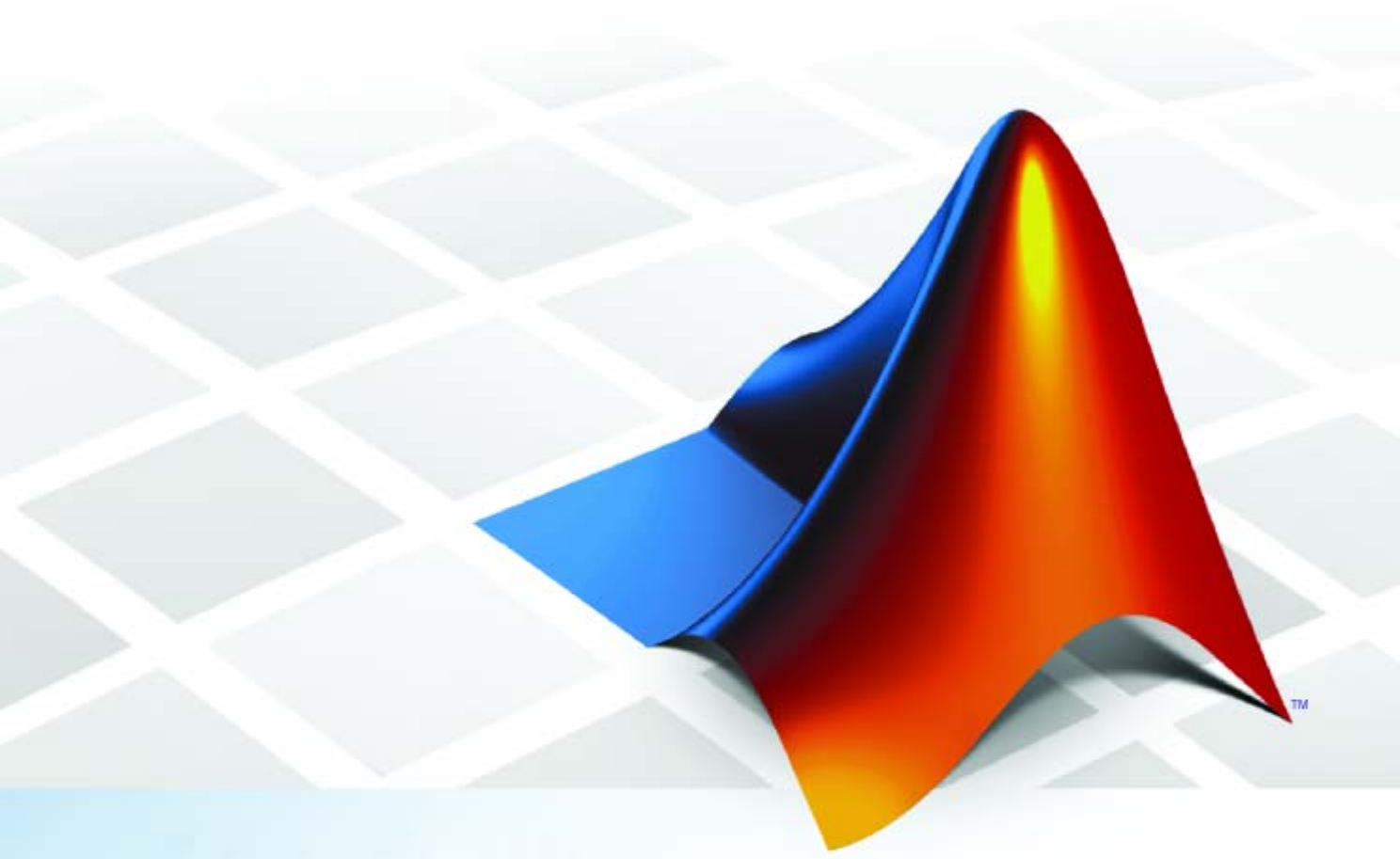
PolySpace® Model Link Products 5

User's Guide



The MathWorks™

Accelerating the pace of engineering and science

How to Contact The MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

PolySpace® Model Link Products User's Guide

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `www.mathworks.com/trademarks` for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Revision History

March 2009 Online Only Revised for Version 5.3 (Release 2009a)

Contents

1 Getting Started with PolySpace Model Link Products

2 Advanced Setup Options

3 PolySpace Utilities

4 Code Generator Specific Information

Glossary

1

Getting Started with PolySpace Model Link Products

Overview of PolySpace Model Link Products

This manual describes how to use PolySpace® for Model-Based Design. The PolySpace Model Link™ SL and PolySpace Model Link TL products allow you to launch a PolySpace C verification from a Simulink® model associated with Real-Time Workshop® Embedded Coder™ software, or dSPACE® TargetLink® software.

PolySpace Model Link SL and PolySpace Model Link TL products provide automatic error detection for code generated from Simulink models. It consists of two principal components:

- A Simulink PolySpace library with associated blocks.
- A "Back to Model" extension in the PolySpace Viewer that allows direct navigation from a runtime error in the auto-generated code to the corresponding Simulink block or Stateflow® chart in the Simulink model.

Getting Started with Model Link Products

In this section...

Overview

In this section, you will:

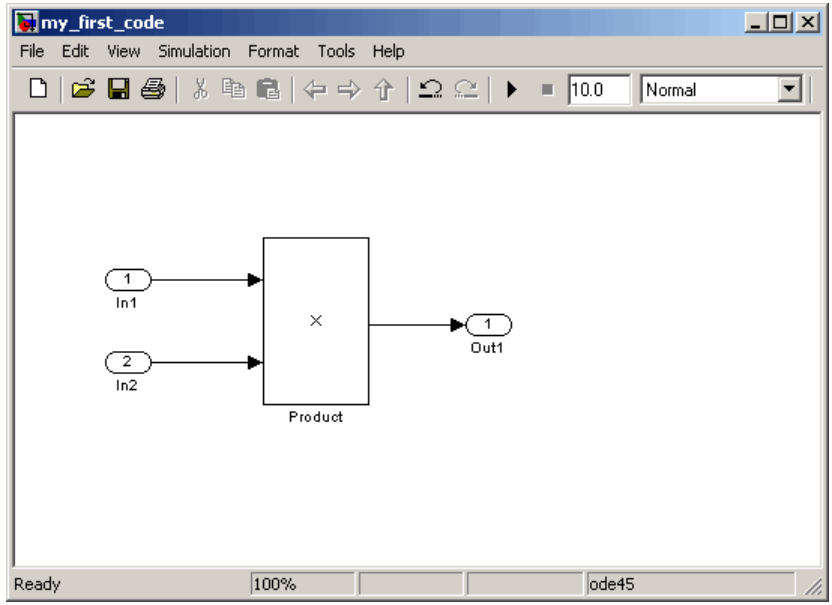
- Create a Simulink model and generate production code (For more information, see the *Real-Time Workshop Embedded Coder Getting Started Guide*)
- Start the PolySpace verification

Creating a Simulink Model and Generating Production Code

To create a Simulink model and generate production code:

1 Open MATLAB®, then start Simulink software.

2 Create a simple Simulink model, similar to the one below.

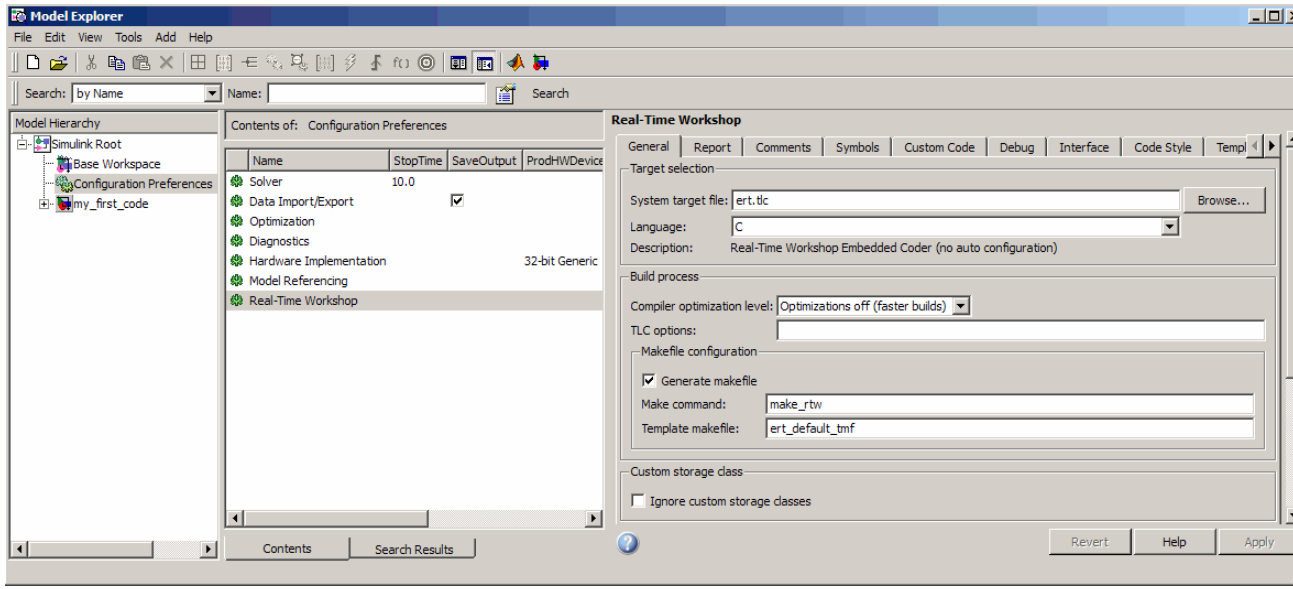


Create the my_first_code model

3 Select **File > Save**, then name the model `my_first_code`.

4 Select **View > Model Explorer**.

The Model Explorer opens.



