

gO:Simulink Guide

Release v3.5

June 2012



gO:Simulink Guide

Release v3.5

June 2012

Copyright © 1997-2012 Process Systems Enterprise Limited

Process Systems Enterprise Limited
6th Floor East
26-28 Hammersmith Grove
London W6 7HA
United Kingdom
Tel: +44 20 85630888
Fax: +44 20 85630999
WWW: <http://www.psenterprise.com>

Trademarks

gPROMS is a registered trademark of Process Systems Enterprise Limited ("PSE"). All other registered and pending trademarks mentioned in this material are considered the sole property of their respective owners. All rights reserved.

Legal notice

No part of this material may be copied, distributed, published, retransmitted or modified in any way without the prior written consent of PSE. This document is the property of PSE, and must not be reproduced in any manner without prior written permission.

Disclaimer

gPROMS provides an environment for modelling the behaviour of complex systems. While gPROMS provides valuable insights into the behaviour of the system being modelled, this is not a substitute for understanding the real system and any dangers that it may present. Except as otherwise provided, all warranties, representations, terms and conditions express and implied (including implied warranties of satisfactory quality and fitness for a particular purpose) are expressly excluded to the fullest extent permitted by law. gPROMS provides a framework for applications which may be used for supervising a process control system and initiating operations automatically. gPROMS is not intended for environments which require fail-safe characteristics from the supervisor system. PSE specifically disclaims any express or implied warranty of fitness for environments requiring a fail-safe supervisor. Nothing in this disclaimer shall limit PSE's liability for death or personal injury caused by its negligence.

Acknowledgements

ModelBuilder uses the following third party free-software packages. The distribution and use of these libraries is governed by their respective licenses which can be found in full in the distribution. Where required, the source code will be made available upon request. Please contact support.gPROMS@psenterprise.com in such a case.

Many thanks to the developers of these great products!

Table 1. Third party free-software packages

Software/Copyright	Website	License
ANTLR	http://www.antlr2.org/	Public Domain
Batik	http://xmlgraphics.apache.org/batik/	Apache v2.0
Copyright © 1999-2007 The Apache Software Foundation.		
BLAS	http://www.netlib.org/blas	BSD Style
Copyright © 1992-2009 The University of Tennessee.		
Boost	http://www.boost.org/	Boost
Copyright © 1999-2007 The Apache Software Foundation.		
Castor	http://www.castor.org/	Apache v2.0
Copyright © 2004-2005 Werner Guttman		
Commons CLI	http://commons.apache.org/cli/	Apache v2.0
Copyright © 2002-2004 The Apache Software Foundation.		
Commons Collections	http://commons.apache.org/collections/	Apache v2.0
Copyright © 2002-2004 The Apache Software Foundation.		
Commons Lang	http://commons.apache.org/lang/	Apache v2.0
Copyright © 1999-2008 The Apache Software Foundation.		
Commons Logging	http://commons.apache.org/logging/	Apache v1.1
Copyright © 1999-2001 The Apache Software Foundation.		
Crypto++ (AES/Rijndael and SHA-256)	http://www.cryptopp.com/	Public Domain
Copyright © 1995-2009 Wei Dai and contributors.		
Fast MD5	http://www.twmacinta.com/myjava/fast_md5.php	LGPL v2.1
Copyright © 2002-2005 Timothy W Macinta.		
HQP	http://hqp.sourceforge.net/	LGPL v2
Copyright © 1994-2002 Ruediger Franke.		
Jakarta Regexp	http://jakarta.apache.org/regexp/	Apache v1.1
Copyright © 1999-2002 The Apache Software Foundation.		
JavaHelp	http://javahelp.java.net/	GPL v2 with classpath exception
Copyright © 2011, Oracle and/or its affiliates.		
JXButtonPanel	http://swinghelper.dev.java.net/	LGPL v2.1 (or later)
Copyright © 2011, Oracle and/or its affiliates.		
LAPACK	http://www.netlib.org/lapack/	BSD Style
libodbc++	http://libodbcxx.sourceforge.net/	LGPL v2

Software/Copyright	Website	License
Copyright © 1999-2000 Manush Dodunekov <manush@stendahls.net>		
Copyright © 1994-2008 Free Software Foundation, Inc.		
lp_solve	http://lpsolve.sourceforge.net/	LGPL v2.1
Copyright © 1998-2001 by the University of Florida.		
Copyright © 1991, 2009 Free Software Foundation, Inc.		
MiGLayout	http://www.miglayout.com/	BSD
Copyright © 2007 MiG InfoCom AB.		
Netbeans	http://www.netbeans.org/	SPL
Copyright © 1997-2007 Sun Microsystems, Inc.		
omniORB	http://omniorb.sourceforge.net/	LGPL v2
Copyright © 1996-2001 AT&T Laboratories Cambridge.		
Copyright © 1997-2006 Free Software Foundation, Inc.		
TimingFramework	http://timingframework.dev.java.net/	BSD
Copyright © 1997-2008 Sun Microsystems, Inc.		
VecMath	http://vecmath.dev.java.net/	GPL v2 with classpath exception
Copyright © 1997-2008 Sun Microsystems, Inc.		
Wizard Framework	http://wizard-framework.dev.java.net/	LGPL
Copyright © 2004-2005 Andrew Pietsch.		
Xalan	http://xml.apache.org/xalan-j/	Apache v2.0
Copyright © 1999-2006 The Apache Software Foundation.		
Xerces-C	http://xerces.apache.org/xerces-c/	Apache v2.0
Copyright © 1994-2008 The Apache Software Foundation.		
Xerces-J	http://xerces.apache.org/xerces2-j/	Apache v2.0
Copyright © 1999-2005 The Apache Software Foundation.		

This product includes software developed by the Apache Software Foundation, <http://www.apache.org/>.

gPROMS also uses the following third party commercial packages:

- **FLEXnet Publisher** software licensing management from Acresto Software Inc., <http://www.acresso.com/>.
- **JClass DesktopViews** by Quest Software, Inc., <http://www.quest.com/jclass-desktopviews/>.
- **JGraph** by JGraph Ltd., <http://www.jgraph.com/>.

Table of Contents

1. Introduction	1
gO:Simulink	1
The export to Simulink capability	2
Importing Simulink models into gPROMS	2
System set-up	2
Licensing	2
MATLAB configuration	3
Confirm environmental variable settings	3
gO:Simulink - Export to Simulink example	3
gO:Simulink - Import to gPROMS example	3
Supported versions of Matlab and Simulink	3
2. Exporting gPROMS models to Simulink	4
gPROMS model preparation	4
The component (gO:Simulink) model	4
The composite model	4
The Export to Simulink utility	5
The Process Entity	5
The Export to Simulink dialog box	5
Exported files	6
3. Using gO:Simulink	8
Loading the Simulink model	8
Executing a simulation	9
Simulink solver settings	10
gO:Simulink options	10
Continuous and discrete-time modes	11
4. Model translation	12
Model translation overview	12
The gO:Simulink library	12
The Export to Simulink dictionary	12
Dictionary files	13
The <Model> tag	14
The <BlockType> tag	14
The <SourceBlock> tag	14
The <SourceType> tag	14
The <Parameter> tag	14
The <Property> tag	14
5. Exporting Simulink flowsheets to gPROMS	15
Software requirements	15
System configuration	15
Exporting a Simulink flowsheet	15
Preparing the Simulink flowsheet for the export	15
Configure the Real-Time Workshop (RTW) for export	16
Exporting the Simulink flowsheet	18
Generation of the Foreign Object containing the Simulink flowsheet	19
6. Using a Simulink flowsheet in gPROMS	21
Methods provided by the Foreign Object containing the Simulink flowsheet	21
NumberOfInputs	21
NumberOfOutputs	21
NumberOfStates	22
NumberOfSamplingRates	22
GetSamplingRates	23
GetSmallestSamplingRate	23
GetStates	24
GetTimeDerivatives	25
GetOutputs	25

Update	27
The gO:Simulink import library - content and examples	28
The AlgebraicSimulinkBlock model	29
The DynamicSimulinkBlock model	30
The DiscreteSimulinkBlock model	31
The MixedSimulinkBlock model	34
Known limitations and troubleshooting	36
Known limitations and workarounds	36
Troubleshooting	39

List of Figures

1.1. A gO:Simulink block embedded in a Simulink model	1
1.2. A Simulink flowsheet embedded in a gPROMS model	1
2.1. A simple distillation column example of a composite model to be exported to Simulink	5
2.2. Export to Simulink dialog box.	6
3.1. The exported Simulink model of the distillation example.	9
3.2. A gO:Simulink Simulation	10
3.3. gO:Simulink block configuration dialog box: Continuous-time case.	10
3.4. gO:Simulink block configuration dialog box: Discrete-time case.	11
4.1. Example of the naming convention relating a gPROMS model to a Simulink block.	13
4.2. The syntax of the dictionary entry file	13
4.3. Specifying more than one <Parameter> tag	14
4.4. Specifying more than one <Property> tag	14
5.1. Example of a flowsheet ready for export.	16
5.2. Configuring the size of the input port In1.	16
5.3. Configuring the Simulink solver settings.	17
5.4. Configuration of the Real-Time Workshop.	18
5.5. Selecting the system target file in the target browser.	18
5.6. Opening a command window using the "Run" window.	19
5.7. Using the "cd" command to change the directory	19
5.8. Directory structure of the exported Simulink model	19
5.9. The generated ForeignObject in form of the SimulinkFOTest.dll file	20
6.1. Example: Using the NumberOfInputs method	21
6.2. Example: Using the NumberOfOutputs method	22
6.3. Example: Using the NumberOfStates method	22
6.4. Example: Using the NumberOfSamplingRates method	23
6.5. Example: Using the GetSamplingRates method	23
6.6. Example: Using the GetSmallestSamplingRate method	24
6.7. Example: Using the GetStates method	24
6.8. Example: Using the GetTimeDerivatives method	25
6.9. Example: Using the GetOutputs method for an algebraic model	26
6.10. Example: Using the GetOutputs method for a model containing continuous states	27
6.11. Example: Using the Update method	28
6.12. Browsing the example projects	28
6.13. Using the AlgebraicSimulinkBlock model	29
6.14. Simulink flowsheet of the SISO gain	30
6.15. Using the DynamicSimulinkBlock model.	31
6.16. Simulink flowsheet of the dynamic (continuous time) model	31
6.17. Using the DiscreteTimeMIMO model.	32
6.18. Simulink flowsheet of the discrete time model	32
6.19. Running the RunOneSamplingRate TASK in the SCHEDULE	33
6.20. The RunOneSamplingRate TASK	33
6.21. Using the MixedSimulinkBlock model.	34
6.22. Simulink flowsheet of the discrete time model	35
6.23. Running the RunOneMixedSamplingRate TASK in the SCHEDULE	35
6.24. The RunOneMixedSamplingRate TASK	36
6.25. Removing a MemoryBlock	37
6.26. Replacing a MemoryBlock	38
6.27. Signal routing for re-initialisations on the Simulink flowsheet	38
6.28. Foreign Object log file example	39

List of Tables

1. Third party free-software packages	3
2.1. gSF file of the distillation column example	7

Chapter 1. Introduction¹

gO:Simulink

The key features of gO:Simulink are:

- Allowing complex non-linear gPROMS process models to be incorporated directly into The MathWorks Simulink environment. The gPROMS model is run from a gO:Simulink block that can be added to any Simulink® model (cf. figure A gO:Simulink block embedded in a Simulink model). No restrictions are imposed on the gPROMS model thereby enabling the user to make full use of all simulation features and numerical capabilities provided by gPROMS.
- Allowing Simulink flowsheets to be incorporated directly into gPROMS. The Simulink flowsheet is exported using the Real-Time Workshop and imported as a ForeignObject. Once imported the model does not need a MATLAB® or Simulink license and will be solely handled by the solvers provided by gPROMS.

Figure 1.1. A gO:Simulink block embedded in a Simulink model

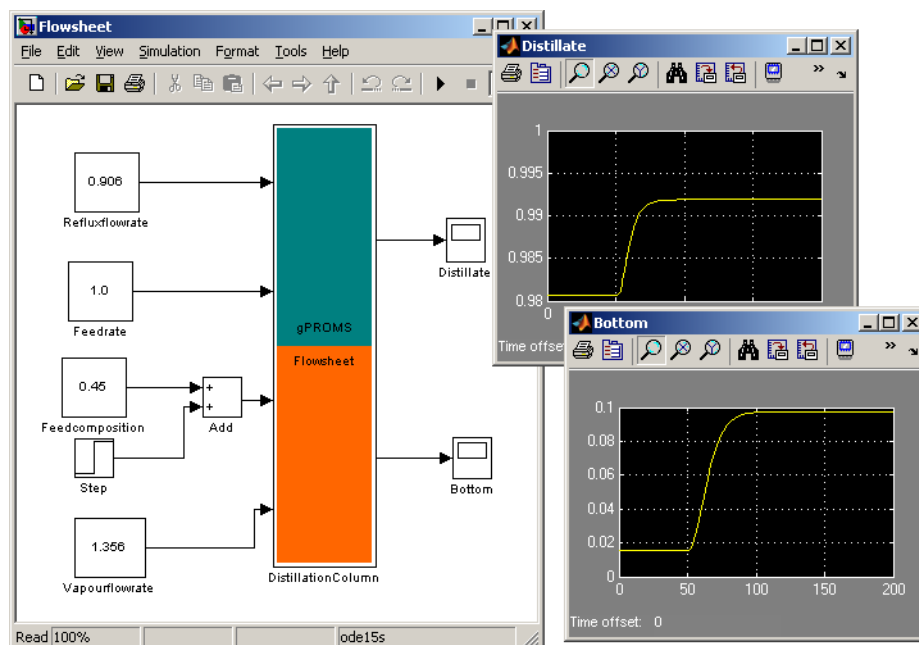
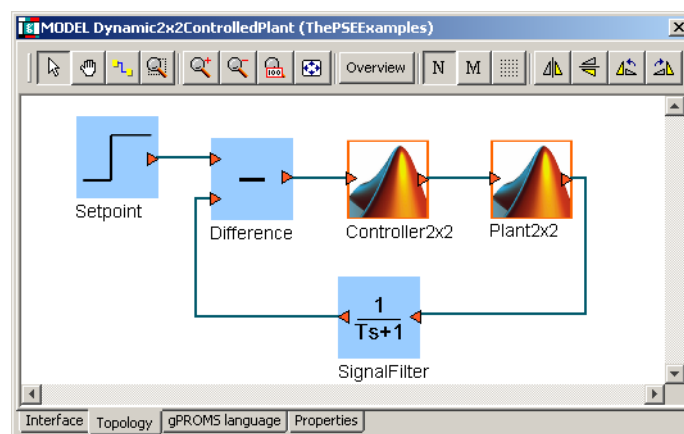


Figure 1.2. A Simulink flowsheet embedded in a gPROMS model



¹Simulink® and MATLAB® are registered trade marks of The MathWorks [<http://www.mathworks.com/>]

Throughout this guide it is assumed that the reader has a good understanding of gPROMS. A good knowledge of both MATLAB and Simulink is also a pre-requisite.

The export to Simulink capability

To promote the integration with Simulink, gPROMS ModelBuilder provides an *Export to Simulink* utility. This utility takes a gPROMS flowsheet model (a composite model) and translates it into a Simulink model: the result is a Simulink (mdl) file that automatically includes an appropriate gO:Simulink block where required.²

In addition, *Export to Simulink* provides the following functionality:

- *Units* on the gPROMS flowsheet that are based on simple flowsheeting models (e.g. sources and sinks) and control system models can be automatically *translated* into equivalent Simulink blocks. This capability is supported by a gO:Simulink library and a translation dictionary (c.f. chapter Model Translation).
- On translation to Simulink the topology of the gPROMS model is maintained.

