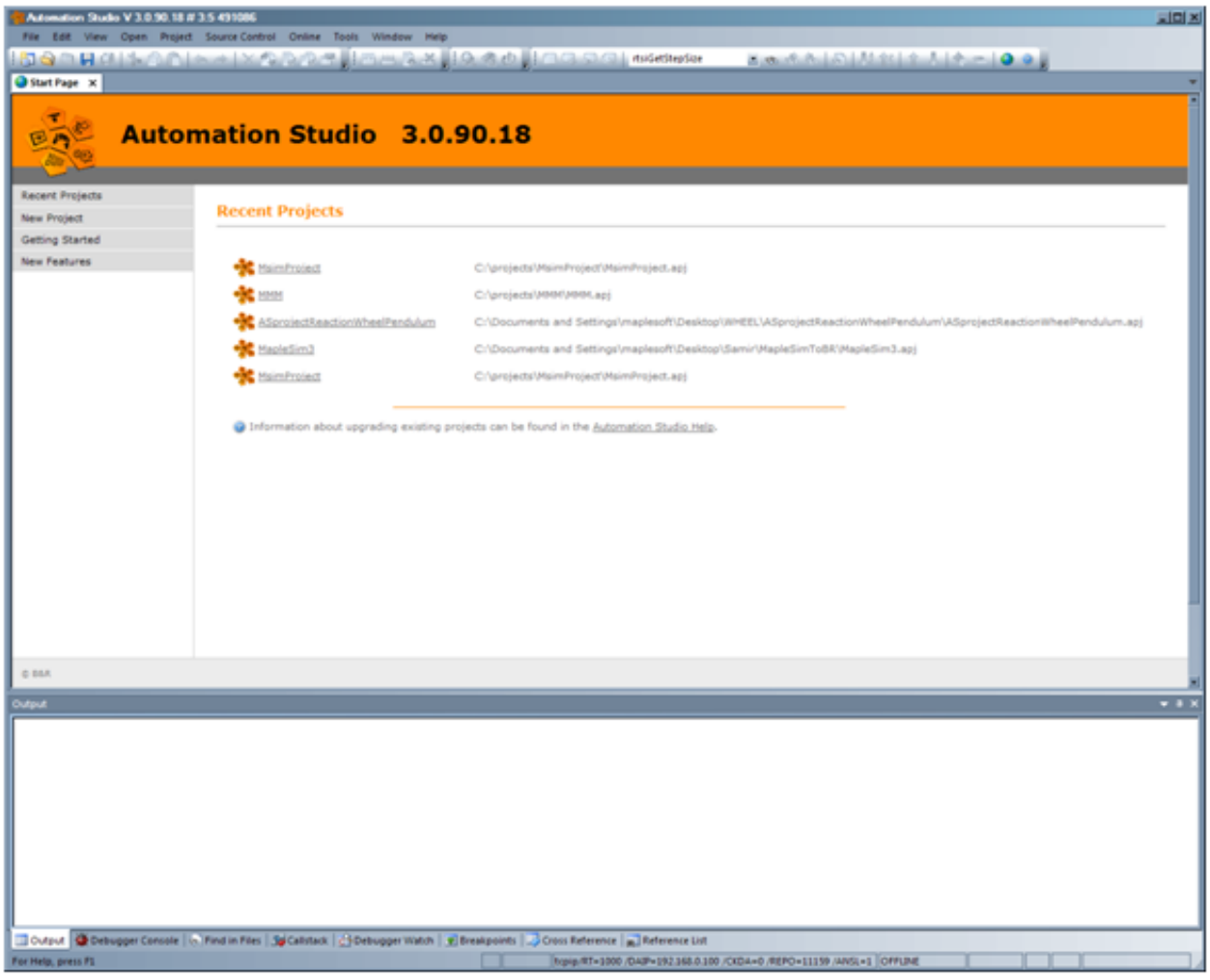
2. Double-click **MSD Destination IP address** and enter the **X20 IP address**. In this example it is **192.168.0.100**.

4.3 Creating a New Project

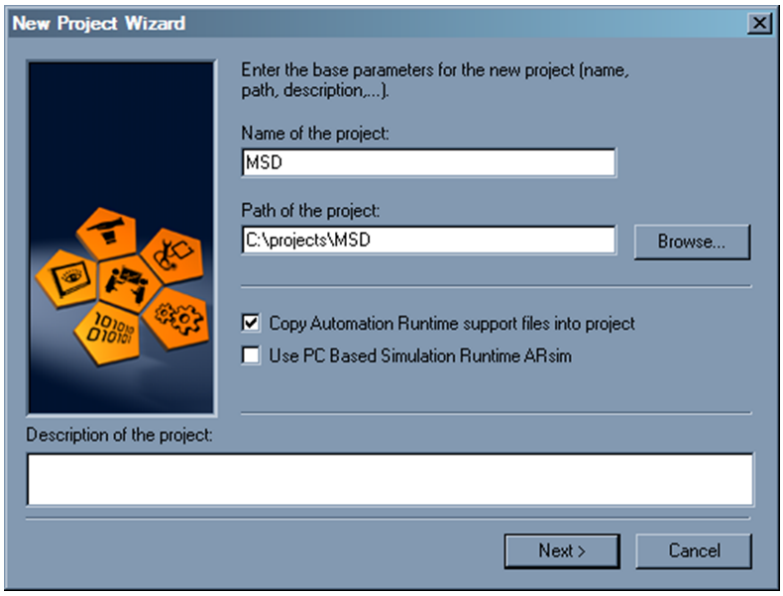
A project contains the application source data (programs, libraries and data objects) and configurations (hardware description, software and I/O mapping data, libraries, and data objects). You must first create a project in order to enter a controller application.

