FMI Kit for Simulink
version 2.4.0

by Dassault Systèmes

April 2017

The information in this document is subject to change without notice.

Dassault Systèmes AB
Ideon Gateway
Scheelevägen 27 – Floor 9
SE-223 63 Lund
Sweden

E-mail: http://www.3ds.com/support
URL: http://www.Dymola.com
Phone: +46 46 270 67 00

Contents

1 FMI Kit for Simulink

1.1 Introduction

1.1.1 FMI

The FMI ("Functional Mock-up Interface") standard allows any modeling tool to generate C code or binaries representing a dynamic system model which may then be seamlessly integrated in another modeling and simulation environment.

FMI started as a key development effort within the MODELISAR project, see

https://itea3.org/project/modelisar.html

The FMI standard is today maintained and developed as a long-term project within the Modelica Association.

Three official FMI specifications have been released. The 'FMI for Model Exchange' specification version 1.0 was released on January 28, 2010, and the 'FMI for Co-Simulation' specification version 1.0 was released on October 12, 2010. FMI 2.0 which merges the model exchange and co-simulation specifications into one document was published on July 25, 2014.

The model exchange specifications focus on the model ODE interface, whereas the co-simulation specifications deal with models with built-in solvers and coupling of simulation tools. A model package implementing the FMI standard is called a Functional Mockup Unit (FMU). For further details visit:

http://www.fmi-standard.org/

1.1.2 FMU Export from Simulink

FMI Kit for Simulink provides a Simulink Coder Target (`rtwsfcnfmi`) to support export of FMUs from MATLAB/Simulink. The FMU export package contains implementations of the FMI standards on top of model code generated by Simulink Coder (formerly Real-Time Workshop). The MATLAB Target Language Compiler (TLC) is used to construct the XML model description.

The package for FMU export from Simulink together with the Dymola support for FMU import facilitates simulation of Simulink models in Dymola.

The utility builds on the Simulink Coder 'S-function Target' configuration that is available in MATLAB. In fact, the same model C code is generated by the 'S-function target with FMI'

as for the Simulink Coder S-function target. In addition, the FMI target performs the following

- Constructs the model description interface, `modelDescription.xml`, from the `<modelname>.rtw` model description
- Compiles the generated model code and the S-function FMI wrapper, and links with required libraries
- Copies resources, such as images and MEX files, to the FMU `resources` folder
- Constructs the FMI zip archive (.fmu) according to the FMI distribution structure

Release History

- Version 1.0, February 10, 2010
    - First version
- Version 1.1, August 20, 2010
    - Supporting MATLAB R2010a
    - Support for S-function blocks written in C
- Version 1.2, June 1, 2012
    - MATLAB support up to R2011b
    - Support for Visual Studio 2010
    - 64-bit support
- Version 1.2.1, March 4, 2013
    - Compliant to FMU Checker ver. 1.0.2
- Version 2.0, March 31, 2015 (included with Dymola 2016)
    - FMI 1.0 and 2.0 support
    - Model Exchange and Co-Simulation
    - Support for all Simulink built-in data types
    - MATLAB support for R2010a - R2014b (32- and 64-bit)
    - Support for Visual Studio 2008 and later compilers
- Version 2.1, May 29, 2015
    - Loading of binary MEX S-functions
    - C++ source S-functions
    - Simulink I/O buses with structured naming
    - Black-box FMU generation
    - Block hierarchy in variable names
- Version 2.1.1, June 24, 2015 (a maintenance version)
- Version 2.1.2, October 9, 2015 (included with Dymola 2016 FD01)
    - Support for MATLAB R2015a and R2015b
- Version 2.2.0, April 15, 2016 (included with Dymola 2017)
    - Released as part of FMI Kit for Simulink (export and import)
    - Support for global tunable workspace parameters
    - Full support for MATLAB R2015b code generation, especially support for parameter references to workspace or mask variables

- Version 2.3.0, October 11, 2016 (included with Dymola 2017 FD01)
    - Support for MATLAB R2016a
    - Improved input handling and support for input interpolation
    - Support for FMU export on Linux
- **Version 2.4.0**, April 7, 2017 (included with Dymola 2018)
    - Support for MATLAB R2016b

1.1.3 FMU Import into Simulink

FMI Kit for Simulink contains a Simulink FMU block, which enables embedding of FMUs into Simulink models. With source code FMUs exported with Dymola 2016 and later, it is also possible to use FMUs in Rapid Accelerator mode and create target code for RSIM, GRT, and dSPACE ds1005, ds1006, and SCALEXIO platforms.

The package for FMU import into Simulink together with the Dymola support for FMU export facilitates simulation of Dymola models in Simulink. In particular, this enables use of Dymola solvers in Simulink through the FMI Co-simulation interface.

1.1.4 Support and Usage

FMI Kit for Simulink has full FMI support for both export and import, which means that both versions 1.0 and 2.0 of the FMI standard are supported for both Model Exchange and Co-Simulation. Supported MATLAB releases are R2010a to R2016b (32- and 64-bit). FMU export supports both Windows and Linux. FMU import is currently only supported on Windows.