Essentially this capability allows the user to solve complex non-linear models using gPROMS' solvers while solving simple flowsheeting models (e.g. sources and sinks) and controls system models directly in Simulink. The main aim of this capability is to support the usage of a gPROMS model during an activity that is best executed using Simulink.

Importing Simulink models into gPROMS

A second option to promote the integration between Simulink and gPROMS is to *import* Simulink flowsheets into gPROMS via *Foreign Objects*. The key step is to generate a C-code representation of the Simulink flowsheet using the Real-Time Workshop. This code is compiled and linked using the Microsoft Visual C++ 6.0 compiler with an interface provided with the gPROMS installation yielding a *Foreign Object* representation of the Simulink flowsheet. Once the system is set up properly, nearly no manual interaction is necessary to generate the *Foreign Object*. In particular no programming language skills are required!

Since the C-code generation depends on the Real-Time Workshop, only Simulink flowsheets consisting of blocks supported by the Real-Time Workshop can be imported to gPROMS.

Once the *Foreign Object* is generated no licenses related to Simulink, MATLAB or to the Real-Time Workshop are required to use the exported Simulink model in gPROMS.

Essentially this feature allows the user to incorporate models implemented in Simulink (like complex control structures) into a gPROMS model. Here the main aim is to support a Simulink model during an activity that is best executed using gPROMS.

System set-up

gO:Simulink is included as part of the standard gPROMS installation process. The following software components are required:

- MATLAB and Simulink are required to use a model exported from gPROMS to gO:Simulink in Simulink.
- MATLAB, Simulink, the Real-Time Workshop and Microsoft Visual C are required to export a Simulink flowsheet and to generate its *Foreign Object* representation to be used in gPROMS.
- No additional software component is required to run a *Foreign Object* representation of a Simulink flowsheet in gPROMS.

Licensing

Both gO:Simulink options require an additional license. Please contact <sales@psenterprise.com> to obtain the appropriate licenses.

²At present only a single gO:Simulink block can be run from a Simulink model.

MATLAB configuration

For MATLAB to load gO:Simulink, the gO:Simulink directory³ must be added to the MATLAB path. In MATLAB this is done by selecting *Set Path* from the *File* menu; this setting can be saved for future MATLAB and Simulink sessions.

Confirm environmental variable settings

The system environment variables *GPROMSHOME* must be set correctly to point to the gPROMS directory of your gPROMS installation⁴. Additionally, the *bin* sub-directory of the gPROMS installation directory must be added to the *PATH* environment variable⁵.

Please contact your system administrator if you are unsure about the location of the gPROMS installation or about the procedure for checking or setting environment variables.

gO:Simulink - Export to Simulink example

The standard gPROMS installation includes a simple distillation column example (*Distillation.gPJ*) for testing the Export to Simulink utility and gO:Simulink⁶. The same example is used in this manual to illustrate the instructions.

Note that to run the example, the gOSimulink_Library.gPJ and the PML Basics.gPJ library projects must also be open in the gPROMS ModelBuilder as these are cross-referenced in the *Distillation.gPJ* project.

gO:Simulink - Import to gPROMS example

The standard gPROMS installation includes a couple of simple examples (*gOSimulinkImportExamples.gPJ*) which

- illustrate the usage of Simulink models in gPROMS
- can be used as templates for more complex applications.

Note that to run the examples the gOSimulink_ImportLibrary.gPJ project must also be open in the gPROMS ModelBuilder as this is cross-referenced by the *gOSimulinkImportExamples.gPJ* project.

The Simulink flowsheets that are used in the various examples are included as part of the gPROMS installation as well.⁷

Supported versions of Matlab and Simulink

Supported versions of Matlab and Simulink are R14 Service Pack 3 (7.1). Later versions are not guaranteed to be compatible. It is already known that exporting Simulink models to gPROMS will fail with R2006a.

gO:Simulink is only available on Microsoft Windows®.

³The gO:Simulink directory is a sub-directory of the gPROMS installation.

⁴The gPROMS installation directory is typically `c:\Program Files\pse`. In such a case *GPROMSHOME* must be set to `c:\Program Files\pse\gPROMS`. Depending on the installation, this variable is already set correctly.

⁵Depending on the installation, the *PATH* already includes this *bin* sub-directory.

⁶The example can be found in the `\examples\gO Product examples\gOSIMULINK` sub-directory of the gPROMS installation (*GPROMSHOME*).

⁷The mdl files can be found in `\examples\gO Product examples\gOSIMULINK\mdl` sub-directory of the gPROMS installation (*GPROMSHOME*).

Chapter 2. Exporting gPROMS models to Simulink

gPROMS model preparation

The following fundamental steps are required to prepare a model for use with gO:Simulink.

1. Develop a component model inside the gPROMS ModelBuilder environment.
2. Build a composite model around an instance of the component model. This composite model may or may not include a topology and instances of other models that you wish to translate to Simulink when the process model is exported.
3. Test that the composite model solves robustly in gPROMS and create a Saved Variable Set that stores the initial values of all the simulation variables.
4. Export the composite model from gPROMS ModelBuilder for use with Simulink.

The composite model will be translated into a Simulink flowsheet model (mdl file) which will include the component model as a gO:Simulink block.

The component (gO:Simulink) model

This model to be solved using gO:Simulink may involve the full range of mathematic systems supported by gPROMS, i.e. in the most general case, integro-partial differential-algebraic equations (IPDEAs) in time and one or more other independent variables, subject to IF and CASE-type discontinuities.

The model must have inlet and outlet Ports defined as these will constitute the inputs and outputs of the Simulink block - refer to the gPROMS ModelBuilder guide for more information on Model Port construction. The definition of the inputs and outputs for the gO:Simulink block will be automatically created during the export.

The composite model

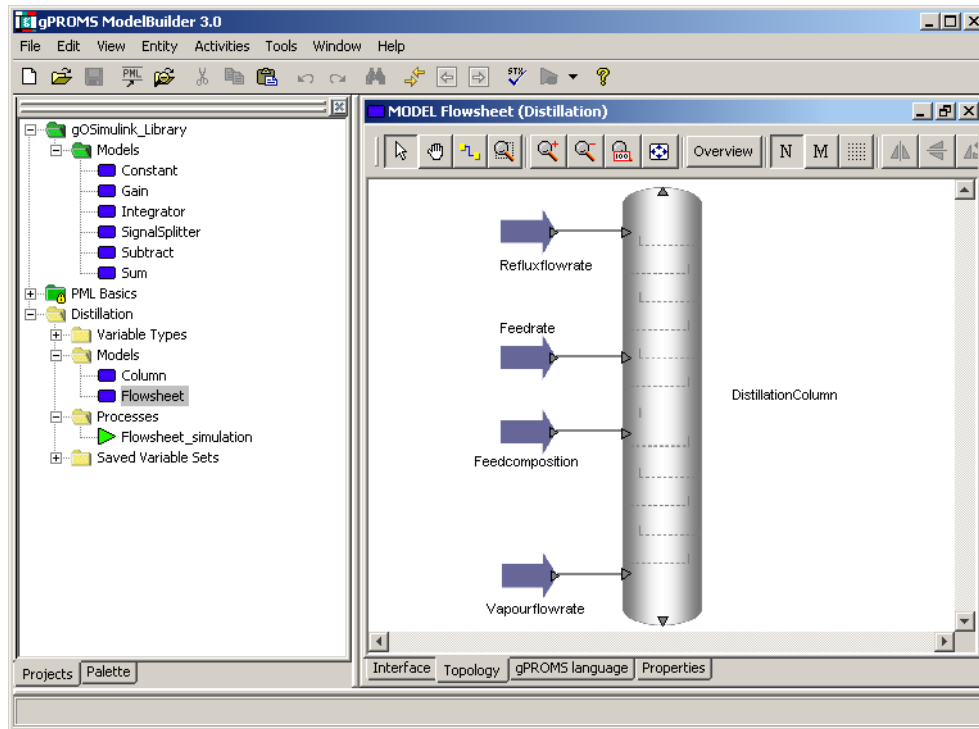
The composite model will typically have a defined topology containing

- no, one or more gPROMS *Units* based on Model from the gO:Simulink library¹ - this capability is described in more detail in the chapter Model translation.
- a graphical topology.

Note that the Export to Simulink utility can also be used to convert a composite model where all Units are translated to Simulink equivalents and there is no requirement for a gO:Simulink block.

¹The gO:Simulink library (*gOSimulink_library.gPJ*) can be found in the gOSimulink sub-directory of the GPROMSHOME directory.

**Figure 2.1. A simple distillation column example
of a composite model to be exported to Simulink**



A simple example is shown in in figure A simple distillation column example of a composite model to be exported to Simulink. The topology of the model shows:

- an instance *DistillationColumn* of a gPROMS Model entity *Column* with four inlet and two outlet Ports. This model will be solved in Simulink as a gO:Simulink block.
- four instances of a gPROMS model *Constant* from the gO:Simulink_Library. They are named *Refluxflowrate*, *Feedrate*, *Feedcomposition* and *Vapourflowrate*. These four instances will be *translated* into equivalent Simulink blocks during the export.

The *Export to Simulink* utility

The Process Entity

The specification for the composite model must be given in an associated Process entity - the specification set and the generic model (taken together) are termed the *process model*. Remember that before exporting the process model to Simulink, it is essential to first check that the model can be robustly simulated inside the gPROMS ModeBuilder environment.

The *Export to Simulink* dialog box

Select the Process Entity forming the process model that is to be exported and choose *Export to Simulink* from the ModeBuilder *Tools* menu. This will activate the Export to Simulink dialog box.

Figure 2.2. Export to Simulink dialog box.

The meaning of the fields of the Export to Simulink dialog box are:

Export Directory	The Simulink working directory to which the Simulink and gPROMS model files should be exported.
Dictionary directory	The directory in which dictionary files are located.
Simulink model (mdl) file	The name to give the Simulink model file that is to be created.
Overwrite previously exported files	Confirm whether to overwrite previously exported files.
gPROMS based Simulink block gPROMS unit entity	The name of the Unit that is to be solved as the gO:Simulink block (any other Units will be automatically translated into equivalent Simulink blocks using the gO:Simulink dictionary).
Use assignments from saved variable sets	This selects the Variable Set entity from which the values for the initial values of input Port variables are taken from (to allow the open model to be run) ^a
Include generated open loop process and model in the current project	Selecting this option will add to the project a Model and a corresponding Process entity equivalent to the process model solved in the Simulink environment via the gO:Simulink block.
Encryption password	An encryption password must be provided to run a model outside the ModelBuilder environment.
Decryption password (optional)	If a decryption password is provided the exported files can be re-imported into ModelBuilder, e.g. for debugging. Not providing an decryption password will encrypt the files without giving the option to decrypt them in ModelBuilder, fully protecting the content from the end-user of the exported files.

^aIf a Saved Variable Set is not available then the initial values for the Port variables cannot be determined automatically and the generated model will not have the correct number of degrees of freedom - therefore it is absolutely necessary that one is always used.

Exported files

The export operation adds a number of files to the specified Export Directory:

- a Simulink model (mdl) and a MATLAB (m) file
- an encrypted gPROMS input (gENCRYPT) and a gO:Simulink (gSF) file (in the input sub-directory)
- (if appropriate) one or more gPROMS Saved Variable Set (gSTORE) files (in the save sub-directory)
- (if appropriate) one or more Miscellaneous Files² that were contained in the gPROMS Project

The gO:Simulink (gSF) file

The gSF file should not be changed by the user, but can be viewed in order to retrieve the information about the inputs and outputs of the gO:Simulink block. The gSF file for the distillation example is shown below.

Table 2.1. gSF file of the distillation column example

INPUTFILE
Column
PASSWORD
gOSimulink
OL_Flowsheet_simulation_0
INPUTPORT
FS.COLUMN.REFLUXFLOWRATE.SIGNAL
INPUTPORT
FS.COLUMN.FEEDFLOWRATE.SIGNAL
INPUTPORT
FS.COLUMN.FEEDCOMPOSITION.SIGNAL
INPUTPORT
FS.COLUMN.VAPOURFLOWRATE.SIGNAL
OUTPUTPORT
FS.COLUMN.DISTILLATCOMPOSITION.SIGNAL
OUTPUTPORT
FS.COLUMN.BOTTOMCOMPOSITION.SIGNAL

The gSF file contains:

- the filename of the encrypted gPROMS input file which is to be used by gO:Simulink
- the password to run the model
- the name of the Process Entity that will be simulated
- the names of the gPROMS variables which are the inputs and outputs of the Simulink block.

²For example, Multiflash configuration (.mfl files).

Chapter 3. Using gO:Simulink

Loading the Simulink model

As outlined in in section Exported files, following the export of the gPROMS process model a Simulink model (mdl) and a MATLAB (m) file are created:

- The Simulink model (mdl) file contains information on all the Simulink blocks.
- The MATLAB (m) file contains the information about the connectivity between the blocks.¹

Loading the MATLAB m-file will automatically open the Simulink model (mdl) file in Simulink and generate the connections between the Simulink blocks. Therefore the following steps should be executed when loading a generated Simulink model for the first time:

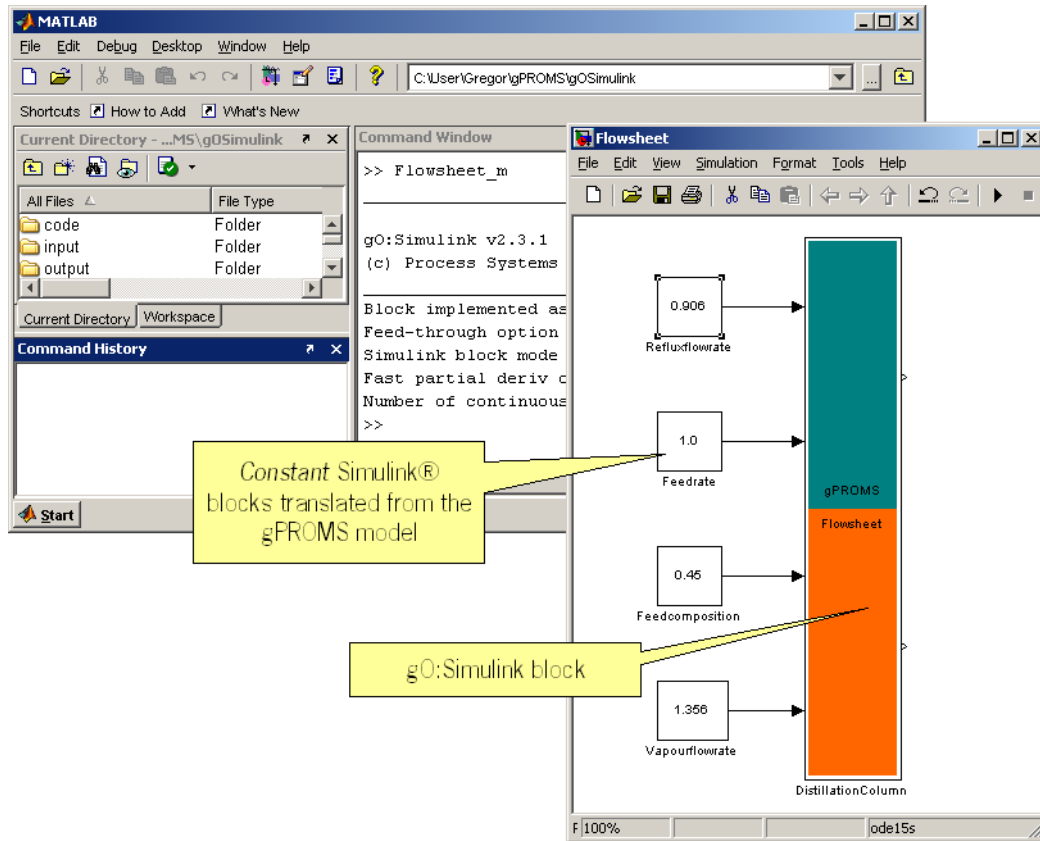
- Start MATLAB and change the directory to the Simulink working directory.
- Load the MATLAB m-file created by the *Export to Simulink* utility.
- This will automatically open the Simulink model (mdl) file and will generate all the connections between the Simulink blocks.
- Save the mdl file by selecting "File->Save" from the menu of the Simulink window.