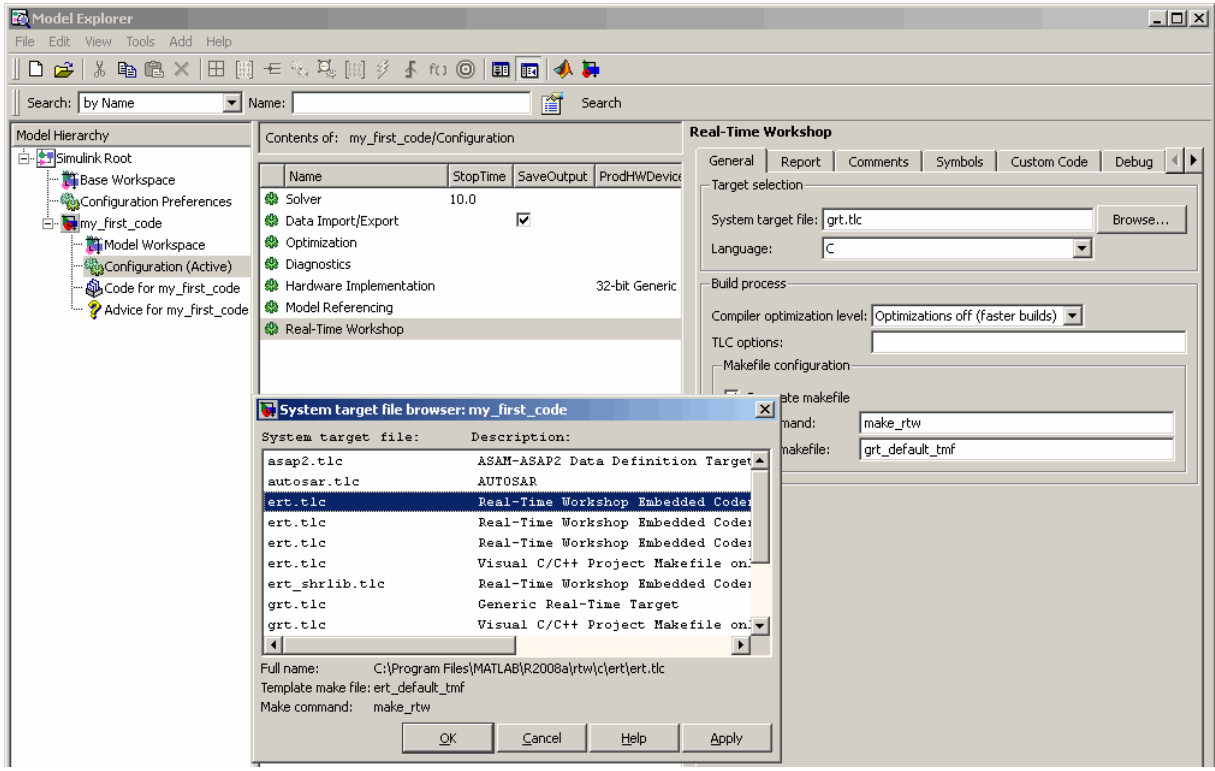
5 Select **Configuration Preferences**, in the Model Hierarchy.

6 Select **Real-Time Workshop** in the Configuration Preferences.

The Real-Time Workshop configuration parameters open.

7 Select the Real-Time Workshop **General** tab.

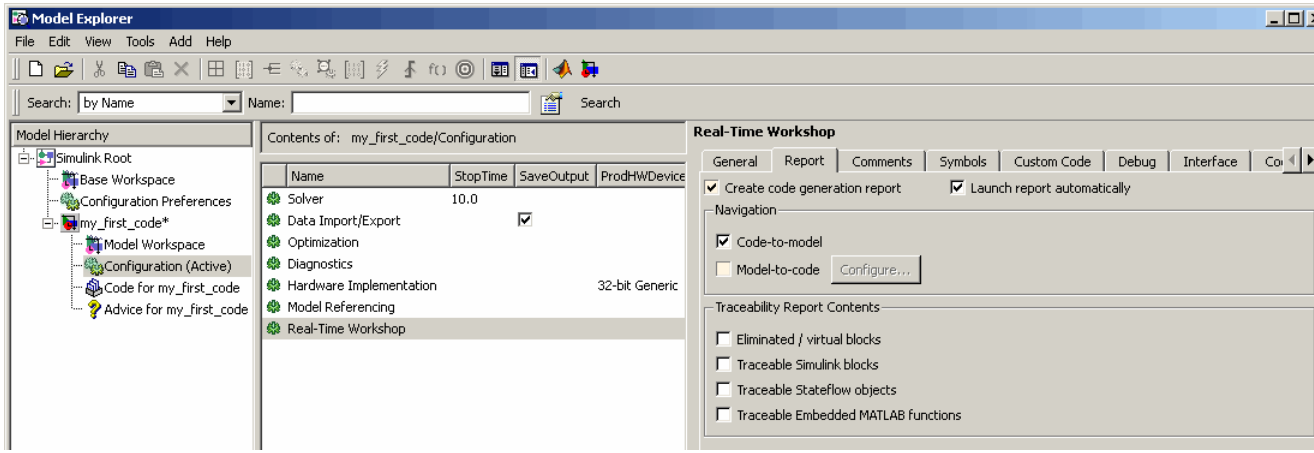
Set the **System target file** to **ert.tlc** (Real Time Workshop Embedded Coder (no auto configuration)).



Change the code generator to Real-Time Workshop® Embedded Coder™ software

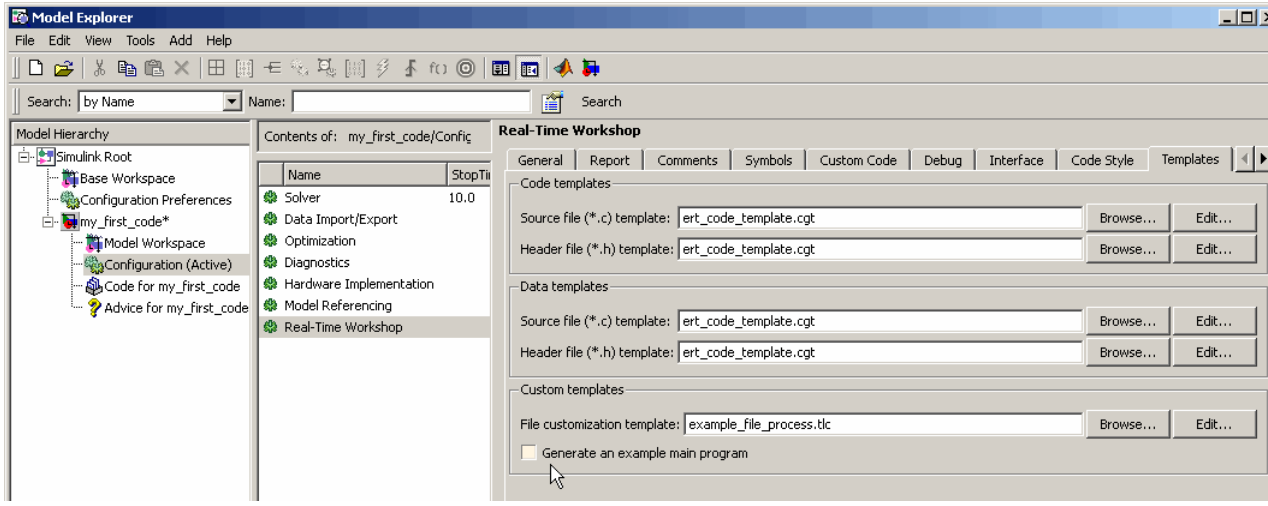
8 Select the **Report** tab.

9 Select **Create code-generation report**, then select **Code-to-model** Navigation.



Set Report Settings

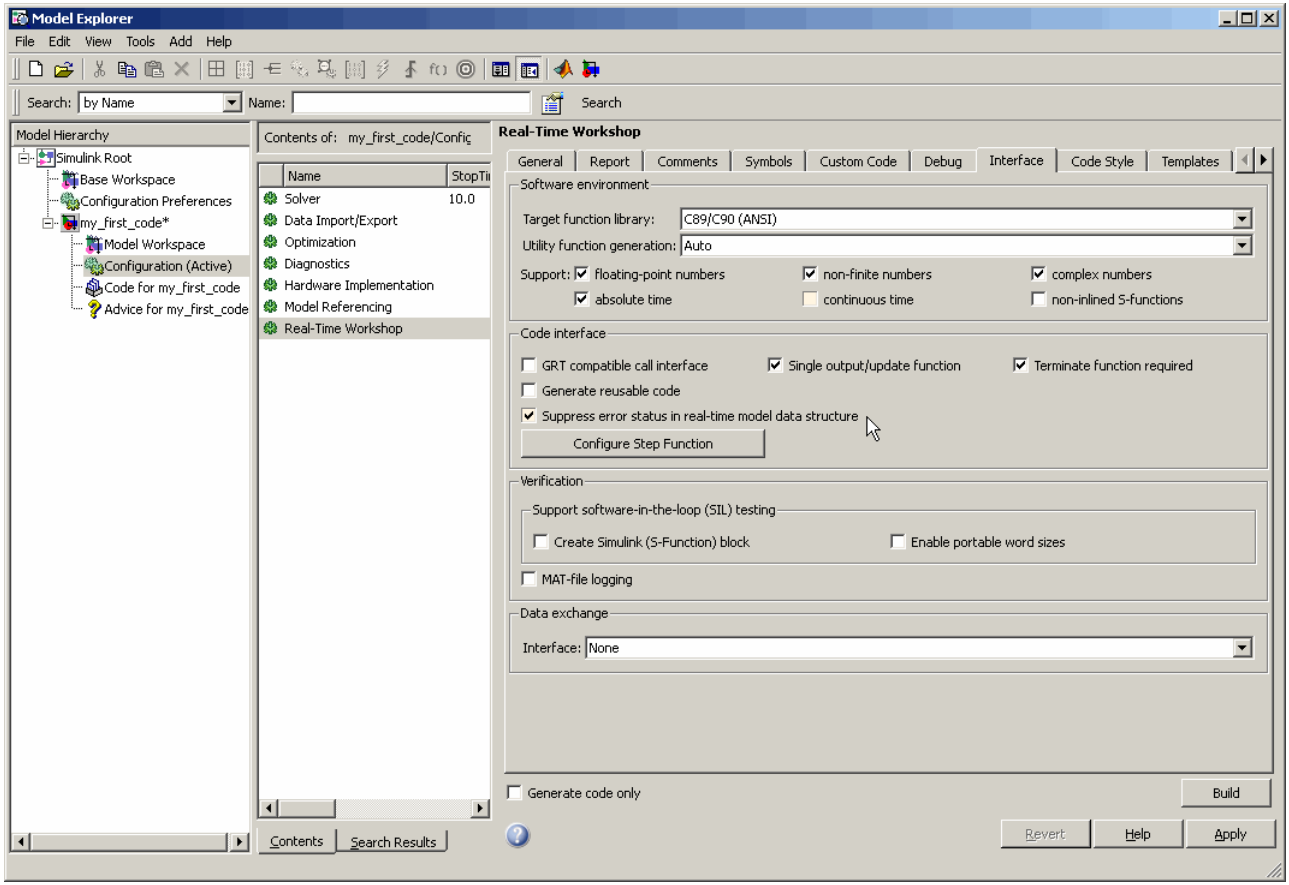
10 Select the **Templates** tab.



