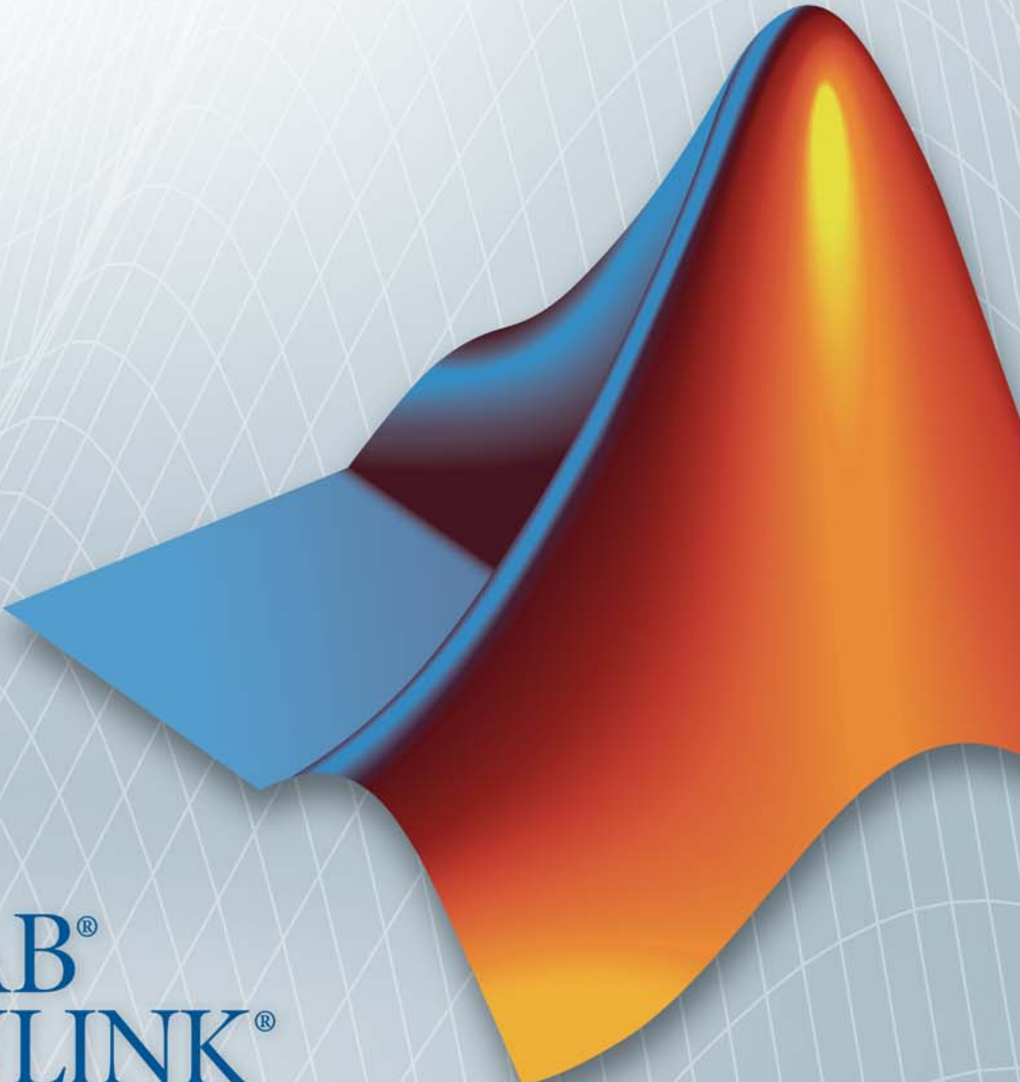
SB2SL

User's Guide

R2012b



MATLAB®
&SIMULINK®

MathWorks®

How to Contact 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.

SB2SL User’s Guide

Trademarks

Patents

Revision History

October 1998	Online only	Revised for Version 2 (Release 11)
January 1999	Online only	Minor revision
September 1999	Online only	Minor revision for Release 11.1
September 2000	Online only	Minor revision for Release 12
June 2004	Online only	Minor revision for Release 14
October 2004	Online only	Minor revision for Release 14SP1
September 2005	Online only	Minor revision for Release 14SP3
March 2006	Online only	Minor revision for Release 2006a+
September 2006	Online only	Minor revision for Release 2006b+
March 2007	Online only	Minor revision for Release 2007a+
September 2007	Online only	Revised for Version 2.7 (Release 2007b+)
March 2008	Online only	Revised for Version 2.7.1 (Release 2008a+)
October 2008	Online only	Revised for Version 2.7.2 (Release 2008b+)
March 2009	Online only	Revised for Version 2.7.3 (Release 2009a+)
September 2009	Online only	Revised for Version 2.7.4 (Release 2009b+)
March 2010	Online only	Revised for Version 2.7.5 (Release 2010a+)
September 2010	Online only	Revised for Version 2.7.6 (Release 2010b+)
April 2011	Online only	Revised for Version 2.7.7 (Release 2011a+)
September 2011	Online only	Revised for Version 2.7.8 (Release 2011b+)
March 2012	Online only	Revised for Version 2.7.9 (Release 2012a+)
October 2012	Online only	Revised for Version 2.7.10 (Release 2012b+)

Contents

1 Converting SystemBuild SuperBlocks to Simulink Models

2 Function Reference

A Conversions

Index

Converting SystemBuild SuperBlocks to Simulink Models

Introduction

In this section...

What Is SB2SL?

You can translate National Instruments® SystemBuild™ SuperBlocks to Simulink® models using the SystemBuild to Simulink Translator (SB2SL). This tool enables you to bring SystemBuild legacy models into Simulink without recreating the original models. SB2SL reads a SystemBuild ASCII format model file and creates a Simulink model that represents the structure and hierarchy of the SystemBuild model.

For each SystemBuild SuperBlock in your model, you can:

- Create a Simulink model that represents the structure and hierarchy of your SystemBuild model.
- Translate National Instruments Xmath® data from the SystemBuild model into MATLAB® variables in the MATLAB workspace.
- Produce a report providing details of the translation.

SB2SL translation is performed on a block-by-block basis. Except for a few blocks, all SystemBuild blocks are translated into either:

- Its Simulink counterpart
- A masked subsystem block containing the computational equivalent if no Simulink counterpart exists

When SB2SL cannot translate a block, it inserts an appropriate blank placeholder block in the resulting Simulink model.

Once you translate your SystemBuild model into the Simulink environment, the results of the Simulink simulation match the results of a SystemBuild simulation. When you convert your model to a Simulink model, optimizations such as vectorization, acceleration modes, and solver selection are also available.

Due to modeling differences between the two environments, you might want to perform further model optimizations to achieve top simulation performance. Validate all models after translation.

Software Requirements

Version 2.7.10 of SB2SL requires MATLAB Version 8.0 and Simulink Version 8.0. For general system requirements, see the installation documentation.

You can apply SB2SL to SystemBuild files saved from SystemBuild Version 5.0 through Version 6.2 on Linux® or PC systems in ASCII format. However, new blocks introduced since SystemBuild Version 6.0 cannot be converted. For more information, see the list of blocks not converted in “Limitations” on page 1-38.

Installation

The SB2SL software is available only through Web download at `http://www.mathworks.com/support/matrixx/transition/sb2sl`. Follow the installation instructions from the Web page to download the version you require. For further instructions on how to install the Linux version of the SB2SL software, see the installation documentation.

Optional Step to Convert to Simulink with SB2SL

Before converting your SystemBuild model to Simulink with SB2SL, consider creating baseline code against which you can verify that the SB2SL converted model returns acceptable results. You can use National Instruments AutoCode™ software to generate C code for existing SystemBuild models. Creating S-function blocks with AutoCode generated C code is a quick way to package and use existing SystemBuild models in the Simulink environment. If your SystemBuild model is large or complicated, consider using the AutoCode software as part of your conversion process.

Using AutoCode:

- Encapsulates existing SystemBuild models for use within Simulink.
- Retains the exact behavior of the AutoCode within Simulink.

To use the AutoCode software, obtain a license for the National Instruments AutoCode product.

Note You can use the AutoCode procedure as part of your workflow only if you do not need to use a scheduler with your model.

Alternatively, you can contact the MathWorks® Consulting Services group about the fee-based support for this process.

A suggested workflow to include the AutoCode software within a Simulink model follows. You must have a working knowledge of the following software:

- National Instruments AutoCode
- Simulink, including S-functions

In addition, decide how you want to partition your SystemBuild model for AutoCode sections.

1. Use the National Instruments AutoCode software to generate C code for these partitions in the existing SystemBuild models.

2. Examine the generated C code and understand the structures and functions for each partition.
3. Modify the generated C file to make sure that the include files list has only:
   - `#include <stdio.h>`
   - `#include <math.h>`
   - `#include  sa_types.h`
4. Copy the `sa_types.h` file into a folder local to the Simulink model.
5. Create a Simulink model.
6. In this new model, wrap the generated code for each partition in its own S-function. If the code is simple, you might be able to use the S-Function Builder tool. Otherwise, use the `sfuntmpl_basic.c` file.
   a. Compile the S-functions.
   b. Add an S-function block for each compiled S-function to the Simulink model.
7. Connect the blocks as appropriate to simulate the original behavior of the SystemBuild model.
8. Configure the model parameters, such as the solver settings and so forth.
9. Simulate the model and record results. You might need to change configuration settings until you are satisfied with the settings.
10. Use the SB2SL tool to convert a partition of the original SystemBuild model to a Simulinkmodel.
11. Replace the AutoCode code S-function for this partition with the SB2SL converted partition.
12. Simulate the Simulink model and record results.
13. Compare the results of the two simulations.
14. As necessary, modify the SB2SL converted partition until you get desired results.

