Getting Started with the MapleSim FMI Connector

Copyright © Maplesoft, a division of Waterloo Maple Inc.
2021

Getting Started with the MapleSim FMI Connector

Copyright

Maplesoft, Maple, and MapleSim are all trademarks of Waterloo Maple Inc.

© Maplesoft, a division of Waterloo Maple Inc. 2012-2021. All rights reserved.
No part of this book may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means — electronic, mechanical, photocopying, recording, or otherwise. Information in this document is subject to change without notice and does not represent a commitment on the part of the vendor. The software described in this document is furnished under a license agreement and may be used or copied only in accordance with the agreement. It is against the law to copy the software on any medium except as specifically allowed in the agreement.

Linux is a registered trademark of Linus Torvalds.

Macintosh is a registered trademark of Apple Computer, Inc.

Windows is a registered trademark of Microsoft Corporation.

All other trademarks are the property of their respective owners.

This document was produced using Maple and DocBook.

Contents

Introduction iv
1 Getting Started 1
1.1 FMI Code Generation Steps 1
1.2 Opening an FMI app 1
1.3 Using the FMU Generation App 1
Step 1: Subsystem Selection 2
Step 2: Inputs, Outputs, and Parameters 2
Step 3: Export Options 3
Step 4: Info Options 5
Step 5: Export 6
Step 6: View Code 6
1.4 Note on Ports for Derivatives 7
1.5 Enabling Extended Visualization Data 7
1.6 Viewing Examples 7
1.7 Example: RLC Circuit Model 8
2 Example: Exporting a Model as an FMU File 9
2.1 Preparing a Model for Export 9
Converting the Model to a Subsystem 9
Defining Subsystem Inputs and Outputs 10
2.2 Defining and Assigning Subsystem Parameters 13
2.3 Exporting Your Model Using the FMU File Generation App 14
Index 16

Introduction

The MapleSim™ FMI Connector and FMI Connector package provides all of the tools you need to prepare and export your dynamic systems models into an FMU (Functional Mock-up Unit) archive file.

You can create a model in MapleSim, simplify it in Maple™ by using an extensive range of analytical tools, and then generate FMU executables that you can incorporate into your toolchain. Using MapleSim Insight (a separate product), you can see live simulation results and 3-D visualizations of your models that run in real-time.

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

Requirements

Windows users need to install a third-party 'zip' utility which must be included in the PATH environment variable in order to successfully generate an FMU.

On 64-bit Windows, some features require Microsoft Visual Studio.

For details on supported platforms and a complete list of system requirements, visit the Maplesoft System Requirements website at **http://www.maplesoft.com/products/system_requirements.aspx**.

For installation instructions, see the **Install.html** file on the product disc or the website at **http://www.maplesoft.com/documentation_center/installation_instructions.aspx**.

Distribution of an FMU

Under the terms and conditions of the Maplesoft End User License Agreement, you have the right to use an FMU created with the MapleSim FMI Connector within your organization. **If you wish to distribute this FMU outside your organization, you must first obtain a FMU Commercial Distribution License from Maplesoft.** For more information, refer to the **FMU Distribution** help page.

Adding External Libraries to Your Search Path

You can export a model that uses an external library as part of the model to an FMU archive. In order to do this, you **first** need to add the directory that contains the external library file (that is, the .dll or .so file) to your search path. This involves appending the external library directory to either your PATH environment variable (for Windows®) or your LD_LIBRARY_PATH environment variable (for Linux® and Macintosh®).

To add an external library directory to your search path

1. Determine the location of the external library directory.

   Note: This is the directory that contains the .dll file (Windows) or the .so file (Linux or Macintosh) that is used in your model.

2. Add the library directory found in step 1 to the appropriate environment variable for your operating system.
   - For Windows, add the library directory to your PATH environment variable.
   - For Linux and Macintosh, add the library directory to your LD_LIBRARY_PATH environment variable.

   Consult the help for your operating system for instructions on how to edit these environment variables.

3. Restart your computer.

1 Getting Started

1.1 FMI Code Generation Steps

This chapter describes how to use the FMI app and in the *Example: RLC Circuit Model (page 8)* section of this chapter, a step by step procedural example shows you how to create an FMU archive file. The **FMU Generation** app consists of the following steps to generate C code:

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

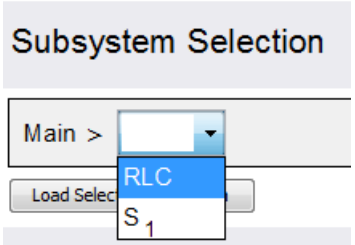
The FMI Connector package

The FMI Connector package is a collection of procedures for manually generating and compiling FMI code from MapleSim models. For information about the FMI Connector package, in Maple, refer to the **FMIConnector** help page.

1.2 Opening an FMI app

To open the FMU Generation app

1. Select the **Add Apps or Templates** tab ( ).
2. Double-click the **FMU Generation** entry in the **Apps** palette. The **Analysis** window opens with **FMU Component Code Generation** app loaded in the **Apps** tab.
3. In the **Subsystem Selection** section, select a subsystem from the drop down list.



1.3 Using the FMU Generation App

The MapleSim FMI Connector provides an **FMU Generation** app for manipulating and exporting MapleSim subsystems as FMU archive files.