FMI Kit for Simulink can be used for free without any license key.

Support and maintenance is offered to Dymola customers through the regular support channel at www.3ds.com/support.

FMI Kit for Simulink is independent of Dymola and updates are sometimes released in between the official Dymola releases. Information about new released versions can be found at www.dymola.com/FMI.

1.2 Installation

FMI Kit for Simulink is located in the `$DYMOLA/Mfiles/FMIKit_for_Simulink/` directory of the Dymola distribution or may be also be downloaded as a zip-archive through DS FileTransfer after contacting your DS support channel. Since the package is independent of Dymola it may be extracted or copied to any location.

Follow these steps to set up the environment in MATLAB:

- Add the `FMIKit_for_Simulink` directory to your MATLAB path and then execute the command `FMIKit.initialize()`
- Optionally, you may add the following to your MATLAB startup script to automatically perform the setup for each new session:

```
addpath('C:\Program Files\FMIKit_for_Simulink');
FMIKit.initialize();
```

(the `addpath` command should be changed to match your system)

1.3 Exporting FMUs from Simulink

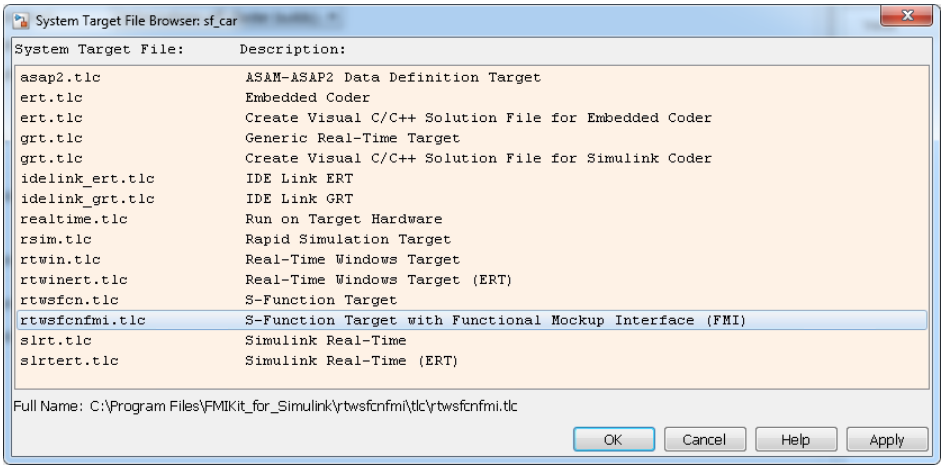
This section describes the procedure to export an FMU from Simulink and the associated settings / configurations.

Adding input and output ports

If the Simulink model to be exported as an FMU should be possible to connect to other components, you need to add external input and/or output ports to your model. These can be found in the Sinks and Sources categories of the Simulink browser. Hierarchical Simulink buses are supported as input and output port types.