You might need to repeat steps 10 to 14 until you are satisfied with the converted SB2SL model.

Use SB2SL

In this section...

Prepare the Model for Conversion

Before translating a SystemBuild model, save it in ASCII format (usually with a file extension `.xmd` or `.sbd`). Also, check that you have write permission for the folder that contains the file to convert.

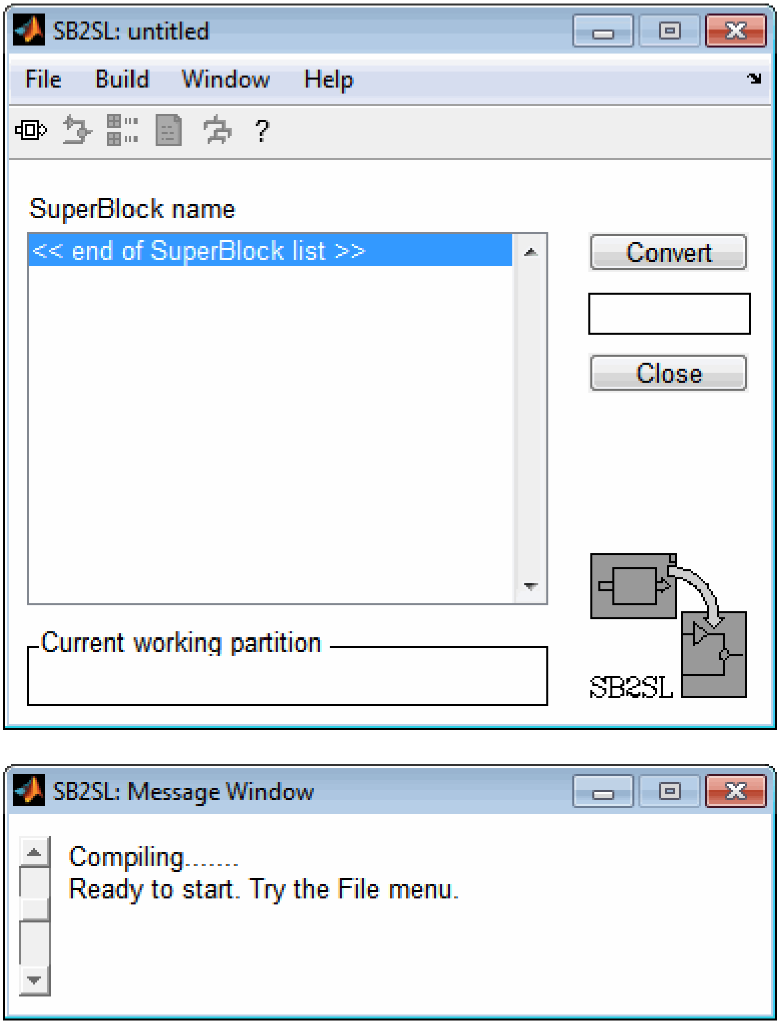
To enable the transfer of parameterized variables (`%vars`) from SB2SL software, make sure the variables are declared and resident in the Xmath workspace. Then save the SystemBuild model with the **Xmath Variables** option set to **Save All**.

Start SB2SL

To start SB2SL, at the MATLAB command prompt, type:

```
sb2sl
```

This opens the main SB2SL graphical user interface (GUI) and an associated message window.



Compiling.......
Ready to start. Try the File menu.

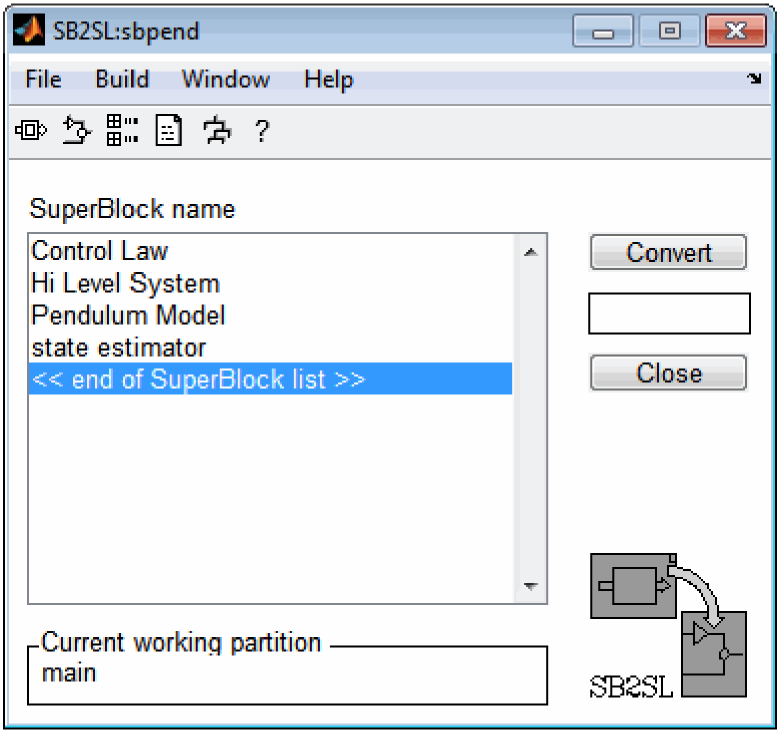
Main SB2SL GUI and Message Window

Load a SystemBuild Model into SB2SL

Select **File > Open** in the SB2SL main GUI to load a SystemBuild model. This opens a file browser from which you can select a SystemBuild model file. Once you select the name of a SystemBuild file in the browser, SB2SL:

- Opens the file
- Loads all of the parameters, if any, into the MATLAB workspace
- Lists the names of all the SuperBlocks in your model in a list

You can follow the process with this tutorial by loading the `.xmd` file, `sbpend.xmd`, included with SB2SL.



SuperBlock name

Control Law
Hi Level System
Pendulum Model
state estimator
<< end of SuperBlock list >>

Current working partition
main

Main SB2SL GUI and Message Window with sbpend

Hint To locate the directory from which to browse for `sbpend.xmd`, type `which sbpend.xmd` at the MATLAB command prompt.

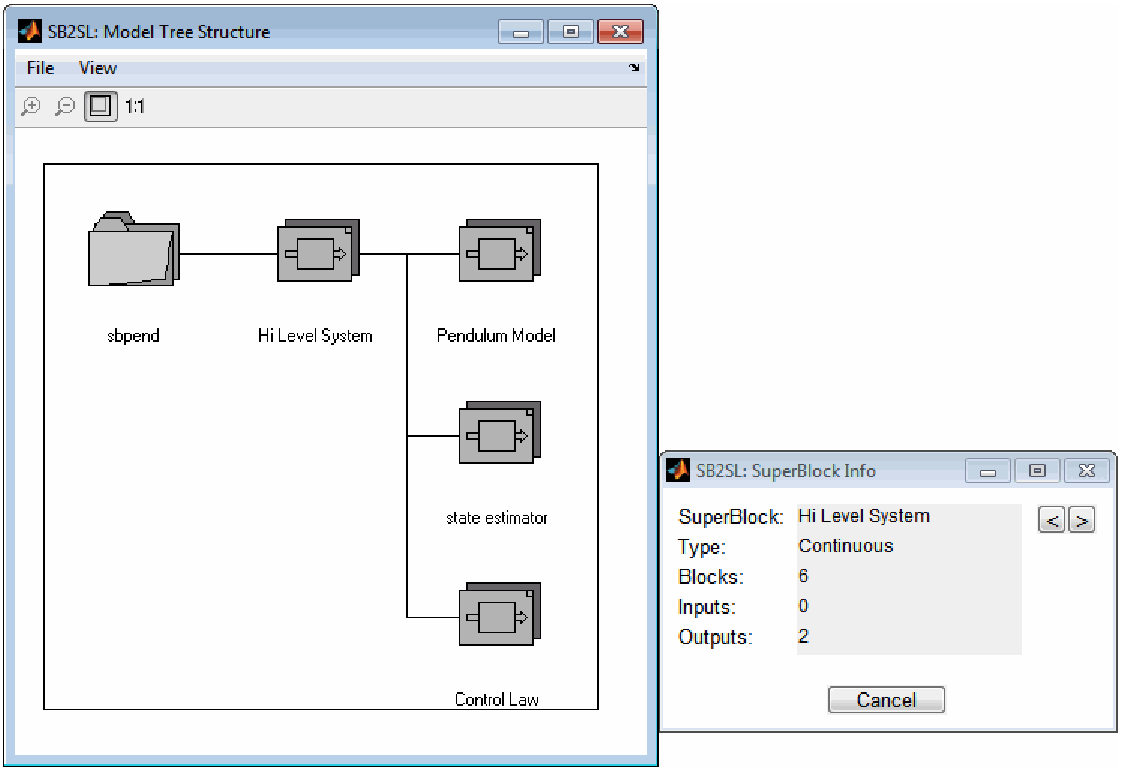
Select SystemBuild SuperBlocks

You can use SB2SL to convert SystemBuild SuperBlocks to Simulink models at any level in the SystemBuild hierarchy. To begin the process of SuperBlock conversion, select the name of a top-level SuperBlock you want to convert from the list in the main SB2SL GUI. This action highlights all SuperBlock names referenced by the selected SuperBlock.