Templates Tab

11 In the Custom templates section, clear **Generate an example main program**.

12 Select the **Interface** tab.



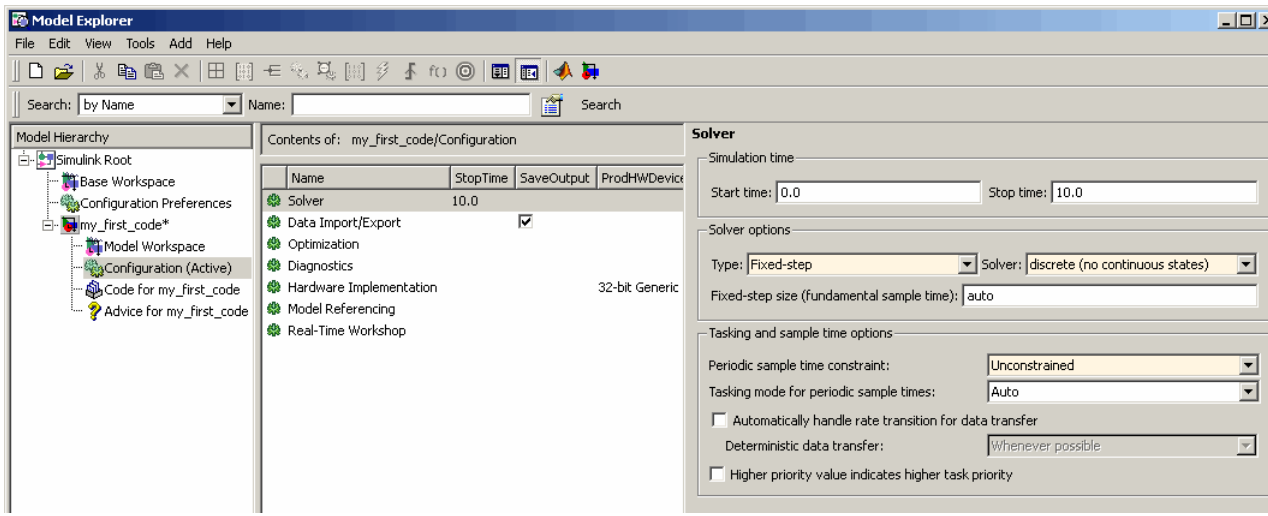
Interface Tab

13 In the Code interface section, select **suppress error status in real-time model data structure**.

14 Click **Apply** in the lower-right corner of the window.

15 In the Configuration Preferences, select **Solver**.

The Solver configuration parameters appear.



Choose Fixed-step Solver

16 In the Solver options section, set the solver **Type** to `Fixed-step`. Then, set the **Solver** to `discrete (no continuous states)`.

17 Click **Apply**.

18 In the Simulink Model Window, select **Tools > Real Time Workshop > Build Model** to generate the production code.

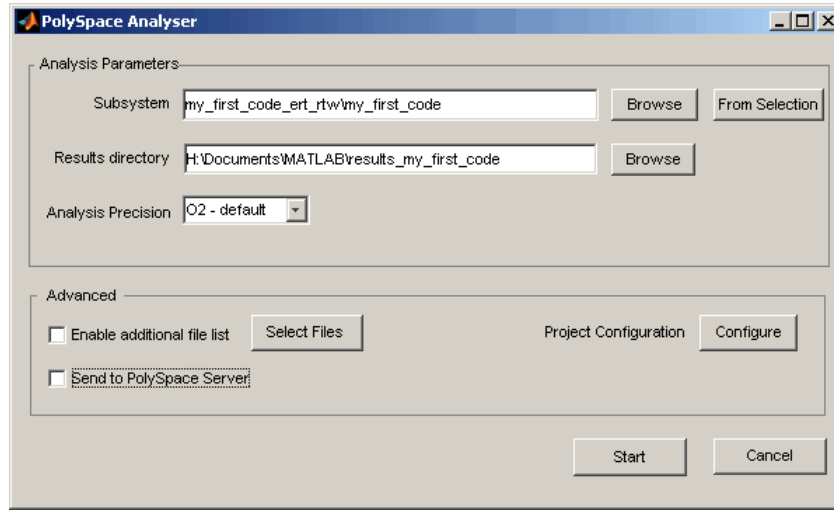
19 Save your Simulink model.

Starting the PolySpace Verification

To Start the PolySpace verification:

1 In the Simulink model window select **Tools > PolySpace > PolySpace for RTW Embedded Coder**.

The PolySpace Analyser dialog box opens.



PolySpace Analyser Dialog Box

Note The subsystem field is automatically populated with the name of the current subsystem, and the results directory is automatically set to `results_`*`subsystem_name`*. If more than one subsystem is present in the model, a subsystem selection dialog opens.

---

2 Click **Start** to start the verification.

The verification starts, and messages appear in the MATLAB Command window:

```
### PolySpace Technologies RTW Embedded Coder integration
### Version 1.4
### Preparing verification
### Locating generated source files:
  ert_main.c ok (c:\MatLAB704\toolbox\rtw\rtwdemos
\rtwdemo_examplemain_ert_rtw)
  rtwdemo_examplemain.c ok (c:\MatLAB704\toolbox\rtw\rtwdemos
\rtwdemo_examplemain_ert_rtw)
### Generating DRS table
  ### Get Parameters
```

```
  ### Get Signals
### Starting verification
```

The exact messages depend on the code generator you use. However, the messages always have the same format:

- Name of code generator
- Version number of the plug-in
- List of source files
- DRS (Data Range Specification) information.

3 Click **Execute** to proceed. You can follow the progress of the verification in the MATLAB Command window, and later using the PolySpace Spooler if you are performing a server verification.

Note Verification of this model takes about 7 minutes. A 3,000 block model will take approximately one hour to verify, or about 15 minutes for each 2,000 lines of generated code.

Fixing an Error in the Design and the Simulink Model

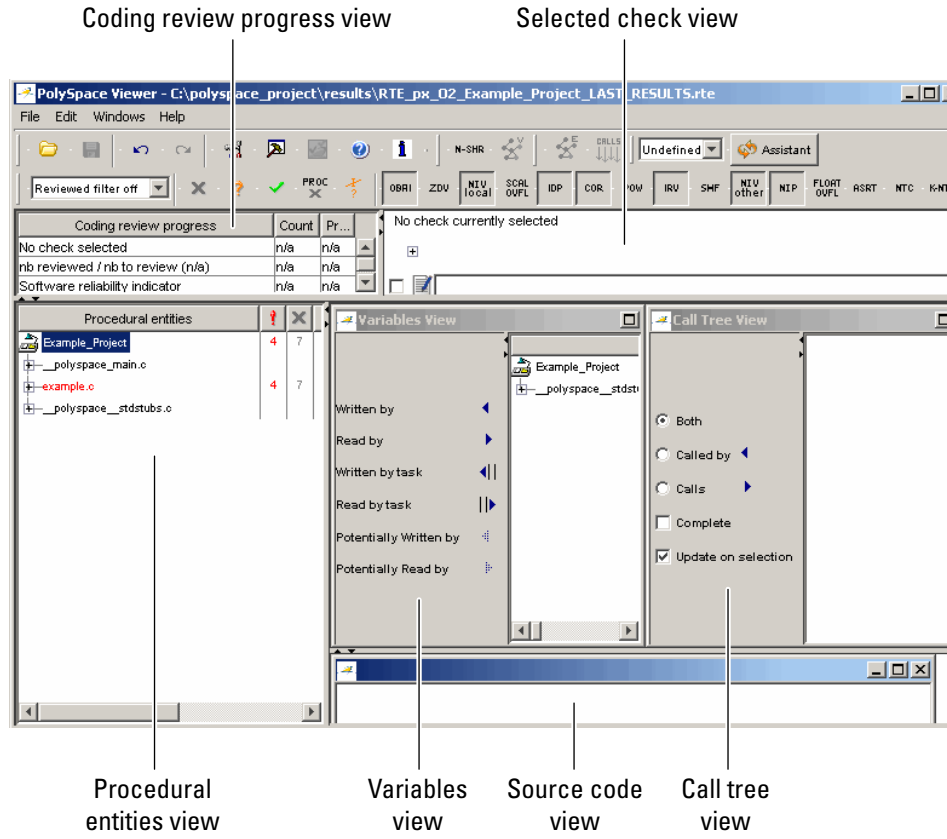
After the verification completes, you can view the results using PolySpace Viewer.

Note If you perform a server verification, you must download your results from the server before you can view them. For more information, see "Downloading Results from Server to Client" in the *PolySpace Products for C User's Guide*.

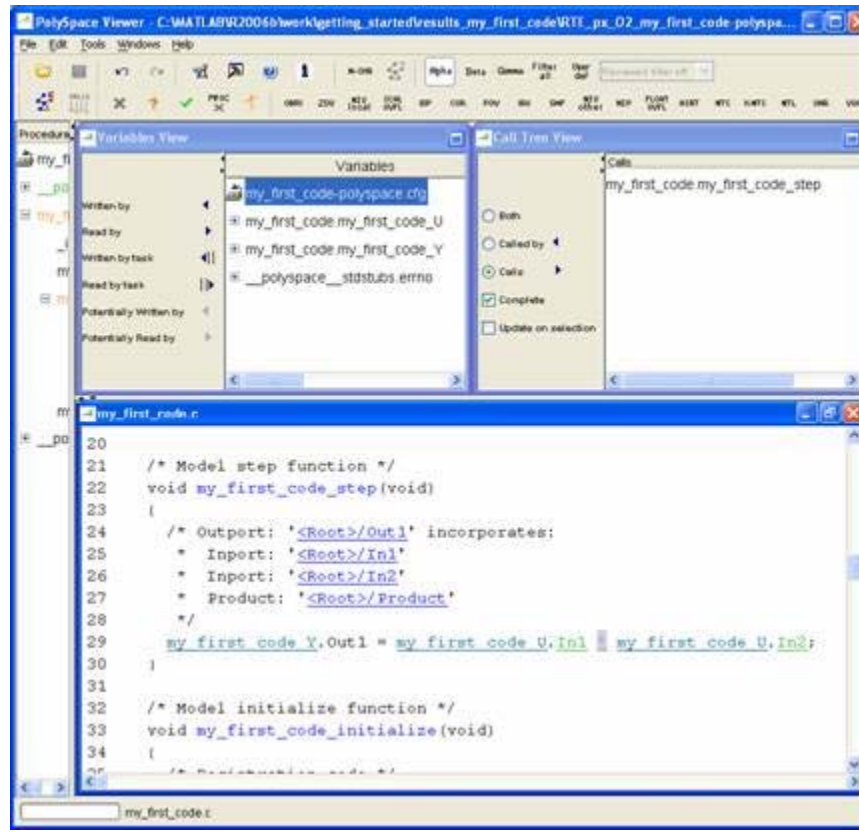
To view your results:

1 In the Simulink model window select **Tools > PolySpace > PolySpace Utilities > Open results**.

After a few seconds, the PolySpace Viewer opens.