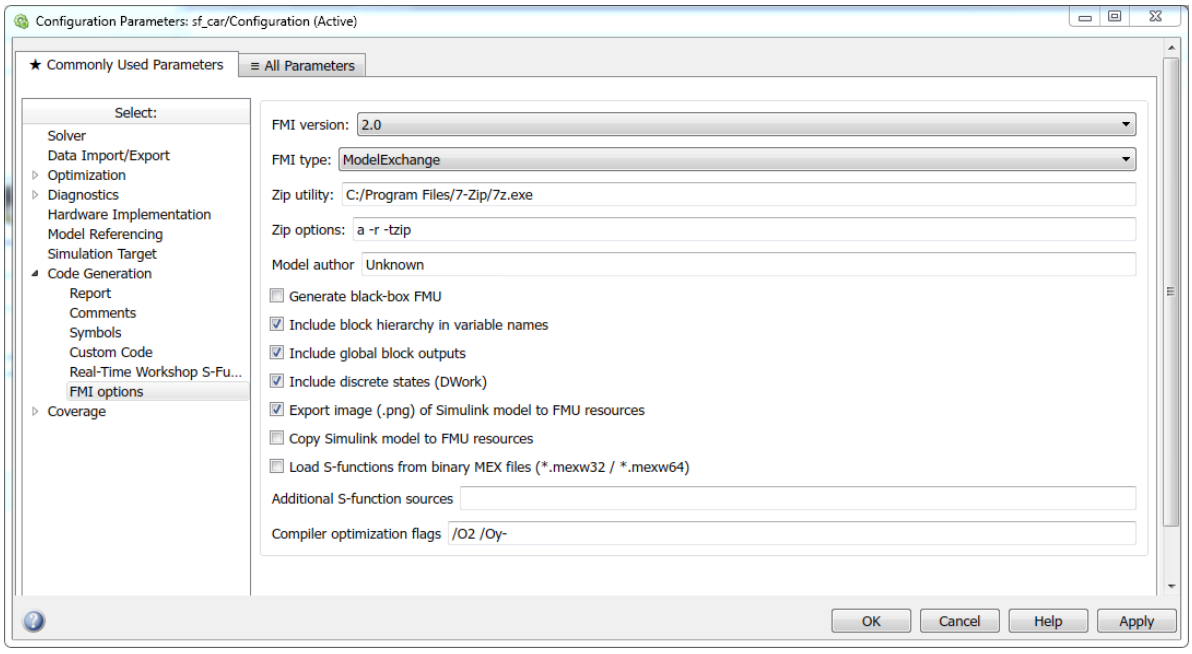
Selecting system target file

In the Simulink **Configuration Parameters** dialog, choose the **Code Generation** tab and click Browse to select a different System Target File. Select `rtwsfcnfmi.tlc` in the list:



Options for FMU export

After selecting the `rtwsfcnfmi.tlc` target, the tab **FMI options** becomes available in the **Code Generation** tab. A description of each option follows below.

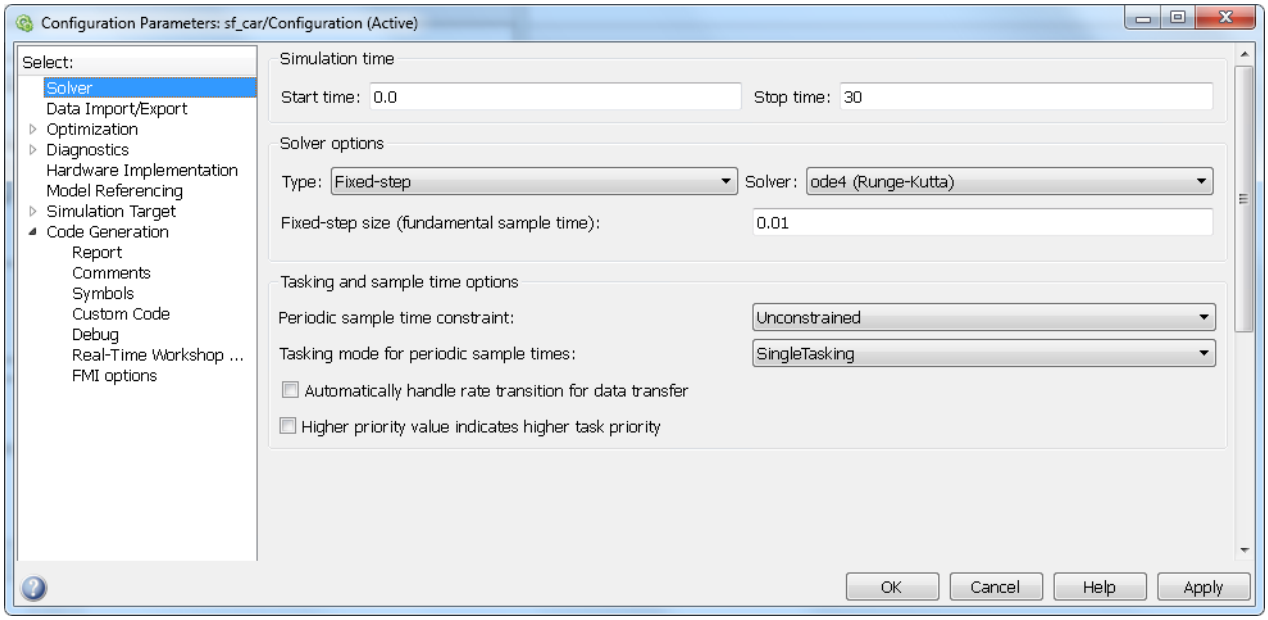


- **FMI version**
    - Selects FMI version for the export (1.0 or 2.0)
- **FMI type**
    - Specifies FMI type (ModelExchange or CoSimulation)
- **Zip utility**
    - Path to Zip utility used to build the FMU archive (the default on Windows is 7-zip, which can be downloaded from www.7-zip.org.)
- **Zip options**
    - Command line options passed to the Zip utility
- **Model author**
    - Specifies the model author for the FMU XML file
- **Generate black-box FMU**
    - Selects if the FMU should be generated as a black box (only inputs and outputs exposed).
- **Include block hierarchy in variable names**
    - Selects if variable names in the FMU XML file should be generated in a structured view using block hierarchy notation. Read more about variable naming below.
- **Include global block outputs**
    - Selects if block outputs should be included in the FMU XML file. Has no effect if black-box export has been selected.
- **Include discrete states (DWork)**
    - Selects if discrete states and modes should be included in the FMU XML file. Has no effect if black-box export has been selected.
- **Export image (.png) of Simulink model to FMU resources**
    - Selects if an image of the top-level Simulink model should be exported with the FMU. The exported image will be located in the subfolder `SimulinkModel` of the FMU resources.
- **Copy Simulink model to FMU resources**
    - Selects if the whole Simulink model should be copied to the FMU. The model will be located in the subfolder `SimulinkModel` of the FMU resources.
- **Load S-functions from binary MEX files**
    - Selects that S-functions in the model will be loaded from pre-compiled binary MEX files instead of using stand-alone compilation of S-function sources (more details on S-functions below). **Note:** This checkbox should only be used if your model has S-function blocks.
- **Additional S-function sources**
    - List of additional user source files for stand-alone S-function compilation. Should be used instead of Custom Code -> Source Files to ensure that the correct compiler options are used.
- **Compiler optimization flags**
    - User-defined optimization flags to be used by the compiler (default: /O2 /Oy- for Visual Studio on Windows and -O2 for GCC on Linux).