To create a new Automation Studio project file

1. Open **B&R Automation Studio**. The **Main Window** appears.



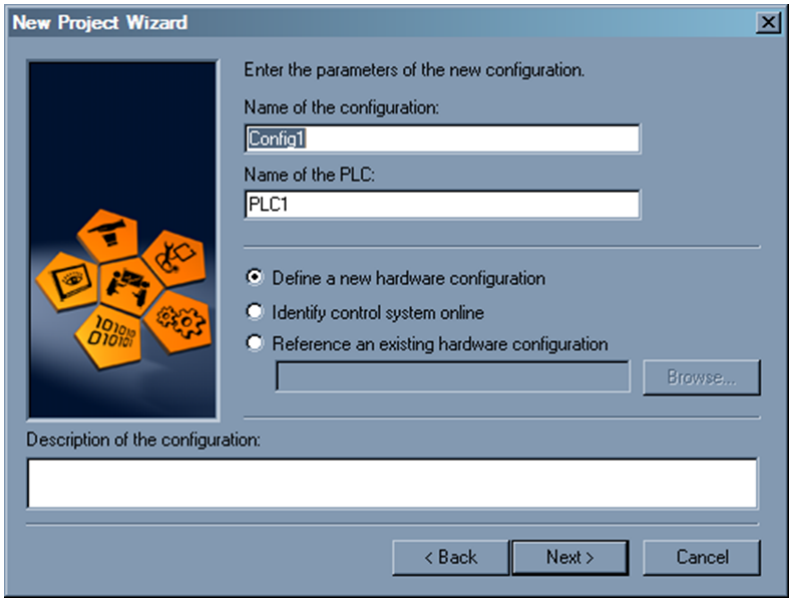
2. From the **File** menu, select **New Project**. The wizard for creating a new project is started.



3. Provide the following basic project data definitions:

Parameter	**Description**
Project name	Project name
Path for the project	Specify the target directory for the generated files. Use the **Browse** button to select a specific directory
Copy Automation Runtime support files into project	Select this option to copy all of the Automation Runtime files to the current project. You can continue to work even if this option is not selected. This setting is used for the next project and must be cleared if the project uses specific target hardware
Use PC Based Simulation Runtime ARsim	Do not select this option
Description of the project	Provide a brief description of the project

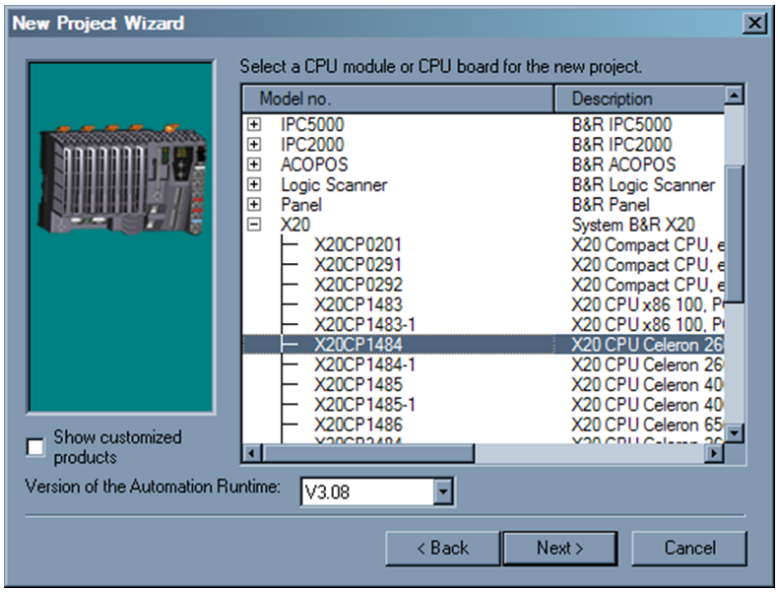
4. Click **Next**. The wizard for defining the new configuration is started.



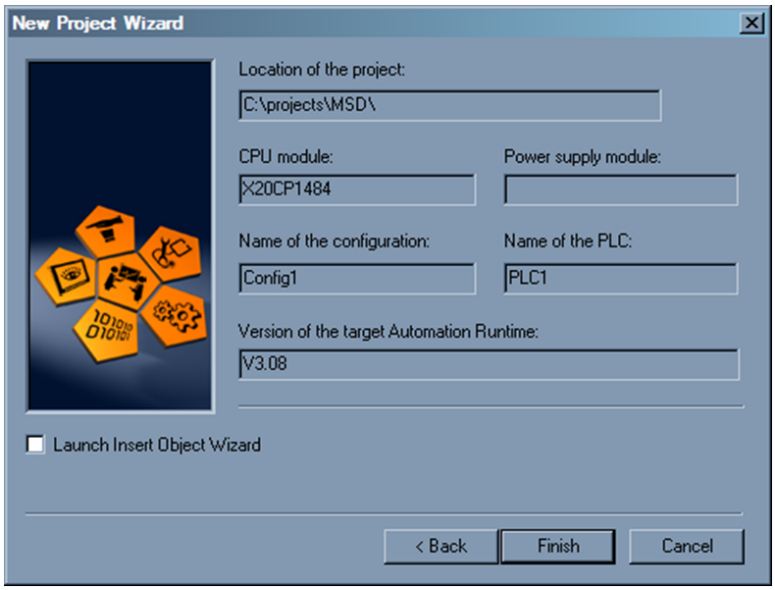
5. Provide the following basic configuration parameters:

Parameter	Description
Name of the configuration	Project a name of the configuration
Name of the PLC	Project a name of the PLC
Define a new hardware configuration	Select this option to provide a new hardware configuration files to the current project. This setting is used for the next project and must be cleared if the project uses specific target hardware
Identify control system online	Do not select this option
Reference an existing hardware configuration	Do not select this option
Description of the project	Provide a brief description of the configuration

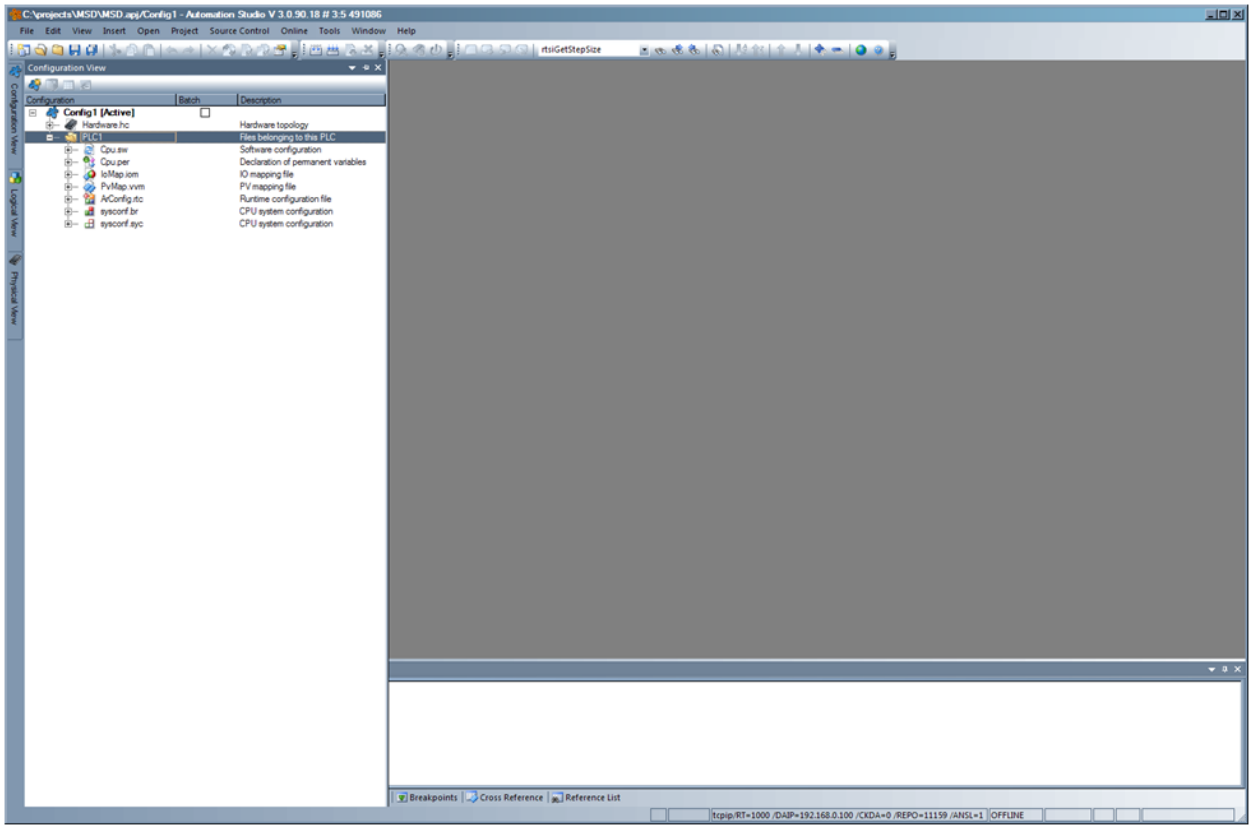
6. Click **Next**. The wizard for selecting a CPU is started.



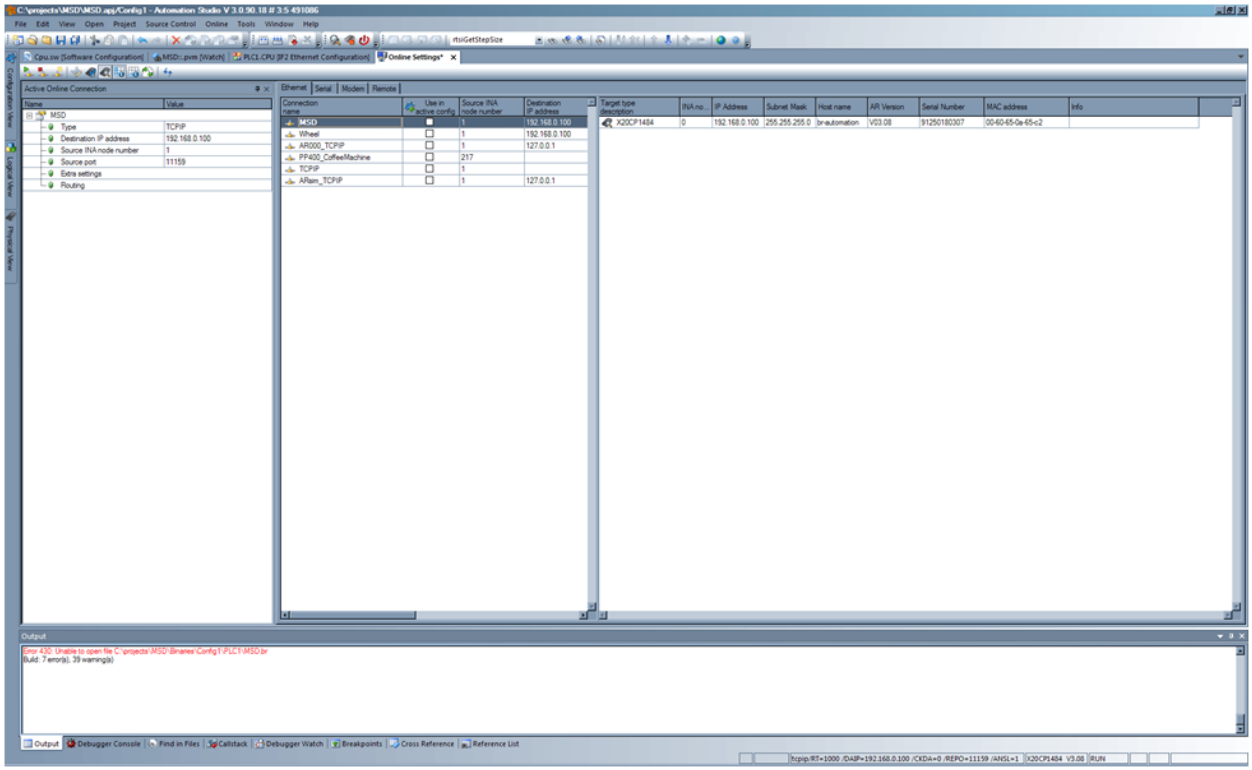
7. Select the appropriate X20 board. Click **Next**. The entered configurations appear.