2 Type **CTRL-N** to go to the next error.



Orange Check in PolySpace® Viewer

3 Click on the orange check.

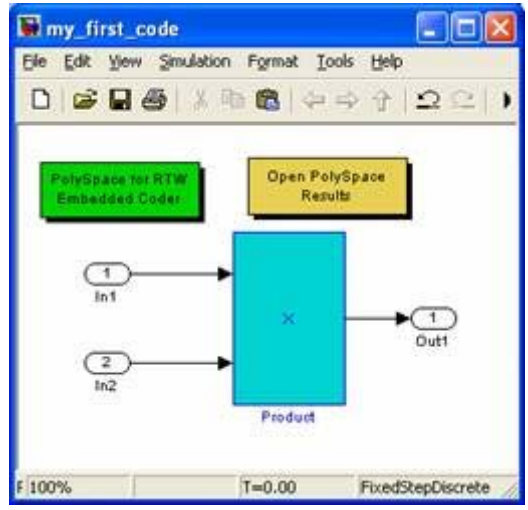
The check shows an overflow of the two entries. PolySpace software assumes that the values for entries are full range, and their multiplication can overflow.

Overflow in the code

4 Now, you must get back to the model to understand what needs to be fixed.

Click the first underlined blue HTML link near the check in the Source Code.

The Simulink model opens, highlighting the block with the error.



Model with Highlighted Block

5 You now must fix the defect in the model. For example, you may come to one of the following conclusions:

- **It is a bug in the design**— The developer should saturate the output, providing this functionally makes sense bound the entries in the model, by adding blocks which will test the input values, and bound them accordingly.
- **It is a bug in the specifications** — The developer should bound the entries, by giving them a range in Simulink software that PolySpace verification can take the ranges into account and turns the code green.

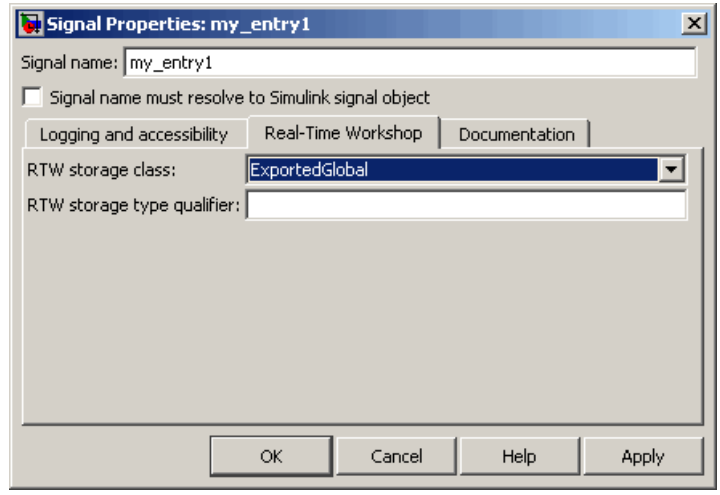
Base Workspace vs. PolySpace Data Ranges

After you examine the model, you can see a block whose signal ranges are not in the expected range.

- If its block is supposed to be robust against this range, it is a design bug. Should the previous block be saturated? Should the signal be bounded with a “switch” block? It is up to the developer to decide the appropriate change in the model
- If the range is an input range of the model, the developer may wish to give this information to the Simulink model, so that PolySpace tools can use that range as an entry.

Prerequisites

Have signals as ExportedGlobal.

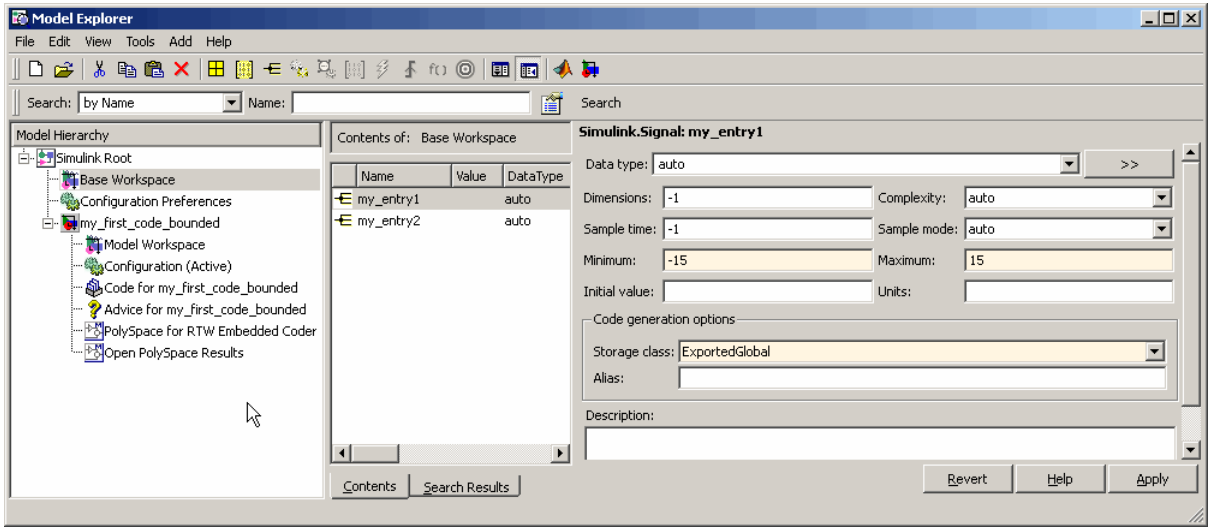


Details of a signal

Update Range of Signals

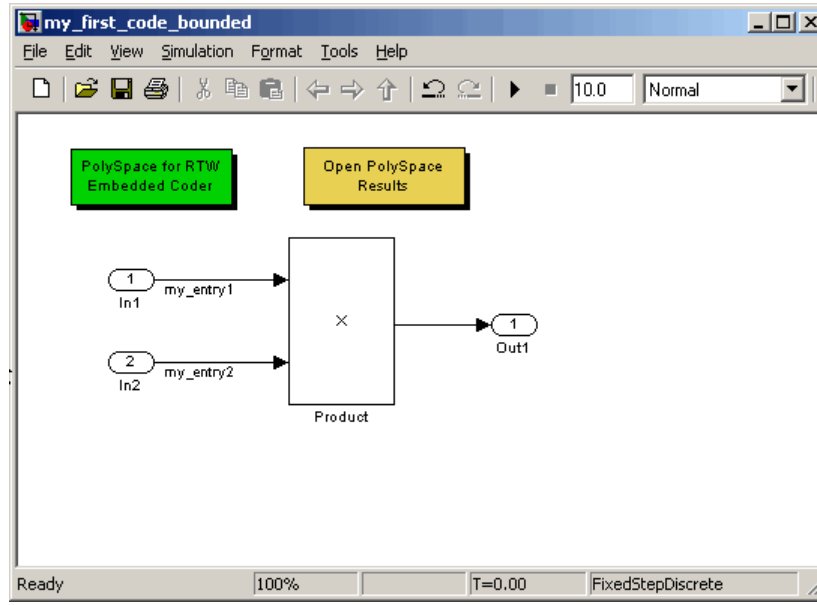
To update your signal ranges:

1. Open model explorer, and go into the Base Workspace tab.
2. Create a signal “`my_entry1`” & “`my_entry2`.”
3. Bound it to `-15` to `15`. Specify its storage class to `ExportedGlobal`.



Signal in the "Base Workspace"

4 Model with signals on entries:



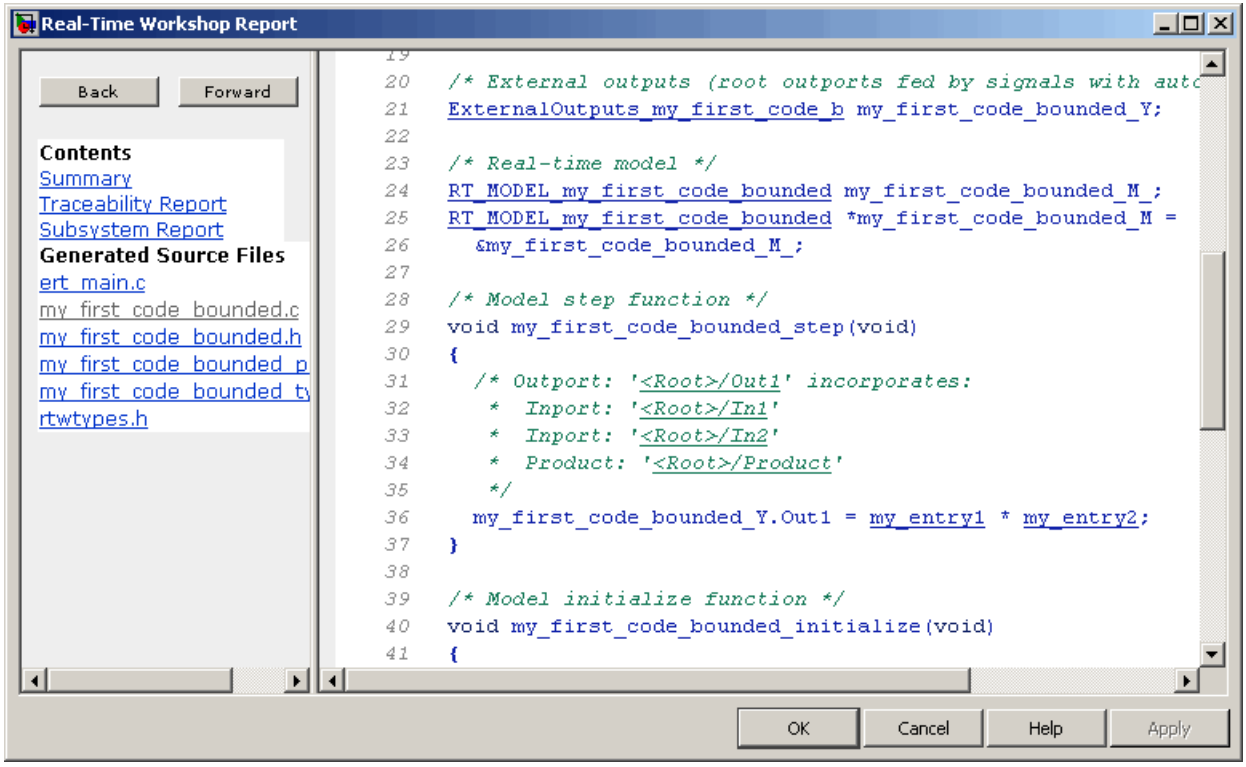
Model with signals my_entry1 and my_entry2 as "ExportedGlobal".