Solver settings

These are the recommended settings for **Configuration Parameters -> Solver**

- **Model Exchange export**: Both Variable-step and Fixed-step solvers supported (recommended to use Variable-step when possible to support accurate event detection using non-sampled zero crossings).
- **Co-Simulation export**: Requires a Fixed-step solver (the selected solver is compiled into the FMU).
- It is also recommended to explicitly set the Tasking mode to SingleTasking.



Including S-functions in the exported FMU

Models containing S-functions can be exported and the S-functions can be included in the FMU either from C/C++ sources or as binary MEX files. Note that the S-functions are not allowed to call into the MATLAB environment, e.g., using `mexCallMATLAB` or `mexEvalString`.

Including S-functions from C/C++ sources

Source compilation of S-functions is default and is used if the option *Load S-functions from binary MEX files* is not selected.

The S-function sources (C or C++) should be available and located in the same directory as the Simulink model. The S-function sources are then automatically compiled and linked to the FMU and no further configuration is needed in the Simulink model.

Note that source compilation of S-functions defines the flag NRT, which is used to indicate that the S-function is generated by Simulink Coder (or user-written) for non-real-time applications using a variable-step (or fixed-step) solver. For S-functions that should be built as MEX files for use in Simulink, it is recommended to use the binary MEX file inclusion as described below.

Including S-functions from binary MEX files

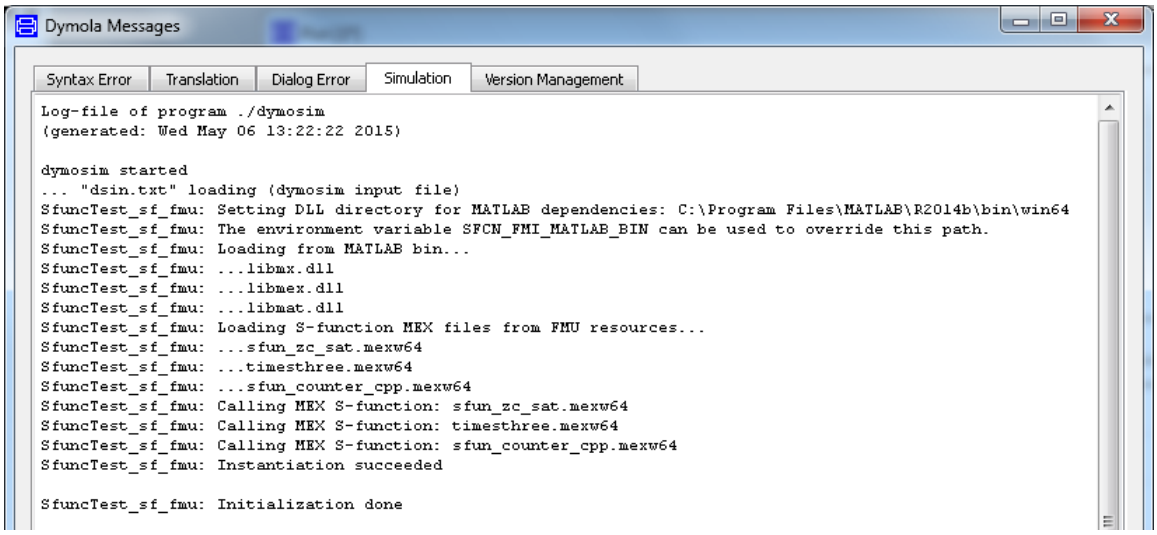
If the option *Load S-functions from binary MEX files* is selected, no compilation of S-function sources is performed. Instead, the S-function MEX files are copied to the FMU (to `resources\SFunctions`) and code is added to dynamically load and call the MEX files

when the FMU is instantiated. This option will also create dependencies on MATLAB binaries (which will not be copied to the FMU).

On Windows, the FMU will by default try to load the MATLAB binaries from the bin directory of the exporting MATLAB installation, which means that export / import on the same computer should work seamlessly. The environment variable `SFCN_FMI_MATLAB_BIN` can be used to specify a different directory from where to load the MATLAB DLLs (for example a MATLAB run-time installation on a different computer).

On Linux, it is required to use the environment variable LD_LIBRARY_PATH to specify the path to the MATLAB binaries.