8. Click **Finish**. The project files are added to the project directory and the **Configuration View** window appears.



9. To see the **Online** settings, from the main **Tool Bar** select **Online > Settings...** . The **Online Settings** window appears displaying the target directory description, ethernet connections and the active online connection.

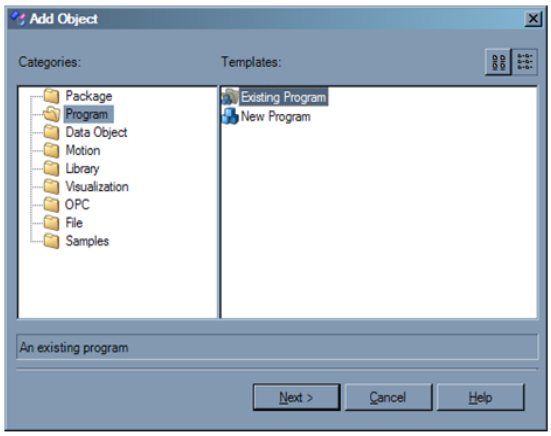


4.4 Adding the Program Object to the CPU

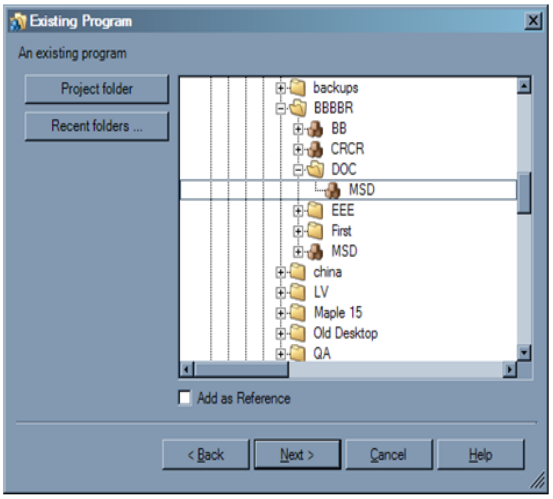
In order to utilize the MapleSim model, the CPU requires the generated C code to be added as a B&R program object.

To add objects to the CPU

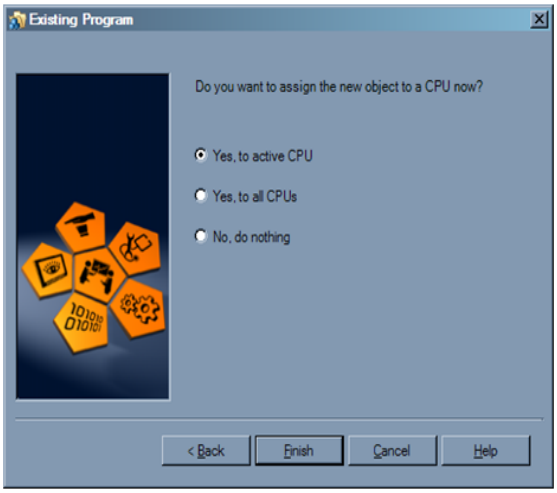
1. Switch to the **Logical View** and from the **Main Tool** bar, select **Insert > Add Object...** (alternatively select the **Logical View**, right-click and select **Add Objects**). The **Add Object** window appears.



2. Select the **Program, Existing Program** folder and click **Next**. The **Existing Program** window appears showing the program list.



3. Select **MSD** and click **Next**. The **Existing Program** window appears showing the object assignment selection.



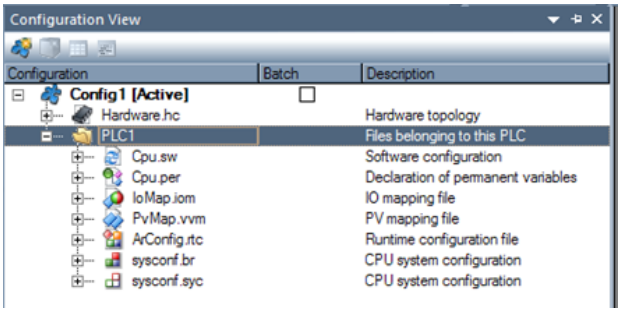
4. Select **Yes, to active CPU** and click **Finish**. The **Project** window appears displaying added objects to the CPU; implementation code (**MSD.c**), local variables (**MSD.var**) and MapleSim model implementation (**cMsimModel.h**).