Alternatively, you can display the SuperBlocks in a tree view by selecting **Window > Tree** in the SB2SL main GUI. This opens the Model Tree Structure window. From this window, you can use your mouse to select the SuperBlock you want to convert to a Simulink diagram. To display all the blocks in the tree, select **View > Level > 2**.

Tip For larger or more complicated SystemBuild models, consider translating the model piecemeal (for example, by subsystem) instead of the entire model all at once.

If you right-click a SuperBlock icon, a window opens that contains additional information related to that SuperBlock (for example, type, number of blocks, etc.).



SuperBlock: Hi Level System
Type: Continuous
Blocks: 6
Inputs: 0
Outputs: 2

Model Tree Structure and SuperBlock Information Windows

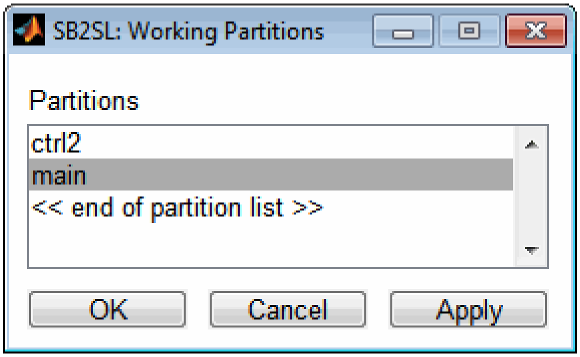
Select a SuperBlock Partition for Conversion

A SystemBuild model can contain data in separate partitions associated with each SuperBlock. When you load a SystemBuild model into SB2SL, all associated partitions are loaded into the MATLAB workspace as MATLAB structures. When you use SB2SL to convert a SuperBlock into a Simulink model, you must select the partition from which to reference the data for building the model.

To choose the data partition:

1 Select **Build > Partition** in the main SB2SL GUI.

  This opens the following window:



2 Select the partition you want your Simulink model to use, and click **Apply**.

Set Translation Options

Before you convert your SystemBuild model to a Simulink one, you can set options for:

- Building the Simulink models ("Translation Build Options" on page 1-13)
- Generating reports from the translation ("Report Generation Options" on page 1-16)
- Converting the reports to various text formats ("Report Formatting Options" on page 1-18)
- Changing GUI font sizes for the translation option dialog boxes ("Window Preferences" on page 1-20)

To save translation option settings for reuse in another SB2SL session, click the **Save** button.

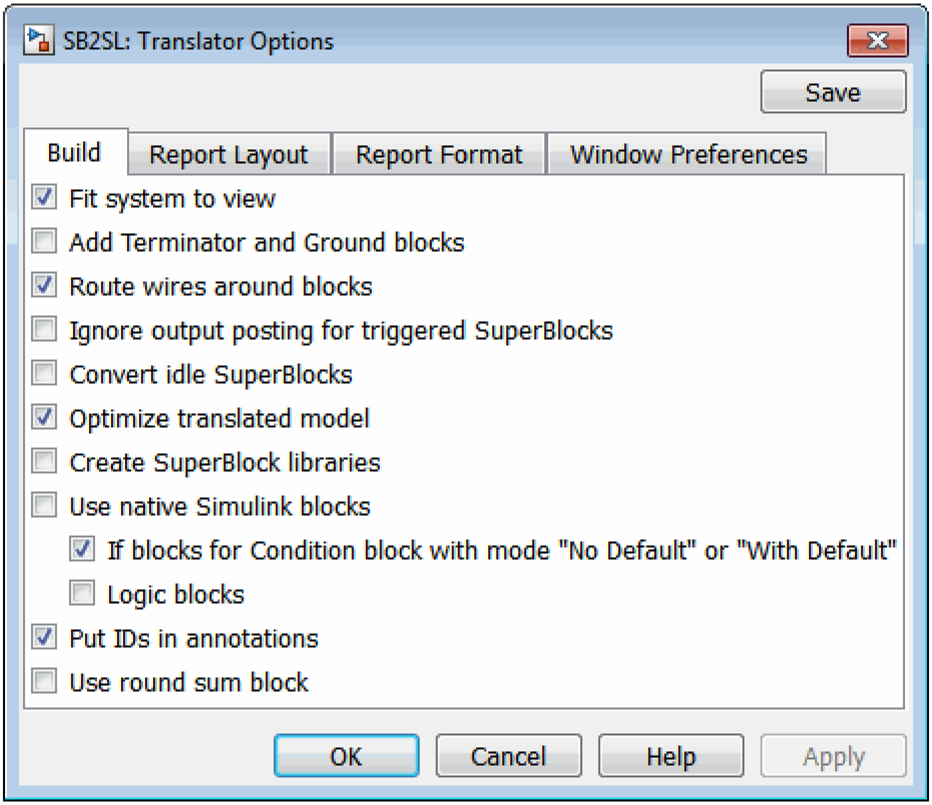
To reset default option settings, in the MATLAB Command Window, type the following:

```
rmpref('SB2SL')
```

Close and restart SB2SL. The default settings are reapplied.

Translation Build Options

To set the translation build options, select **File > Preferences** in the main SB2SL GUI.



☑ Fit system to view

□ Add Terminator and Ground blocks

☑ Route wires around blocks

□ Ignore output posting for triggered SuperBlocks

□ Convert idle SuperBlocks

☑ Optimize translated model

□ Create SuperBlock libraries

□ Use native Simulink blocks

- ☑ If blocks for Condition block with mode "No Default" or "With Default"
- □ Logic blocks

☑ Put IDs in annotations

□ Use round sum block

The following build options are available:

Option	Description
Fit system to view	Select this check box to scale the model to fit the window size. Clear this check box if you want to use the original block sizes.
Add Terminator and Ground blocks	Select this check box to terminate unconnected block inputs or outputs with Simulink Terminator or Ground blocks. By default, SB2SL does not terminate unconnected block inputs or outputs.
Route wires around blocks	Select this check box to minimize crossing blocks with signal lines in the Simulink model resulting from SB2SL translation.
Ignore output posting for triggered SuperBlocks	When you select this check box: • All triggered SystemBuild outputs are posted in “as soon as finished (SAF)” mode. • Triggered SuperBlocks assigned to “after timing requirement (ATR)” and “at next trigger (ANT)” output posting modes are ignored.
Convert idle SuperBlocks	If your model contains enabled or triggered SuperBlocks that are also nested, one or more of these blocks might never execute. To convert these idle SuperBlocks, select this check box. By default, SB2SL does not convert idle SuperBlocks.
Optimize translated model	Select this check box to maximize the use of standard Simulink blocks when translating the following SystemBuild blocks: • Data store blocks • Algebraic/logical expression blocks • Integrator blocks

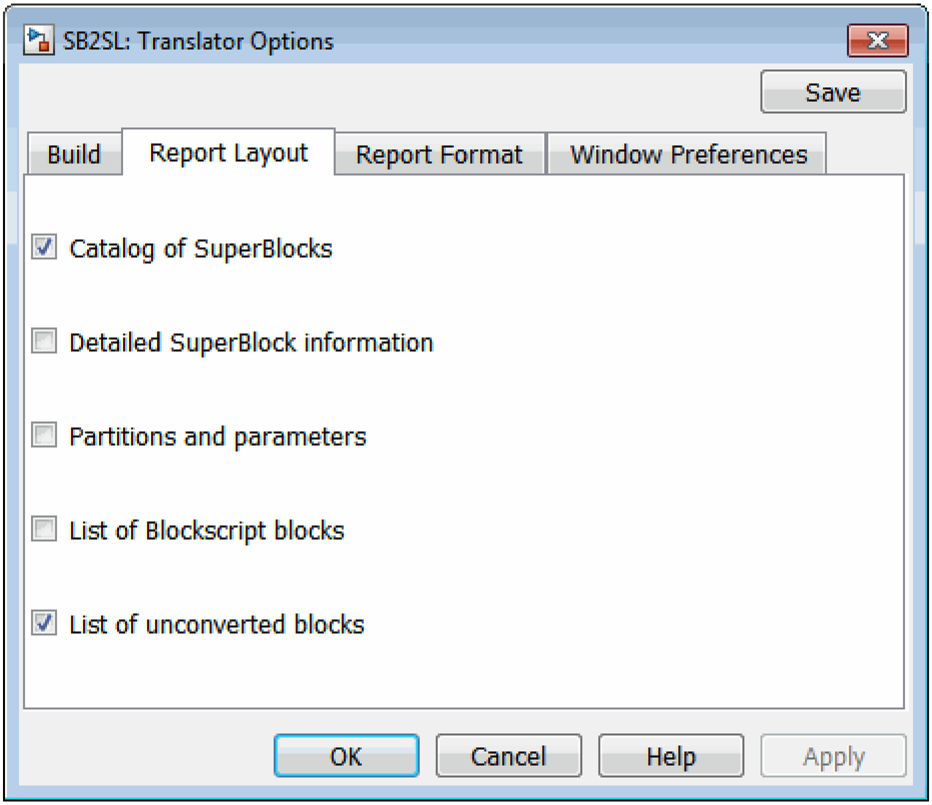
Option	Description
Create SuperBlock libraries	Select this check box to create Simulink library files that contain one Subsystem block per library for each SuperBlock. This option creates Simulink library files in the current directory. SB2SL creates library links from the top-level model and subsequently nested library links. Select this option if you want to use a component-based modeling approach in the Simulink environment. **Note** If you have a library from a previous conversion, SB2SL will use that library. If you want to convert a model that reuses names from an earlier model conversion, you should convert the new model into an empty directory. Converting this model into the same directory as the earlier conversion might cause unexpected links.
Use Simulink native blocks	Select this check box to convert using native Simulink blocks. This option allows the Simulink environment to provide additional optimization and configuration ability in simulation and code generation. Alternately, when you select this check box, the dialog selects both of the following options by default: • `If blocks for Condition block with mode No Default or With Default` Select this check box to convert the Condition block using native Simulink if-else blocks and action subsystems for the Condition block **Mode** parameter set to `With Default` and `No Default`. • `Logic blocks` Select this check box to convert using native Simulink logic blocks.