Re-Generate Code and Launch the PolySpace Verification Again

To regenerate the code and relaunch the PolySpace verification:

1 Regenerate the code

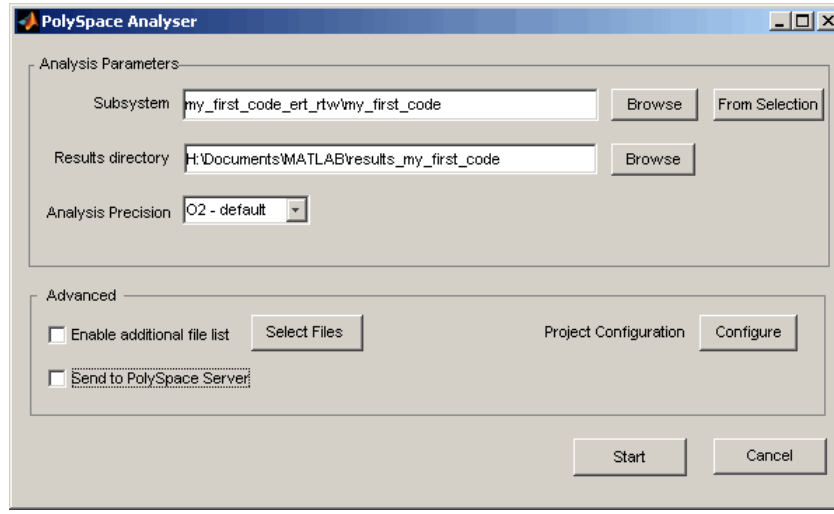
The entries are no longer part of a structure, they are separated global each.



Html report generator from Real-Time Workshop® Embedded Coder™ Software

2 Select **Tools > PolySpace > PolySpace for RTW Embedded Coder**.

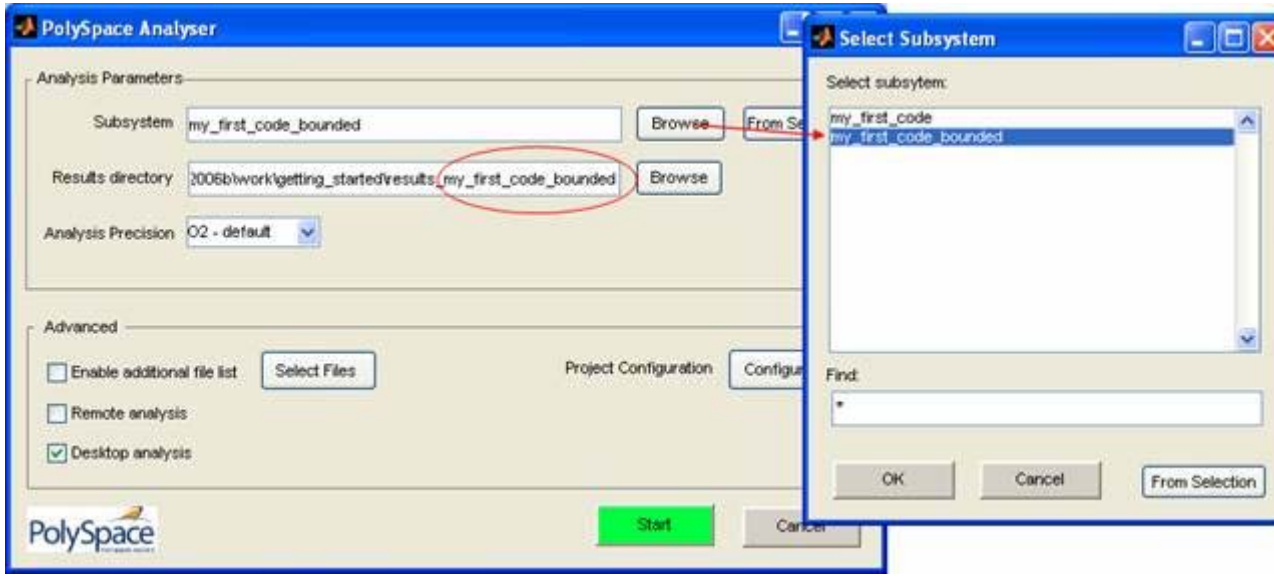
The PolySpace Analyser dialog box opens.



PolySpace Analyser Dialog Box

3 In results directory field enter `results_my_first_code_bounded`.

4 In the subsystem name field, enter `my_first_code_bounded`.

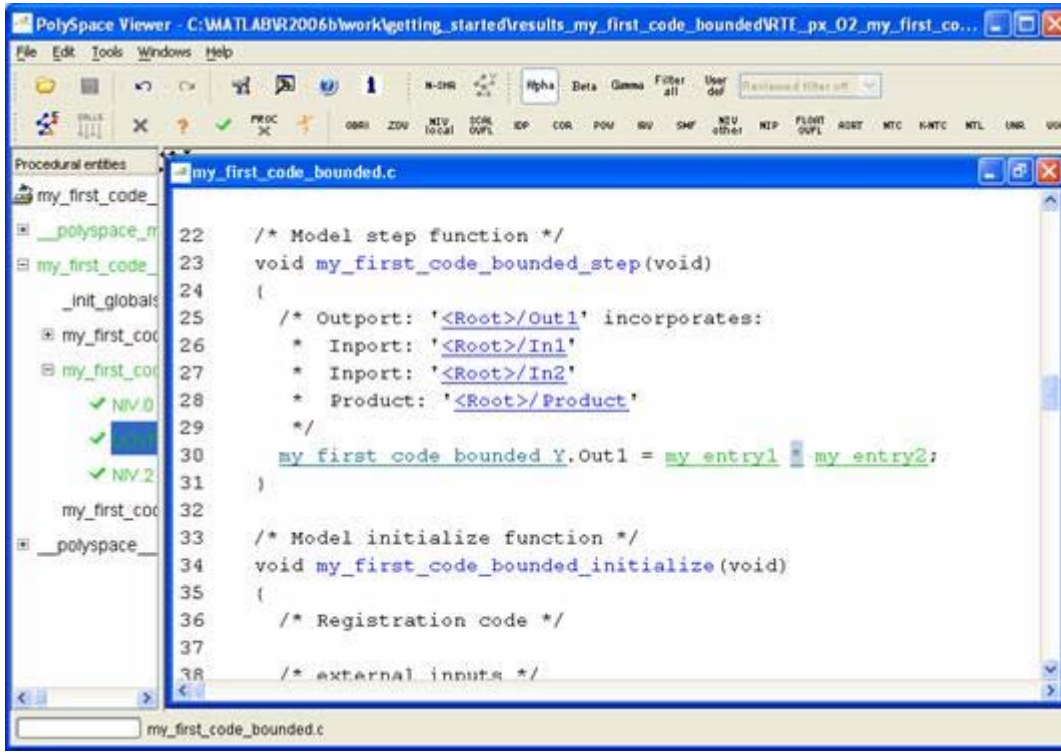


Results directory

5 Click Start to start the verification.

6 Once verification is complete, select **Tools > PolySpace > PolySpace Utilities > Open results** to view your results.

7 Examine the generated files in the PolySpace Viewer:



Detail of generated files viewed in PolySpace® Viewer

Everything is green. PolySpace verification has confirmed that no Runtime Errors are present in the model.

Can You Find More Bugs in the Model?

To answer this question, we need to now more about the tool, such as:

- Which windows of the PolySpace Viewer contain what information
- Which colors hide which messages
- How to find bugs using PolySpace Viewer

For more information, see “Reviewing Verification Results” in the *PolySpace Products for C User’s Guide*.

2

Advanced Setup Options

Advanced Setup

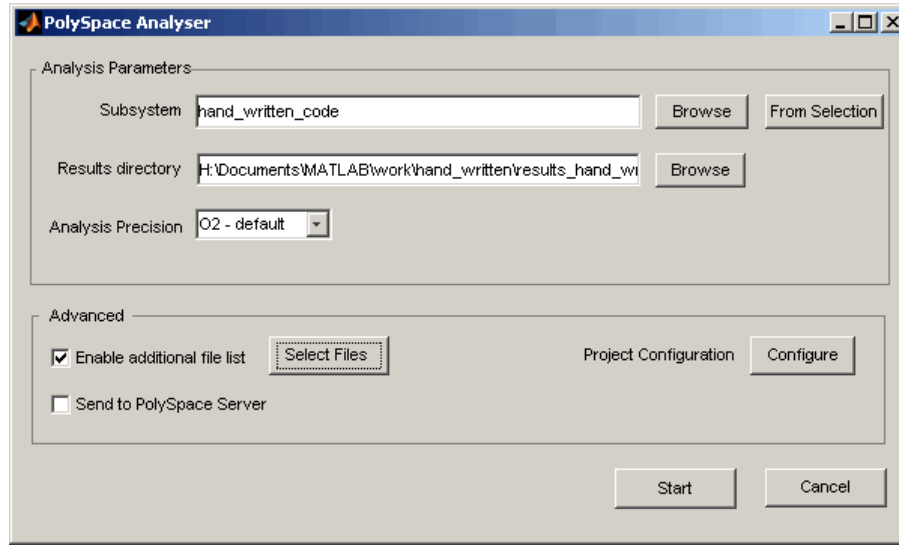
In this section...

Handwritten Code

Files such as S-function wrappers are, by default, not part of the PolySpace verification. They should be added manually.

