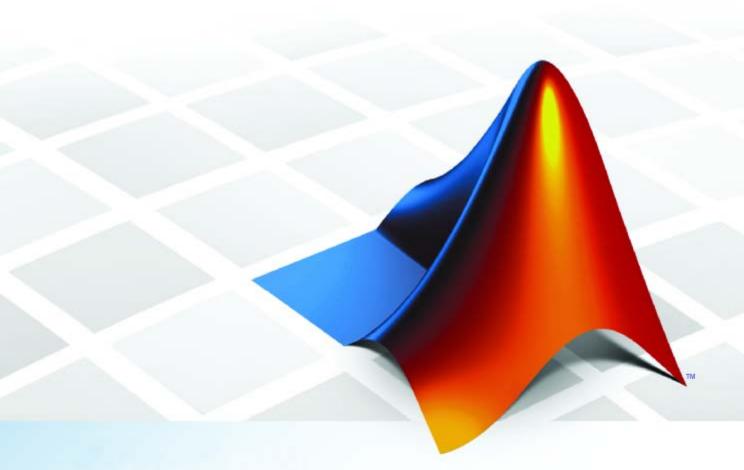
PolySpace® Model Link Products 5 User's Guide





How to Contact The MathWorks



(a)

www.mathworks.comWebcomp.soft-sys.matlabNewsgroupwww.mathworks.com/contact_TS.htmlTechnical Support

suggest@mathworks.com bugs@mathworks.com doc@mathworks.com service@mathworks.com info@mathworks.com Product enhancement suggestions Bug reports Documentation error reports Order status, license renewals, passcodes 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

© COPYRIGHT 1999–2009 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

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

Getting Started with PolySpace Model Link Products

Overview of PolySpace Model Link Products	1-2
Getting Started with Model Link Products	1-3
Overview	1-3
Creating a Simulink Model and Generating Production	
Code	1-3
Starting the PolySpace Verification	1-9
Fixing an Error in the Design and the Simulink Model	1-11
Base Workspace vs. PolySpace Data Ranges	1-14

Advanced Setup Options

2

1

Advanced Setup	2-2
Handwritten Code	2-2
Target Production Environment	2-3
Creating a PolySpace Configuration File Template	2-6

PolySpace Utilities

3

PolySpace Utilities Library
Overview of PolySpace Utilities Library
Open PolySpace Results
PolySpace Enable COM Server
PolySpace Menu
PolySpace Project Configuration
PolySpace Utilities Menu

PolySpace Commands Available in Batch Mode as M-Functions	3-9
Archives Files Produced for the PolySpace Verification	3-11
directory\polyspace\	3-11
Files used in the model directory	3-12
Auto-generated files in the model directory	3-12

Code Generator Specific Information

4

PolySpace Model Link SL Product		
Overview		
Subsystems		
Default Options		
Data Range Specification		
Code Generation Options		
PolySpace Model Link TL Product		
• -		
Overview		
Subsystems	 	
Overview Subsystems Data Range Specification	 	
Overview Subsystems	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·

Glossary

Getting Started with PolySpace Model Link Products

- "Overview of PolySpace Model Link Products" on page 1-2
- "Getting Started with Model Link Products" on page 1-3

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" on page 1-3

"Creating a Simulink Model and Generating Production Code" on page 1-3

"Starting the PolySpace Verification" on page 1-9

"Fixing an Error in the Design and the Simulink Model" on page 1-11

```
"Base Workspace vs. PolySpace Data Ranges" on page 1-14
```

Overview

In this section, you will:

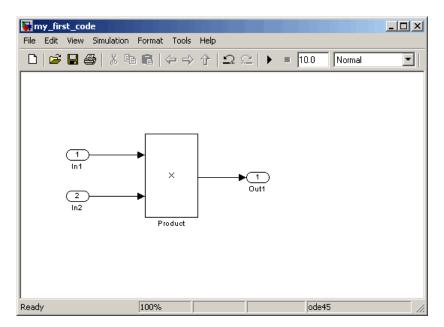
- Create a Simulink model and generate production code (For more information, see the *Real-Time Workshop Embedded Coder Getting Started Guide*)
- Start the PolySpace verification

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.

1



Create the my_first_code model

- $\textbf{3} \ \text{Select} \ \textbf{File} > \textbf{Save}, \ \text{then name the model } \textbf{my_first_code}.$
- 4 Select View > Model Explorer.