Option	Description
Put IDs in annotations	Select this check box to insert the block ID into the annotation of subsystem blocks instead of the block name (the annotation parameter name is `AttributesFormatString`). The ID is still visible just below the block name. This option does not affect the block ID of SuperBlocks; they always have the block ID in the subsystem annotation to help componentization. **Note** If you want to insert a block ID into the annotation of a model that was converted in a release before SB2SL 2.7.2, use the `sbid2anno` function.
Use round sum block	Select this check box to use a round summing junction instead of a square one in the Simulink model. This change is only visual.

Report Generation Options

You can use report generation options to select the portions of the SystemBuild data you want to include in a build report. To create a build report, select the SB2SL **Build > Report** option. This option saves the build report in the current directory in a file named `*xmdfilename*.html`, for example, `sbpend.html`.

To specify the data portion options, click the **Report Layout** tab in the Translator Options window.



☑ Catalog of SuperBlocks

□ Detailed SuperBlock information

□ Partitions and parameters

□ List of Blockscript blocks

☑ List of unconverted blocks

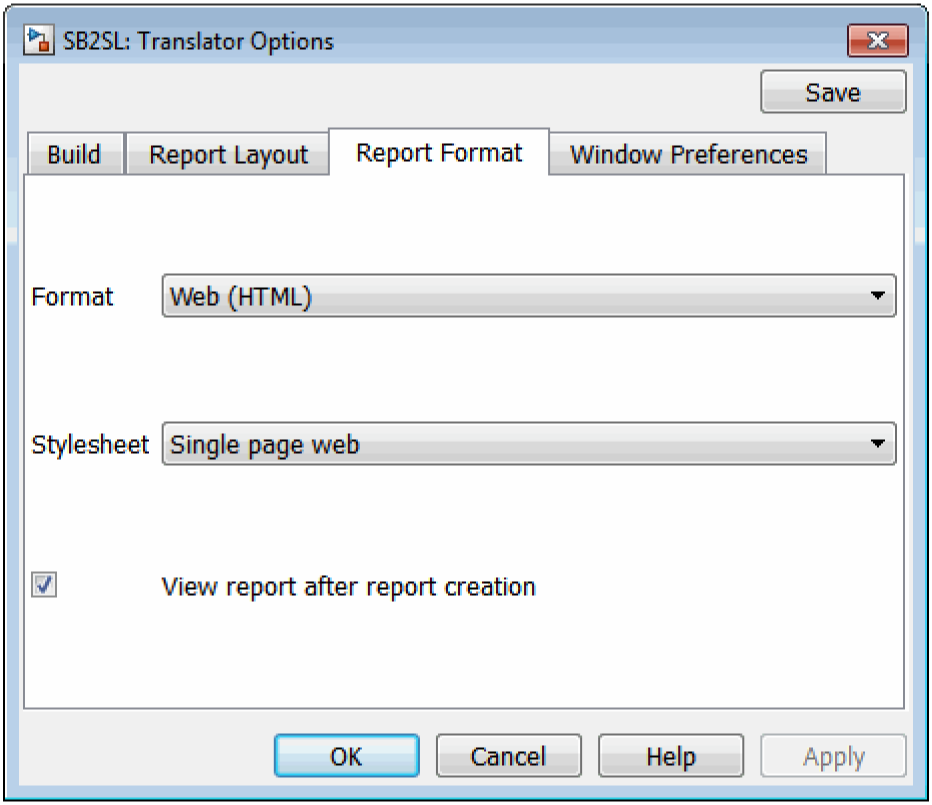
The following report options are available for inclusion in the build report:

Option	Description
Catalog of SuperBlocks	Select this check box to include a list of the SuperBlocks in the model.
Detailed SuperBlock information	Select this check box to include detailed information about the SuperBlocks in the model.
Partitions and parameters	Select this check box to include the partitions and parameters in the model.

Option	Description
List of Blockscript blocks	Select this check box to include a list of the BlockScript blocks in the model.
List of unconverted blocks	Select this check box to include a list of the missing (unconverted) blocks from the model. If all blocks were converted, the report indicates that SB2SL has converted all blocks.

Report Formatting Options

You have the following options for specifying the format of generated reports. Click the **Report Format** tab in the Translator Options window to access these options. Click the **Report Format** tab in the Translator Options window to access these options.



Format: Web (HTML)

Stylesheet: Single page web

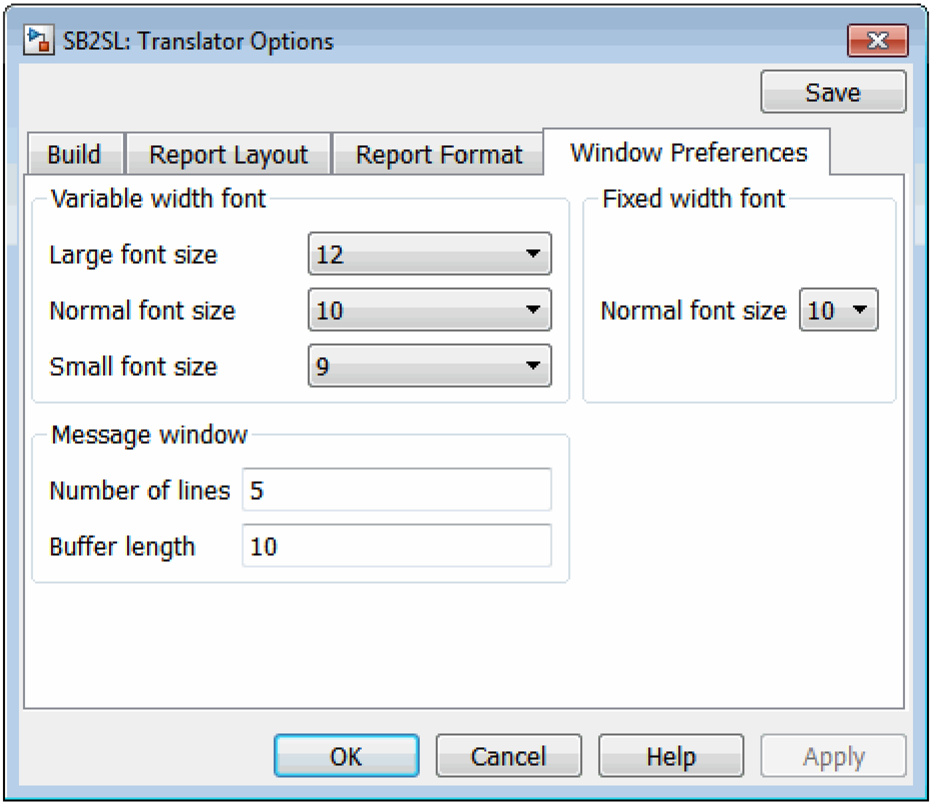
☑ View report after report creation

Option	Description
Format	Select the output format: - `Web (HTML)` - `Rich Text Format 95 (RTF)` - `Rich Text Format 97 (RTF)` - `LaTex (TEX)`

Option	Description
	Selecting `Web (HTML)` enables the **View report after creation** check box.
Stylesheet	The choices for this option depend on the setting of **Format**. • If **Format** is `Web (HTML)`, select `Single page web` or `Multi page web` output • If **Format** is `Rich Text Format 95 (RTF)`, `Rich Text Format 97 (RTF)`, or `LaTex (TEX)`, select `Standard print`, `Simple print`, or `Large type print`.
View report after conversion	Select this check box to display the report after it is created. This check box is enabled only when **Format** is `Web (HTML)`.

Window Preferences

You can use window preferences to customize the look of your SB2SL windows. Click the **Window Preferences** tab in the Translator Options window to access these options.



Variable width font

Large font size 12

Normal font size 10

Small font size 9

Fixed width font

Normal font size 10

Message window

Number of lines 5

Buffer length 10

Under...	Do...
Variable width font	From the drop-down lists, select fonts to change the large, normal, and small font size of the SB2SL window labels.
Fixed width font	From the drop-down list, select the font size for fixed-width displays.
Message window	In **Number of lines**, enter the number of display lines. In **Buffer length**, enter the number of lines you want to keep in the message buffer.