Next time the Simulink model is opened the Simulink model (mdl) file can be loaded directly without using the MATLAB m-file.²

The distillation column example is shown in figure The exported Simulink model of the distillation example. The gO:Simulink model is clearly visible on the flowsheet model. Four Simulink *constant* blocks can also be seen. These have been *translated* from the gO:Simulink library models - compare the translated model with its gPROMS equivalent shown earlier in figure A simple distillation example of a composite model to be exported to Simulink.

¹Currently it is not possible to export the blocks and the connectivity into a single mdl file.

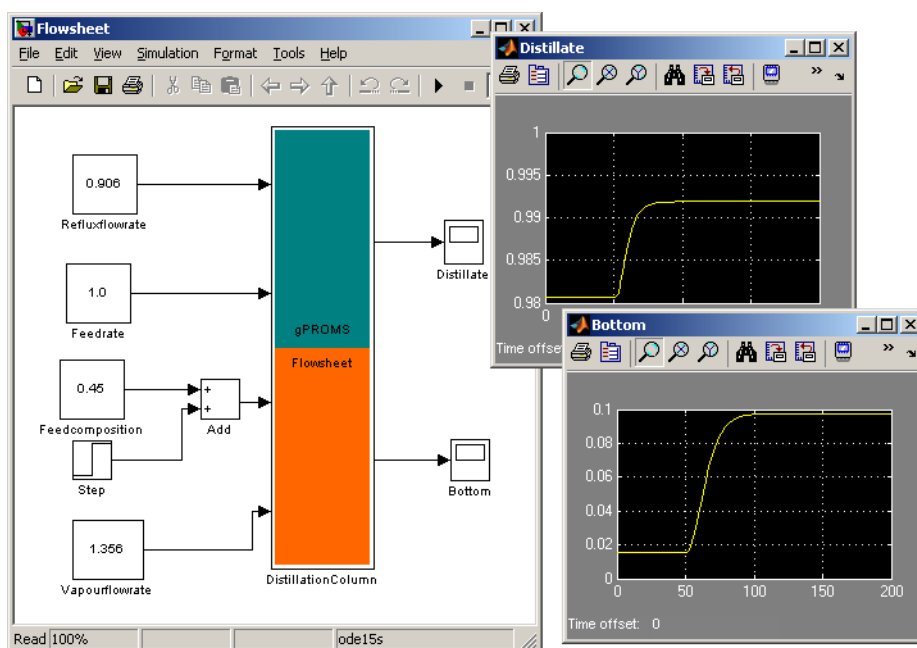
²If the Matlab m-file is loaded for a second time, after the Simulink model (mdl) file has been saved, several warning messages will be issued.

Figure 3.1. The exported Simulink model of the distillation example.

Executing a simulation

A gO:Simulink simulation is started in the normal way using the *play* button on the Simulink toolbar.

Results can be inspected using standard Simulink facilities such as *displays* or *scope* blocks. Of course, additional Simulink blocks can be added to the Simulink flowsheet, making full use of all features Simulink offers to the user. For the distillation column example two scopes have been added to display the simulation results as well as a step function and a sum block to impose a step on the feed composition input variable. The modified Simulink model and the simulation results are shown in the figure A gO:Simulink Simulation [9].

Figure 3.2. A gO:Simulink Simulation

Simulink solver settings

As most gPROMS model of interest tend to be relatively stiff, it is recommended that the user selects the *ode15s* solver in Simulink (especially if making use of the continuous-time mode of gO:Simulink). The solver selection and other Simulink *Simulation Parameters* can be adjusted from the *Simulation* menu. The Simulink model (mdl) file created by the *export to Simulink* utility will use the *ode15s* by default.

gO:Simulink options

The gO:Simulink block can be configured by the user. To access the configuration utility simply double-click the gO:Simulink block (see the figures gO:Simulink configuration dialog box: *Continuous-time* case and gO:Simulink configuration dialog box: *Discrete-time* case)

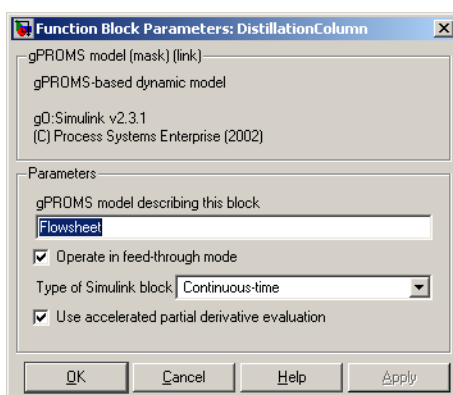
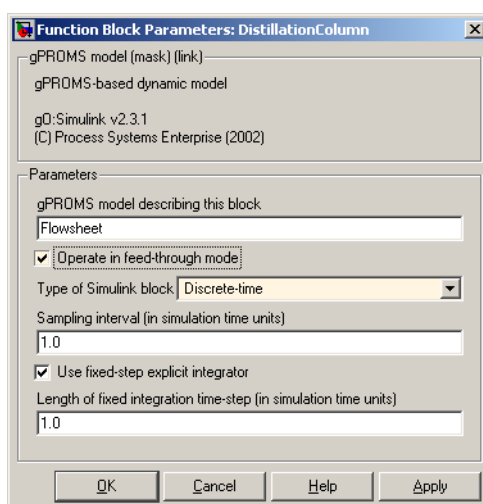
Figure 3.3. gO:Simulink block configuration dialog box: *Continuous-time* case.

Figure 3.4. gO:Simulink block configuration dialog box: *Discrete-time* case.

The explanation of the various items in the configuration dialog boxes is:

- *gPROMS model describing the block* - the .gSF file being used (it is not recommended to change this).
- *feed-through mode* - if selected this means that the block informs Simulink that its outputs are directly dependent on its inputs. Leaving this box unchecked often accelerates the solution process without introducing significant inaccuracies.
- A drop-down list is used to indicate whether the block is to be solved using the *continuous-time* or *discrete-time* solution methods (cf. chapter Continuous and discrete-time modes).
- In *Continuous-time* mode a checkbox is available to select whether to use accelerated partial derivative calculation or not. It is recommended that this is used.
- If *Discrete-time* mode is selected, gPROMS' own integrator is used to integrate the gO:Simulink block over a single sampling interval. The length of the sampling time interval needs to be specified. You may also choose to use one of gPROMS' variable-step variable-order implicit integrators (recommended), or a much simpler fixed step explicit one. In the latter case a positive value for the time step must be provided.

Continuous and discrete-time modes

- When solved as a *continuous-time* block: gO:Simulink uses gPROMS' powerful nonlinear equation solvers to automatically transform the implicit DAE (gPROMS) model into an explicit ODE model that Simulink can use directly.
- When solved as a *discrete-time* block: gPROMS solvers integrate the gPROMS model over each sampling interval. Simulink's solver sees only the block input and output variables - the gPROMS model itself is invisible to Simulink.

The selection of the most appropriate option depends on the system under consideration and, in particular, the process time constant: for systems where a very small sampling interval would be required the discrete-time mode may be inappropriate.

Chapter 4. Model translation

Model translation overview

Units on the gPROMS flowsheet model that are based on simple flowsheeting models (e.g. sources and sinks) and control system models can be automatically translated into equivalent Simulink blocks. This capability is supported by a gO:Simulink library and a translation dictionary.

The dictionary defines how a gPROMS model can be translated into an equivalent Simulink block. The following fundamental pre-requisitions have to be met:

- A gPROMS model with the same name as a Simulink block (with equivalent mathematical properties) must be present in the ModelBuilder project (or cross-referenced library).
- A dictionary entry (again with the same name) that relates the gPROMS model to its Simulink equivalent.

Note that while the gO:Simulink block of an *exported* process model contains the full mathematical model, the *translation* of Units via a dictionary entry does not involve any conversion of the gPROMS equations into equivalent Simulink expressions. Instead during the *Export to Simulink* the gPROMS Unit is effectively replaced with an existing Simulink block¹.

The gO:Simulink library

The gPROMS installation includes a gO:Simulink library (*gOSimulink_Library.gPJ*) and associated dictionary which includes a number of gPROMS models equivalent to some standard Simulink blocks:

- Constant
- Demux
- Gain
- Integrator
- Mux
- SignalSplitter
- Subtract
- Sum

The gO:Simulink library and dictionary are located in the gPROMS\gOSimulink folder of the gPROMS installation directory².

The *Export to Simulink* dictionary

The default dictionary used when a process model is exported to Simulink contains entries for all models in the gO:Simulink library: a dictionary entry corresponds to a single xml file. Note that all dictionary entries needed during the export have to be located in a single directory - the default dictionary directory can be changed, if desired, by the user in the *Export to Simulink* dialog box at the time of the export (see the Export to Simulink dialog).

¹Thus, the behaviour of the gPROMS model is only identical with that of the Simulink block if the user supplies a Simulink block which is a mathematically identical representation of the gPROMS model.

²The gPROMS installation directory is typically: c:\program files\pse

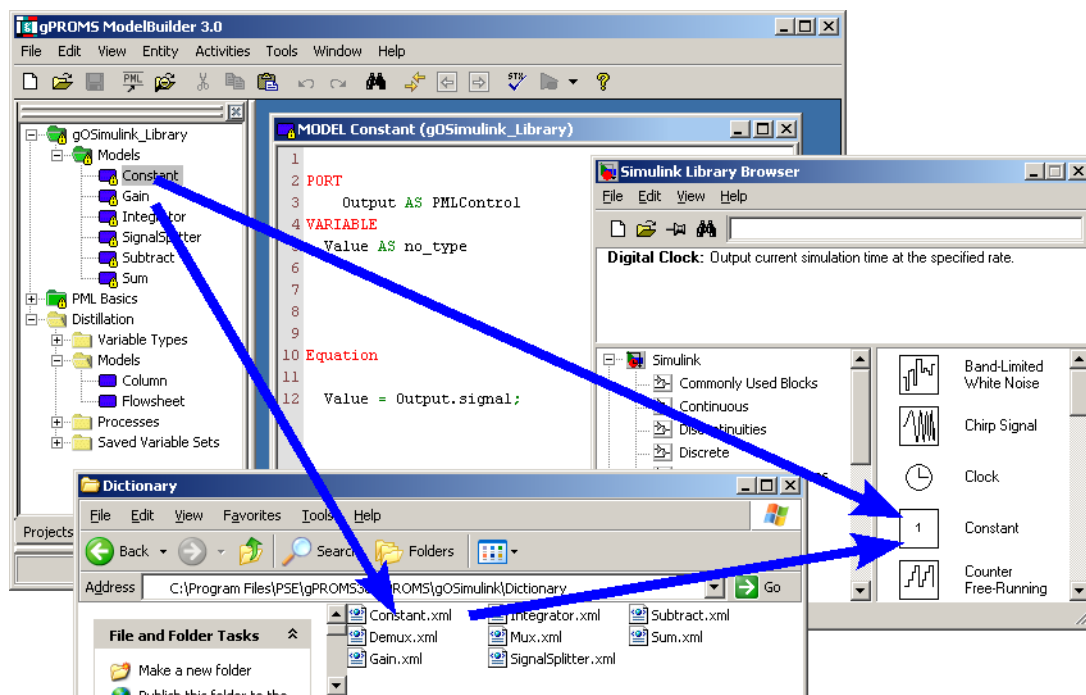
The main function of each dictionary entry is to determine which input quantities should be passed from the gPROMS model to the Simulink model. Values can only be passed for those gPROMS parameters and variables whose names are identical to the corresponding names in the Simulink block.

For example, the *Constant* gPROMS model contains a variable named *Value*. The Simulink block *Constant* has a parameter called *Value*. During the *Export to Simulink* the input value of the gPROMS variable *Value* is written to the Simulink model (mdl) file, thus defining the value of the corresponding quantity in the Simulink *Constant* block.

Dictionary files

To enable the translation of any appropriate gPROMS Model to Simulink the dictionary used by gPROMS can be fully extended by the user. This is done by defining some additional xml dictionary files: remember that the name of the xml file must be identical to the names of the gPROMS model and the Simulink block. So for example the *Constant* gPROMS model in the gO:Simulink library relates to the *Constant* Simulink block and the dictionary entry is *Constant.xml*.

Figure 4.1. Example of the naming convention relating a gPROMS model to a Simulink block.



The xml file format of the dictionary entry file is:

Figure 4.2. The syntax of the dictionary entry file

```
<Model>
  <BlockType>
  </BlockType>

  <SourceBlock>
  </SourceBlock>

  <SourceType>
  </SourceType>

  <Parameters>
    <Name>
    </Name>
  </Parameters>

  <Property>
  </Property>
</Model>
```

The `<Model>` and `<BlockType>` tags are compulsory. All other tags (`<SourceBlock>`, `<SourceType>`, `<Parameter>` and `<Property>`) are optional.

The `<Model>` tag

The file must start with the `<Model>` tag and end with the `</Model>` tag.

The `<BlockType>` tag

This is the type of the Simulink block like Constant, Integrator, etc. If the block is a reference from a library the `BlockType` should be specified as a *Reference*.

The `<SourceBlock>` tag

This entry is needed only if the `BlockType` is specified as a *Reference*. This should give the full path of the block, for example:

- `gOSimulink/Splitter` - `gOSimulink` is the Simulink library name and `Splitter` is the block name.
- `fixpt_lib_4/Bits/Bitwise\nOperator` - here `fixpt_lib_4` is the Simulink library name, `Bits` is the sub-library and `Bitwise\nOperator` is the block.

The `<SourceType>` tag

This is the short description of the source type - e.g.: "Signal Splitter" or "Fixed-point Bitwise Operator".

The `<Parameter>` tag

Parameters are the names of the gPROMS Parameters/Variables that are inputs to the model (i.e. SET or ASSIGNED in the process model). The parameter names should be specified within the `<Name></Name>` tag. More than one parameter can be specified:

Figure 4.3. Specifying more than one `<Parameter>` tag

```
<Parameters>
  <Name>
    paramName1
  </Name>
  <Name>
    paramName2
  </Name>
</Parameters>
```

Remember that the gPROMS parameter and variable names should be identical to the corresponding names in the Simulink block.

The `<Property>` tag

Block properties can be specified using the `<Property>` tags. More than one property can be specified, but each one should have its own `<Property></Property>` tag.

Figure 4.4. Specifying more than one `<Property>` tag

```
<Property>
  DisplayOption "bar"
</Property>
<Property>
  Inputs "+-"
</Property>
```

Chapter 5. Exporting Simulink flowsheets to gPROMS

Software requirements

In order to export a model from Simulink to gPROMS you need to have:

- Matlab.¹
- Simulink.
- The Real-Time Workshop (RTW).
- The Microsoft Visual C++ 6.0 compiler.
- gPROMS v3.0.3 or later.