The Model Explorer opens.

Model Explorer		
📙 🗅 🚅 🐰 🖻 🛍 🗙 🖽 [明 壬 🗞 現 🗐 🖇 🌆 🔘 🔳 🖬 👘 👘	
Search: by Name	Name: Search	
Model Hierarchy	Contents of: Configuration Preferences	Real-Time Workshop
🗄 - 🚼 Simulink Root		General Report Comments Symbols Custom Code Debug Interface Code Style Templ ()
👸 Base Workspace	Name StopTime SaveOutput ProdHWDevice	- Target selection
	Solver 10.0	
🗄 🖳 🙀 my_first_code	🕸 Data Import/Export 🔽	System target file: ert.tlc Browse
	Optimization	Language: C
	Diagnostics	Description: Real-Time Workshop Embedded Coder (no auto configuration)
	Ardware Implementation 32-bit Generic	- Build process
	Model Referencing	Build process
	😢 Real-Time Workshop	Compiler optimization level: Optimizations off (faster builds)
		TLC options:
		- Makefile configuration
		Generate makefile
		Make command: make_rtw
		Template makefile: ert_default_tmf
		Custom storage dass
		Ignore custom storage classes
	Contents Search Results	Revert Help Apply
	Contents Search Results	

- 5 Select Configuration Preferences, in the Model Hierarchy.
- 6 Select Real-Time Workshop in the Configuration Preferences.

The Real-Time Workshop configuration parameters open.

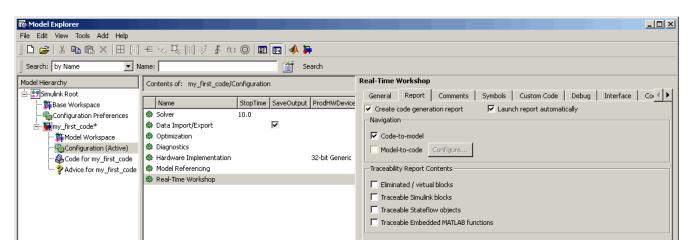
7 Select the Real-Time Workshop General tab.

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

Model Explorer File Edit View Tools Add Help		
] 🛨 🔍 🖳 🗐 🕖 🖡 fo 🔘 🔳 🔃 📣 🕽	
	Name: Search	
Model Hierarchy	Contents of: my_first_code/Configuration Name StopTime SaveOutput ProdHWDevice Solver 10.0 Data Import/Export Import/E	Real-Time Workshop General Report Comments Symbols Custom Code Debug Target selection System target file: grt.tlc Language: C
Code for my_first_code	Hardware Implementation 32-bit Generic Model Referencing Real-Time Workshop	Build process Compiler optimization level: Optimizations off (faster builds) TLC options: Makefile configuration TLC aptions:
	System target file browser: my_first_code System target file: Description: asap2.tlc ASAM-ASAP2 Data Definiti autosar.tlc AUTOSAR ert.tlc Real-Time Workshop Embed grt.tlc Visual C/C++ Project Mak ert.tlc Visual C/C++ Project Mak ert.tlc Visual C/C++ Project Mak Full name: C:\Program Files\MATLAB\R2008a\rtw\c\ert\ert.tlc Template make file: ert_defauk_tmf Make command: QK Cancel Help	And: make_rtw nakefile: grt_default_tmf ded Code: ded Code: efile on:- ded Code:

Change the code generator to Real-Time Workshop® $\mathbf{Embedded}\ \mathbf{Coder^{\mathsf{TM}}}\ \mathbf{software}$

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



Set Report Settings



🚳 Model Explorer			
File Edit View Tools Add Help			
📙 🗅 🥔 👗 🐚 🛍 🗶 🖽 📖] = % 및 [ii] ∮ ∱ fo @ [II 📧 📣 🖬	
Search: by Name	Name:	🖀 Search	
Model Hierarchy	Contents of: my_first_code/Config	Real-Time Workshop	
⊡- ∰Simulink Root ∰Base Workspace 	Name StopTin	General Report Comments Symbols Custom Code Debug Interface Code Style	Templates
	Data Import/Export	Source file (*.c) template: ert_code_template.cgt Browse	Edit
···· 🎁 Model Workspace ···· 🍓 Configuration (Active)	OptimizationDiagnostics	Header file (*.h) template: ert_code_template.cgt Browse	Edit
Code for my_first_code	Hardware Implementation Model Referencing	Data templates	
• ·	Real-Time Workshop	Source file (*.c) template: ert_code_template.cgt Browse	Edit
		Header file (*.h) template: ert_code_template.cgt Browse	Edit
		Custom templates	
		File customization template: example_file_process.tlc Browse	Edit
		Generate an example main program	

Templates Tab

- 11 In the Custom templates section, clear Generate an example main program.
- **12** Select the **Interface** tab.