With this app, 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 FMU component and library code. You can assign model equations to a variable, group inputs and outputs, and define additional input and output ports for variables.

Note: FMU component 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 MapleSim. To access these, from the **Help** menu, select **Examples > FMI Connector Examples**.

Step 1: Subsystem Selection

This part of the app identifies the subsystem modeling components that you want to export as a block component.

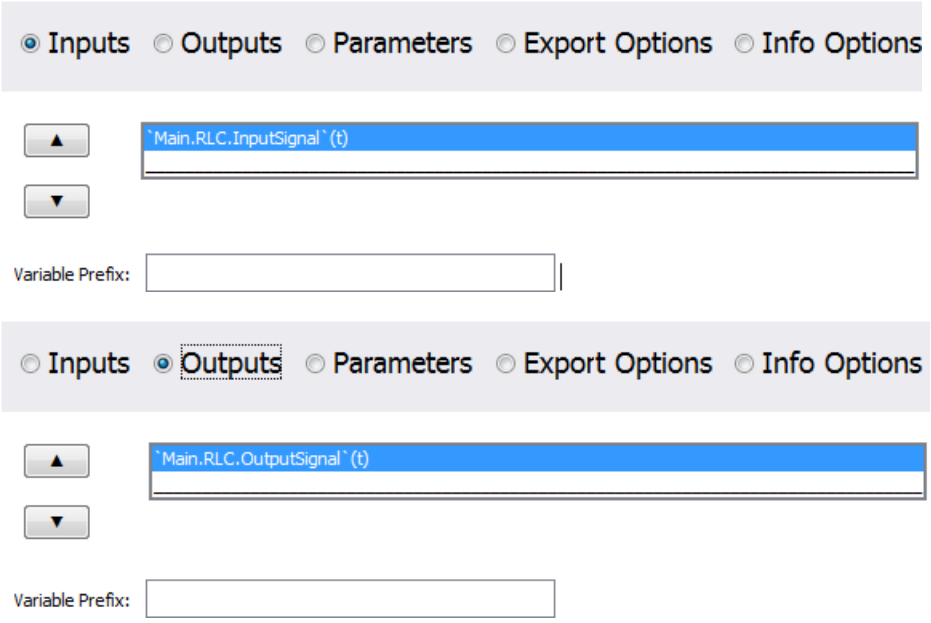
To connect a subsystem to modeling components outside of its boundary, you add subsystem ports to your model. A subsystem port is an extension of a component port in your subsystem. The resulting signals can then be directed as inputs and outputs for the FMU archive files. By creating a subsystem you improve the visual layout of a system in the model workspace and also prepare the model for export. The example in Chapter 2 shows you how to group all of the components into a subsystem.

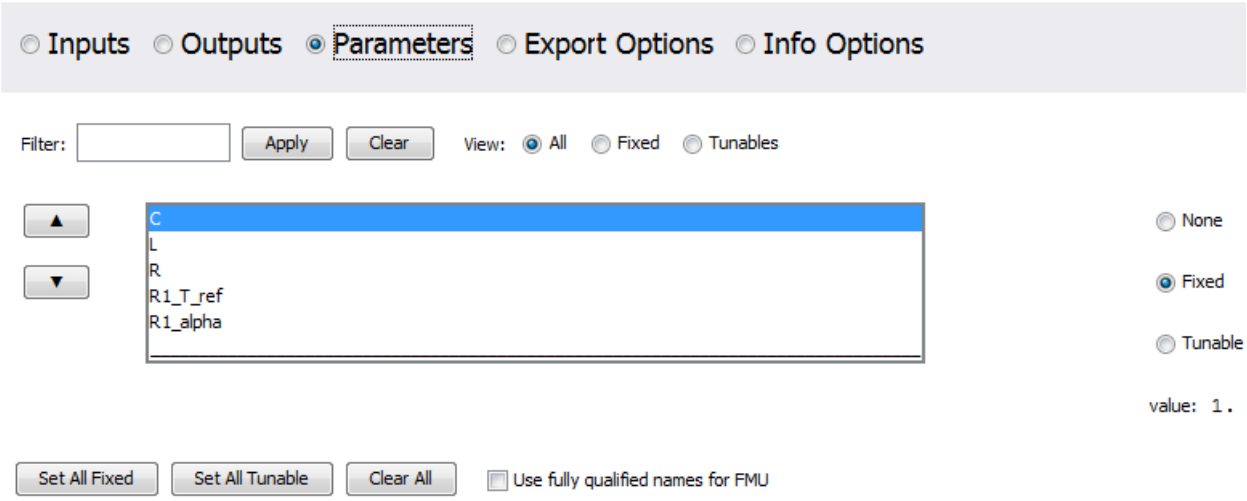
You can select which subsystems from your model you want to export to an FMU archive file. After selecting a subsystem, click **Load Selected Subsystem**. All defined input and output ports are loaded.

Step 2: Inputs, Outputs, and Parameters

The **Inputs**, **Outputs**, and **Parameters** sections let 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. You can specify prefixes for both input and output port variables. The prefix for the input (output) port variables will be applied to all the variables in the input ports (outputs) table.