Convert SuperBlocks to Simulink Models

Before converting your SuperBlock to a Simulink model, you can set options for building the model and recording the translation. See “Set Translation Options” on page 1-12 for more information.

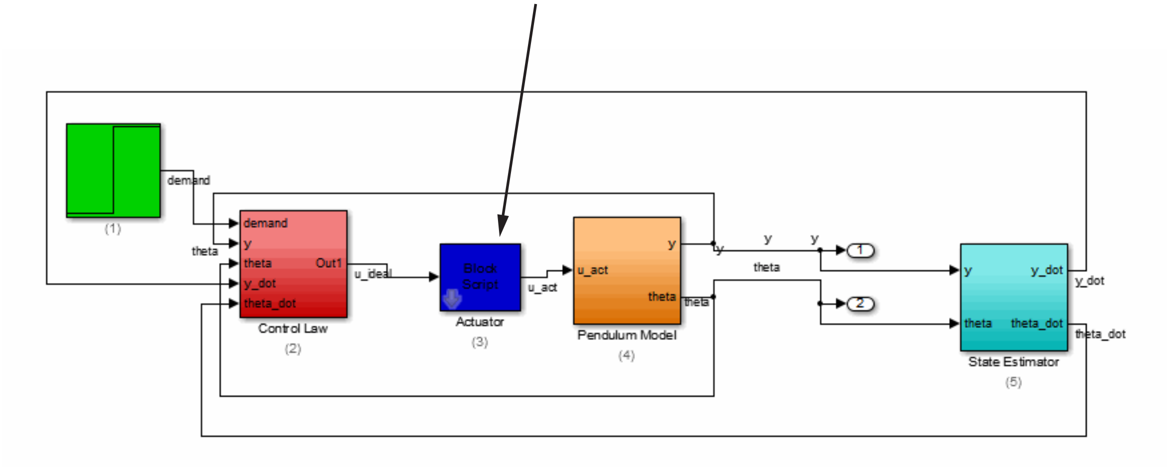
You are ready to convert your model to a Simulink one after you have:

- Selected the top-level SuperBlock and the partition you want to translate
- Set any desired translation options (see “Set Translation Options” on page 1-12)

To begin the translation, click the **Convert** button on the main SB2SL GUI. This begins the translation process and the resulting Simulink model is opened when it is finished. During the translation:

- The progress bar beneath the **Convert** button on the main SB2SL GUI slides toward completion.
- The message window displays actions describing the translation.

Note, there is an S-function in this model.
You must compile this with the Build > Compile command.



Simulink® Model for sbpend.xmd

After you convert your model to a Simulink one, some blocks on the Simulink diagram might be labeled `Unconverted`. See "Blocks Not Converted to Simulink Models" on page 1-39 and "Suggestions for Handling Unconverted Blocks" on page 1-40 for information about unconverted blocks.

Default Conversion Results

SB2SL performs the following during a default conversion:

- Creates a top-level model with nondefault model-level parameter settings
- Converts SystemBuild SuperBlocks to Simulink atomic subsystems

SB2SL creates a top-level model with the following nondefault model-level Configuration Parameters dialog box parameter settings:

Configuration Parameter	Value	Command–Line Parameter	Value
Optimization > Signals and Parameters pane: **Inline parameters**	`On`	`'InlineParams'`	`'on'`
Solver pane: **Type**	`Variable-step`	`'SolverType'`	`'Variable-step'`
Solver pane: **Solver**	`ode45`	`'SolverName'`	`'ode45'`
Diagnostics > Connectivity pane: **Mux blocks used to create bus signals**	`error`	`'StrictBusMsg'`	`'ErrorLevel1'`
Diagnostics > Data Validity pane: **Signal resolution**	`Explicity only`	`'SignalResolution-Control'`	`'UseLocalSettings'`
Model Referencing pane: **Rebuild**	`If any changes in known dependencies detected`	`'UpdateModelReference-Targets'`	`'IfOutOf-Date'`

Noncontinuous SuperBlocks (discrete, procedural, and triggered) correspond most closely to atomic subsystems in the Simulink environment because atomic subsystems are a semantically closer match to SuperBlocks. SB2SL creates atomic subsystems with the following additional Atomic Subsystem block parameter settings to improve readability, componentization potential, and scalability.

Atomic Subsystem Block Parameters	Value	Command Line Parameter	Value
Show port labels	`SignalName`	`'ShowPortLabels'`	`'SignalName'`
Treat as atomic unit	`On`	`'TreatAsAtomicUnit'`	`'on'`
Function packaging	`Function`	`'RTWSystemCode'`	`'Function'`

For continuous SuperBlocks, SB2SL creates atomic subsystems with the following parameters:

Atomic Subsystem Block Parameters	Value	Command Line Parameter	Value
Show port labels	`SignalName`	`'ShowPortLabels'`	`'SignalName'`
Treat as atomic unit	`Off`	`'TreatAsAtomicUnit'`	`'off'`
Function packaging	`Function`	`'RTWSystemCode'`	`'Function'`

SB2SL also enters the block ID string in the Atomic Subsystem block property SB2SL **Block Annotation** tab.

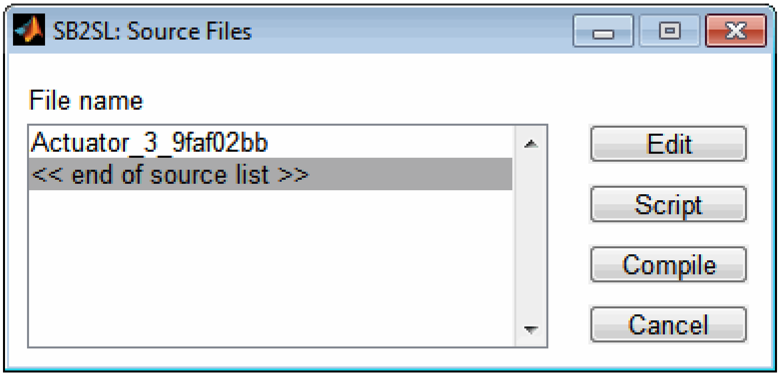
Atomic Subsystem Block Properties	Value	Command-Line Parameter	Value
Block Annotation	Block ID string	`'AttributesFormat-String'`	Block ID string

Alternatively, if the subsystem is atomic and the subsystem contents meet the criteria for model reference (see “Simulink Model Referencing Requirements”), you can convert the subsystem to a referenced model. See the Converting Subsystems to Model Reference example for an example of this.

Note By default, SB2SL does not create Simulink library files with one Subsystem block per library for each SuperBlock. If you want to transition to component-based modeling in the Simulink environment, set the SB2SL main GUI **Build > Option Create SuperBlock libraries** option (see "Translation Build Options" on page 1-13). This option enables your SystemBuild conversion to create Simulink library files with one Subsystem block per library for each SuperBlock. This option can help you transition to component-based modeling in the Simulink environment.

Compile Converted BlockScript

When you convert using SB2SL, SB2SL converts SystemBuild BlockScript blocks into C code and places them into Simulink S-functions automatically. Select **Build > Compile** in the main SB2SL GUI to open the Source Files window. This window lists the S-functions generated by the translated SystemBuild BlockScript blocks.



From the Source Files window:

1 Select the files you want to compile.

2 Click the **Compile** button, and the MATLAB standard `mex` command compiles these C code S-functions.

Save Translated Models and Data

Once the translation is complete, select **File > Save** in the main SB2SL GUI to save either your model or your data:

- Select **File > Save > Model** to save the Simulink model to a file so that it can be reloaded directly from the MATLAB and Simulink environment.
-
- Select **File > Save > Dave** to save the model data read from the SystemBuild file during the translation.

Note You can set the `PreLoadFcn` callback on the Simulink block diagram to reload the model data file the next time the Simulink model is opened. For details on model callbacks, see "Callback Functions".

Generate a Report

You can generate a report recording the details of your translation after you convert a model with SB2SL. There are several report options you might want to set beforehand. See "Report Generation Options" on page 1-16 for information on these options.

To generate a report with the default option settings, select **Build > Report** after converting your model.

Conversion Strategies

In this section...

Componentization

Converting SystemBuild models to Simulink models enables you to simulate sections of the overall model. It also allows you to more easily run existing SystemBuild level tests and confirm the validity of the conversion. You can componentize your converted SystemBuild model using library link and model reference conversion capabilities. If you are creating multiple models during the conversion process, either through multiple conversion invocation or subsequent conversions of atomic subsystems into model references, having a single configuration set object (see "Manage a Configuration Reference") with your desired configurations for all models can simplify conversions.

The benefits of componentization of your SystemBuild model include:

- Ability to convert your SystemBuild model using library links and model reference

  Converting components to be referenced models instead of library links permits simplified testing. Because a referenced model is simply a model that can be simulated, component tests can be brought to the MATLAB or Simulink environment in a straightforward manner.

- Visually cleaning up the resulting model and addressing any issues with unconverted blocks
- Testing the converted models using existing SuperBlock level tests

  The next step is to get the new model to simulate with the same results as the original model. This step might involve changing solver settings and zero-crossing controls for models with continuous states employing variable-step solvers.