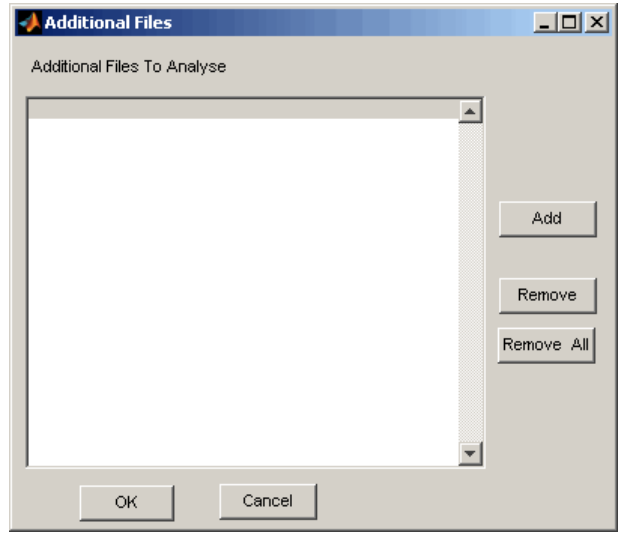
To add a file manually:

1. When starting the PolySpace verification, browse and add c-files to your verification:



2. Select additional files by ticking "Enable additional file list," then click on "Select Files".

A C File browser appears to add files to the PolySpace verification.



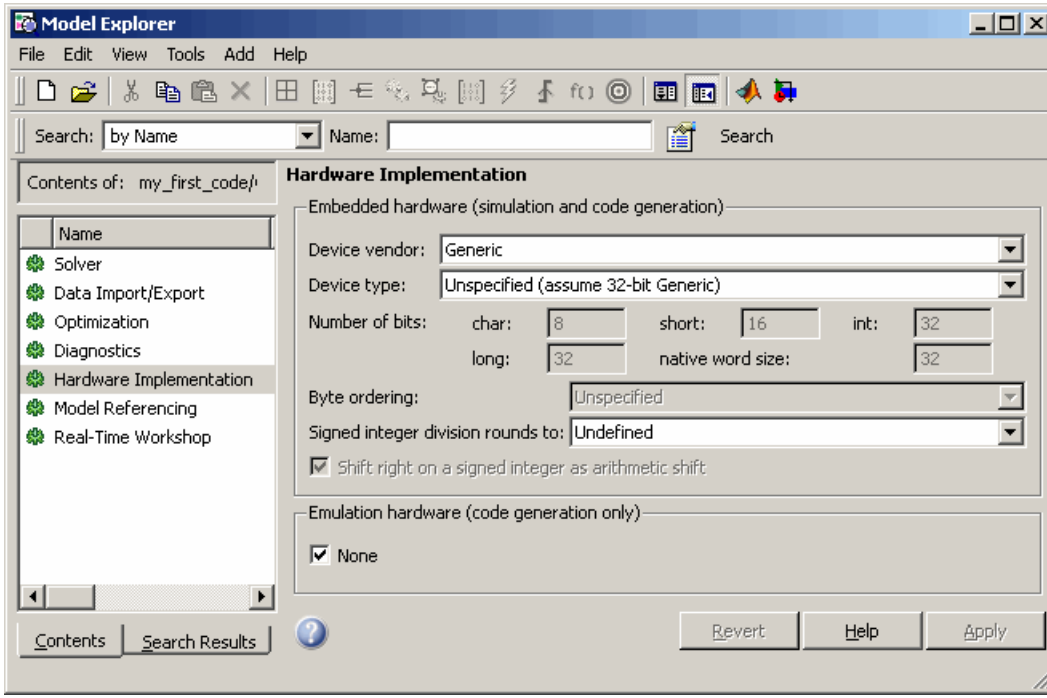
3 Select the appropriate c file and then start the verification.

Target Production Environment

In Simulink software, you need to configure the target and cross-compiler specificities.

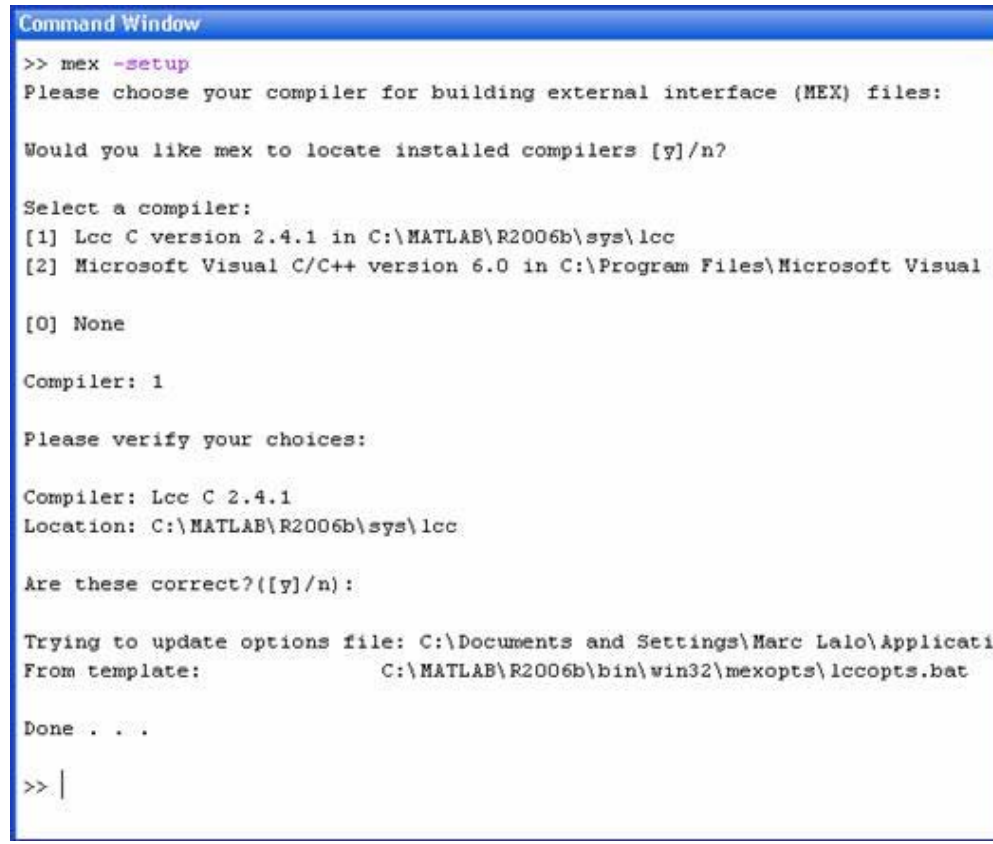
These parameters include:

- Size of the types for char, short, int (see Hardware implementation of the model explorer)



Target selection in Simulink® Configuration Parameters

- Cross compiler flag (-D), and library include (-I), implicitly defined when - for instance - the cross compiler is setup via the “mex -setup” command.



Cross compiler settings in MATLAB® Command Window

PolySpace settings work exactly the same way, you will need to perform the following tasks (they will be detailed step by step in the next sections).

1. define the same parameters for your cross compiler and target.
2. save this in a template PolySpace configuration file and set this template to be the default configuration file for every PolySpace verification.

Why does this matter?

- For the PolySpace verification, an overflow on an integer type does not mean the same when the size of an integer is 16 bits or 32 bits.

- PolySpace software needs the cross compiler header files, as they contain definitions of types, macros, used by the application, whether the application made of generated code or hand written code.

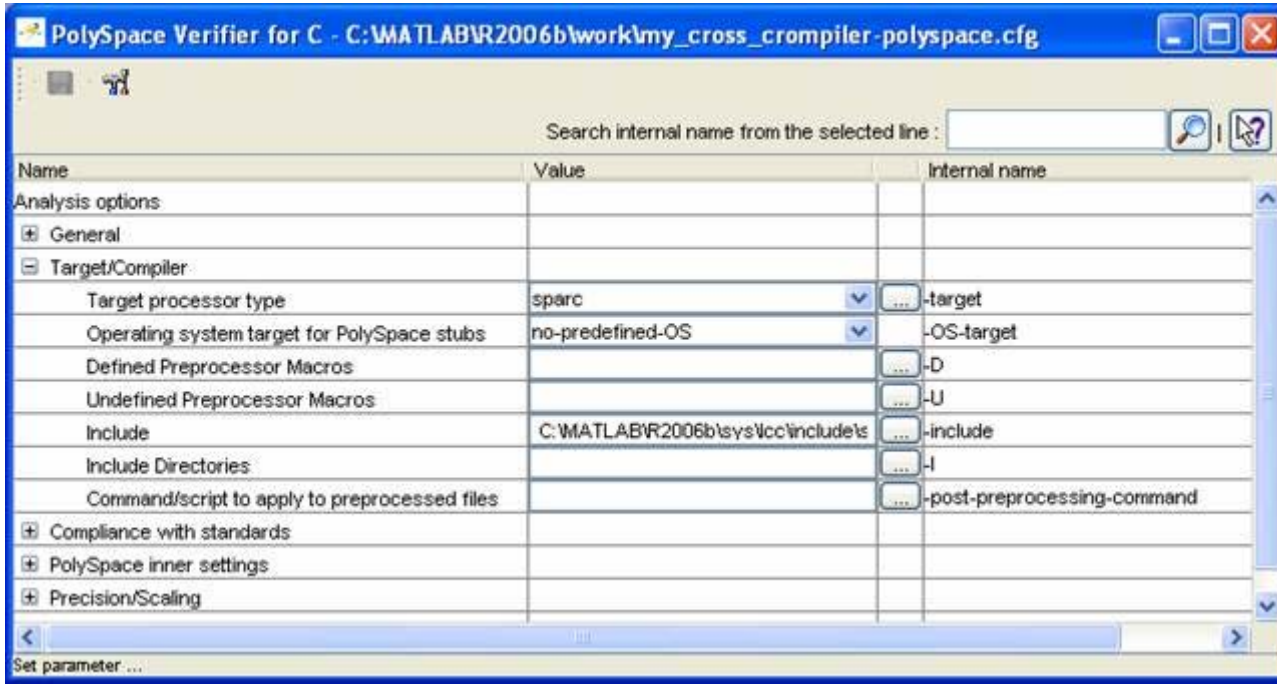
For more information, refer to and "Option Descriptions" in the *PolySpace Products for C Reference*.

Creating a PolySpace Configuration File Template

To Create a configuration file template:

1 In the Simulink model window, select **Tools > PolySpace > PolySpace Utilities > Configure project**.

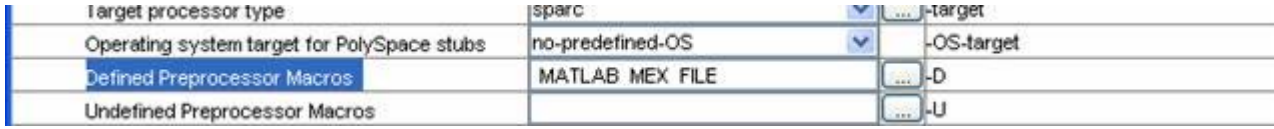
The PolySpace Launcher interface opens, allowing you to customize the target and cross compiler.