1

🐼 Model Explorer		
File Edit View Tools Add Help		
] 🗅 😅 X 📭 🛍 🗙 🖽 [ii]	モミ現間多手の圖【	II 💀 🔺 🖡
Search: by Name	Jame:	Search
Model Hierarchy Model Hierarchy Smulnk Root State Workspace State Work		Real-Time Workshop General Report Comments Symbols Custom Code Debug Interface Code Style Templates Image: Style Style Software environment Target function library: C89/C90 (ANSI) Image: Style St
4 •	✓ ✓	MAT-file logging Data exchange Interface: None Interface: None Generate code only Build Image: Bailed Build

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.

🔯 Model Explorer				
File Edit View Tools Add Help				
🗅 😅 % 🖻 🛍 🗙 🖽 [+E 🐁 🖳 [0] 🖗 🖡 ft) 🔘 🔳 🖪 📣 🕽		
Search: by Name 💌 N	lame:	Search Search		
Model Hierarchy	Contents of: my_first_code/0	Configuration		Solver
🗄 🖅 Simulink Root				Simulation time
🎁 Base Workspace	Name	StopTime SaveOutput ProdHW	Device	Start time: 0.0 Stop time: 10.0
···· 🎭 Configuration Preferences	🔅 Solver	10.0		
⊡- Se my_first_code*	Data Import/Export			Solver options
	Optimization Diagnostics			Type: Fixed-step Solver: discrete (no continuous states)
	Hardware Implementation	32-bit G	operic	
Advice for my_first_code	Model Referencing	52 bit G	anone	Fixed-step size (fundamental sample time): auto
And the Lot My Turst Toge	Real-Time Workshop			Tasking and sample time options
				Periodic sample time constraint:
				Tasking mode for periodic sample times: Auto
				T Automatically handle rate transition for data transfer
				Deterministic data transfer: Whenever possible 🔽
				Higher priority value indicates higher task priority

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.

A PolySpace Analys	er		<u>_ </u>
Analysis Parameters			
Subsystem	my_first_code_ert_rtw\my_first_code	Browse	From Selection
Results directory	H:\Documents\MATLAB\results_my_first_code	Browse	
Analysis Precision	02 - default		
- Advanced			
Enable additional	I file list Select Files	Project Configuration	Configure
E Send to PolySpa	ce Server		
		Start	Cancel

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.

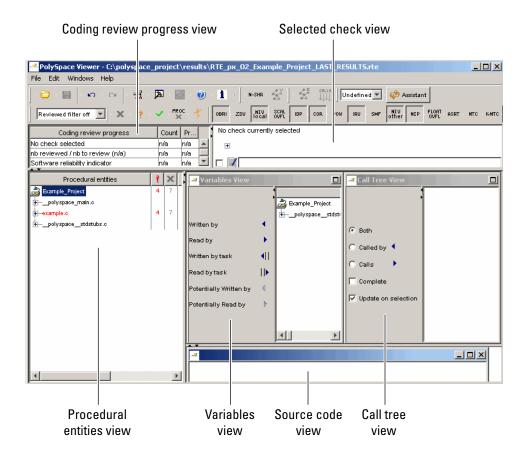
Note If you perform a server verification, you must download your results from the server before you can view them. For more information, see "Downloading Results from Server to Client" in the *PolySpace Products for C User's Guide*.

To view your results:

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

After a few seconds, the PolySpace Viewer opens.

T



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

11 × + - 3	5 1 ann 200 品語 m an	now and now and the first state with some
Writen by 4 Read by 4 Writen Dytesk 4() Read bytesk () Potentially Writen by 4	Variables	Calls my_first_code.my_first_code_str Called by 4 Colerate F Constants
	t >	C Updee on selection
25 • 1: 26 • 1: 27 • 2 28 */	:pert: ' <u><reet>/Outl</reet></u> ' inco apert: ' <u><reet>/Inl</reet></u> ' apert: ' <u><reet>/In2</reet></u> ' reduct: ' <u><reet>/Product</reet></u> '	
29 <u>my fi</u> 30 j 31	rst code Y.Out1 = sy firs	t code U.Ini 🛛 my first code U.In2

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.