SB2SL creates one top-level model per conversion. By default, it configures the converted model to work with the Simulink Model block to allow for the creation of a model reference component in another model or library.

If you do not want to use referenced models but do want to use design components, convert the top-level model into an atomic subsystem:

1. Open a new or existing library.

2. Drag an Atomic Subsystem block into that library.

3. In the Simulink model editor window of the top-level model, select **Edit > Select all**.

4. In the Simulink model editor window of the top-level model, select **Edit > Copy**.

5. In the new or existing library, double-click the Atomic Subsystem block.

   The subsystem is displayed.

6. In the Simulink model editor of the Atomic Subsystem block, select **Edit > Paste**.

   The contents of the top-level model are now in the Atomic Subsystem block.

7. Close the Atomic Subsystem block.

8. Save and close the top-level model and library.

Unconverted SuperBlocks

If the SystemBuild model contains a SuperBlock that SB2SL cannot convert (for example, an external SuperBlock that is referenced by the SystemBuild model), you can still create a link to that unconverted block by doing one of the following:

- Replace the empty subsystem that is in place for the unconverted block with a Simulink Model block to create a link:

  1. Assuming that model A has an unconverted external SuperBlock, find the file that contains the unconverted SuperBlock (for example, file B).

2. Using SB2SL, translate the file that contains the unconverted SuperBlock (for example, file B) to a Simulink model.
3. Leave file B as its own model, model B.
4. Drag a Model block into model A to reference model B.

- Copy a translated model into a Subsystem block in a library:

  1. Assuming that model A has an unconverted external SuperBlock, find the file that contains the unconverted SuperBlock (for example, file B).
  2. Using SB2SL, translate the file that contains the unconverted SuperBlock (for example, file B) to the Simulink model.
  3. Open a new or existing library.
  4. Drag an Atomic Subsystem block into this library.
  5. Copy and paste the contents of model B into the new Atomic Subsystem block and save the library.
  6. Drag a copy of the new Atomic Subsystem block into A.

Improve Signal Line Wiring Results

When SB2SL converts a SystemBuild model into a corresponding Simulink model, it connects the blocks as best as it can. If you are dissatisfied with these results, you can improve the wiring results of the signal lines by:

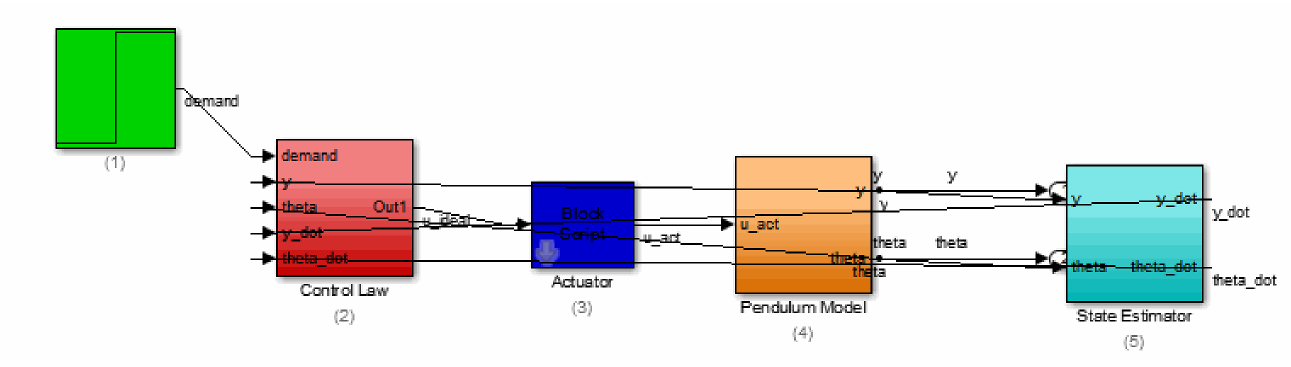
- Manually cleaning up the wiring using the tips in “Wiring Cleanup Tips” on page 1-29
- Converting block and system interfaces to native Simulink modeling styles: vectorization, matrix signals, and buses using the guidelines in “Migrate to a Native Simulink Modeling Style” on page 1-32

Wiring Cleanup Tips

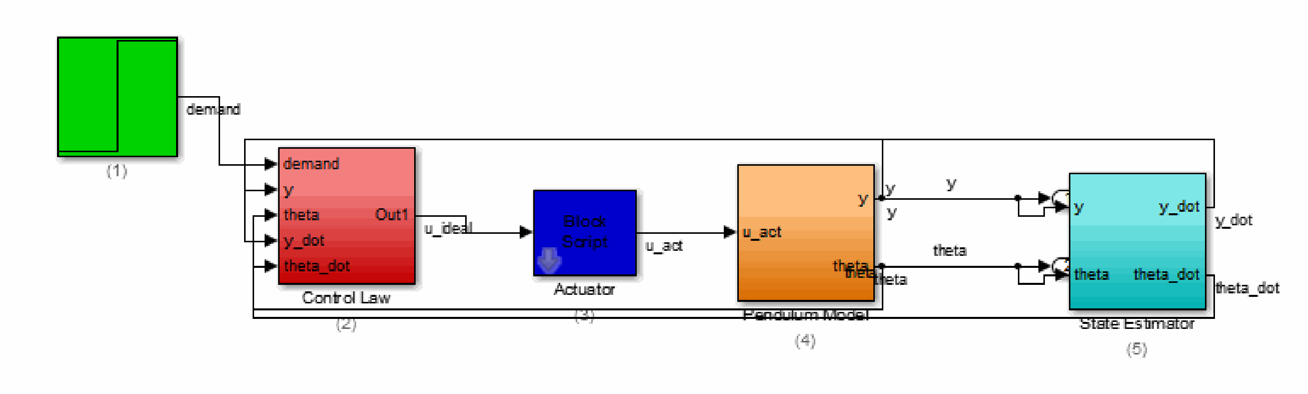
The following guidelines describe how you can visually clean a Simulink model that results from a SystemBuild model translation:

Modeling Pattern	In the Simulink Model Editor...
Multiple lines in parallel going to multiple destinations can cause visually undesired wiring in your model. The use of Mux and Demux blocks can cause these issues.	Perform one or all of the following: • Route a single line to a copy of the Demux block next to the destination. This line enables one wire to be used for the majority of the routing instead of multiple wires. • Rotate and resize blocks and connectors. • Select the Mux or Demux block and use the **Diagram > Rotate & Flip > Flip Block** command to rotate the block 180 degrees to change the wiring visually.
Excessively autorouted lines can cause visually undesired wiring.	Perform one or all of the following: • Turn off autorouted lines in the SB2SL GUI (**File > Preferences**, click **Build** tab, and clear the **Route wires around blocks** check box). • Resize Mux and Demux blocks to line up their corresponding ports. This alignment helps remove diagonal wiring.

The following example shows the appearance of the `sbpend` model when you turn off autorouted lines.



The following example shows the appearance of the sbpend model when you turn on autorouted lines.



Use the **Format** menu commands on the Simulink model editor for basic graphical cleanup of a model, such as block mass alignments and relative alignments.

Silence Unconnected Port Warnings

After conversion, SB2SL might generate a model with unconnected blocks. By default, unconnected blocks cause warnings each time you update the model diagram. To avoid these warnings, use one of the following:

- Before conversion, enable the addition of Terminator and Ground blocks in the SB2SL GUI (**File > Preferences**, click **Build** tab, and select the **Add Terminator and Ground blocks** check box).
- After conversion, use the addterms function to add terminators to the unconnected ports in the model.

If you do not want the unconnected lines to be terminated, and you do not want to display the warnings in your MATLAB Command Window, you can suppress these messages with the following:

1. Before conversion, disable the addition of Terminator and Ground blocks in the SB2SL GUI (**File > Preferences**, click **Build** tab, and clear the **Add Terminator and Ground blocks** check box).

2. In the MATLAB Command Window, type the following:

   ```
   warning('off','Simulink:Engine:InputNotConnected')
   warning('off','Simulink:Engine:OutputNotConnected')
   ```

3. When you want to reenable the warnings, type the following:

   ```
   warning('on','Simulink:Engine:InputNotConnected')
   warning('on','Simulink:Engine:OutputNotConnected')
   ```

These commands are session-wide commands that affect all Simulink models until you exit the MATLAB environment or change the warning settings.

Migrate to a Native Simulink Modeling Style

Once you have a functioning baseline model, consider the following guidelines to take advantage of the Simulink software capabilities. There are no SystemBuild correlations.

- To reduce wiring clutter and simplify interfaces:
  - Use the Simulink single-wire vector and matrix support. The SystemBuild software uses row-major 2-D matrices in some cases, whereas the Simulink software uses column-major arrays for all matrix dimensions. This means that to translate some 2-D calculations, you might need to account for a design transpose from time to time (an actual transpose block is not needed because the entire algorithm is transposed).
  - Create single-wire bundles using the Bus Creator and Bus Selector blocks. The SystemBuild software has a graphical wire bundling capability. However, you use this only for visual presentation; you do not use it to define interfaces or semantic operations. Simulink bus signals are more like real signals; they can:
    - Feed into nonarithmetic operator blocks such as Inport, Outport, Switch, and so on.
    - Have nested hierarchies (buses within buses).