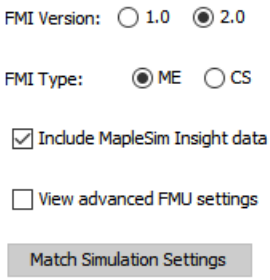
Note: If the parameters are not marked for export they will be numerically substituted. Parameters marked as tunable will also be marked for export by default. However, tunable parameters will be ignored if FMI Version and Environment (see Step 3: Export Options) are anything other than FMI 2.0 and Co-Simulation. To use fully qualified parameter names for the generated FMU, check the corresponding check box; otherwise, short parameter names will be used by default.

Step 3: Export Options

Select **Export Options** to specify the advanced options for the code generation process. Default settings are provided for both Model Exchange and Co-Simulation export options.

FMI Version and Environment

Select the FMI version and environment for your code. You can choose between **FMI 1.0** and **FMI 2.0** for the version, and between Model Exchange and Co-Simulation for the environment.



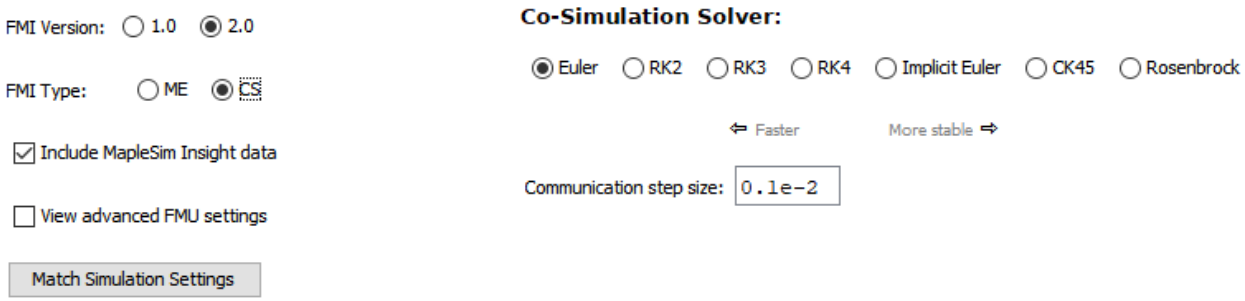
Select **Include MapleSim Insight data** to include the necessary data for the FMU to be run with MapleSim Insight (available as a separate product).

Select **View advanced FMU settings** to see Contraint Handling, Event Handling, and Code Generation options.

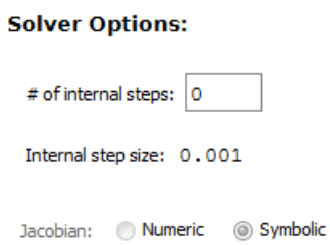
To match the MapleSim model's current simulation settings, click the **Match Simulation Settings** button. (Note that you can revert to the default settings for Model Exchange and Co-Simulation using the refresh button at the top of the app window.)

Cosimulation Solver Settings

For cosimulation, the Co-Simulation Solver can be set to Euler, RK2, RK3, RK4, Implicit Euler, CK45, or Rosenbrock. The communication step size can also be specified.

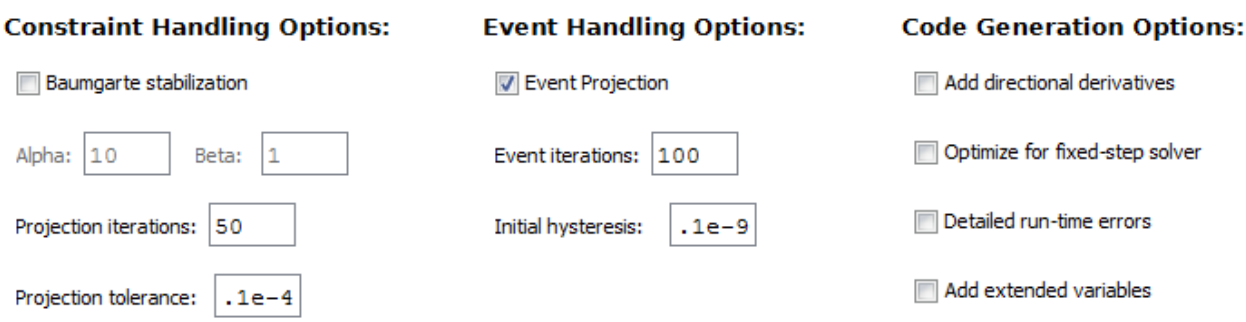


Specify Solver Options for the chosen solver.



Jacobian: In the case of the Implicit Euler solver, specify either numeric or symbolic formulation for the system Jacobian.

Advanced FMU Settings



The Constraint Handling Options specifies whether the constraints are satisfied in a DAE system by using constraint projection in the generated FMU archive file. 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.

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 stabilization: Apply the Baumgarte constraint stabilization.

Alpha: Set the derivative gain for Baumgarte constraint stabilization.

Beta: Set the proportional gain for Baumgarte constraint stabilization.

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

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

The Event Handling Options specifies whether the events are satisfied in a DAE system by using event projection in the generated FMU archive file. 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 **Event iterations** to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

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

Select **Event projection** to perform event projection 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.

Note: Currently, if the model has events, they are handled using the event handling functions in the generated Msim-Model.c file, and not the FMI provided Event Handling routines.

The Code Generation Options control details of the generated code and run-time errors.