4.5 Configuring the MSD Program Object on the Active CPU

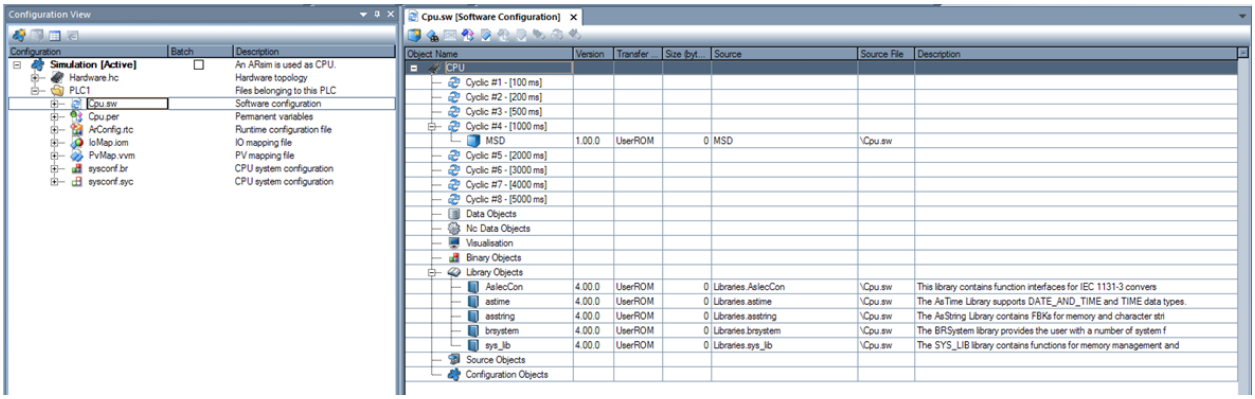
The CPU requires the object program duration to run at the same rate as the baserate specified in the **B&R Template** (see *Baserate (page 5)*).

To configure the MSD program object on the active CPU

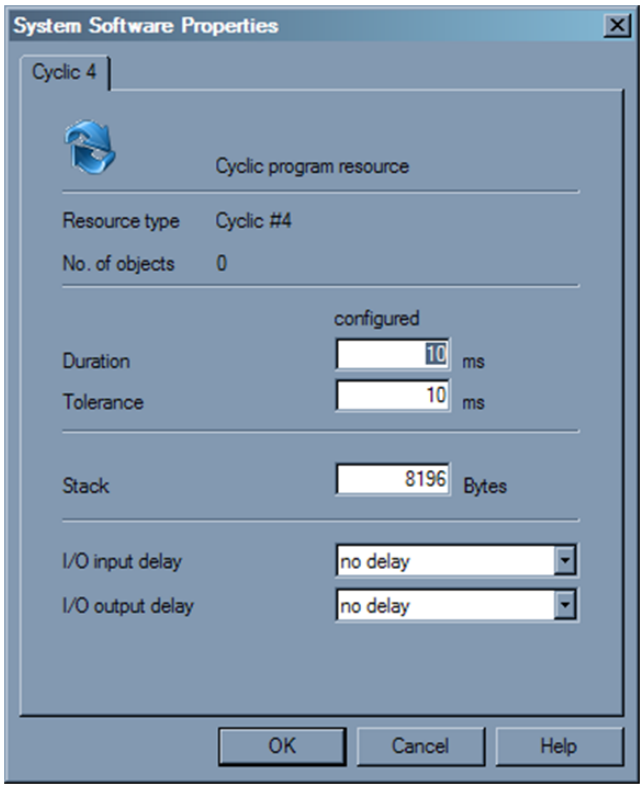
1. Switch to the **Configuration View** and expand **PLC1**.



2. Double-click **CPU.SW**. The **CPU.SW** window appears.



3. Right-click on the **Cyclic #4** object program. The **System Software Properties** window appears.



4. Set the **Cyclic #4** object program **Duration** to 10 ms.

Note: By default the B&R **Cyclic #4** object is 100 ms. Ensure that the **Cyclic #4** object program duration matches the baserate specified in the B&R Template (see *Baserate (page 5)*). For this example the cyclic duration is 10ms. To change the default duration value for the B&R **Cyclic #4** object, right-click on **Cyclic #4 > Properties** and change **Duration** to 10ms and **Tolerance** to 10ms. For addition information click **Help**.

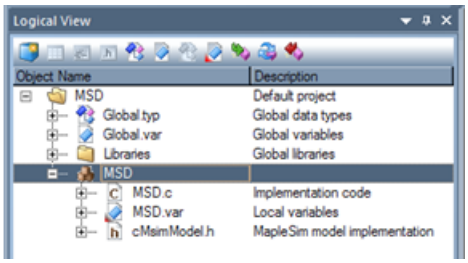
5. Click **OK**. The **Cyclic #4** object program duration value changes to 10 ms and the project is automatically saved.