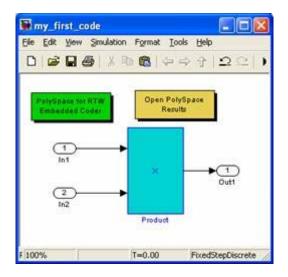


Overflow in the code

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.

🙀 Signal Properties: my_	_entry1		×
Signal name: my_entry1			
🔲 Signal name must resolve	to Simulink signal object		
Logging and accessibility	Real-Time Workshop	Documentation	
RTW storage class:	ExportedGlobal		
RTW storage type qualifier:			
	OK Cancel	Help	Apply

Details of a signal

Update Range of Signals

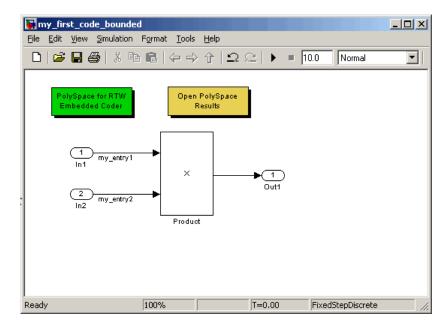
To update your signal ranges:

- 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.

Model Explorer			
File Edit View Tools Add Help			
🗅 😅 👗 🗈 🛍 🗙 🎛 🏢 🗲 🗞 🖻	🕹 [0] 🖗 🛧 fo 🔘 🖡	II 🗖 🗆 📣	k 🙀
Search: by Name Name:			Search
Model Hierarchy	Contents of: Base Worksp	ace	Simulink.Signal: my_entry1
- Simulink Root	, Name Value	DataType	- Data type: Jauto 💽 💦
Configuration Preferences	€ my_entry1	auto	Dimensions: -1 Complexity: auto
- Tirst_code_bounded	← my_entry2	auto	Sample time: -1 Sample mode: auto
			Minimum: -15 Maximum: 15
			Initial value: Units:
···· ? Advice for my_first_code_bounded ····· ? PolySpace for RTW Embedded Coder			Code generation options
			Storage class: ExportedGlobal
			Alias:
			Description:
	•	Þ	
	<u>C</u> ontents <u>S</u> earch Resu		<u>R</u> evert <u>H</u> elp <u>A</u> pply
,			

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.

1

🙀 Real-Time Workshop Report		
Back Forward Contents Summary	19 20 21 22 23 24	<pre>/* External outputs (root outports fed by signals with autc ExternalOutputs my_first_code_b my_first_code_bounded_Y; /* Real-time model */ DT_WODEL wy_first_code_bounded_wy_first_code_bounded_W_;</pre>
Traceability Report Subsystem Report Generated Source Files ert main.c	24 25 26 27 28	<pre>RT_MODEL_my_first_code_bounded my_first_code_bounded_M_; RT_MODEL_my_first_code_bounded *my_first_code_bounded_M =</pre>
<u>my first code bounded.c</u> <u>my first code bounded.h</u> <u>my first code bounded p</u> <u>my first code bounded ty</u>	29 30 31 32	<pre>void my_first_code_bounded_step(void) { /* Outport: '<<u>Root>/Out1</u>' incorporates: * Inport: '<root>/In1'</root></pre>
rtwtypes.h	33 34 35 36	<pre>* Inport: '<<u>Root>/In2</u>' * Product: '<u><root>/Product</root></u>' */ my_first_code_bounded_Y.Out1 = my_entry1 * my_entry2;</pre>
	37 38 39	<pre>} /* Model initialize function */</pre>
	40 41	<pre>void my_first_code_bounded_initialize(void) {</pre>
		OK Cancel Help Apply

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.

Analys PolySpace Analys	ser			_ 🗆 🗡
_ Analysis Parameters				
Subsystem	my_first_code_ert_rtw/my_first_code	•	Browse	From Selection
Results directory	H:\Documents\MATLAB\results_my_f	irst_code	Browse	
Analysis Precision	02 - default			
Advanced				
Enable additiona	al file list Select Files	Project Con	figuration	Configure
Send to PolySp	ace Server			
		S	Start	Cancel

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.