System configuration

If gPROMS and Matlab are installed in the default directories no system configuration should be necessary. But it is strongly recommended to check the system configuration before proceeding with the export, compilation and linking.

The system configuration can be changed (if necessary) in the BuildFO.bat file residing in the gPROMS \bin sub-directory of the gPROMS installation directory (typically c:\Program Files\PSE). Open the BuildFO.bat file using a text editor (like notepad.exe). Check the following lines of the batch file to confirm that the batch file is compatible with your installation:

- Line 6: INSTALDIR should point to your gPROMS installation directory.²
- Line 20: MATLAB_HOME should point to your matlab installation directory.³

If necessary change the settings according to your installations.

Exporting a Simulink flowsheet

Exporting a Simulink flowsheet is done in the steps:

1. Preparing the Simulink flowsheet for the export.
2. Configuring the Real-Time Workshop for the export.
3. Exporting of the Simulink flowhseet.
4. Generation of the Foreign Object containing the exported flowsheet.⁴

Once the Foreign Object containing the Simulink flowsheet is generated it is ready to use in gPROMS.

Preparing the Simulink flowsheet for the export

The following steps need to be taken to perpare the Simulink model for exoprt:

1. The Simulink model to be exported must be saved in a mdl file naked "SimulinkTogPROMSTemplate.mdl".⁵

¹The currently supported Matlab release is R14 Service Pack 3 (Matlab 7.1)

²The default value is c:\Program Files\PSE.

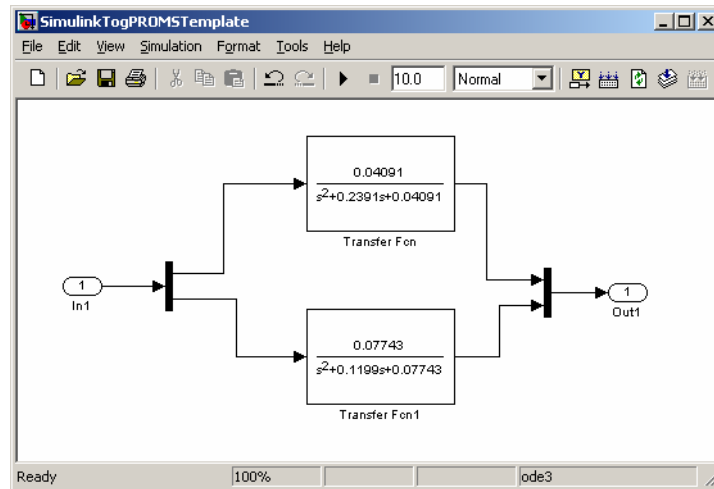
³The default value for Matlab 7.1 is c:\Program Files\MATLAB71.

⁴The exported flowsheet is represented in gPROMS as a Foreign Object.

⁵This is essential because this name is used during the RTW code generation. Any other name will not be compatible with the naming convention in the source code provided by PSE.

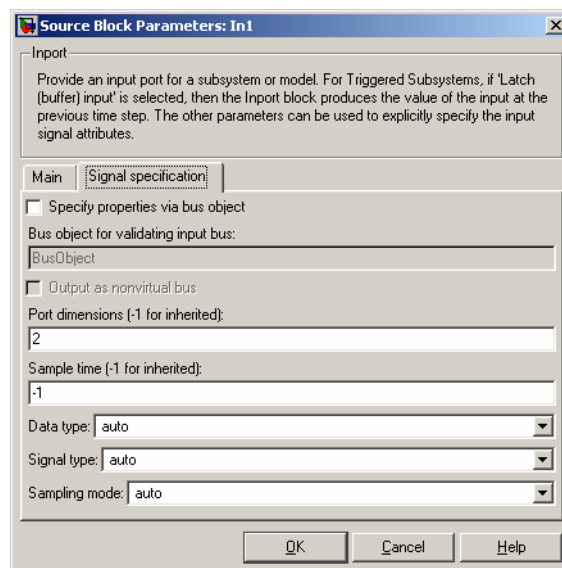
2. The Simulink flowsheet may contain any number of blocks.

Figure 5.1. Example of a flowsheet ready for export.



3. The Simulink flowsheet must contain exactly one input port called "In1". The port may be used to receive either a scalar value or a vector from gPROMS. The size of the port needs to be configured by the user. Double click on the In1 port to access the port parameters and select the "Signal specification" tab. Enter the total number of values passed from gPROMS to Simulink in the "Port dimensions" field.

Figure 5.2. Configuring the size of the input port In1.



4. The Simulink flowsheet must contain exactly one output port called "Out2". The port may be used to send either a scalar value or a vector to gPROMS. The size of the output port is automatically determined during the RTW code generation.
5. Save the SimulinkTogPROMSTemplate.mdl file containing the flowsheet.

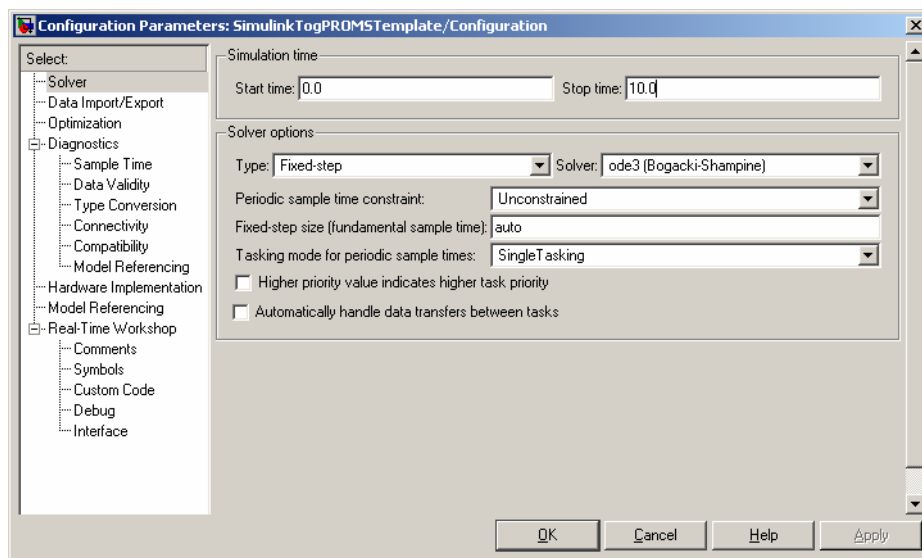
The flowsheet is now ready for export.

Configure the Real-Time Workshop (RTW) for export

Specific settings and options are required to export the flowsheet using the Real-Time Workshop. The configuration dialogue of the RTW can be accessed by selecting "Configuration Parameters" from the "Simulation" menu of the SimulinkTogPROMSTemplate.mdl window. The configuration steps are:

1. Setting of the Simulink solver options⁶.
 - a. Set the "Type" in the "Solver options" to "Fixed-step".
 - b. Set the "Solver" in the "Solver options" to "ode3 (Bogacki-Shampine)".
 - c. Set the "Tasking mode for periodic sample times" to "SingleTasking".⁷

Figure 5.3. Configuring the Simulink solver settings.



2. Setting of the Real-Time Workshop configuration.
 - a. Set the "System target file" in the "Target selection" to "grt_malloc.tlc" in the "System target file browser" by clicking the "Browse..." button.
 - b. Check that the "Language" is set to "C".
 - c. Select the option "Generate code only".
 - d. De-select the option "Generate HTML report".
 - e. Click the "Apply" button to apply the settings.

⁶The selections of these options has no impact on the gPROMS solver settings or on the numerical accuracy of the solution obtained in gPROMS. These settings are required to make the exported code compatible with the Real-Time Workshop requirements.

⁷It is essential to set the "SingleTasking" mode in case that more than one sampling rate is used in the model. Using other settings will result in an error message during the export step.

Figure 5.4. Configuration of the Real-Time Workshop.

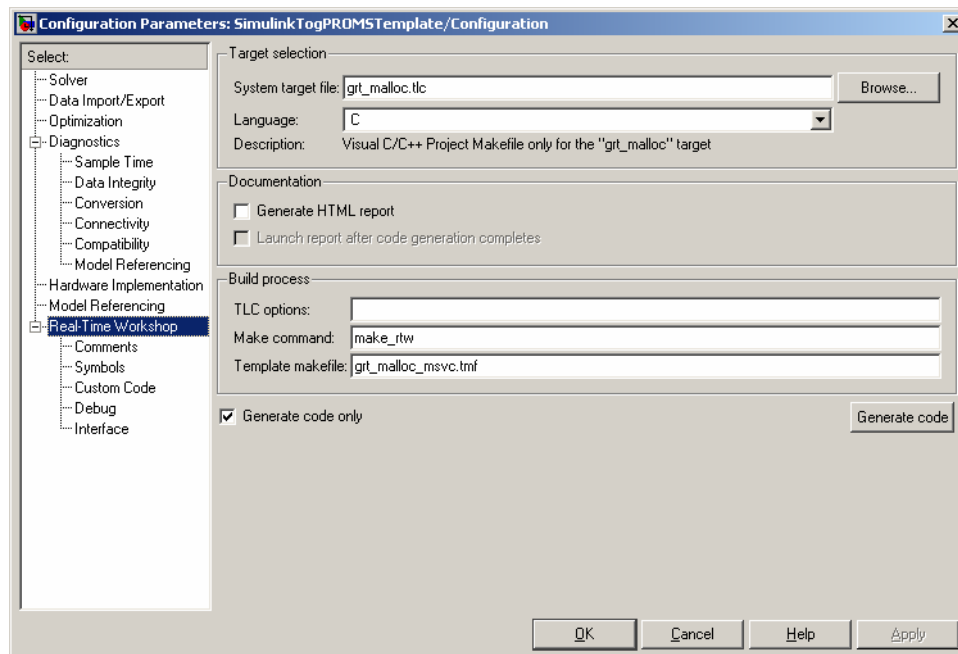
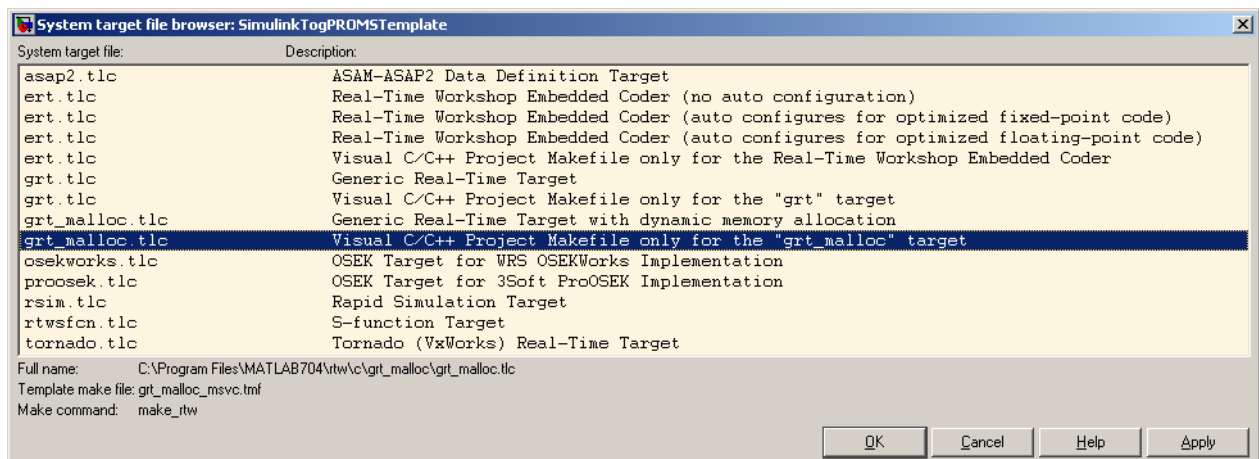


Figure 5.5. Selecting the system target file in the target browser.



Once this configuration has been set the system is ready for exporting the Simulink flowsheet using the Real-Time Workshop.

Exporting the Simulink flowsheet

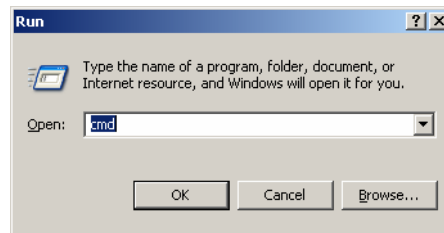
Before exporting the flowsheet make sure that the model and the Real-Time Workshop are configured correctly. Export the model selecting "Real-Time Workshop->Build Model" from the "Tools" menu of the Simulink model.

The Real-Time Workshop will create a sub-directory SimulinkTogPROMSTemplate_grt_malloc_rtw containing the exported Simulink flowsheet.

Generation of the Foreign Object containing the Simulink flowsheet

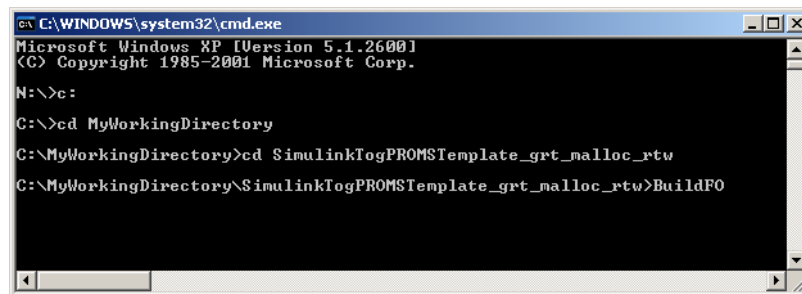
Once the Simulink flowsheet is exported a Foreign Object can be created using the BuildFO.bat [15] file. The batch file needs to be called from the Windows cmd window. To open the cmd window run cmd.exe from the "Run" window in Windows. To open the "Run" window click on "Start" in the Windows XP taskbar and select "Run".

Figure 5.6. Opening a command window using the "Run" window.



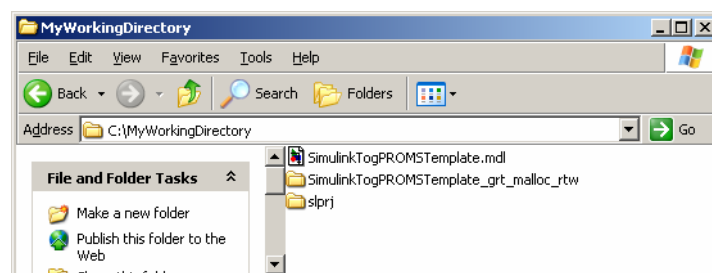
Once the cmd window is open change to the directory where the SimulinkTogPROMSTemplate.mdl file is located by using the "cd" command:

Figure 5.7. Using the "cd" command to change the directory



Change to the directory SimulinkTogPROMSTemplate_grt_malloc_rtw which is automatically generated during the export of the model using the Real-Time Workshop.