Target and cross compiler settings in PolySpace® tools

2 The `-target` option defined the size of types. You can configure a custom target by selecting `mcpu (advanced)` at the bottom of the drop-down list

3 You can configure cross compiler settings by clicking on the `-D` options.



Note `MATLAB_MEX_FILE` is a directive option that is needed when the LCC cross-compiler is specified. Defining templates can be use in all subsequent verification.

4 Save the configuration file and close the interface.

5 Copy the file in `<matlabroot>/polyspace/cfg` directory.

6 Rename it in `my_cross_compiler.cfg` (It could be any other name).

7 Type in the MATLAB command window:

```
PolySpaceSetTemplateCFGFile
('C:\MATLAB\R2006b\polyspace\cfg\my_cross_compiler.cfg')
```

Create a template configuration file

This configuration file can now be used as a template for all subsequent verification.

3

PolySpace Utilities

PolySpace Utilities Library

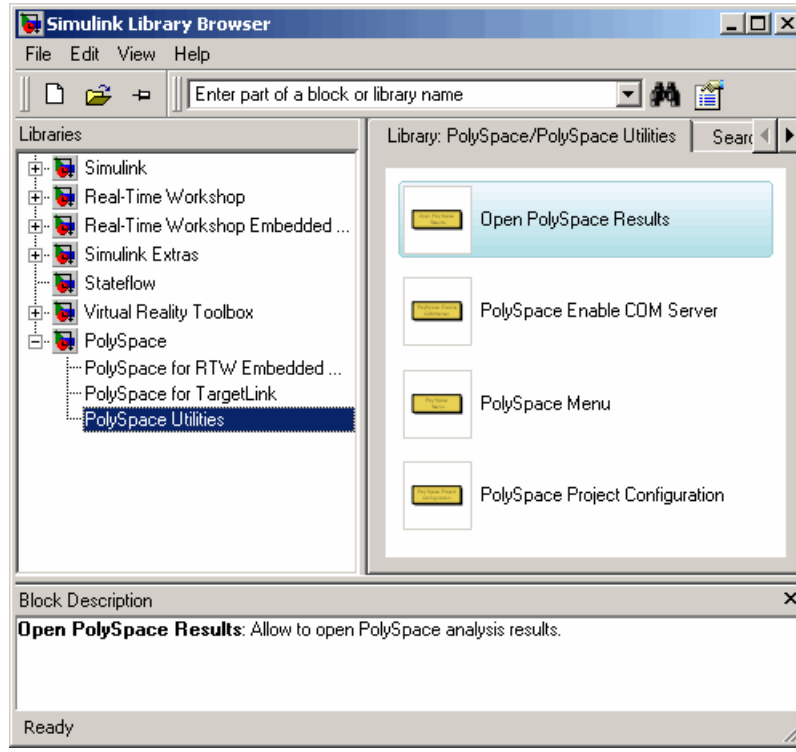
In this section...

Overview of PolySpace Utilities Library

The PolySpace Utilities Library consists of four blocks:

- **Open PolySpace Results**
- **PolySpace Enable COM Server**
- **PolySpace Menu**
- **PolySpace Project Configuration**

They can either be run directly from the Simulink library browser or dragged into a Simulink model (see next figure).



PolySpace® Utilities

Open PolySpace Results

This option allows the results of the PolySpace verification to be viewed and easy navigation with a right click from the PolySpace results to an element in the Simulink model.

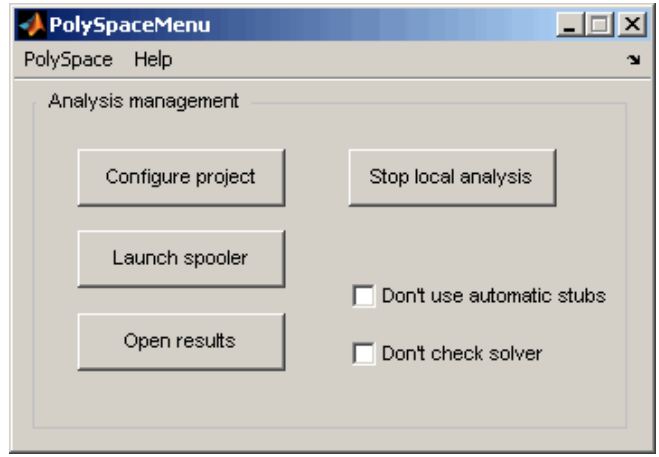
PolySpace Enable COM Server

This block is called by default with the “Open PolySpace Results” block. This block is mandatory when The PolySpace Viewer has been opened outside a Simulink session to enable the Back to Model feature inside the Viewer.

Note If you do not have administrator rights to your system, you may receive an error when using the Back to Model feature in the Viewer. This occurs because the PolySpace Viewer cannot call the `regserver` command. If you do not have administrator rights to your system, ask your system administrator to run the command `matlab /regserver` on the version of MATLAB you use with PolySpace software.

PolySpace Menu

The menu consists of two sections, the first for managing the verification and the second for configuring the tools and documentation.



PolySpace® Menu

Analysis Management

Analysis management contains the following options:

- **Configure project** – Opens the PolySpace configuration dialog, for more information see next section.
- **Launch spooler** – Opens the PolySpace spooler. For more information, see "Running Verifications on PolySpace Server"in the *PolySpace Products for C User's Guide*.

- **Open results** – Opens the PolySpace Viewer with the last available results. If the verification has been done on the server, downloading them first is required before clicking on this button. It is recommended to not change the proposed directory during download.
- **Stop local verification** – Stops a verification running on the local machine. If the verification has been remotely spooled this option will only work during the compilation phase before the verification is sent to the server. However, you can click the **Launch spooler** button and stop the verification from the spooler dialog.

General Options

Tools & Documentation contains the following options:

- **Don't use automatic stubs** – Enables/disables PolySpace automatic stubbing of certain blocks behavior. This behavior depends on the code generator being used and is described in the documentation specific to your code generator below.
- **Don't check solver** – Disables the check of the solver used with Real-Time Workshop Embedded Coder software.

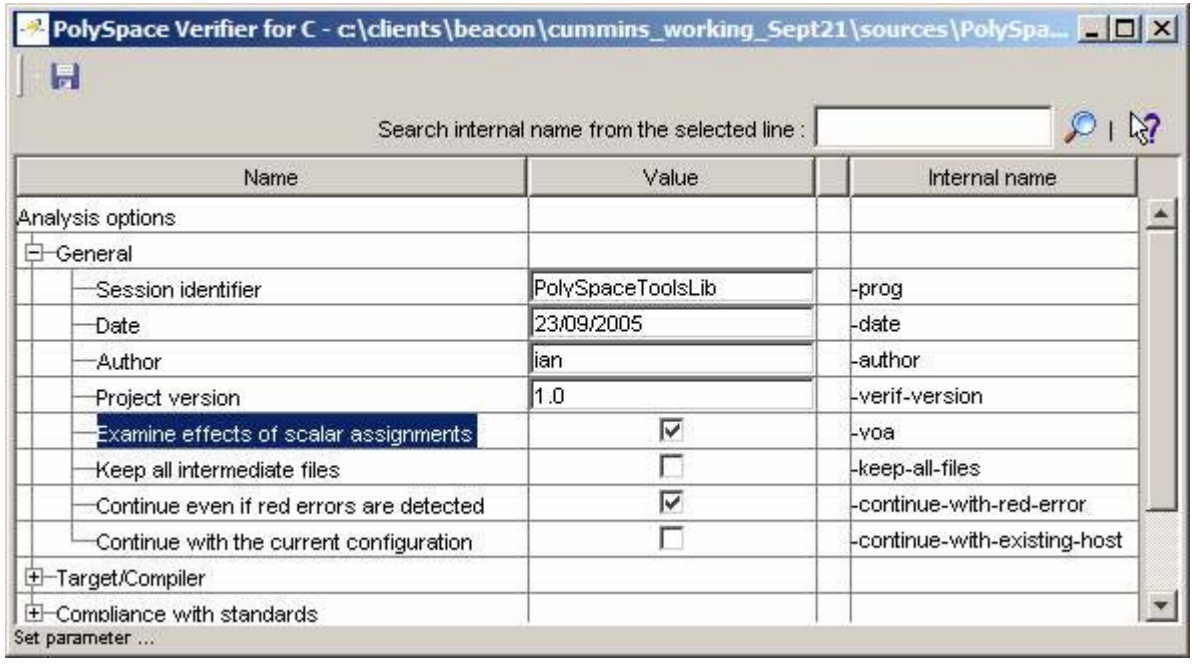
PolySpace Project Configuration

Clicking on "Project configuration" starts a cut-down PolySpace launcher (see next figure).

Next figure allows the configuration of the PolySpace project. For example setting items such as the processor type the code has been generated for, Compilation flags etc. The first time the tool is run a template configuration is created with the following options set:

```
-continue-with-existing-host
-ignore-float-rounding
-OS-target no-predefined-OS
-allow-ptr-arith-on-struct
-results-dir results
```

Other options are automatically set depending on the code generator being used. See the documentation for specific code generators below for more information.

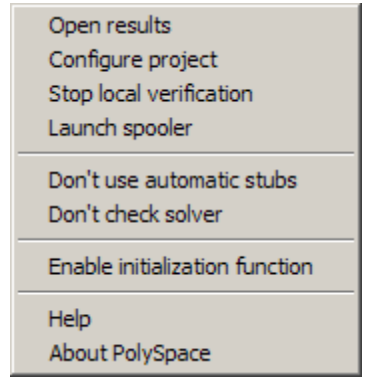


Project Configuration Interface

PolySpace Utilities Menu

The PolySpace utilities menu allows you to access the options in the PolySpace Library directly from the Simulink model window.

To access the utilities menu, select **Tools > PolySpace > PolySpace Utilities** from the model window.



The Utilities menu contains the following options:

- **Open results –** Opens the PolySpace Viewer with the last available results.

  Configure project – Opens the PolySpace configuration dialog, for more information see next section.

- **Stop local verification** – Stops a verification running on the local machine. If the verification is run on the server, this option only works during the compilation phase before the verification is sent to the server. However, you can click the **Launch spooler** button and stop the verification from the spooler dialog.

- **Launch spooler** – Opens the PolySpace spooler. For more information, see "Running Verifications on PolySpace Server"in the *PolySpace Products for C User's Guide*.

- **Don't use automatic stubs** – Specifies that the verification will not generate stubs. By default, PolySpace verification stubs all functions. Using this option allows you to use manual stubbing. For more information, see "Stubbing"in the *PolySpace Products for C User's Guide*.