With logging enabled the FMU outputs information about the loading of binaries and MEX files during instantiation. The following is an example of importing an 64-bit FMU with MEX file dependencies into Dymola on Windows:



Configuring Visual Studio Compiler

The FMI binary is built using the same version of compiler as used when building MEX files in MATLAB. The compiler is configured in MATLAB using the command

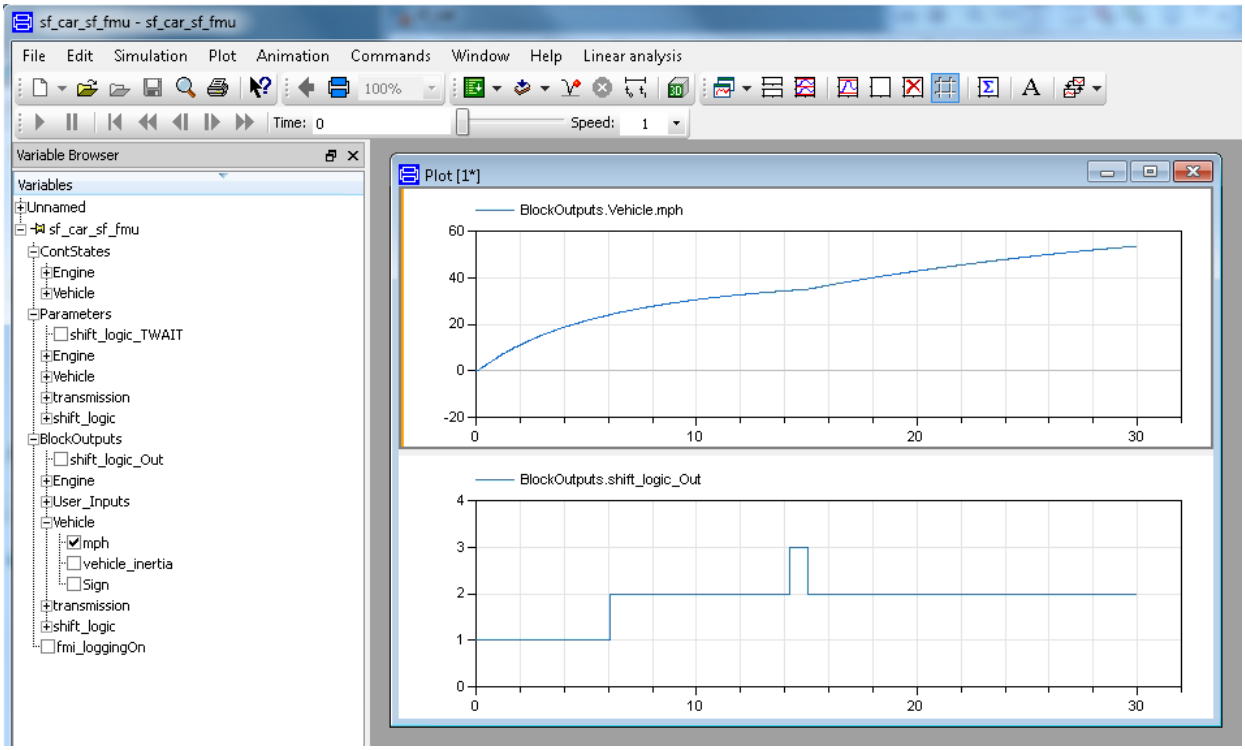
```
>> mex –setup
```

Variable naming

Two options are available for naming of variables in the FMU XML file. With the option *Include block hierarchy in variable names* selected, variable names are generated with block-hierarchical notation and the XML model description specifies the attribute `variableNamingConvention="structured"`. Alternatively, with the box de-selected, the Simulink Coder C code identifiers (not traceable back to model) are used as variable names and the XML specifies `variableNamingConvention="flat"`.

The variable names for continuous-time states, discrete states, parameters, and block outputs are separated into the top-level categories *ContStates*, *DiscStates*, *Parameters*, and *BlockOutputs* in the structured view (see example from Dymola structured FMU import below). This is to ensure unique variable names in the FMU XML file, since variable names from different categories are not guaranteed to be unique within a block. In the flat view, the variable names are appended with `_xc`, `_xd`, `_pm`, and `_wb`, respectively.

The flat view is guaranteed to generate unique variable names in all cases, whereas the structured view in some rare cases could produce name conflicts (on limitations, see section “Limitations and Trouble-Shooting” below).



Building the FMU

Start the build process by pressing **Ctrl-B** (or through the Simulink Code menu).