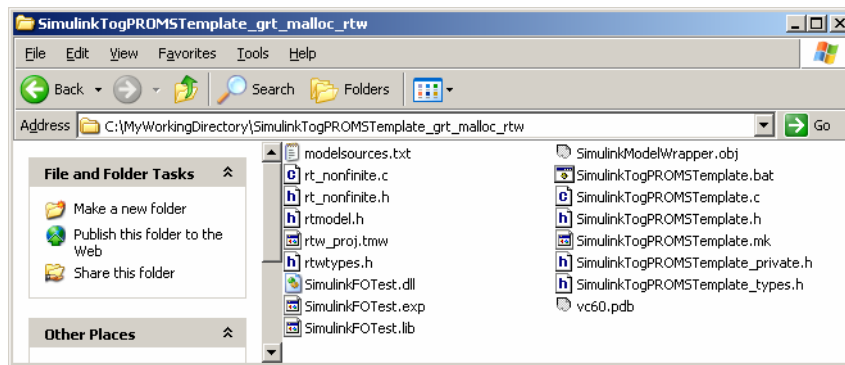
Figure 5.8. Directory structure of the exported Simulink model



Invoke BuildFO.bat file from the cmd window to generate the Foreign Object.⁸ The Foreign Object will be generated as a file named SimulinkFOTest.dll residing in the SimulinkTogPROMSTemplate_grt_malloc_rtw directory.

⁸Depending on your gPROMS installation the gPROMS\bin sub-directory of your gPROMS installation directory (typically c:\program files\pse) will not be in your Windows PATH. In that case you either need to include this directory manually in the PATH or call the BuildFO.bat file by using the full path to the file.

Figure 5.9. The generated ForeignObject in form of the SimulinkFOTest.dll file



Of course the dll can be renamed if necessary. The easiest way to make the Foreign Object accessible from within a gPROMS model is to put the file into the `gPROMS\fo` sub-directory of the gPROMS installation directory.

Chapter 6. Using a Simulink flowsheet in gPROMS

Once the dll containing the Foreign Object representing the exported Simulink model is generated it is ready to use within a gPROMS model. Like all Foreign Objects the Foreign Object supports a set of methods supporting various activities necessary to use it within a gPROMS model.

There are three ways to make use for the Foreign Objects containing the exported Simulink models:

1. Using the gOSimulink_Import library and the examples provided with the installation.
2. Writing new user models using the various methods provided by the Foreign Object.
3. Using the library and the examples as templates for new user models.

In many cases the first option will be sufficient since the library and the examples are generic and will cover many standard applications. However, extending the models (third option) or even writing completely new models may be necessary in specific applications. It is recommended to get a general understanding of the various methods provided to get a better understanding of the library and the examples.

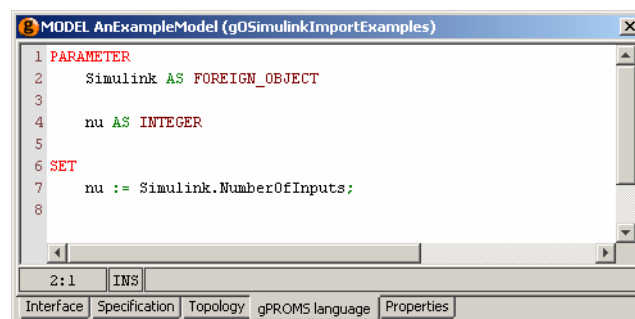
Methods provided by the Foreign Object containing the Simulink flowsheet

NumberOfInputs

The method NumberOfInputs returns structural information about the Simulink flowsheet: It gives the number of values expected by the Foreign Object to be passed from gPROMS. This is equivalent to the number defined in the In1 block of the Simulink flowsheet.

- Number of arguments to be passed to the method: None.
- Number of values passed back from the method: One.
- Data type returned: INTEGER

Figure 6.1. Example: Using the NumberOfInputs method

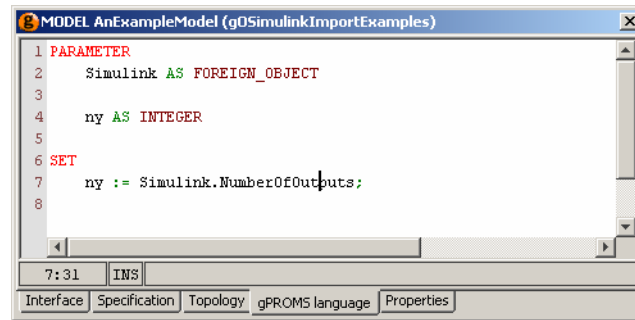


NumberOfOutputs

The method NumberOfOutputs returns structural information about the Simulink flowsheet: It gives the number of values returned by the Foreign Object to gPROMS. This is equivalent to the number of values in the Out1 block of the Simulink flowsheet.

- Number of arguments to be passed to the method: None.
- Number of values passed back from the method: One.
- Data type returned: INTEGER

Figure 6.2. Example: Using the NumberOfOutputs method

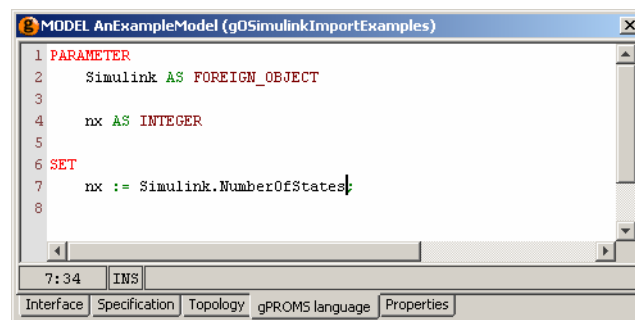


NumberOfStates

The method NumberOfStates returns structural information about the Simulink flowsheet: It gives the number of continuous states of the Simulink model.

- Number of arguments to be passed to the method: None.
- Number of values passed back from the method: One.
- Data type returned: INTEGER

Figure 6.3. Example: Using the NumberOfStates method

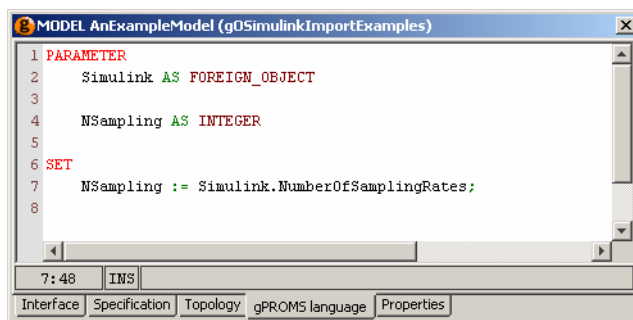


NumberOfSamplingRates

The method NumberOfSamplingRates returns structural information about the Simulink flowsheet: It gives the number of sampling rates used in the Simulink model.

- Number of arguments to be passed to the method: None.
- Number of values passed back from the method: One.
- Data type returned: INTEGER

Figure 6.4. Example: Using the NumberOfSamplingRates method



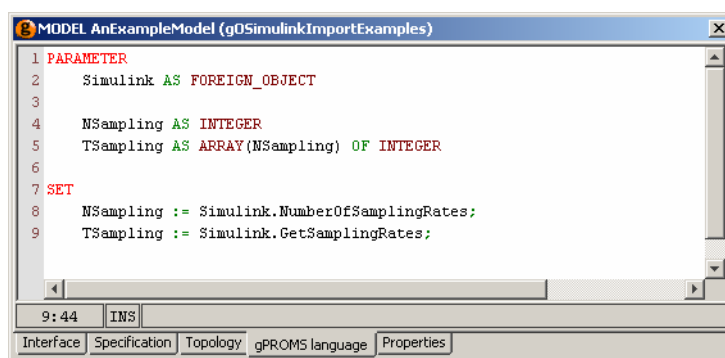
The number of sampling rates included in the model embedded in the Foreign Object may be larger than the number of sampling rates specified by the user in the Simulink flowsheet due to the code generation of the Real-Time Workshop. The most likely reason is that more than one sampling rate was specified by the user. During the export the smallest common sampling rate is added automatically by the Real-Time Workshop. This has no further impact on the usage of the model (except for the fact that one more sampling rate will be reported by the Foreign Object).

GetSamplingRates

The method GetSamplingRates returns parametric information about the Simulink flowsheet: It returns the values of all sampling rates used in the Simulink model.

- Number of arguments to be passed to the method: None.
- Number of values passed back from the method: ARRAY: Depends on the number of sampling rates used in the model.
- Data type returned: INTEGER

Figure 6.5. Example: Using the GetSamplingRates method



GetSmallestSamplingRate

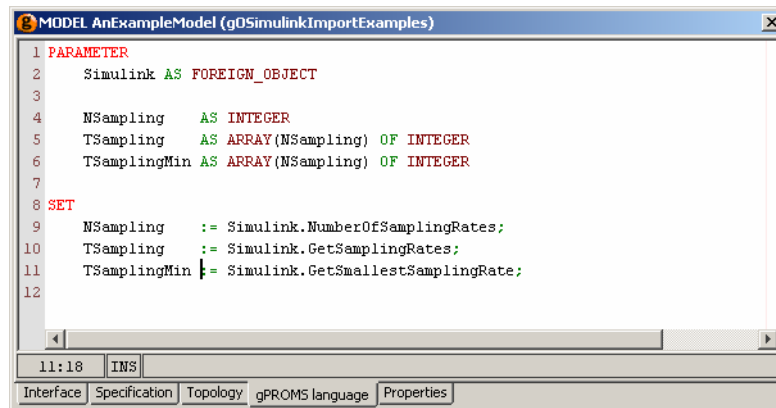
The method GetSmallestSamplingRate returns parametric information about the Simulink flowsheet: It returns the smallest sampling rate used in the Simulink model.¹

- Number of arguments to be passed to the method: None.
- Number of values passed back from the method: One.

¹The information about the smallest sampling rate may not be necessary in most applications. For convenience this method was introduced into the Foreign Object.

- Data type returned: INTEGER

Figure 6.6. Example: Using the GetSmallestSamplingRate method

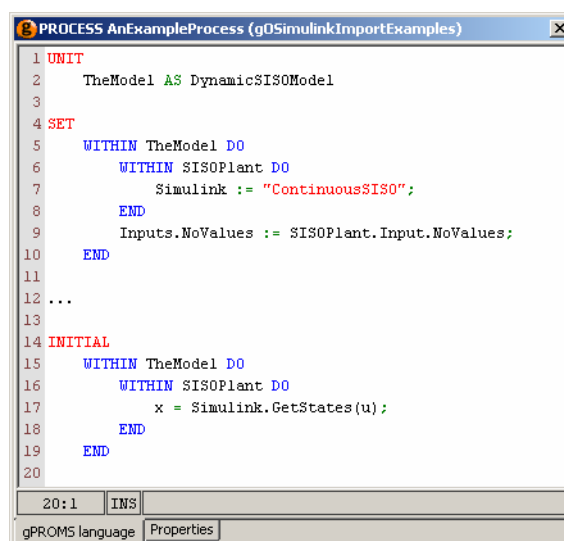


GetStates

The method GetStates can be used for models containing continuous time states. It returns the information about the initial conditions of the continuous states embedded in the Simulink flowsheet.²

- Number of arguments to be passed to the method: One - the values of the inputs in the input port In1 (number of inputs).
- Size of the argument to be passed to the method: Scalar or array - depending on the size of the input port In1.
- Data type: REAL.
- Number of values passed back from the method: Depends on the number of continuous states included in the Simulink flowsheet.
- Data type returned: REAL.

Figure 6.7. Example: Using the GetStates method



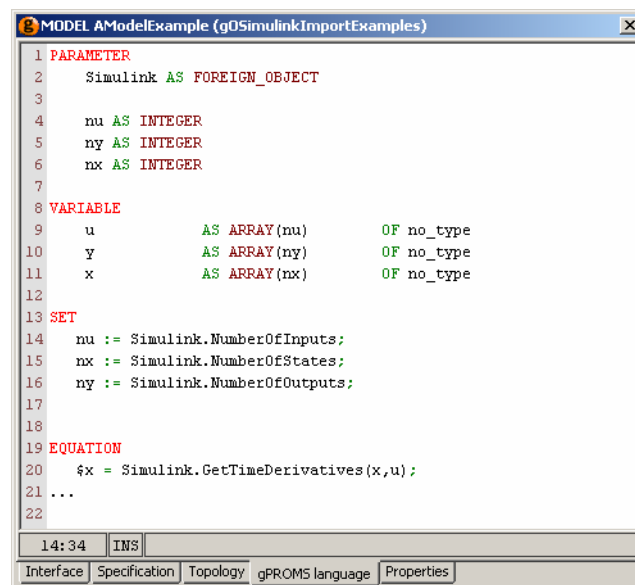
²Some Simulink blocks allow to define the initial conditions of the states. This information may or may not be used in the gPROMS model. The method GetStates allows to access this information. The values of the states may be calculated by the Simulink block and may depend on the input values.

GetTimeDerivatives

The method `GetTimeDerivatives` can be used for models containing continuous time states. It returns the derivatives of the continuous states with respect to time.

- Number of arguments to be passed to the method: Two:
 1. The current values of the inputs in the input port In1
 2. The current values of the states.
- Size of the argument to be passed to the method: Scalar or array - depending on the size of the input port In1 (number of inputs) and the number of continuous states:
 1. The size of the first argument must be equivalent to the number returned by the method `NumberOfStates`.
 2. The size of the second argument must be equivalent to the number returned by the method `NumberOfInputs`.
- Data type: REAL.
- Number of values passed back from the method: Depends on the number of continuous states included in the Simulink flowsheet.
- Data type returned: REAL.

Figure 6.8. Example: Using the `GetTimeDerivatives` method



```
1 PARAMETER
2   Simulink AS FOREIGN_OBJECT
3
4   nu AS INTEGER
5   ny AS INTEGER
6   nx AS INTEGER
7
8 VARIABLE
9   u      AS ARRAY(nu)      OF no_type
10  y      AS ARRAY(ny)      OF no_type
11  x      AS ARRAY(nx)      OF no_type
12
13 SET
14   nu := Simulink.NumberOfInputs;
15   nx := Simulink.NumberOfStates;
16   ny := Simulink.NumberOfOutputs;
17
18
19 EQUATION
20   $x = Simulink.GetTimeDerivatives(x,u);
21 ...
22
```

GetOutputs

The method `GetOutputs` returns the current values of the outputs in the port Out1 of the Simulink flowsheet. The usage of the method depends on if the the Simulink model contains any continuous time states.

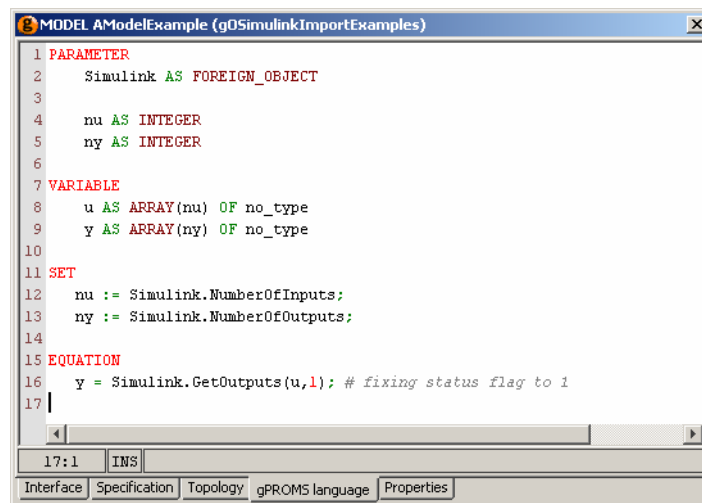
Simulink models containing no continuous states

Typically Simulink models containing no continuous states are purely algebraic models or time discrete models. In these cases the outputs of the Simulink model are functions of the current values of the inputs In1 only.