    In addition, you can:

    - Create bus objects in the MATLAB workspace to define and enforce interfaces.

  - Use the bus editor to graphically edit bus objects.

  See “Bus Objects”.

- Instead of the SystemBuild logical concept (positive, negative), use the Simulink Boolean data type (false, true). To create native Simulink models with full efficiency and diagnostic capability, consider moving from the SB2SL logical blocks to the Simulink native logic blocks. In the converted model, consider replacing the LOG sublibrary NOT block with its Simulink equivalent (Logical Operator block with **Operator** parameter set to `NOT`).
- The SystemBuild Algebraic Expression block supports inlined and production code generation, but it does not currently support some of the Simulink Coder™ code generation optimizations. Consider replacing the SystemBuild Algebraic Expression block with the MATLAB Function block to improve production code generation.

Compatibility Between SystemBuild and Simulink Software

In this section...

Introduction

SB2SL performs a block-by-block translation of the SystemBuild model. For SystemBuild blocks for which a clear Simulink equivalent exists, SB2SL places the equivalent built-in Simulink block into the resulting Simulink model. The Gain block is an example in which there is a clear equivalent between SystemBuild and Simulink blocks.

Other SystemBuild blocks have no clear Simulink equivalents. However, through the use of Simulink masking and library features, equivalent implementations of these blocks have been created and are in a Simulink library called `libsb2sl`.

An example of this type of block is the Ramp block in the SystemBuild SNG library. This block supports limits on the output and a relative start time for the ramp. The standard Simulink Ramp block does not inherently support these features. SB2SL translates this block into a masked subsystem that includes a collection of existing Simulink blocks. This masked subsystem behaves the same as the SystemBuild Ramp block.

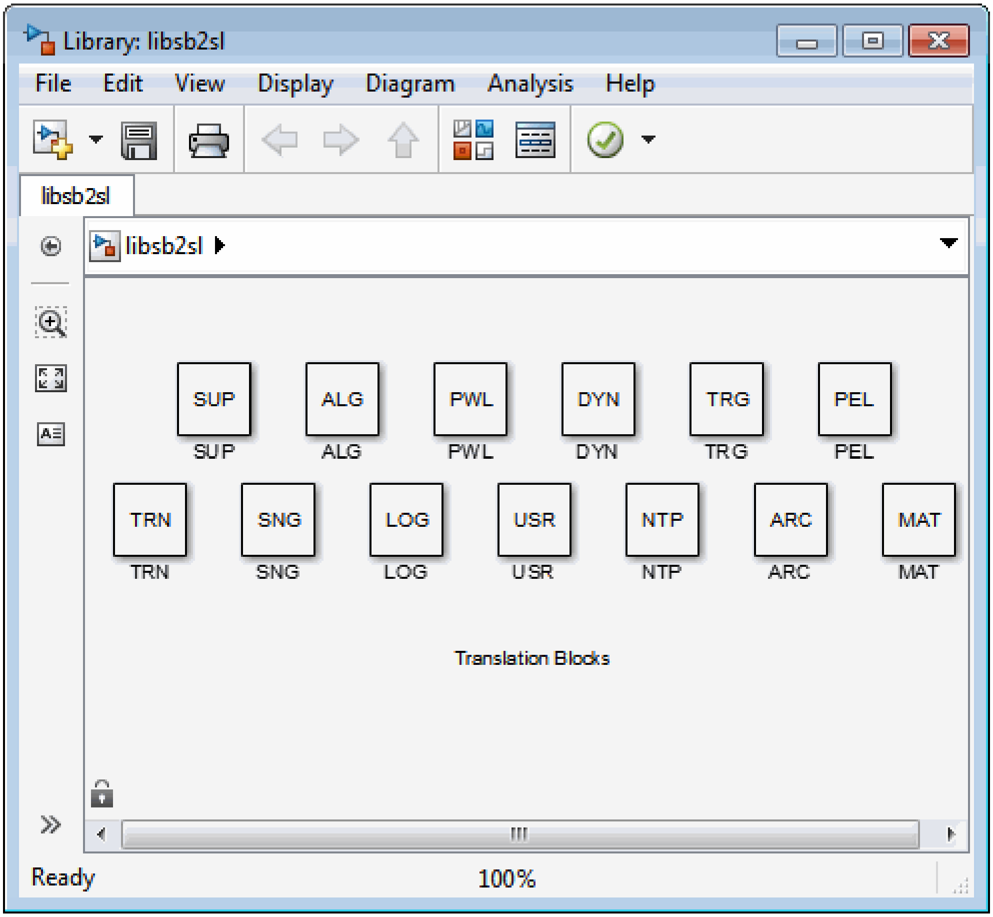
SB2SL Simulink Library

You can find all of the masked blocks generated by SB2SL that are not in any of the other Simulink libraries in the library `libsb2sl`. This library is provided as part of the Simulink environment. (You need to download and install the SB2SL software only if you want to use the SB2SL tool to convert

SystemBuild models.) You can open the library at the MATLAB command line by typing:

```
libsb2sl
```

After SB2SL translation, some blocks that appear in the resulting Simulink model may be from this library.



Open any mask of the Simulink blocks in this library to see the exact implementation of each SystemBuild equivalent used by SB2SL. For example, the Simulink equivalent to the SystemBuild Ramp block is in

```
libsb2sl/SGN/LimRamp
```

For these blocks:

1VarPoly
ConditionBlock
DAxisRotation
Decoder
Encoder
IAxisRotation
LogExpression
ZILogExpression
General
General0

the following equivalents are enabled:

- Code reuse
- Variable-step solvers in referenced models
- Improved performance with accelerated models
- Simulink Normal mode for model reference

Simulink Coder Software and Converted SB2SL Models

You can use the Simulink Coder software to generate code for models you have converted from the SystemBuild to the Simulink environment (using SB2SL). Code is generated for most translated blocks in the model. Code generation is also supported for converted models that contain noninlined BlockScript blocks.

The blocks that do *not* support code generation through the Simulink Coder software are:

- GainScheduler

- Interp Table (Archive library)
- ShiftRegister

Referenced Models in Normal Mode with Converted SB2SL Models

You can use converted SB2SL models in referenced models and execute those models in Simulink Normal mode. Normal mode is one of two modes in which Simulink software can execute a referenced model. For further details, see “Simulation Modes for Referenced Models”.

Limitations

In this section...

Unsupported Conversions

No translator can completely convert an *optimally* designed SystemBuild model into an optimally designed Simulink model. There are subtle differences in the way that the two models work that prevent faithful translation of all capabilities. However, this tool does convert basic blocks and hierarchy from one tool to the other in a form that can be simulated. The following are limitations of SB2SL:

- Does not translate binary SystemBuild files.
- Only double data types are supported. Other data types are not supported.
- Write to Variable and Read from Variable blocks do not support the element- or bit-addressing option.
- The SystemBuild simulation parameter `cdelay` is not supported.
- The timing of triggered subsystems with the "as soon as finished" output posting requirement differs from the SystemBuild implementation:
  - SystemBuild updates the outputs at the beginning of the next minor numerical integration step.
  - In the Simulink environment, the outputs are available immediately.
- Simulink models obtained from SB2SL conversions of SuperBlocks containing any triggered SuperBlocks with both of the following attributes will not run:
  - The output posting is selected as "at timing requirement."
  - The triggered SuperBlock is nested within another triggered SuperBlock.

- BlockScripts with scalar parameters cannot generate embedded real-time (ERT) target code.
- BlockScripts cannot be used in referenced models.

For more information on the correspondence between SystemBuild and Simulink blocks, see “MATRIXx Feature to MathWorks Feature Mapping” on page A-2.

File Format Support

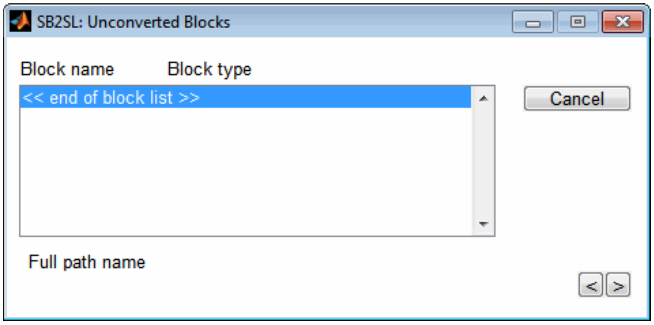
SB2SL cannot read SystemBuild files stored in the binary file format.

Blocks Not Converted to Simulink Models

SB2SL converts the following SystemBuild blocks into empty placeholder blocks in Simulink models. You may want to replace these with various Simulink blocks you have developed that are equivalent.

- State transition diagrams
- MathScript blocks
- UserCode blocks (see “Suggestions for Handling UserCode Blocks” on page 1-41 for a workaround)
- Interactive Animation blocks
- Any new blocks introduced since SystemBuild Version 6.0

These blocks are converted into blocks labeled `Unconverted`. To view a complete listing of the blocks not translated, select **Build > Unconverted Blocks** from the SB2SL GUI.