4.6 Adding Library Objects to the Project

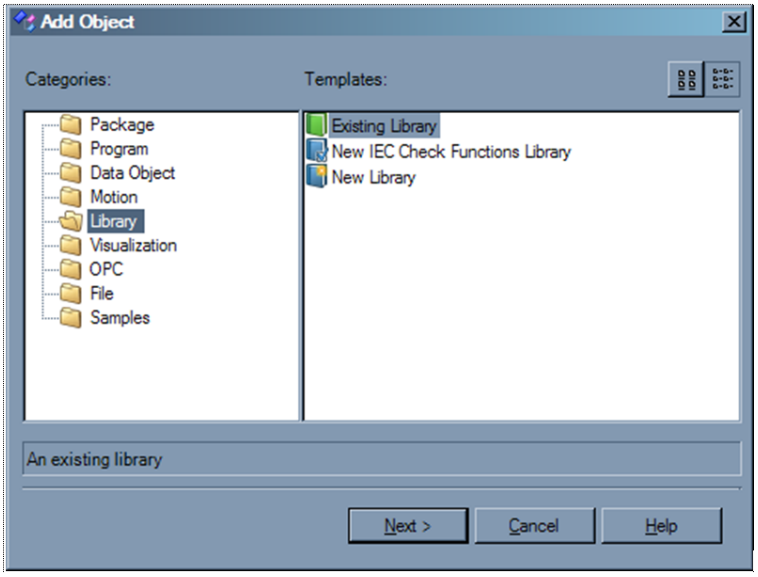
A library is a collection of project specific reusable functions grouped into functional categories.

To add a library object to a project

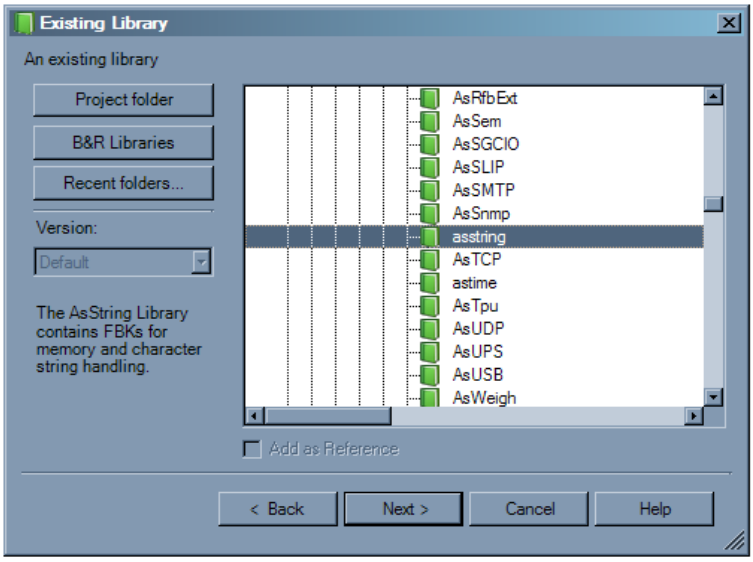
1. Switch to the **Logical View** and select **MSD**.



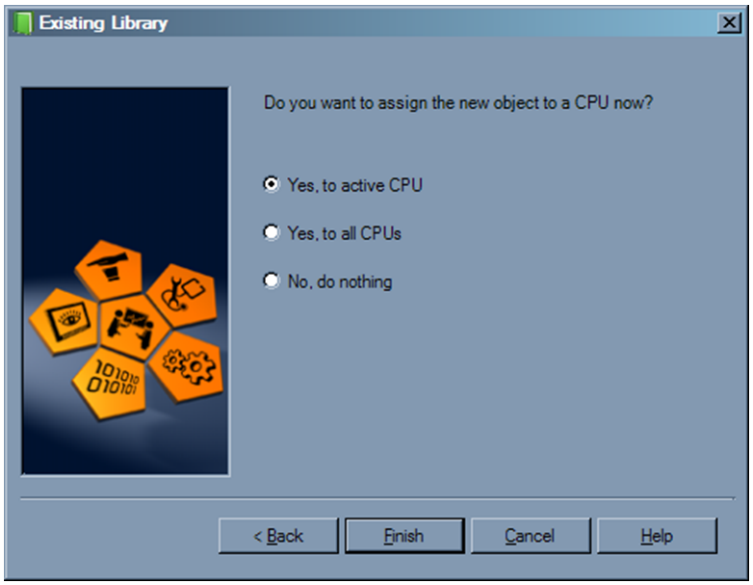
2. From the **Main Toolbar**, select **Insert > Add Object...** (alternatively select the **Logical View**, right-click and select **Add Objects**). The **Add Object** window appears.



3. Select **Library > Existing Library**. Click **Next**. The **Existing Library** window appears.



4. Select the following files:
   - asstring
   - brsystem
   - syslib
5. Click **Next**. The **Existing Library** window appears showing the object assignment selection.



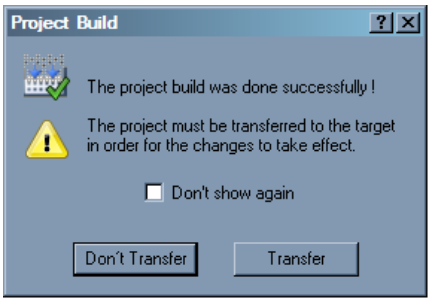
6. Select **Yes, to active CPU** and click **Finish**. The libraries are successfully added to the Project and assigned to the CPU.

4.7 Building the Configuration and Transferring the Project to the Target

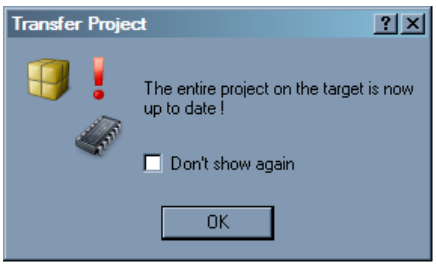
This procedure builds and transfers the project files to the target device.

To build the configuration and transfer the project to the target

1. Switch to the **Configuration View** and expand **PLC1**.
2. From the **Main Tool** bar, select the **Project > Build Configuration** (alternatively press **F7**). When the project is built and the **Project Build** window appears.