- Number of arguments to be passed to the method: Two:
 1. The current values of the inputs in the input port In1

2. The value of a status flag. This is necessary to synchronise the gPROMS numerics with the evaluation of the discrete-time states of the Simulink model embedded in the Foreign Object. A change in the status flag forces a synchronisation. In case of a purely algebraic model (containing no discrete states) this is not necessary and the second argument can be set to a fixed value (e.g. 1). The value itself has no impact on the calculation, only a change ensures a proper synchronisation.³
- Size of the argument to be passed to the method: Scalar or array - depending on the size of the input port In1 (number of inputs):
 1. The size of the first argument must be equivalent to the number returned by the method NumberOfInputs.
 2. The second argument is an array of the size returned by the method NumberOfSamplingRates. In case that no discrete states exist the second argument is a scalar.
 - Data type: REAL.
 - Number of values passed back from the method: Depends on the number of output values of the Simulink model.
 - Data type returned: REAL.

Figure 6.9. Example: Using the GetOutputs method for an algebraic model



Simulink models containing continuous states

In difference to Simulink models without continuous states Simulink models are functions of the current values of the inputs In1 and of the continuous states.

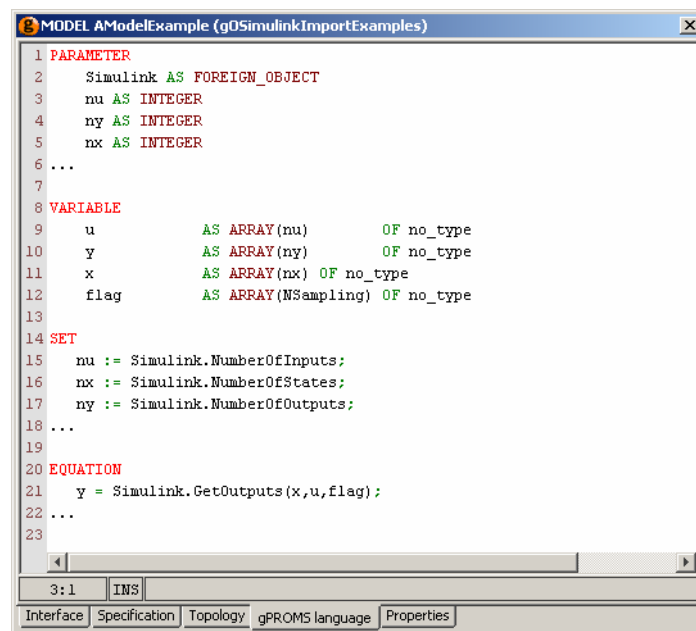
- Number of arguments to be passed to the method: Three:
 1. The current values of the inputs in the input port In1
 2. The current values of the continuous states.
 3. The value of a status flag. This is necessary to synchronise the gPROMS numerics with the evaluation of the discrete-time states of the Simulink model embedded in the Foreign Object. A change in the status flag forces a synchronisation. If the model does not contain time discrete states this is not necessary and the second argument can be set to a fixed value (e.g. 1). The value itself has no impact on the calculation, only a change ensures a proper synchronisation.⁴

³Typically this synchronisation is done in a TASK and not on the MODEL level.

⁴Typically this synchronisation is done in a TASK and not on the MODEL level.

- Size of the argument to be passed to the method: Scalar or array - depending on the size of the input port In1 (number of inputs):
 1. The size of the first argument must be equivalent to the number returned by the method NumberOfInputs.
 2. The size of the second argument must be equivalent to the number returned by the method NumberOfStates.
 3. The third argument is an array of the size returned by the method NumberOfSamplingRates. In case that no discrete states exist the second argument is a scalar.
- Data type: REAL.
- Number of values passed back from the method: Depends on the number of output values of the Simulink model.
- Data type returned: REAL.

Figure 6.10. Example: Using the GetOutputs method for a model containing continuous states

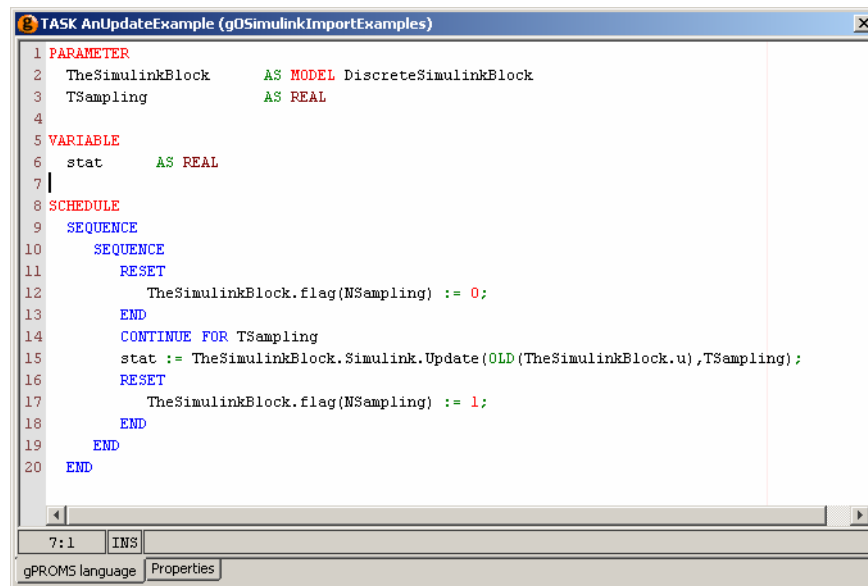


Update

The method Update can be used for models containing discrete time states. It enforces an update of the discrete time states of the Simulink model embedded into the Foreign Object.

- Number of arguments to be passed to the method: one: The current values of the inputs in the input port In1.
- Size of the argument to be passed to the method: Scalar or array - depending on the size of the input port In1 (number of inputs).
- Data type: REAL.
- Number of values passed back from the method: One. The value returned indicates if an update of the states has been successful. This information may or may not be used in a model or in a TASK.
- Data type returned: REAL.

Figure 6.11. Example: Using the Update method



The example TASK illustrating the usage of the Update method assumes that the MODEL of type "DiscreteSimulinkBlock" contains a ForeignObject with the name "Simulink". The update of the inputs u to the discrete time Simulink block is done in line 15. The synchronisations of the gPROMS and Simulink calculations is enforced using the change of the value of the flag variable.

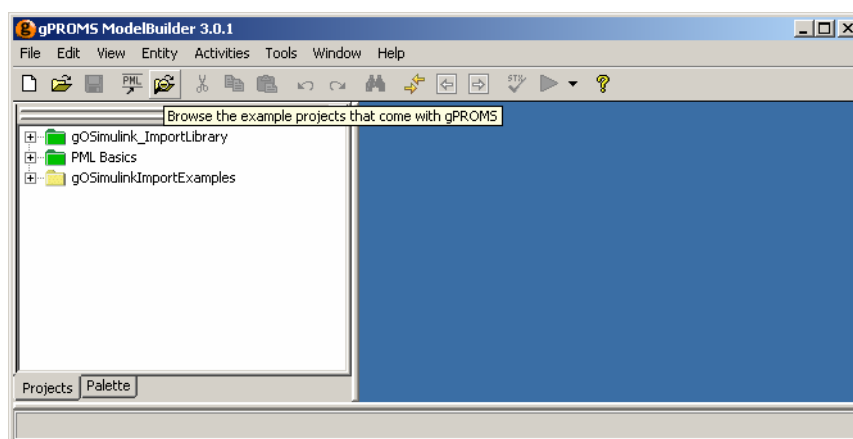
The gO:Simulink import library - content and examples

The gPROMS installation includes a library plus a couple of examples

- illustrating how to import models from Simulink and how to make use of the various methods provided by the Foreign Object and
- serving as templates to generate more sophisticated applications.

The gO:Simulink import library is located in the gPROMS/gOSimulink sub-directory of the gPROMS installation directory (by default: c:\Program Files\PSE). The examples can be accessed by browsing the examples projects. The Simulink models used in the examples are located in the directory gPROMS\examples\gO Product Examples\gOSIMULINK\mdl.

Figure 6.12. Browsing the example projects



The main content of the gO:Simulink import library are four models and two tasks.

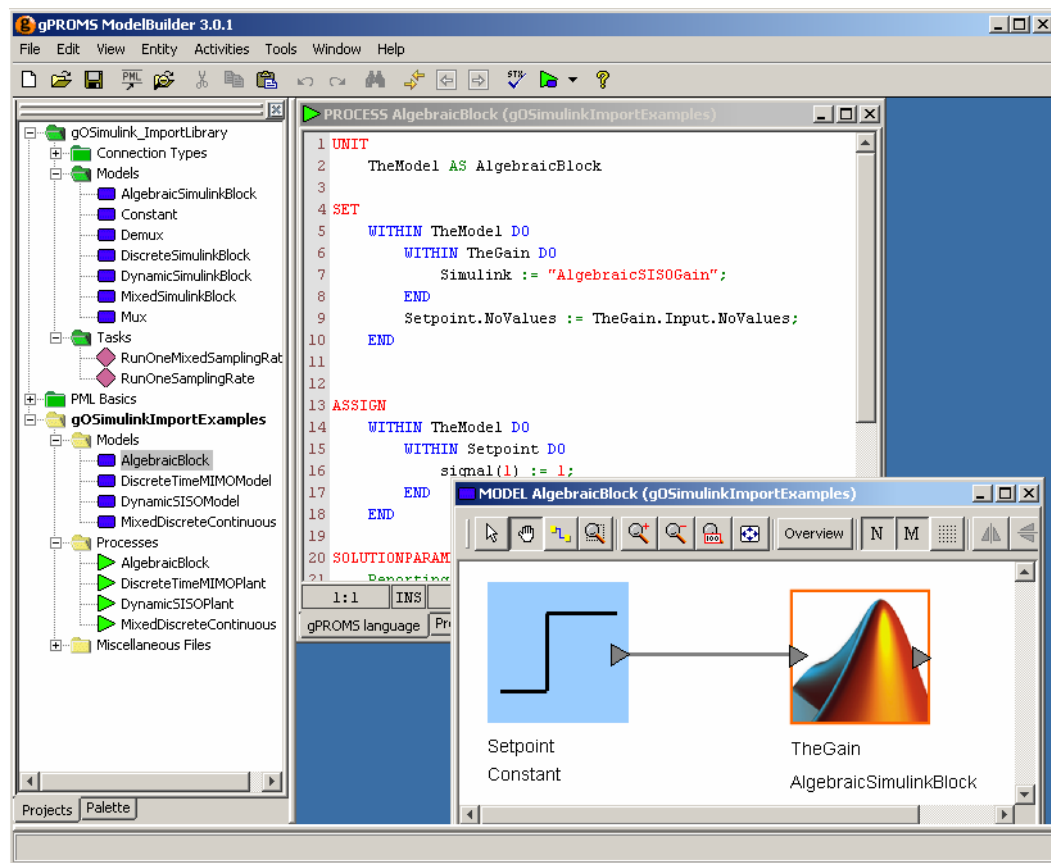
The AlgebraicSimulinkBlock model

The AlgebraicSimulinkBlock model can be used for any algebraic Simulink model (containing no continuous or discrete time states). The configuration of the model involves two steps:

1. Specifying the Foreign Object containing the Simulink model by setting the PARAMETER Simulink.
2. Specifying the input of the model by
 - ASSIGNing an appropriate number of values to the signal variable of the input PORT or
 - making an appropriate connection on a flowsheet to the input PORT of the model.

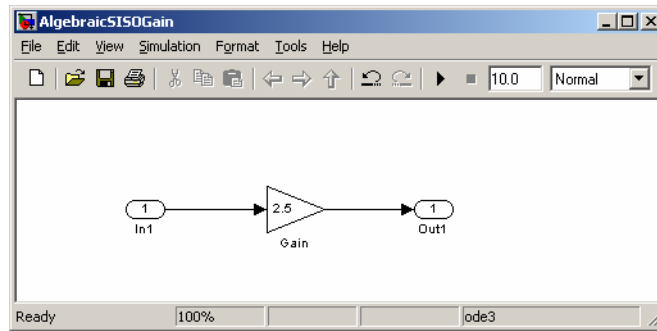
The usage of the model is illustrated by the example AlgebraicBlock.

Figure 6.13. Using the AlgebraicSimulinkBlock model



The Foreign Object used in the example is a static gain with one input and one output.

Figure 6.14. Simulink flowsheet of the SISO gain



The DynamicSimulinkBlock model

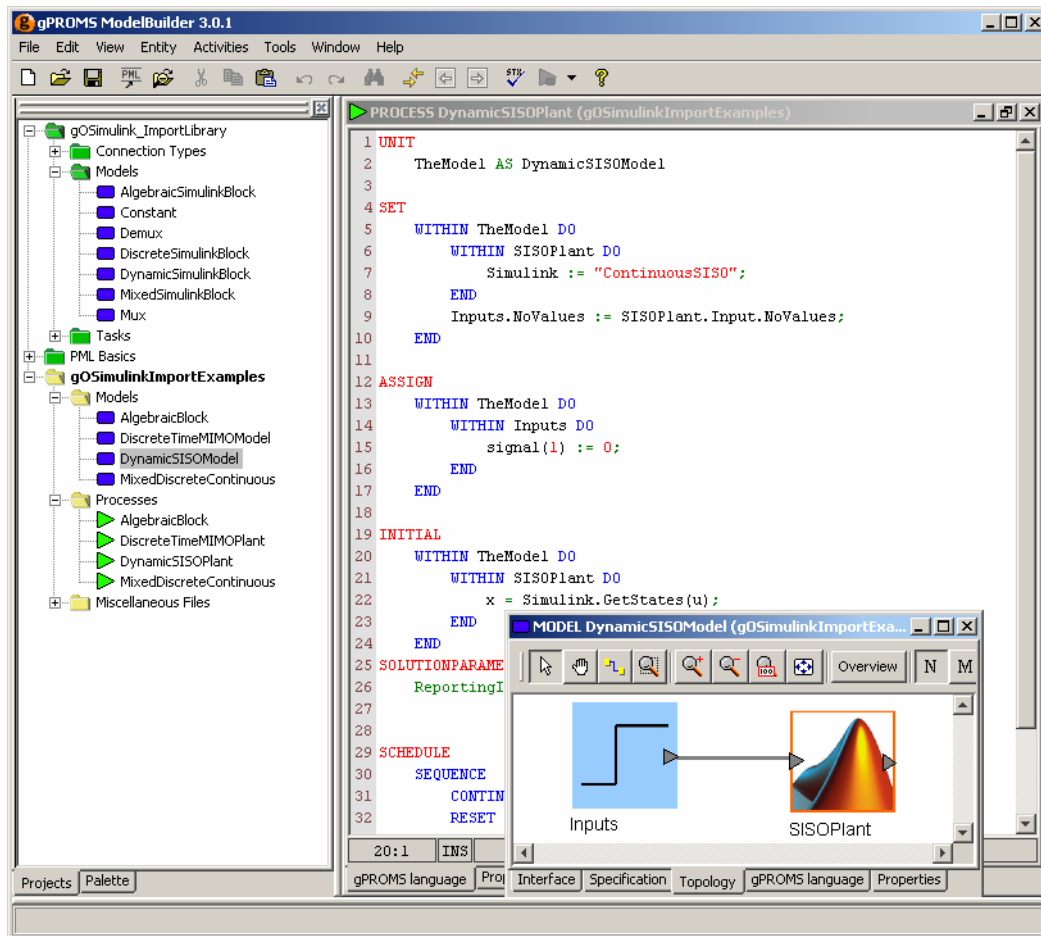
The DynamicSimulinkBlock model can be used for Simulink models which include continuous time dynamic blocks (like integrators, transfer function blocks,...).