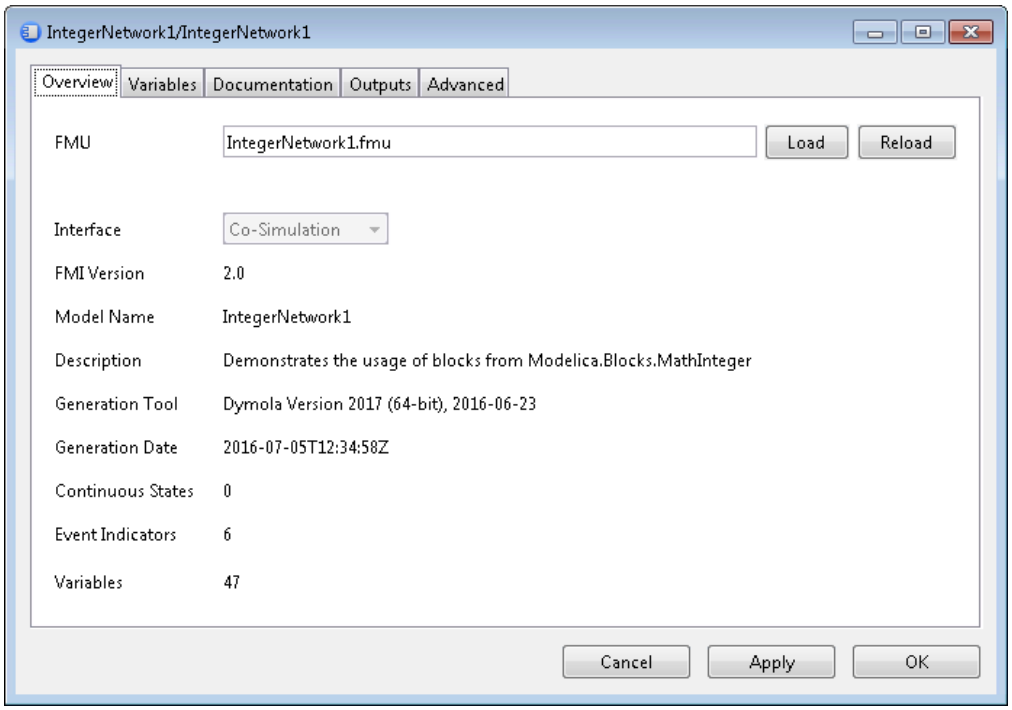
The build process will compile the generated model code using the FMI Simulink wrapper and link with the required MATLAB and system libraries to create the FMU binaries. The build process will also create the FMI XML model description, `modelDescription.xml`, and construct the FMI archive, `<modelname>_sf.fmu` in the current working directory.

1.4 Importing FMUs into Simulink

This section describes the procedure to import an FMU into Simulink and the associated settings / configurations in the user interface. There is also a set of MATLAB commands to interact with the FMI Kit import. These are described in more detail in the HTML documentation accessed through `FMIKit_for_Simulink/html/fmikit.html`

Adding FMUs to a model

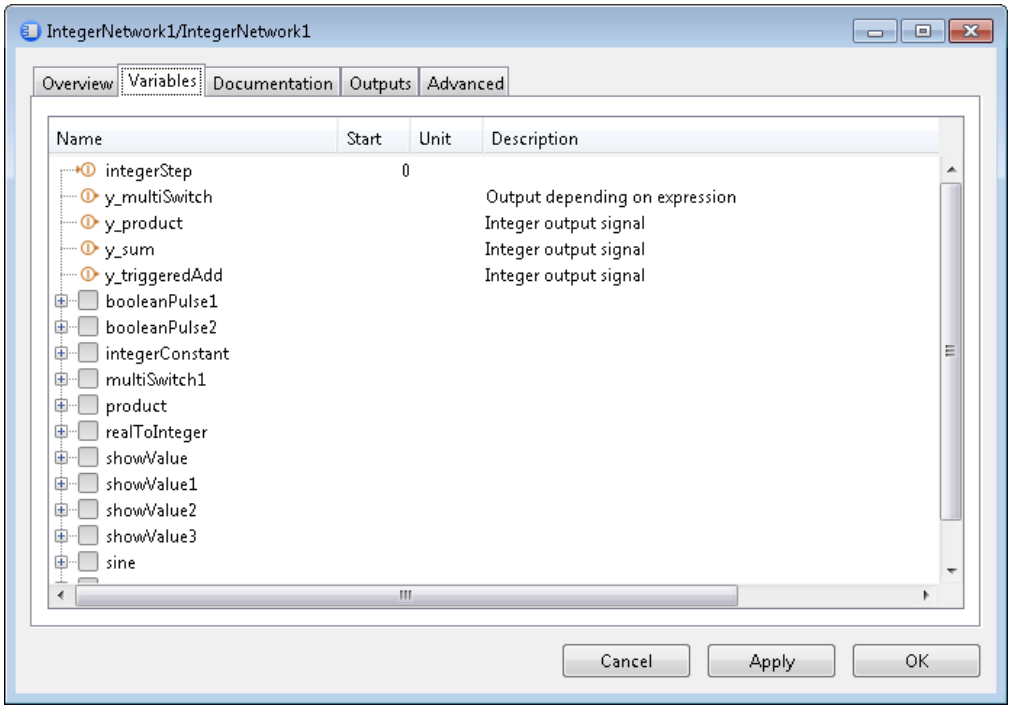
- Open the Simulink library browser (**View > Library Browser**) and drag the FMU block from the FMI Kit library into your model.
- Double-click the FMU block, select **Load** and choose the FMU.
- Click **OK**.