Add directional derivatives: Select the check box to generate directional derivatives code.

Optimize for fixed-step solver: Specifies if the generated code should be optimized for use with fixed-step solvers.

Detailed run-time errors: Report run-time errors in detail.

Add extended variables: Choose between minimal and extended variable tracking during the simulation.

Step 4: Info Options

In the generated FMU archive file, model information (for example, variable names, units and initial values) are stored in a text file in XML format. Additional information about the model, such as the model author, description, and version, can also be included in this file.

Author: Name and organization of the model author (for example, J. Smith, Maplesoft http://www.maplesoft.com).

Description: Brief description of the model (for example, Model of a lithium-ion battery).

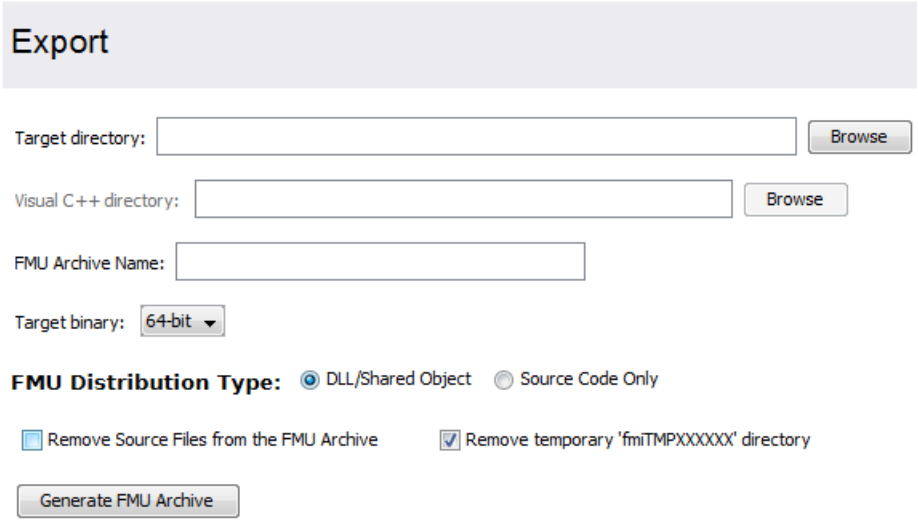
Version: Model version or FMU version.

Copyright: Intellectual property copyright (for example, (C) Maplesoft 2019).

License: Intellectual property licensing (for example, Proprietary, Public Domain, or BSD License).

Step 5: Export

Generating the C code creates temporary files for viewing purposes.



Specify the locations for the **Target Directory** and **Visual C++ Directory**.

Note: The Visual C++ Directory is not required for Linux and Mac® platforms. For Windows platforms, a typical location for the Visual C++ Directory is: **C:\Program Files (x86)\Microsoft Visual Studio\version\WDExpress\VC**. If necessary, navigate to the correct Visual C++ Directory for your platform using the **Browse** button.

Provide a name for the generated FMU archive in **FMU Archive Name**.

If required, set the binary target to **32-bit** or **64-bit**.

Select **Remove Source Files from the FMU Archive:** to remove source files after code generation.

Select **Remove temporary `fmiTMPXXXXXX`directory** to remove temporary files after code generation.

To generate an FMU archive click **Generate FMU Archive**. The FMU is generated.

Note: If your model contains an external library, you must add the directory that contains the external library to your search path. See *Adding External Libraries to Your Search Path (page iv)* for instructions on how to do this.

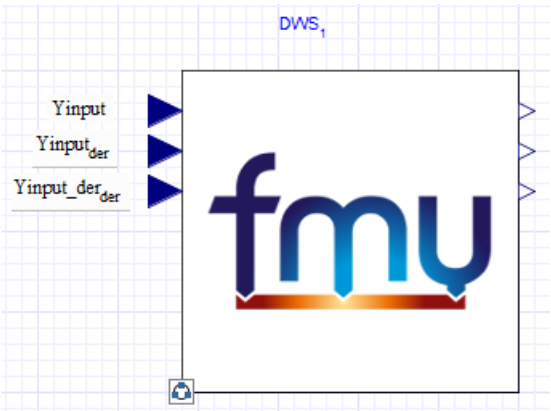
Step 6: View Code

After the C code is generated, the **FMI C Code** and **MsimModel.c** components can be viewed.

1.4 Note on Ports for Derivatives

When the derivative of an input is required in the FMU, the FMU Generation app adds additional ports to the FMU that allow the user to directly define these. This avoids the numerical differentiation of these values, which can be unstable at the start of a simulation.

If ports for derivatives are created, their names will be of the form $VariableName_{der}$ (first derivative) or $VariableName_{der_{der}}$ (second derivative).



1.5 Enabling Extended Visualization Data

You can enable the generation of extended visualization data in your FMU. Extended visualization data is used to produce FMU simulation results (that is, plots and 3-D animations) that can be imported back into MapleSim. To do this, you need to set two parameters in the FMU generated by the MapleSim FMI Connector using a third-party FMU tool that allows you to view and edit parameters.