The configuration of the model involves three steps:

1. Specifying the Foreign Object containing the Simulink model by setting the PARAMETER Simulink.
2. Specifying the input of the model by
 - ASSIGNing an appropriate number of values to the signal variable of the input PORT or
 - making an appropriate connection on a flowsheet to the input PORT of the model.
3. Specifying the initial conditions of the model by
 - Setting the conditions manually in the INITIAL section or
 - Retrieving the values of the initial conditions from the Simulink model using the GetStates method.

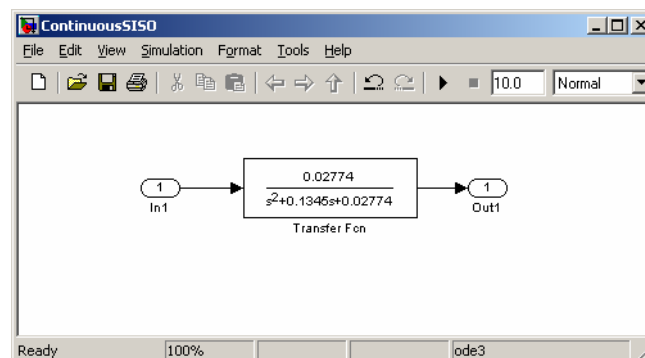
The usage of the model is illustrated by the example DynamicSISOModel.

Figure 6.15. Using the DynamicSimulinkBlock model.



The Foreign Object used in the example is a dynamic model with one input and one output.

Figure 6.16. Simulink flowsheet of the dynamic (continuous time) model



The DiscreteSimulinkBlock model

The DiscreteSimulinkBlock model can be used for Simulink models which include discrete time dynamic blocks but no continuous time elements.

The configuration of the model involves three steps:

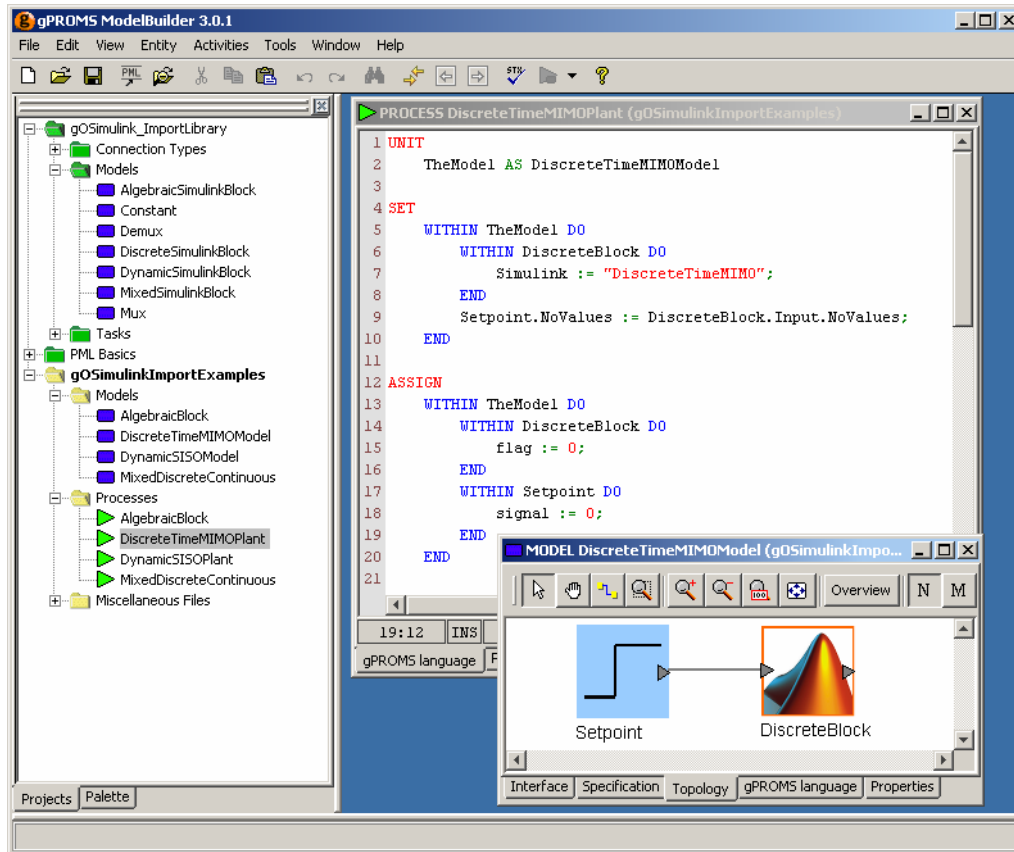
1. Specifying the Foreign Object containing the Simulink model by setting the PARAMETER Simulink.
2. Specifying the input of the model by

- ASSIGNing an appropriate number of values to the signal variable of the input PORT or
- making an appropriate connection on a flowsheet to the input PORT of the model.

3. Specifying the values of the flag variable in the ASSIGN section.

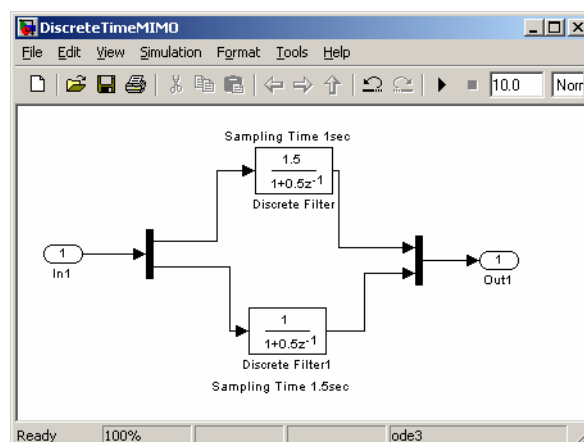
The usage of the model is illustrated by the example DiscreteTimeMIMOModel.

Figure 6.17. Using the DiscreteTimeMIMO model.



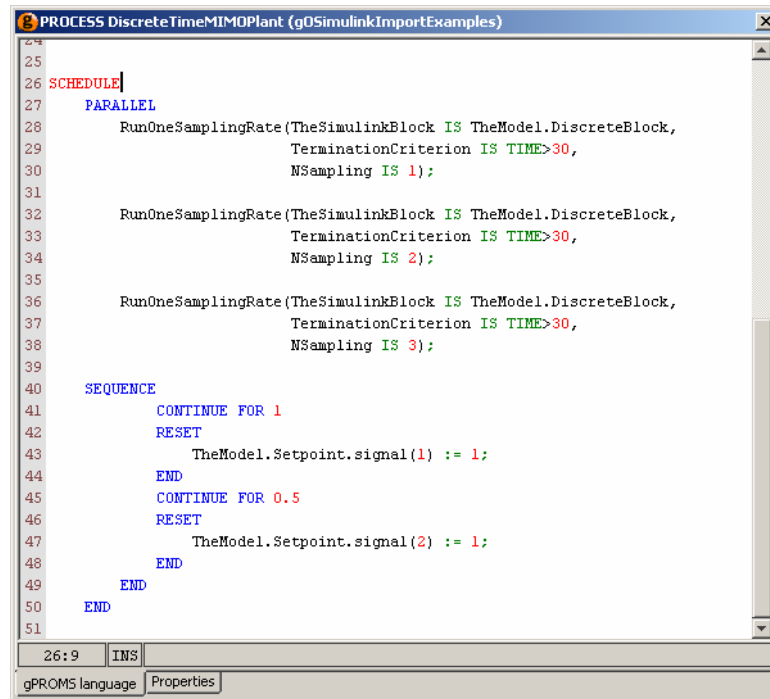
The Foreign Object used in the example is a time discrete dynamic model with two inputs and two outputs.

Figure 6.18. Simulink flowsheet of the discrete time model



Handling the sampling times of the Simulink model is done using the TASK RunOneSamplingRate in the SCHEDULE of the PROCESS.

Figure 6.19. Running the RunOneSamplingRate TASK in the SCHEDULE



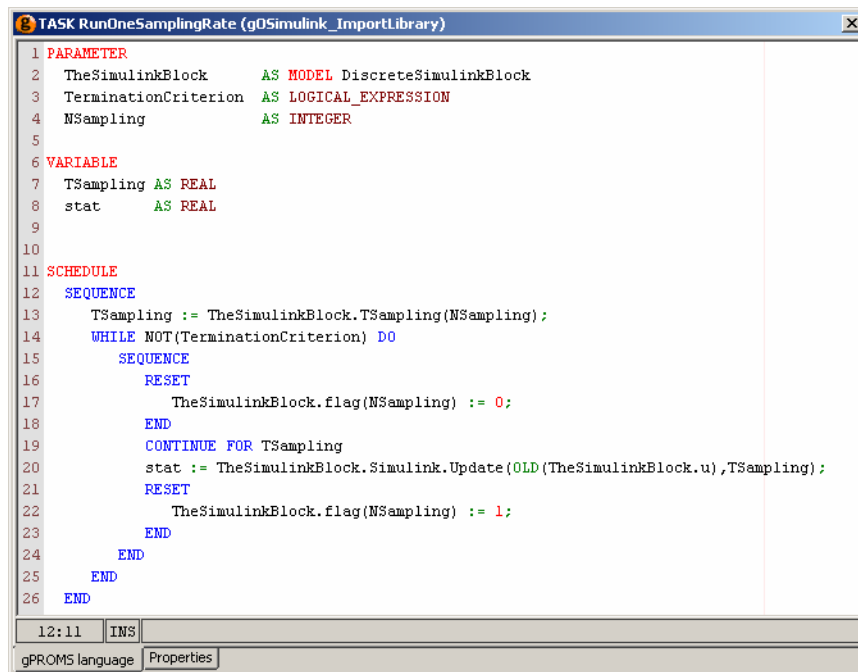
```

PROCESS DiscreteTimeMIMOPlant (gOSimulinkImportExamples)
25
26 SCHEDULE
27   PARALLEL
28     RunOneSamplingRate(TheSimulinkBlock IS TheModel.DiscreteBlock,
29                       TerminationCriterion IS TIME>30,
30                       NSampling IS 1);
31
32     RunOneSamplingRate(TheSimulinkBlock IS TheModel.DiscreteBlock,
33                       TerminationCriterion IS TIME>30,
34                       NSampling IS 2);
35
36     RunOneSamplingRate(TheSimulinkBlock IS TheModel.DiscreteBlock,
37                       TerminationCriterion IS TIME>30,
38                       NSampling IS 3);
39
40   SEQUENCE
41     CONTINUE FOR 1
42     RESET
43       TheModel.Setpoint.signal(1) := 1;
44     END
45     CONTINUE FOR 0.5
46     RESET
47       TheModel.Setpoint.signal(2) := 1;
48     END
49   END
50 END
51
26:9 INS
gPROMS language Properties
  
```

5

The RunOneSamplingRate TASK can be used to handle one sampling rate of a Simulink model. Multiple sampling rates can be handled by running multiple instances of this TASK in parallel.

Figure 6.20. The RunOneSamplingRate TASK



```

TASK RunOneSamplingRate (gOSimulink_ImportLibrary)
1
2 PARAMETER
3   TheSimulinkBlock AS MODEL DiscreteSimulinkBlock
4   TerminationCriterion AS LOGICAL_EXPRESSION
5   NSampling AS INTEGER
6
7 VARIABLE
8   TSampling AS REAL
9   stat AS REAL
10
11 SCHEDULE
12   SEQUENCE
13     TSampling := TheSimulinkBlock.TSampling(NSampling);
14     WHILE NOT(TerminationCriterion) DO
15       SEQUENCE
16         RESET
17           TheSimulinkBlock.flag(NSampling) := 0;
18         END
19         CONTINUE FOR TSampling
20         stat := TheSimulinkBlock.Simulink.Update(OLD(TheSimulinkBlock.u),TSampling);
21         RESET
22           TheSimulinkBlock.flag(NSampling) := 1;
23         END
24       END
25     END
26   END
12:11 INS
gPROMS language Properties
  
```

The key elements of this TASK are:

- Updating the discrete states in the Simulink model at the end of the sampling time using the method Update.

⁵Although the Simulink model initially contained only two sampling rates a third sampling rate is added during the code generation of the Real-Time Workshop.

- Synchronising the Simulink and gPROMS numerics using a change in the variable flag.

The MixedSimulinkBlock model

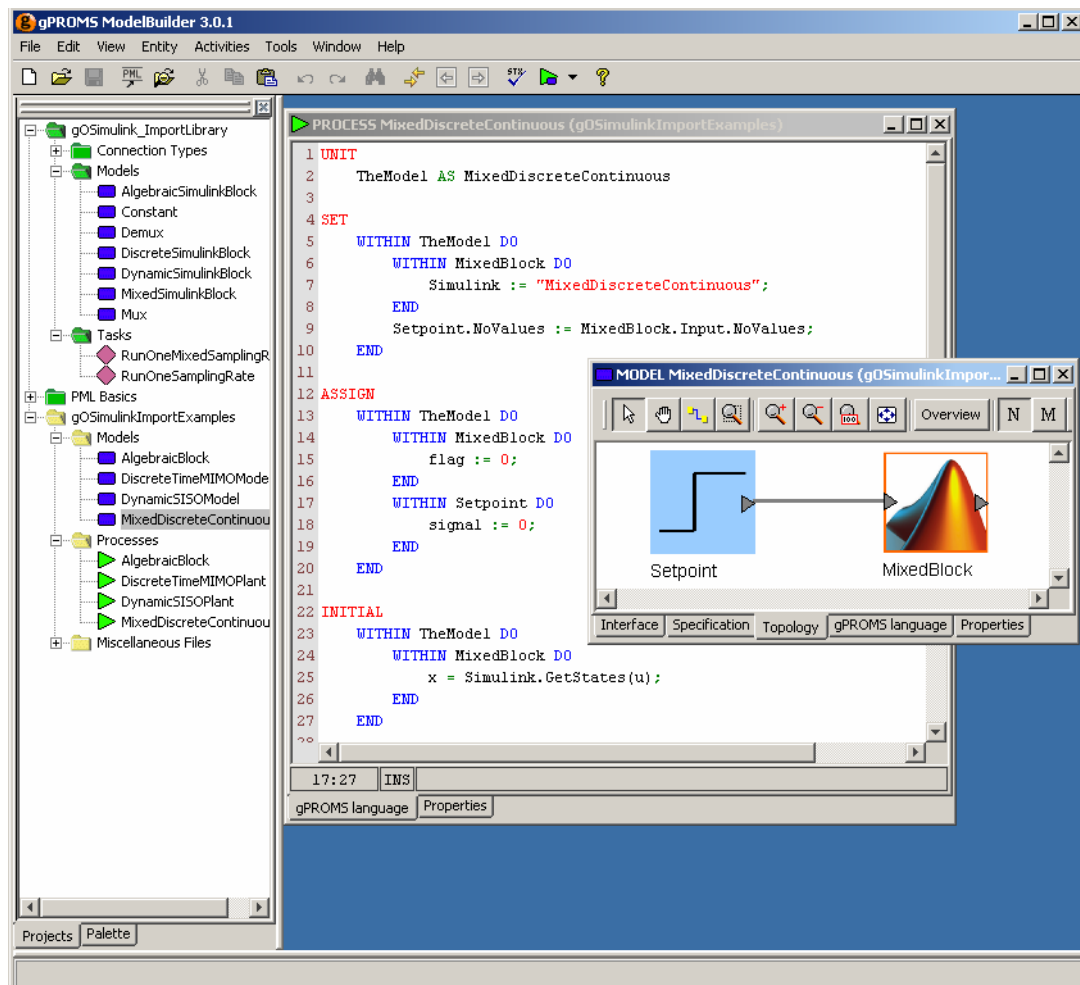
The MixedSimulinkBlock model can be used for Simulink models which include discrete and continuous time dynamic blocks at the same time. It combines the features of the DynamicSimulinkBlock and DiscreteSimulinkBlock models.