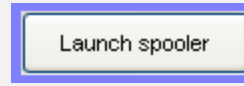
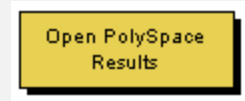
- **Don't check solver** – Specifies that the model be verified regardless of the type of solver selected. To ensure optimal precision and performance, the software checks that the model uses a fixed-step discrete solver. Selecting this option disables this check.

- **Enable initialization function** – This option enables the Back to Model feature in the Viewer, when you open the results outside a Simulink session.

- **Help** – Opens the PolySpace documentation.

PolySpace Commands Available in Batch Mode as M-Functions

You can also run the following commands from the command line.

Command	Description	Icon
PolySpaceForEmbeddedCoder	Launch PolySpace verification on code generated by Real-Time Workshop Embedded Coder software	
PolySpaceForTargetLink	Launch PolySpace on code generated by TargetLink	
PolySpaceSpooler	Inspect the queue of the remotely sent verification over the server	
PolySpaceViewer	Launch PolySpace Viewer	
PolySpaceSetTemplateCFGFile	Select a template file in batch mode	
PolySpaceGetTemplateCFGFile	Get the currently selected template file (empty by default)	
PolySpaceReconfigure	In case of a PolySpace release update without enabling the MATLAB plug-in	

Example with EmbeddedCoder:

Suppose that you open a Simulink model with the name `example.mdl`.

Enter `PolySpaceForEmbeddedCoder(example')` in the MATLAB Command window.

The verification starts.

Archives Files Produced for the PolySpace Verification

In this section...

Template files located in MATLAB installation directory\polyspace\

When a verification is first performed the tool copies the following two files into the local model directory. On subsequent verifications the files are not copied again meaning it is OK to model the copies in the model directory.

- `cfg\templateEmbeddedCoder.cfg` — This file is copied to the model_directory/model_name-polyspace.cfg at the start of the first verification of the model. It contains the template PolySpace configuration settings to support the TargetLink code generator. The templateTargetLink.cfg file can be updated with site specific settings, to ease verification of new models.

  A MATLAB command exists to change the name/location of the file which contains the template configuration:

  ```
  PolySpaceSetTemplateCFGFile(config_filename)
  ```

  This is most useful when the PolySpace verification is started as part of an automated process. Here the process would set the template configuration file to use, erase the local copy in the model directory and then start the PolySpace verification.

- `stub\ppcom_ec.sh` — This file is copied to the model_directory/ppcom_ec.sh at the start of the first verification of a model. The file is not recopied for subsequent verifications. It is used to stub lookup table types (only interpolation, not extrapolation) to improve the accuracy of verification results.

Files used in the model directory

- `model-name-polyspace.cfg` — As mentioned above this file is copied from the MATLAB installation directory\polyspace\cfg\templateEmbeddedCoder.cfg file the first time a verification is run on a model. It is subsequently modified by the Project Configuration block, or the Configure button in the option in the PolySpace Analyzer dialog. It contains the PolySpace settings for verifying the current model.
- `ppcom_ec.sh` — The PolySpace Embedded Coder post preprocessing command.
- `polyspace_additional_file_list.txt` — This file is created if the Advanced option, Select Files is used in the PolySpace Analyzer dialog box. This option allows files that are not part of the model to be analyzed together with the model. For example these files could contain custom lookup table code, custom stubs, device driver code etc. The Enable additional file list option needs to be set together with configuring the list of extra files to analyze.

Auto-generated files in the model directory

These files are generated from the model for each verification when it is started, and do not need archiving:

- `model name_drs.txt` — The DRS information extracted automatically from the model.
- `polyspace_include_dir_list.txt` — List of compilation include directories extracted from the mode.
- `polyspace_file_list.txt` — List of file contained in the model to analyze
- `model name_last_parameter.txt` — The last set of parameters used in the PolySpace Analyzer dialog box.

4

Code Generator Specific Information

PolySpace Model Link SL Product

In this section...

Overview

The PolySpace Model Link SL product has been tested with Real-Time Workshop Embedded Coder software — see the Installation Guide for more information.

Subsystems

A dialog will be presented after clicking on the PolySpace for Embedded Coder block if multiple subsystems are present in a diagram. Simply select the subsystem to analyze from the list. The subsystem list is generated from the directory structure from the code that has been generated.

Default Options

The following default options are set by the tool:

```
-I path to source code
-desktop
-D PST_ERRNO
-I matlabroot\polyspace\include
-I matlabroot\extern\include
-I matlabroot\rtw\c\libsrc
-I matlabroot\simulink\include
-I matlabroot\sys\lcc\include
```

Note *matlabroot* is the MATLAB tool installation directory.

Data Range Specification

The software automatically creates a PolySpace Data RangeSpecification (DRS) file using information from the MATLAB workspace. This DRS information is used to initialize each global variable to the range of valid values, as defined by the min-max information in the workspace.

The main sources of information are Simulink.signals and Simulink.parameters.

You can also manually define a DRS file using the PolySpace Launcher. If you define a DRS file, the software appends the automatically generated information to the DRS file you create. Manually defined DRS information overrides automatically generated information for all variables.

Code Generation Options

The Real-Time Workshop® configuration parameters settings must be configured as follows for optimum use of the tool.

Note These are the options recommended by The MathWorks for generating target code.

- **Real Time Workshop** tab:

  1. Select “Generate HTML report” and set “Include hyperlinks to model”. **Note** that if this is not set navigation from PolySpace results to the model will not work.
  2. Set the system target file to be an appropriate ert.tlc (use the browse button to locate). This is an indication that the code generator is Real-Time Workshop Embedded Coder software (and not just Real-Time Workshop software, used for rapid prototyping).

3 Set the Solver parameters "Type" to *Fixed-step*, and "Solver to *discrete* (no continuous state). It illustrates that the code has been generated for a target, and not for a simulation based on continuous timing.

- Optionally, on "*Interface panel"* tab, make sure that "Generate reusable code" is unselected. Setting this option will generate more warnings in the PolySpace results.

PolySpace Model Link TL Product

In this section...

Overview

The PolySpace Model Link TL product has been tested with the some release of the dSPACE Data Dictionary version and TargetLink Code Generator - see the Installation Guide for more information.

As the PolySpace Model Link TL product extracts information from the dSPACE Data Dictionary remember to regenerate the code before performing a PolySpace verification. This ensures that the Data Dictionary has been correctly updated.

Subsystems

A dialog will be presented after clicking on the PolySpace for TargetLink block if multiple subsystems are present in a diagram. Simply select the subsystem to analyze from the list.

Data Range Specification

The tool automatically creates PolySpace Data RangeSpecification (DRS) information using the dSPACE Data Dictionary for each global variable. This DRS information is used to initialize each global variable to the range of valid values as defined by the min-max information in the data dictionary. This allows PolySpace software to model every value that is legal for the system during its verification. Further the Boolean types are modeled having a minimum value of 0 and a maximum of 1. Defining the min-max information carefully in the model can help PolySpace verification to be more precise significantly because only range of reels values are analyzed.

You can also manually define a DRS file using the PolySpace Launcher. If you define a DRS file, the software appends the automatically generated information to the DRS file you create. Manually defined DRS information overrides automatically generated information for all variables.

DRS cannot be applied to static variables. Therefore, the compilation flags `-D static=` is set automatically. It has the effect of removing the static keyword from the code. If you have a problem with name clashes in the global name space you may need to either rename one of or variables or disable this option in PolySpace configuration.

Lookup Tables

The tool by default provides stubs for the lookup table functions. This behavior can be disabled from the PolySpace menu — see "PolySpace Menu" on page 3-4 for more information. The dSPACE data dictionary is used to define the range of their return values. Note that a lookup table that uses extrapolation will return full range for the type of variable that it returns.

Default Options

The following default options are set by the tool:

```
-I path to source code
-desktop
-D PST_ERRNO
-I dspaceroot\matlab\TL\SimFiles\Generic
-I dspaceroot\matlab\TL\srcfiles\Generic
-I dspaceroot/matlab\TL\srcfiles\i86\LCC
-I matlabroot\polyspace\include
-I matlabroot\extern\include
-I matlabroot\rtw\c\libsrc
-I matlabroot\simulink\include
-I matlabroot\sys\lcc\include
```

Note *dspaceroot* and *matlabroot* are the dSPACE and MATLAB tool installation directories respectively.

Code Generation Options

From the TargetLink Main Dialog, it is recommended to set the option “Clean code” and deselect the option “Enable sections/pragmas/inline/ISR/user attributes”.

When installing the PolySpace Model Link TL product, the tlcgOptions variable has been updated with ’PolyspaceSupport’, ’on’ (see variable in ’C:\dSPACE\Matlab\Tl\config\codegen\tl_pre_codegen_hook.m’ file).

Glossary

Atomic
In computer programming, atomic describes a unitary action or object that is essentially indivisible, unchangeable, whole, and irreducible.

Atomicity
In a transaction involving two or more discrete pieces of information, either all of the pieces are committed or none are.

Batch mode
Execution of PolySpace from the command line, rather than via the launcher Graphical User Interface.

Category
One of four types of orange check: *potential bug, inconclusive check, data set issue* and *basic imprecision*.

Certain error
See "red check."

Check
A test performed by PolySpace during a verification and subsequently colored red, orange, green or gray in the viewer.

Code verification
The PolySpace process through which code is tested to reveal definite and potential runtime errors and a set of results is generated for review.

Dead Code
Code which is inaccessible at execution time under all circumstances due to the logic of the software executed prior to it.

Development Process
The process used within a company to progress through the software development lifecycle.

Green check
Code has been proven to be free of runtime errors.

Gray check
Unreachable code; dead code.

Imprecision
Approximations are made during a PolySpace verification, so data values possible at execution time are represented by supersets including those values.

mcpu
Micro Controller/Processor Unit

Orange check
A warning that represents a possible error which may be revealed upon further investigation.

PolySpace Approach
The manner of use of PolySpace to achieve a particular goal, with reference to a collection of techniques and guiding principles.

Precision
An verification which includes few inconclusive orange checks is said to be precise

Progress text
Output from PolySpace during verification to indicate what proportion of the verification has been completed. Could be considered as a “textual progress bar”.

Red check
Code has been proven to contain definite runtime errors (every execution will result in an error).

Review
Inspection of the results produced by a PolySpace verification.

Scaling option
Option applied when an application submitted to PolySpace proves to be bigger or more complex than is practical.

Selectivitiy

The ratio (green checks + gray checks + red checks) / (total amount of checks)

Unreachable code

Dead code.

Verification

The PolySpace process through which code is tested to reveal definite and potential runtime errors and a set of results is generated for review.