The FMU is automatically extracted to the directory specified under Advanced -> Unzip Directory. This directory must remain in the same relative path when the model is moved to a different directory or machine.

For FMI 2.0 FMUs that support both model exchange and co-simulation the interface kind can be selected.

Variables & start values

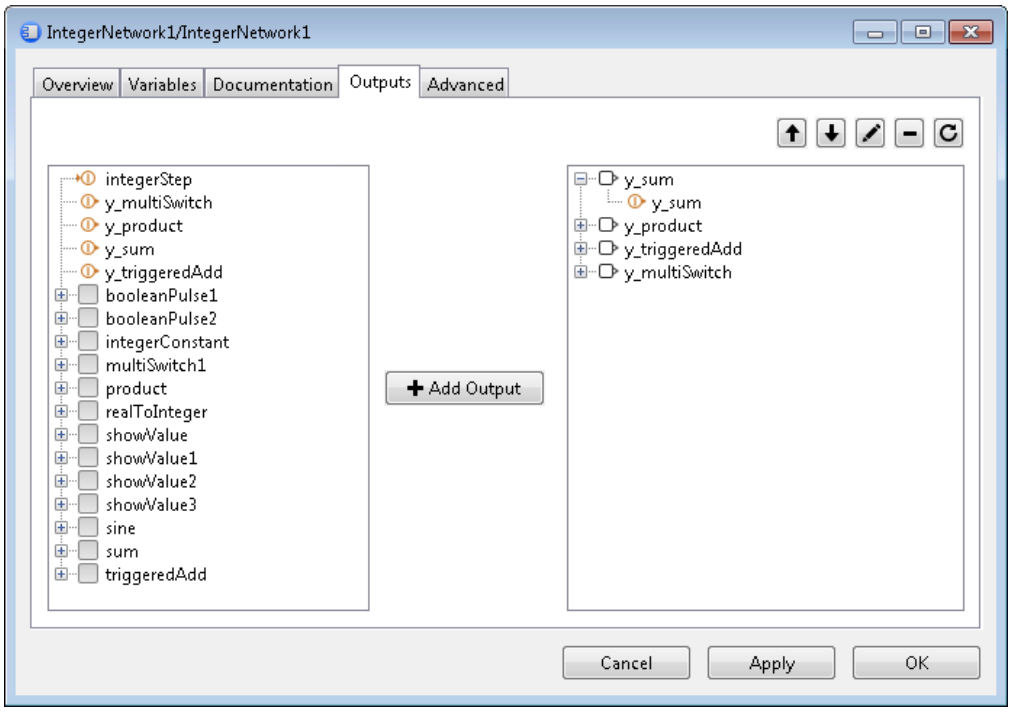


The Variables tab shows all variables of the FMU. Input variables are marked with an arrow on the left, output variables with an arrow on the right of the icon.

The start value, unit and description of the variable (if provided) are displayed in the Start, Unit and Description columns. Start values that were explicitly set are displayed as bold text.

To change the start value of a variable click into the respective field in the "Start" column and enter an expression that evaluates to the respective type of variable. Changed start values are indicated by bold text. To reset the start value to its default simply clear the "Start" field.