PolySpace Analyser	Jelect Subsystem
Analysis Parameters	Select subsytem:
Subsystem my_first_code_bounded Browse From S	my_first_code
Results directory 2006b/work/getting_started/vesults_my_first_code_bounded Browse	
Analysis Precision 02 - default	
Advanced	
Enable additional file list Select Files Project Configuration Configu	A second s
Remote analysis	·
Cesktop analysis	OK Cancel From Selection
PolySpace Car	

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:

File Edit Tools Windows	C:\MATLAB\R2006b\work\getting_started\results_my_first_code_bounded\RTE_px_02_my_first_co 📳 🗖 🔯
0 ■ 0 0 ≰ III × ?	
Procedural entities	my first code_bounded.c
amy_first_code_	
I polyspace m 2	2 /* Model step function */
E my_first_code_ 2	
2	
_init_globals 2	5 /* Outport: '< <u>Root>/Out1</u> ' incorporates:
E my_first_cod 2	6 * Inport: ' <root>/Inl'</root>
8 my_first_cot 2	7 * Inport: ' <root>/In2'</root>
V NIV.0 2	8 * Product: '< <u>Root>/Product</u> '
	9 */
and a second sec	0 my first code bounded Y.Out1 = my entry1 my entry2;
✓ NV.2 3	1)
my_first_coc 3	
putyspace	3 /* Model initialize function */
3	
	5 (
	6 /* Registration code */
3	
3	R /* external innuts */

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*.

Advanced Setup Options

Advanced Setup

In this section ...

"Handwritten Code" on page 2-2

"Target Production Environment" on page 2-3

"Creating a PolySpace Configuration File Template" on page 2-6

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:

A PolySpace Analys	ser			
F Analysis Parameters	,			
Subsystem	hand_written_code		Browse	From Selection
Results directory	H:\Documents\MATLAB\work\hand	_written\results_hand_wi	Browse	
Analysis Precision	O2 - default			
C Advanced				
Enable additiona	al file list Select Files	Project C	onfiguration	Configure
Send to PolySpa	ace Server			
			Start	Cancel
			Start	

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

📣 Additional Files	
Additional Files To Analyse	
	Add
	Remove
	Remove All
_	
OK Cancel	

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)

🔯 Model Explorer	
File Edit View Tools Add He	elp
🗅 🚅 % 🖻 🛍 🗙 8	∃ 囲 モ 稔 戛 囲 彡 ♪ භ ◎ 囲 画 参 🖡
Search: by Name	Name: Search
Contents of: my_first_code/	Hardware Implementation
	Embedded hardware (simulation and code generation)
Name	Device vendor: Generic
🏶 Solver	
🏟 Data Import/Export	Device type: Unspecified (assume 32-bit Generic)
Optimization	Number of bits: char: 8 short: 16 int: 32
Diagnostics	long: 32 native word size: 32
🏟 Hardware Implementation	Byte ordering:
🏟 Model Referencing	
🏟 Real-Time Workshop	Signed integer division rounds to: Undefined
	Shift right on a signed integer as arithmetic shift
	Emulation hardware (code generation only)
	✓ None
	Revert Help Apply
<u>Contents</u> earch Results	
	li li

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.

```
Command Window
>> mex -setup
Please choose your compiler for building external interface (MEX) files:
Would you like mex to locate installed compilers [y]/n?
Select a compiler:
[1] Lcc C version 2.4.1 in C:\MATLAB\R2006b\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft Visual S
[0] None
Compiler: 1
Please verify your choices:
Compiler: Lcc C 2.4.1
Location: C:\MATLAB\R2006b\sys\lcc
Are these correct?([y]/n):
Trying to update options file: C:\Documents and Settings\Marc Lalo\Applicatio
From template:
                            C:\MATLAB\R2006b\bin\win32\mexopts\lccopts.bat
Done . . .
>>
```

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:

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.

PolySpace Verifier for C - C:\MATLAB\R20	006b\work\my_cross_	crompiler	pol	yspace.cfg	
hr 🔳	Search internal name fr	om the selecte	ed lin	e: 🔎 I	2
Name	Value			Internal name	
Analysis options					^
闭 General					
Target/Compiler			1		
Target processor type	sparc	*		-target	
Operating system target for PolySpace stubs	no-predefined-OS	~		-OS-target	
Defined Preprocessor Macros				-D	
Undefined Preprocessor Macros				-0	
Include	C:WATLAB/R2006b/sys	Vcc'linclude's		-include	
Include Directories				l <mark>-1</mark>	
Command/script to apply to preprocessed files				-post-preprocessing-command	
Compliance with standards					
PolySpace inner settings					
Precision/Scaling					~
<	al. Ome		4		>
Set parameter					

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.