3. Click **Transfer**. The project files are transferred to the device and the **Transfer Project** window appears.



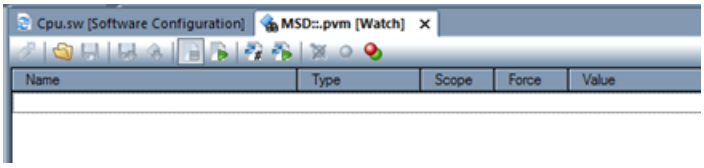
4. Click **OK**.

4.8 Adding a Watch

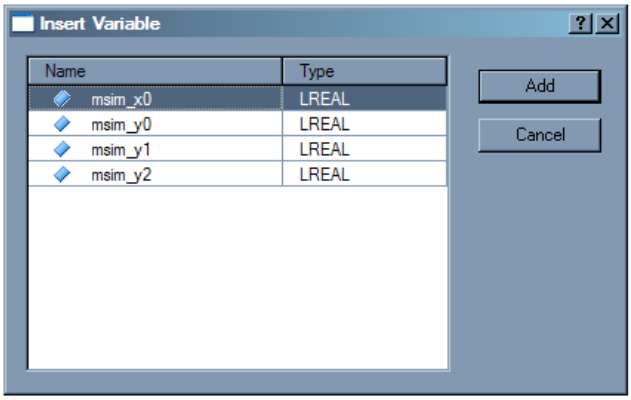
Adding a **Watch** lets you monitor the program variables during the simulation or on the PLC.

To add a watch

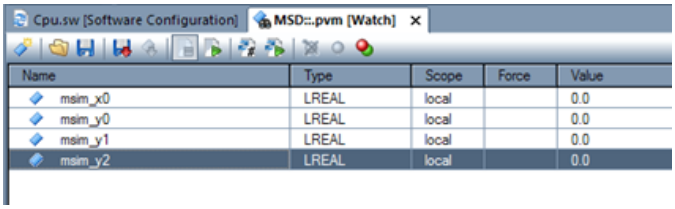
1. Switch to the **Logical View** and right-click on **MSD** program object. A drop-down menu appears.
2. Select **Open > Watch**. The **MSD::pvm (Watch)** window appears.



3. Right-click inside the window and select **Insert Variable...**. The **Insert Variable...** window appears.



4. Select the four variables and click **Add**. The four variables are loaded into the **MSD::pvm (Watch)** window.



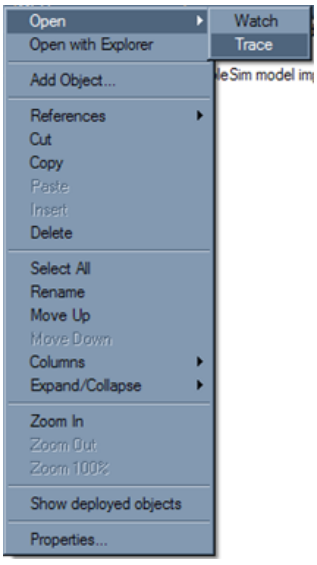
Note: msim_x0 is the input and requires a nonzero value in order to trace the simulation (see *Simulating the Project (page 44)*).

4.9 Adding a Trace

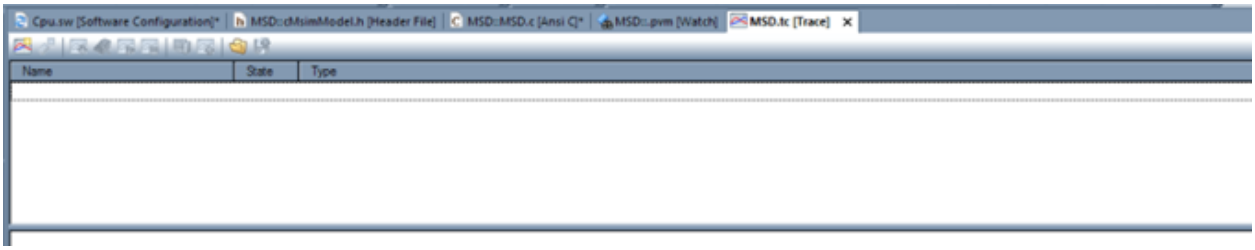
The **Trace** tool records and plots program values.

To add a Trace

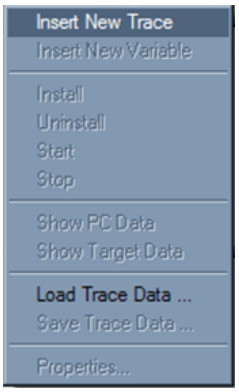
1. Switch to the **Logical View** and right-click on **MSD** program object. A drop-down menu appears.