To enable extended visualization data, set the following FMU parameters:

- **binfilename:** Enter a file name with a *.bin* file extension for the extended visualization data file. The extended visualization data file is necessary in order to import the FMU simulation results into MapleSim. Enter *DISABLED* if you do not want to generate extended visualization data.
- **minsamplestep:** Enter the minimum sample step, in seconds, for the extended visualization data in the *.bin* file. The default value is 2.5e-2 *s* (that is 1/40 of a second). This sample step is used for both plots and 3-D animations (for 3-D models). For example, a sample step of 1/40 of a second generates plot points every 1/40 of a second and gives you a frame rate of 40 frames per second in your 3-D animation. Smaller values for the minimum step size give smoother plots and animations, but result in much larger *.bin* files.

After you execute your FMU, the extended visualization data file is created in the same directory that your FMU is in. See **MapleSim > Using MapleSim > Simulating a Model > Importing FMU or Simulink(R) S-function Results** in the MapleSim help system for information on how to import these results into MapleSim.

1.6 Viewing Examples

Within MapleSim there are many examples for you to view.

To view an example:

1. From the **Help** menu, select the **Examples > FMI Connector Examples** menu, and then 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 **Attached Files** tab, expand **Documents**. You can open any of these documents by right-clicking (**Control**-clicking for Mac) its entry in the list and clicking **View**. After you add a template to a model, it will be available from this list.

1.7 Example: RLC Circuit Model

In this example, you will generate an FMU archive file using an RLC circuit model created in MapleSim.

To generate an FMU archive file:

1. From the **Help** menu, select **Examples > FMI Connector Examples**, and then select the **RLC Parallel Circuit** example.
2. Select the **Add Apps or Templates** tab ( ).
3. Double-click on the **FMU Generation** entry in the **Apps** palette. The **Analysis** window opens with the **FMU Component Code Generation** app loaded in the **Apps** tab.
4. Select the **RLC** subsystem from the drop-down list in the **Subsystem Selection** section. This menu displays all of the subsystems and components in your MapleSim model.
5. Click **Load Selected Subsystem**. All of the app 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.
6. In the **Export** section of the app, specify the target directory, the Visual C++ directory, and the FMU archive name.
7. Click **Generate FMU Archive.** The .fmu zip file is created and saved in the target directory.

Note: Generating a block may require a few minutes.

2 Example: Exporting a Model as an FMU File

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 an FMI file.

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 FMI file Generation app.

To open the slider-crank mechanism example:

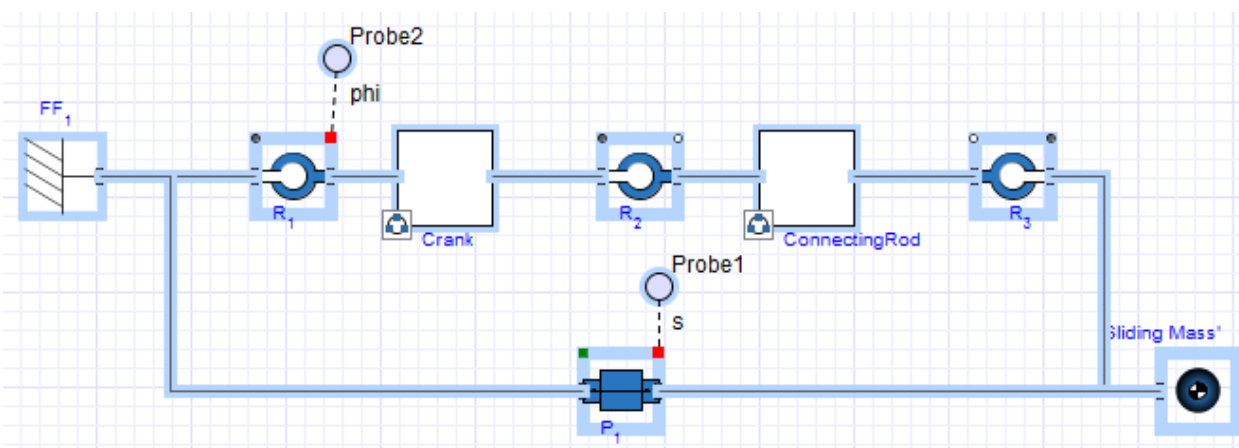
1. In MapleSim, click the **Help** menu item.
2. Select **Examples > User's Guide Examples > Chapter 6**, and then select **Planar Slider-Crank Mechanism**.

Converting the Model to a Subsystem

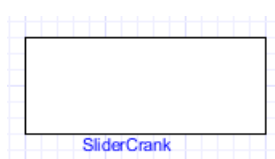
By converting your entire model or part of your model into a subsystem, you identify which parts of the model you want to export. Since the FMI connector supports data signals and properties on acausal connectors such as mechanical flanges and electrical pin, in this example, you will group all of the components into a subsystem and use an acausal input into the subsystem.

To create a subsystem:

1. Draw a box around all of the components in the model by dragging your mouse over them.



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.



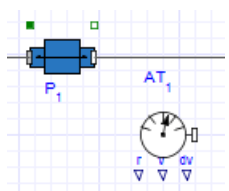
Defining Subsystem Inputs and Outputs

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. In this example, since FMI supports causal and acausal ports, a torque driver is used as an input to the subsystem in MapleSim and for the FMU file.