The configuration of the model involves four steps:

1. Specifying the Foreign Object containing the Simulink model by setting the PARAMETER Simulink.
2. Specifying the input of the model by
 - ASSIGNing an appropriate number of values to the signal variable of the input PORT or
 - making an appropriate connection on a flowsheet to the input PORT of the model.
3. Specifying the values of the flag variable in the ASSIGN section.
4. Specifying the initial conditions of the model by
 - Setting the conditions manually in the INITIAL section or
 - Retrieving the values of the initial conditions from the Simulink model using the GetStates method.

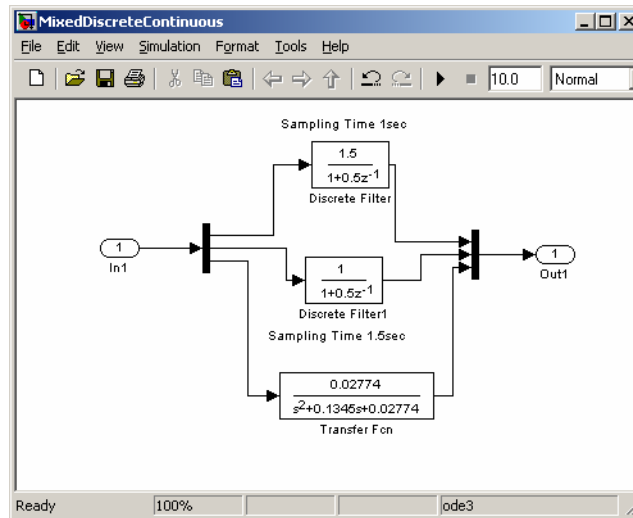
The usage of the model is illustrated by the example MixedDiscreteContinuous.

Figure 6.21. Using the MixedSimulinkBlock model.



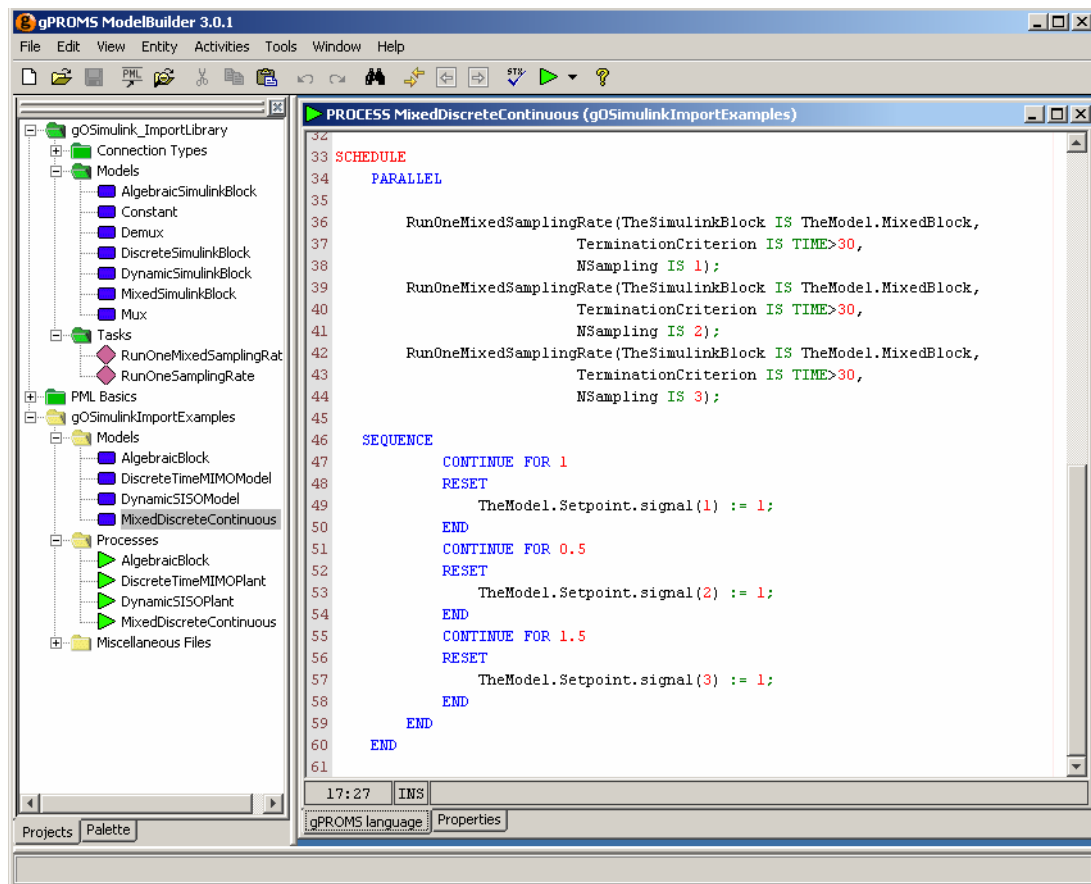
The Foreign Object used in the example is a mixed discrete/continuous time dynamic model with three inputs and three outputs.

Figure 6.22. Simulink flowsheet of the discrete time model



Handling the sampling times of the Simulink model is done using the TASK RunOneMixedSamplingRate in the SCHEDULE of the PROCESS.

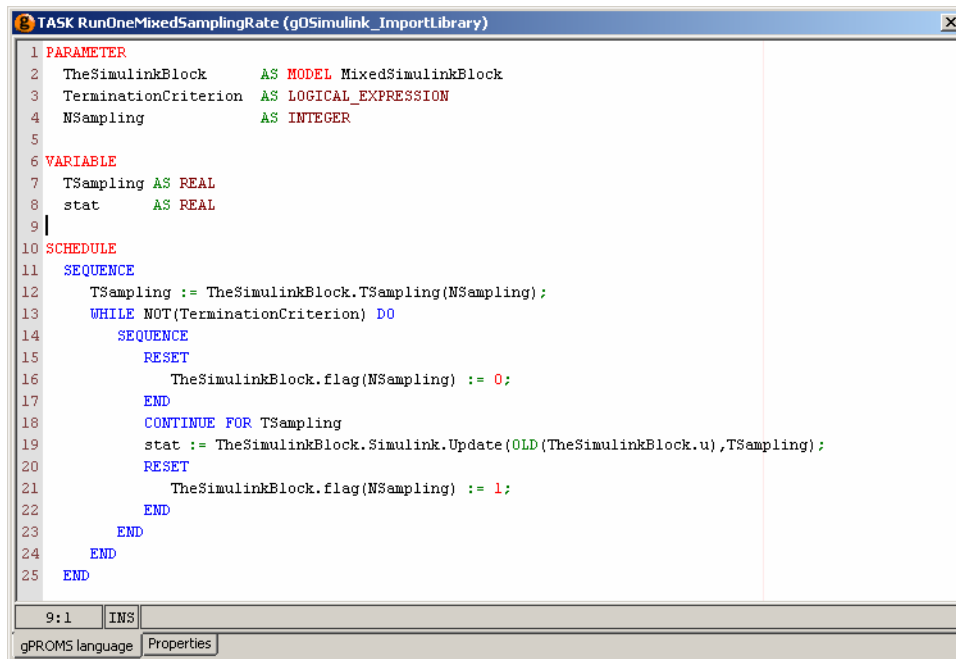
Figure 6.23. Running the RunOneMixedSamplingRate TASK in the SCHEDULE



⁶Although the Simulink model initially contained only two sampling rates a third sampling rate is added during the code generation of the Real-Time Workshop.

The RunOneMixedSamplingRate TASK can be used to handle one sampling rate of a Simulink model. Multiple sampling rates can be handled by running multiple instances of this TASK in parallel.

Figure 6.24. The RunOneMixedSamplingRate TASK



The key elements of this TASK are:

- Updating the discrete states in the Simulink model at the end of the sampling time using the method Update.
- Synchronising the Simulink and gPROMS numerics using a change in the variable flag.

Known limitations and troubleshooting

Known limitations and workarounds

The three known limitations are:

1. The technology used for importing Simulink models into a gPROMS simulation is based on the Real-Time Workshop. Thus only Simulink flowsheets consisting purely of blocks supported by the Real-Time Workshop can be used.
2. Although MemoryBlocks are supported by the Real-Time Workshop these blocks cannot be used when importing the Simulink flowsheet to gPROMS.⁷
3. Simulink supports the re-initialisation of dynamic states of the model on a flowsheet level (e.g. for the continuous-time Integrator block). gPROMS in the other hand does not support handling of re-initialisations on the MODEL level (though in the SCHEDULE of the PROCESS or a TASK).

The first limitation is inherent to the technology used so there is no workaround available. But the two other limitations can be overcome using workarounds.

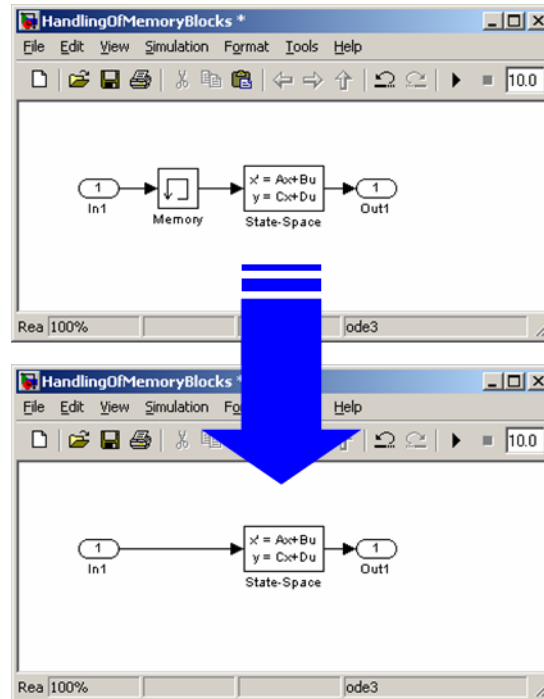
Workarounds for MemoryBlocks

The most common application for a MemoryBlock in Simulink is the breaking of an algebraic loop. In case that this loop is not part of the Simulink model to be exported to gPROMS the solution is:

⁷The MemoryBlock is a workaround on the Simulink side to handle algebraic loops in the Simulink flowsheet. Since the gPROMS numerics fully supports models containing implicit equations (the equivalent of an algebraic loop in Simulink) there is no need to support a concept like a MemoryBlock in gPROMS.

1. Remove MemoryBlock.
2. Connect the block which were formerly separated by the memory block.

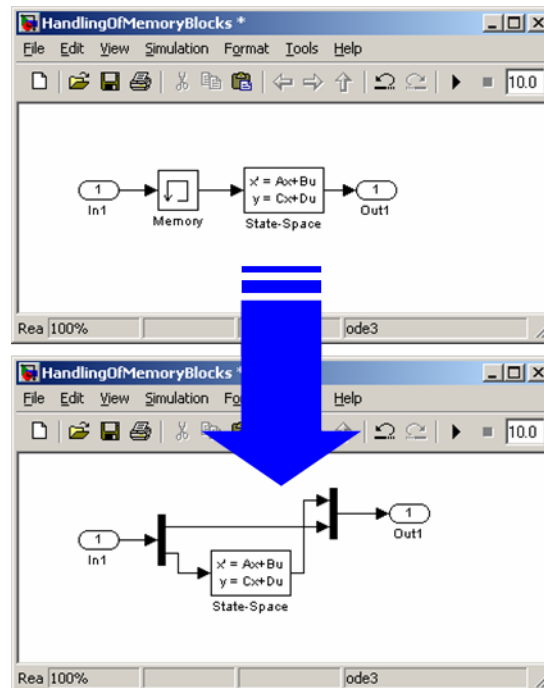
Figure 6.25. Removing a MemoryBlock



If the MemoryBlock is still necessary to break in algebraic loop in the model to be exported the solution is:

1. Remove the MemoryBlock.
2. Add a new value to be passed from gPROMS to the input port In1.
3. Add a new value to be passed to gPROMS to the output port Out1.
4. Add a new connection from the new value of the In1 port to the block that was formerly connected to the output of the MemoryBlock.
5. Add a new connection from the output of the block that was formerly connected to the input of the MemoryBlock to the new value of the output port Out1.
6. Add an equation in the gPROMS model which enforces that the two new values (added under the points 2 and 3) are equal.

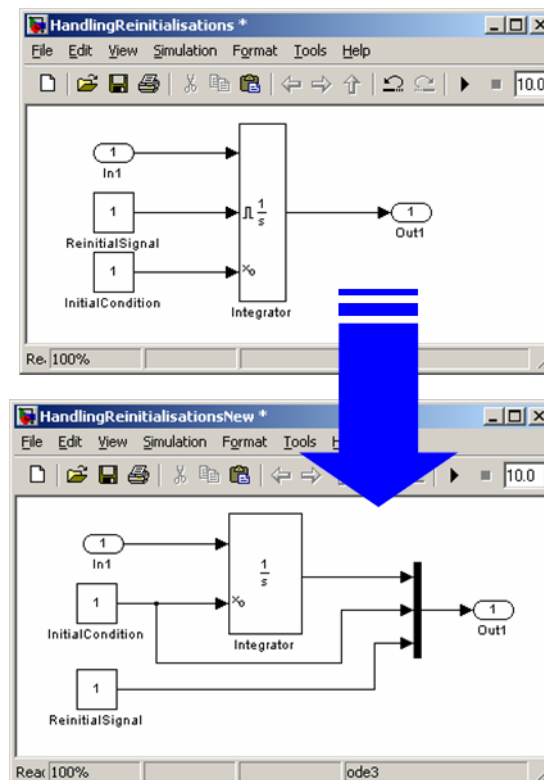
Figure 6.26. Replacing a MemoryBlock



Workarounds for re-initialisations on the Simulink flowsheet

If the Simulink model contains a mechanism to re-initialise continuous states on the model level, this can be handled by exposing the respective signals to the gPROMS model and forcing a re-initialisation using a TASK in the gPROMS simulation.

Figure 6.27. Signal routing for re-initialisations on the Simulink flowsheet



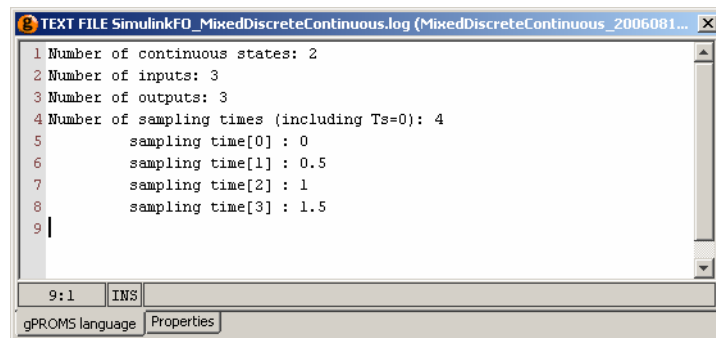
Troubleshooting

Log files created by the Foreign Objects

During the simulation a log file is automatically created by the foreign object. This log file contains structural information about the Simulink model embedded into the Foreign Object:

- The number of continuous state
- The number of inputs
- The number of outputs
- The number of sampling rates
- The sampling rates

Figure 6.28. Foreign Object log file example



The information included in the log file can be used to check if the Foreign Object is compatible with the MODELS, TASKs and PROCESS used in combination with the Foreign Object.