2. Select **Open > Trace**. The **MSD.tr (Trace)** window appears.

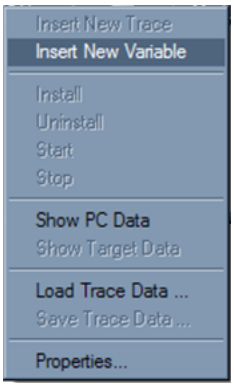


3. Right-click anywhere in the top frame of the **Trace** window. A drop-down menu appears.

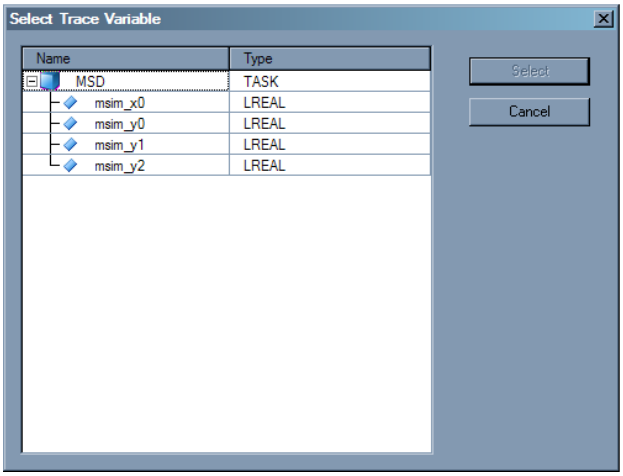


4. Select **Insert New Trace**. The **Trace** window appears with the **TARGET_CONFIGURATION**.

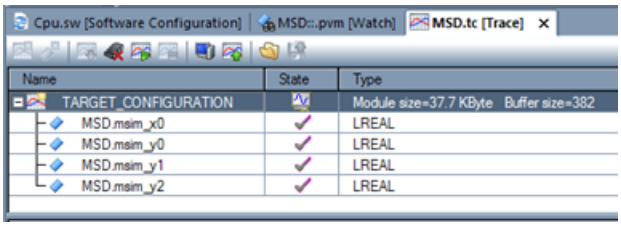
5. Right-click on **TARGET_CONFIGURATION**. A drop-down menu appears.



6. Select **Insert New Variable**. The **Select Trace Variable** window appears.



7. Select all the variables and click **Select**. The **Trace** window appears with the **TARGET_CONFIGURATION** variables.



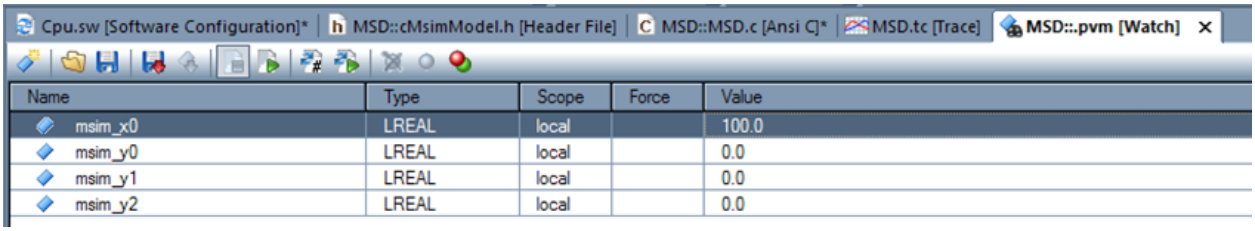
8. Right-click on **TARGET_CONFIGURATION > Install**. The **Trace** is correctly installed on the Target.

4.10 Simulating the Project

At this point the project is ready for simulation.

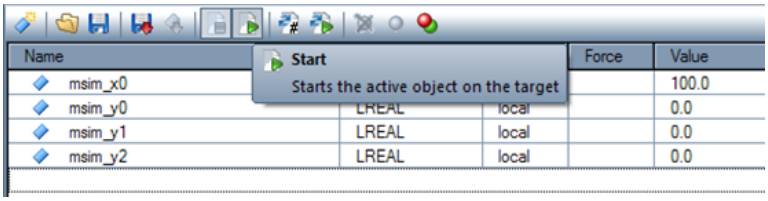
To simulate the project

1. Switch to the **MSD::pvm (Watch)** window and double click on the **msim_x0 Value** cell. Enter an input value of 100.

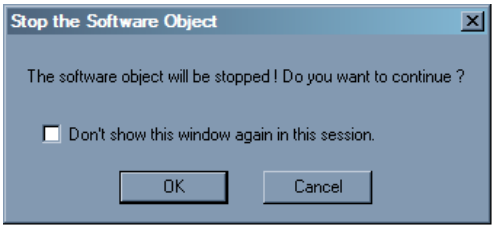


Note: In order for the simulation results to correspond with the MapleSim model use the same input signal values.

2. From the **Toolbar** click **Start** ( ). (Alternatively, right-click inside the **Watch** window and select **Start**.) The simulation starts and the computed values appear.



3. Run the simulation until the values reach a steady state for approximately 30 seconds.
4. To stop the simulation click **Stop** ( ). The **Stop the Software Object** window appears.



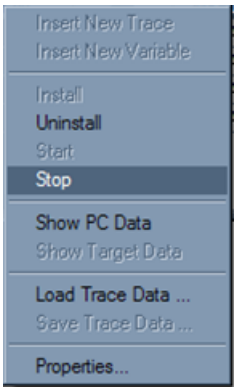
5. Click **OK**.

4.11 Viewing the Simulation Results

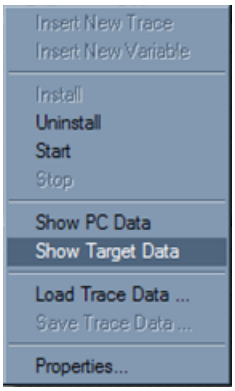
After simulation the results can be viewed.

To view the project

1. Switch to the **MSD.tc (Trace)** window and right-click on **TARGET_CONFIGURATION**. A drop-down menu appears.



2. Click **Stop** to stop the trace.
3. Right-click on **TARGET_CONFIGURATION**. A drop-down menu appears.



4. Select **Show Target Data**. The real-time results for each of the target configuration variables appear.


Cpu.sw [Software Configuration]
MSD::pvm [Watch]
MSD.tc [Trace]
Name
State
Type
TARGET_CONFIGURATION
Module size=37.7 KByte Buffer size=382
MSD.msim_x0
MSD.msim_y0
MSD.msim_y1
MSD.msim_y2
LREAL
TARGET_DATA
UnitX
UnitY
x-pos. 0.000000
y-pos. 0.000000
x-diff. 0.000000
y-diff. 0.000000

Index