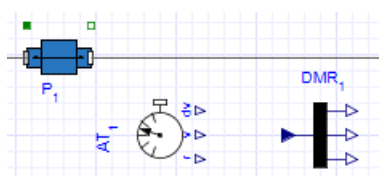
You will convert the displacements of the slider and the joint between the crank and connecting rod to output signals. The input will be a torque driver that is applied to the revolute joint that represents the crank shaft.

To create a subsystem output port:

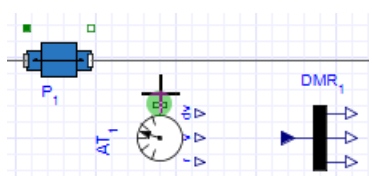
1. Double-click the subsystem block to view its contents. The broken line surrounding the components indicates the subsystem boundary, which can be resized by clicking and dragging its sizing handles.
2. Delete the probes that are attached to the model.
3. In the **Library Components** tab ( ) on the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



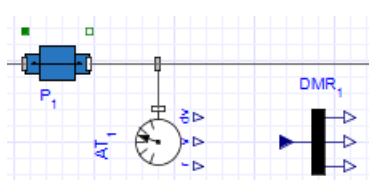
5. Right-click (**Control**-click for Mac®) the **Absolute Translation** component and select **Rotate Counterclockwise**.
6. From the **Signal Blocks** > **Routing** > **Demultiplexers** menu, drag a **Real Demultiplexer** component to the **Model Workspace** and place it to the right of the **Absolute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



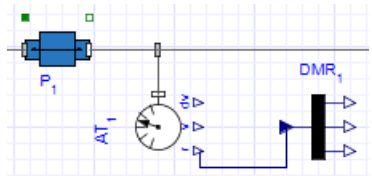
7. To connect the **Absolute Translation** component to the model, click the frame_b connector. The frame is highlighted in green when you hover your pointer over it.



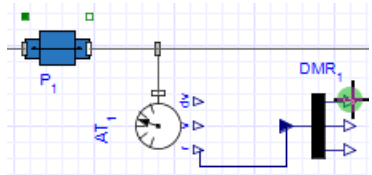
8. Draw a vertical line and click the connection line directly above the component. The sensor is connected to the rest of the diagram.



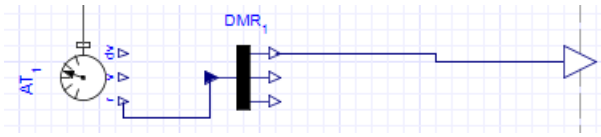
9. In the same way, connect the **r** output port (*TMOutputP*) of the **Absolute Translation** component to the input port of the demultiplexer. This is the displacement signal from the sensor in x, y, and z coordinates. Since the slider only moves along the x axis, the first coordinate must be an output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



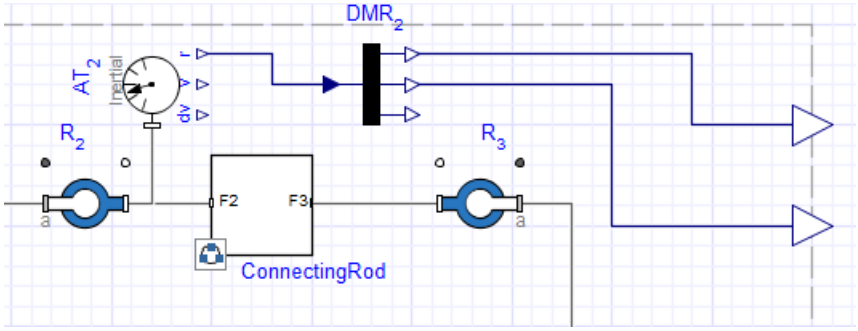
10. Hover your pointer over the first demultiplexer port and click your mouse button once.



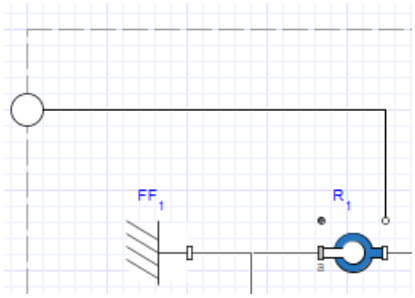
11. Drag your pointer to the subsystem boundary and then click the boundary once. A real output port is added to your subsystem.