Output ports

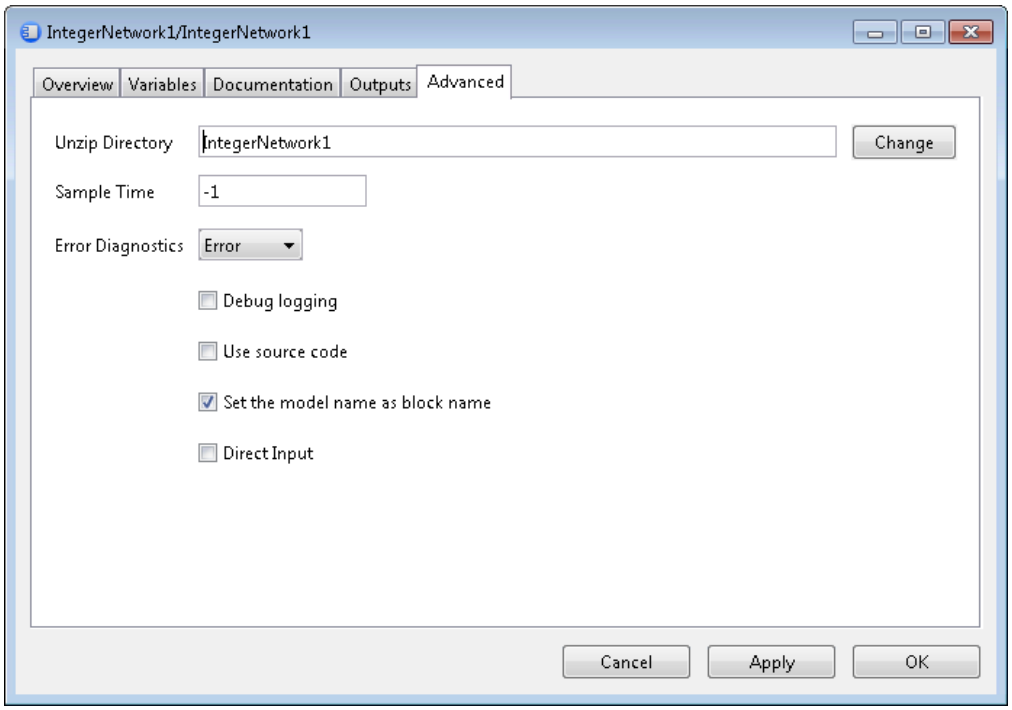


By default the block has the output ports defined by the FMU.

- To add output ports select one or more variables in the left view and click “Add Output”.
- To remove output ports select the ports in the right view and click “-“.
- To move an item in the right view, select it and use the up and down buttons.
- To restore the default output ports, click the reset button.

Advanced settings

On the Advanced tab you can change additional settings for the FMU block:



- **Unzip Directory**
    - The folder where the FMU is extracted. The path can be absolute or relative to the model file. To use a custom path, change this field before loading an FMU.
- **Sample Time**
    - The sample time for the block (use -1 for inherited)
- **Error Diagnostics**
    - Determines how to handle errors reported by the FMU
- **Debug Logging**
    - Enables the debug logging to the MATLAB console
- **Use Source Code**
    - If checked a source S-function `sfun_<model_name>.c` is generated from the FMU's source code which gets automatically compiled when the Apply or OK button is clicked. For FMI 1.0 this feature is only available for FMUs generated with Dymola 2016 or later.
- **Set Model Name**
    - Use the model name of the FMU as block name
- **Direct Input**
    - If checked `ssSetInputPortDirectFeedThrough(true)` is set for all input ports of the FMU and the value of the block's inputs `u` at `t+1` is applied to the input variables of the FMU at time `t`. This gives better results for FMUs that contain direct terms and do not support input interpolation.

      If not checked `ssSetInputPortDirectFeedThrough(true)` is only set for input ports whose input variables have output variables with a direct dependency. The derivative `der_u` for these input variables is set such that u(t) + der_u(t) * step_size = u(t+1) if the FMU supports input interpolation. Variables that are manually added to the block's output ports are assumed to depend on all input variables.

Source code FMUs

With source code FMUs you can use advanced simulation targets that require code generation.

To use FMU source code, open the block dialog an on the "Advanced" tab select **Use Source Code**. After clicking **OK** FMI Kit generates a source S-function `.c` and builds S-function MEX file `.mex32` (or `.mexw64` on a 64 bit platform). You can now use the following additional simulation targets: Rapid Accelerator, RSIM, GRT, ds1005, ds1006.

1.5 Limitations and Trouble-Shooting

FMU Export

The following relates to version 2.4.0 of the `rtwsfcnfmi` target:

On Windows, the export supports Visual Studio 2008 (9.0) and later compilers as supported with the respective MATLAB releases.

On Linux, the package should support the versions of gcc supported with the respective MATLAB releases. The object files shipped in the package have all been compiled using gcc 4.3.4

The FMU is compiled with dynamic loading of the C run-time on Windows. This may require installation of corresponding Visual Studio redistributables on the target platform.

The option *Include block hierarchy in variable names* could in very rare cases give rise to name conflicts in the XML variable names. For example, any special characters in Simulink block names will be converted to underscore which may lead to name conflicts. It is recommended to avoid using special characters in block names with this option (carriage return and space are safe to use).

For multiple instances of conditionally executed nonvirtual subsystems or Stateflow charts, it is required to select "Treat as atomic unit" and set "Function packaging" to "Inline" for the subsystems/charts.

S-functions in the exported model are not allowed to call into the MATLAB environment, e.g., using `mexCallMATLAB` or `mexEvalString`.

The FMU export target is not model reference compliant.

The package is subject to the same limitations as the standard S-Function Target.

1.6 File Structure

The `rtwsfcnfmi` target folder (`FMIKit_for_Simulink\rtwsfcnfmi`) consists of six sub-directories and the included files are described briefly below.

bin

Pre-compiled Visual Studio and GCC binaries of the FMI implementation for the supported MATLAB releases.

c

This directory holds C source files to include and compile the Simulink Coder-generated model code. The standard FMI header files are located in the sub-directory `fmi`.

m

This directory contains MATLAB help files called from the TLC scripts. These are used to construct the date, GUID, and value reference attributes used in the XML model description.

tlc

The TLC scripts used for code generation and for constructing FMI-specific files are included in this directory. The template makefiles and compiler-dependent settings can also be found here.

2 Index