Target processor type	sparc	V	<u> </u>	-target	
Operating system target for PolySpace stubs	no-predefined-OS	Y		-OS-target	
Defined Preprocessor Macros	MATLAB MEX FILE			-D	
Undefined Preprocessor Macros				-U	

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')
```

		ug [Deswop	Window	Leib	
> Do	1 mgma	ang.	of Town	lateCE	CELLA (IC.) MATLARY POOR (Inchange	alatalar aroas compiler stall
Po	olySpa	ceS	etTemp	plateCF	GFile('C:\MATLAB\R2006b\polyspac	ce/cfg/my_cross_compiler.cfg')

Create a template configuration file

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

PolySpace Utilities

- "PolySpace Utilities Library" on page 3-2
- "PolySpace Commands Available in Batch Mode as M-Functions" on page $3{\text{-}9}$
- "Archives Files Produced for the PolySpace Verification" on page 3-11

PolySpace Utilities Library

In this section...

"Overview of PolySpace Utilities Library" on page 3-2

"Open PolySpace Results" on page 3-3

"PolySpace Enable COM Server" on page 3-3

"PolySpace Menu" on page 3-4

"PolySpace Project Configuration" on page 3-5

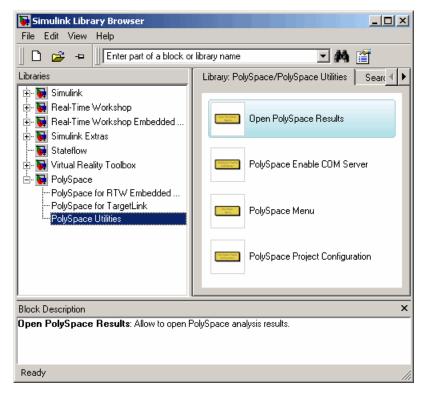
"PolySpace Utilities Menu" on page 3-6

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.

📣 Po	ySpaceMenu	
PolySp	ace Help	ند
An	alysis management —	
	Configure project	Stop local analysis
	Launch spooler	Don't use automatic stubs
	Open results	Don't check solver

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.

Search interr	nal name from the selected line : $ig \lceil$	ا 🔍
Name	Value	Internal name
Analysis options		
⊟-General		
-Session identifier	PolySpaceToolsLib	-prog
—Date	23/09/2005	-date
-Author	lian	-author
-Project version	1.0	-verif-version
Examine effects of scalar assignments		-voa
Keep all intermediate files		-keep-all-files
-Continue even if red errors are detected	V	-continue-with-red-error
Continue with the current configuration		-continue-with-existing-host
 ──Target/Compiler		

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

Command	Description	lcon
PolySpaceForEmbeddedCoder	Launch PolySpace verification on code generated by Real-Time Workshop Embedded Coder software	PolySpace for RTW Embedded Coder
PolySpaceForTargetLink	Launch PolySpace on code generated by TargetLink	PolySpace for Targetlink
PolySpaceSpooler	Inspect the queue of the remotely sent verification over the server	Launch spooler
PolySpaceViewer	Launch PolySpace Viewer	Open PolySpace Results
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	

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

Example with EmbeddedCoder:

Suppose that you open a Simulink model with the name $\tt example.mdl.$

 $\label{eq:commanded} Enter \ \texttt{PolySpaceForEmbeddedCoder(example')} \ in the \ MATLAB \ Command \ window.$

The verification starts.

Archives Files Produced for the PolySpace Verification

In this section...

"Template files located in MATLAB installation directory \polyspace \" on page 3-11

"Files used in the model directory" on page 3-12

"Auto-generated files in the model directory" on page 3-12

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.

Code Generator Specific Information

- "PolySpace Model Link SL Product" on page 4-2
- "PolySpace Model Link TL Product" on page 4-5

PolySpace Model Link SL Product

In this section ...

"Overview" on page 4-2

"Subsystems" on page 4-2

"Default Options" on page 4-2

"Data Range Specification" on page 4-3

"Code Generation Options" on page 4-3

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:
 - 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" on page 4-5

"Subsystems" on page 4-5

"Data Range Specification" on page 4-5

"Lookup Tables" on page 4-6

"Code Generation Options" on page 4-7

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).



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.