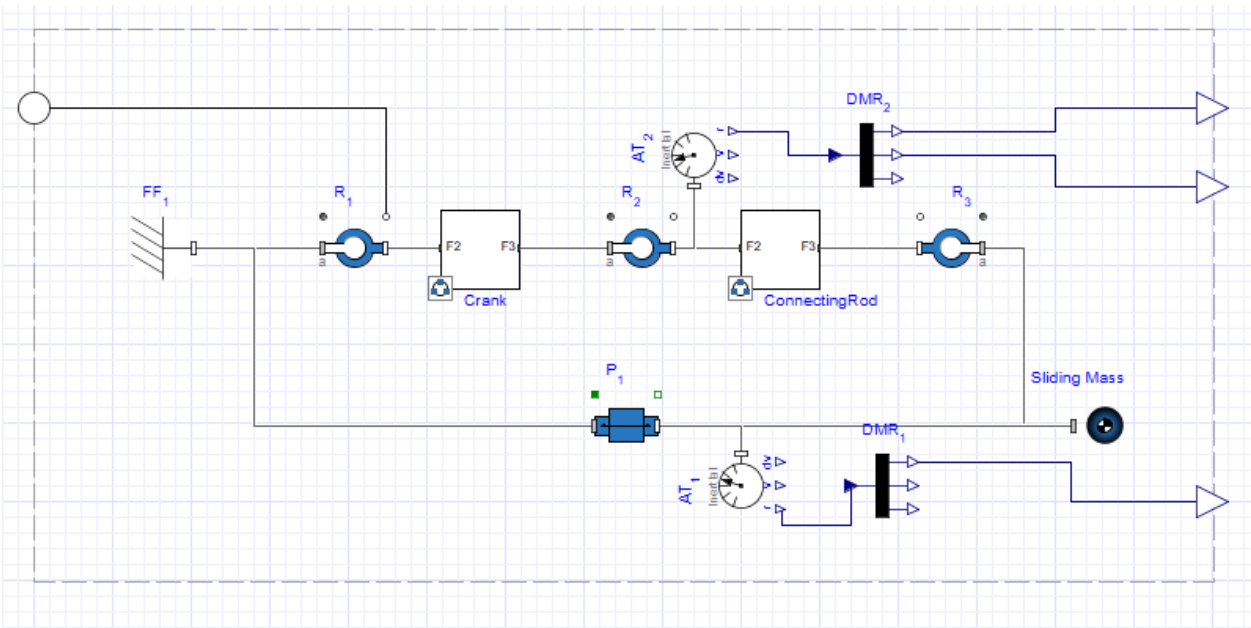
12. Add another **Absolute Translation** component above the **Connecting Rod** subsystem.
13. Right-click (**Control**-click for Mac) the **Absolute Translation** component and select **Flip Vertical**. Right-click the **Absolute Translation** component again and select **Rotate Clockwise**.
14. Add a **Real 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 are now ready to add a real input port to your subsystem to control the torque on the crank shaft.



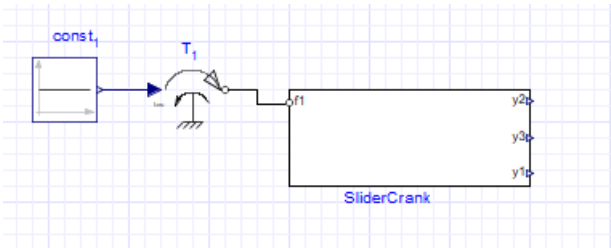
15. Click on the white flange of the leftmost **Revolute Joint** and drag the pointer to the left boundary. This will act as the acausal input connection to the subsystem.



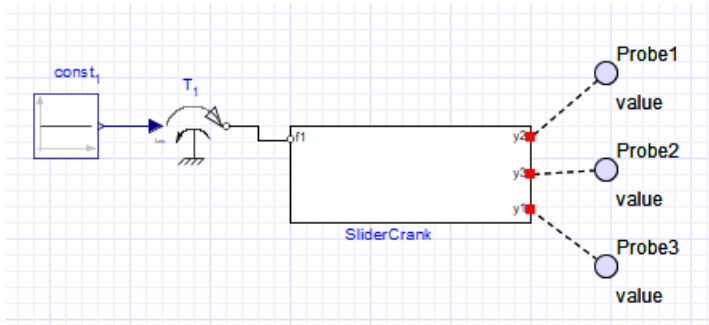
The complete subsystem appears below.



16. Click **Main** ( ) in the **Model Workspace** toolbar to browse to the top level of the model.
17. From the **1-D Mechanical** > **Rotational** > **Torque Drivers** menu, add a **Torque** component to the **Model Workspace** and place it next to the subsystem.
18. From the **Signal blocks>Common** menu, add a **Constant** signal component to the **Model Workspace** and next to the Torque component and connect it to the Torque component.



19. Click **Attach Probe** ( ) above the **Model Workspace** toolbar and then click the top output port of the **SliderCrank** subsystem.
20. In the **Model Workspace**, click the probe once to position it.
21. 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.

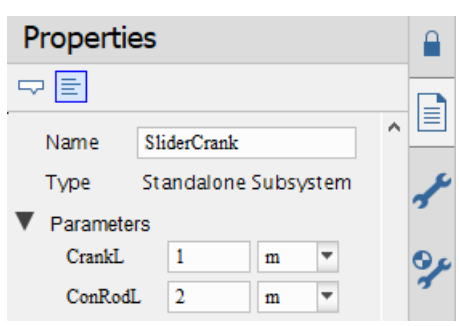
To edit parameters

1. Double-click the **SliderCrank** component on the Model Workspace to see the detailed view of the **SliderCrank** subsystem, and then click **Parameters** ( ) in the **Model Workspace** toolbar. The parameter editor appears.
2. In the first **Name** field, type **CrankL** and press **Enter**.
3. Change the **Type** to Length from the drop-down menu.
4. Specify a default value of **1m** and enter **Crank length** as the description.
5. In the second row of the table, define a parameter called **ConRodL** and press **Enter**.
6. Change the **Type** to Length from the drop-down menu.
7. Specify a default value of **2m** and enter **Connecting Rod Length** as the description.

Standalone Subsystem default settings

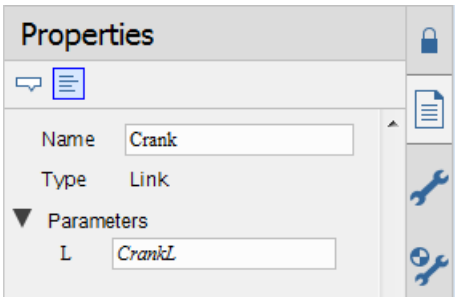
Name	Type	Default Value	Default Units	Description
CrankL	Length [m]	1	m	Crank Length
ConRodL	Length [m]	2	m	Connecting Rod Length

8. Click **Diagram View** ( ) to switch to the diagram view, and then click **Main** ( ).
9. Select the **SliderCrank** subsystem. The parameters are defined in the **Properties** tab ( ).

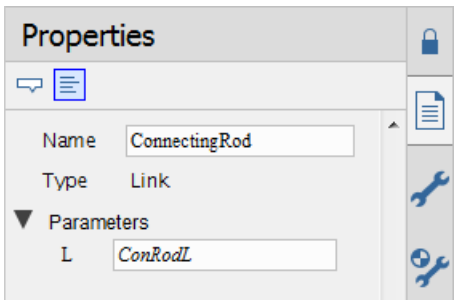


10. Double-click the **SliderCrank** subsystem, and then select the **Crank** subsystem.

11. In the **Properties** tab ( ), change the length value (**L**) to **CrankL**. The **Crank** subsystem now inherits the numeric value of **CrankL** that you defined.



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



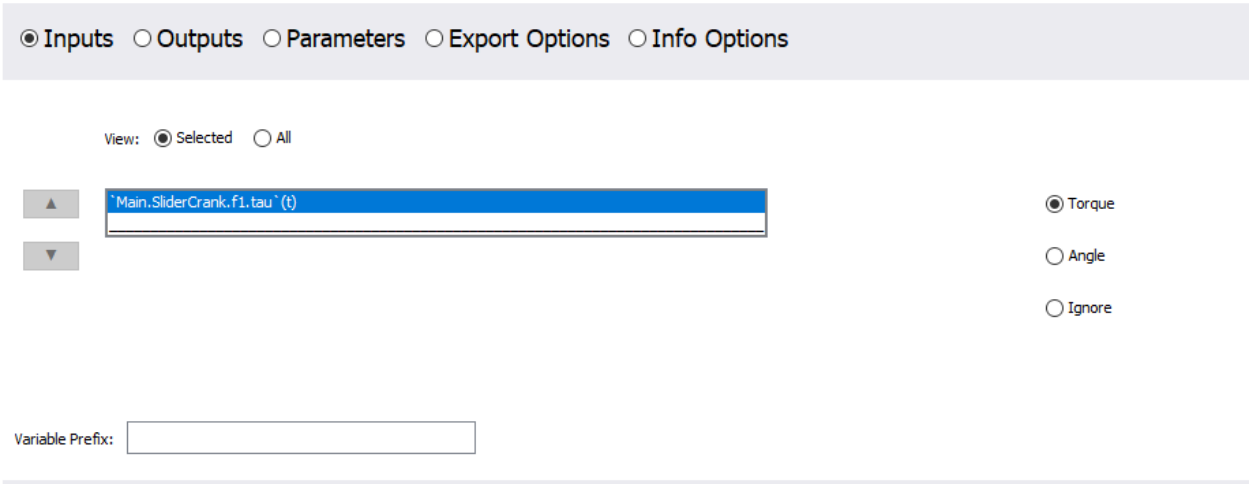
13. Click **Main** ( ) in the **Model Workspace** toolbar to navigate to the top level of the model. You will include these parameter values in the model that you export. You are now ready to convert your model to an FMI block.

2.3 Exporting Your Model Using the FMU File Generation App

After preparing the model, you can use the FMU Generation app to set export options and convert the model to an FMU file.

To generate an FMU file

1. Select the **Add Apps or Templates** tab ( ).
2. Double-click on the **FMU Generation** entry in the **Apps** palette. The **Analysis** window opens with the **FMU Component Code Generation** app loaded in the **Apps** tab.
3. In the **Subsystem Selection** section, select the **SliderCrank** subsystem from the drop-down list, and then click **Load Selected Subsystem**. All of the app fields are populated with information specific to the subsystem.
4. Since this example uses an acausal 1D - mechanical rotational flange, on selecting the **input** option, you have the option to either choose a torque or an angle input that can be fed into the subsystem.



5. In the **Export** section, specify the location for the **Target Directory**, **Visual C++ Directory** and provide a name for the generated **FMU Archive.**
6. Click **Generate FMU Archive** to generate the .FMU zip file.

Index

A

Apps, 1
    FMU File Generation, 14

D

Der
    Ports, 7

E

Examples
    RLC circuit model, 8
    slider-crank model, 9
Export Options, 3
Exporting, iv
Extended Visualization Data, 7
External Libraries, iv

F

FMI
    exporting, 9
FMI Connector Examples, 1

G

Generate
    External Libraries, 6
    FMU Archive, 6, 15

L

License
    Distribution of FMU, iv

M

Models using external libraries, iv

P

Port and Parameter Management, 2
Ports
    for Derivatives, 7

S

Subsystem parameters, 13
Subsystem Selection, 2