Suggestions for Handling Unconverted Blocks

You can implement all of the SystemBuild operations represented by the unconverted blocks on your Simulink diagram using the MATLAB software, Simulink, and, in some cases, other related products. Here are some suggestions for replacing the unconverted blocks with ones usable for simulation with the Simulink environment:

- You can replace MathScript blocks with Interpreted MATLAB Function or MATLAB Function blocks. MATLAB Coder is used to run MATLAB files. You must write your own files to execute the equivalent MathScript.
- You can replace UserCode blocks with S-Function blocks. These are blocks you can use to run C code or Fortran.
- You can use a variety of blocks in the Simulink Sinks library to replace Interactive Animation blocks, depending on the function of that block. For a greater variety of animated blocks, see the Gauges Blockset™ documentation.
- You can replace state transition diagrams with Stateflow® charts. This requires you to purchase Stateflow in addition to MATLAB and Simulink software.

To replace an unconverted block in your Simulink model with the correct Simulink block:

1 Open an unconverted block in the Simulink model by double-clicking it.

This opens a window listing the SystemBuild component that caused the unconverted block to be created.

2 Either:

- Delete the unconverted block and copy an appropriate standard Simulink block into its place.
- Use the Simulink function `replace_block` to replace the unconverted block in the Simulink model.

Suggestions for Handling UserCode Blocks

SB2SL does not directly convert UserCode blocks to Simulink blocks. As a workaround, you can manually convert the UserCode block contents to equivalent Simulink S-function methods and SimStruct functions.

You should have the following background:

- Good C programming skills
- Good understanding of SystemBuild UserCode blocks

For more information about S-functions:

See...	For...
"How S-Functions Work"	General information on how S-functions work.
"About Writing C S-Functions"	General information on writing C S-functions.
"Templates for C S-Functions"	Descriptions of the available C MEX S-function templates, the minimum required S-function methods, and the S-function data types.
"DWork Vector Basics"	Description of DWork vectors that you can use to allocate blocks of memory from within S-functions.

This topic describes how to create custom Simulink S-function files. The action you choose depends on whether or not your UserCode block code is simple. Simple UserCode block code has only `INIT`, `STATE`, `OUTPUT`, and/or `LAST` modes:

- If your UserCode block code is simple, consider using the S-Function Builder block to create an S-function. See "Use the S-Function Builder Block to Convert Simple UserCode Block Code" on page 1-42.
- If your UserCode block is more complex, consider manually converting your code to an S-function. See "Manually Convert More Complex UserCode Block Code" on page 1-43. If a UserCode block has `MONIT`, `EVENT`, and/or `LIN` modes, it is more complex.
- If your UserCode block contains Fortran code, see "Convert UserCode Block Fortran Code" on page 1-44.

Use the S-Function Builder Block to Convert Simple UserCode Block Code

Before you start, see "Build S-Functions Automatically". That topic describes how to use the S-Function Builder block to create an S-function.

The S-Function Builder block supports the following S-function methods:

- `mdlInitializeConditions`
- `mdlInitializeSampleTimes`
- `mdlInitializeSizes`
- `mdlCheckParameters`
- `mdlProcessParameters`
- `mdlDerivatives` (continuous states)
- `mdlUpdate` (discrete states)
- `mdlOutputs`
- `mdlTerminate`

The S-Function Builder block has a GUI that guides you in the generation of an S-function. To use it, copy the argument information code from your simple UserCode block into the S-Function Builder block dialog box.

Manually Convert More Complex UserCode Block Code

To convert UserCode block code using existing C MEX S-function templates:

1. In the SystemBuild model, open the UserCode block to access the code contents.

2. From the available templates, copy the most appropriate C MEX S-function template to your working directory:

   - `sfuntmpl_basic.c`
   - `sfuntmpl_doc.c`

3. Rename your template copy with a unique name. This renamed file is your C MEX S-function file.

4. Open the UserCode block code file and your C MEX S-function file.

5. Copy the contents of the mapping modes in the UserCode block code file to the corresponding S-function method in the C MEX S-function file. Use the following mapping table as a guide. Modify the C code to ensure that it has the correct syntax in the S-function.

UserCode Block Mode	S-Function Method

`INIT`	`mdlInitializeConditions`
	`mdlInitializeSampleTimes`
	`mdlInitializeSizes`
	`mdlCheckParameters`
	`mdlProcessParameters`
	`mdlStart`

`STATE`	`mdlDerivatives` (continuous states) `mdlUpdate` (discrete states)
`OUTPUT`	`mdlOutputs`
`MONIT`	`mdlGetTimeOfNextVarHit`
`EVENT`	`mdlZeroCrossings`

UserCode Block Mode	S-Function Method
`LIN`	`mdlProjection`
`LAST`	`mdlTerminate`

6 Using the following mapping table, reimplement the number of inputs, outputs, and states in the S-function method, `mdlInitializeSizes`. Use the corresponding SimStruct function.

Arguments	S-Function Method
`NU`	`ssSetNumInputPorts`
`NX`	`ssSetNumContStates` (continuous states) `ssSetNumDiscStates` (discrete states)
`NY`	`ssSetNumOutputPorts`

7 Save your file.

Convert UserCode Block Fortran Code

If your UserCode block code contains Fortran code, see “Create Level-2 Fortran S-Functions”. That topic provides guidelines on how you can create an S-function to interact with your Fortran code.

2

Function Reference

sbid2anno

Purpose

Convert block names with ID to traditional names

Syntax

```
sbid2anno('sys')
sbid2anno('sys','ShowIdString','off')
sbid2anno('sys','ShowIdString','on','ReplaceDepth',inf)
```

Description

For the blocks and subsystems in the top level of `sys`, `sbid2anno('sys')` moves the appended SuperBlock IDs from the block names to the block annotation fields. It uses the following guidelines:

- The function assigns canonical, unique, hidden names to blocks with no name before the SuperBlock ID.
- The function assigns canonical, unique, numeric suffixes to blocks with appended SuperBlock IDs that make them unique.
- The function ignores blocks with no appended SuperBlock IDs.

`sbid2anno('sys','ShowIdString','off')` performs the same function as `sbid2anno('sys')`, but does not set the block annotation in the block property, `AttributesFormatString`.

`sbid2anno('sys','ShowIdString','on','ReplaceDepth',inf)` performs the same function as `sbid2anno('sys')`, but does not set the block annotation in the block property, `AttributesFormatString`. This function also replaces all block names with appended SuperBlock IDs in the model, regardless of the number of nested system levels. If the value `ReplaceDepth` is invalid (nonnumeric), this function ignores the value and uses `1`.

Examples

This example assumes a previously translated SuperBlock, FltLevel. It converts all blocks and subsystems with appended SuperBlock IDs at the root level of FltLevel.

```
open_system('FltLevel')
sbid2anno('FltLevel')
```

A

Conversions

MATRIXx Feature to MathWorks Feature Mapping

Simulink can replicate almost all SystemBuild functionality through basic blocks or through the use of additional blocksets or S-function code. This topic contains tables that might be helpful in the translation of SystemBuild models to Simulink models:

In this section...

Corresponding SystemBuild and Simulink Blocks

Transition SystemBuild Access Commands to MATLAB and Simulink

Xmath Module	Xmath Function	MathWorks Function	MathWorks Product	Notes
xms	`copydatastore`	Data Store Read and Data Store Write blocks	Simulink — Signal Routing	Use these blocks to create data stores. To add these blocks, at the command line, enter `add_block`.
xms	`copystd`	NONE	Stateflow	Cannot copy Stateflow states from the command line.
xms	`copysuperblock`	`add_block`	Simulink built-in	

Xmath Module	Xmath Function	MathWorks Function	MathWorks Product	Notes
xms	`createblock`	`add_block`	Simulink built-in	
xms	`createbubble`	NONE	Stateflow	Cannot create Stateflow states from the command line.
xms	`createconnection`	`add_line`	Simulink built-in	
mx_pns	`createmcd`	NONE	NONE	Simulink does not have an equivalent RTF file type.
xms	`creatertf`	Build code with Simulink Coder	Simulink Coder	Simulink does not have an equivalent RTF file type.
xms	`createstd`	NONE	Stateflow	Cannot create Stateflow states from the command line.
xms	`createsuperblock`	`add_block`	Simulink built-in	
xms	`createsuperbubble`	NONE	Stateflow	Cannot create Stateflow states from the command line.
xms	`createtransition`	NONE	Stateflow	Cannot create Stateflow transitions from the command line.

Xmath Module	Xmath Function	MathWorks Function	MathWorks Product	Notes
xms	`createusertype`	NONE	NONE	Use Simulink Fixed Point™ to define new data types.
xms	`deleteblock`	`delete_block`	Simulink built-in	
xms	`deletebubble`	NONE	NONE	In Stateflow, cannot remove states from the command line.
xms	`deletecomponent`	NONE	NONE	
xms	`deleteconnection`	`delete_line`	Simulink built-in	
xms	`deletedatastore`	Data Store Read & Data Store Write blocks	Simulink — Signal Routing	You must use these blocks to create data stores. To remove these blocks, at the command line, enter `delete_block`.
xms	`deletestd`	NONE	NONE	In Stateflow, cannot remove states from the command line.
xms	`deletesuperblock`	`delete_block`	Simulink built-in	
xms	`deletetransition`	NONE	NONE	
xms	`deleteusertype`	NONE	NONE	

Xmath Module	Xmath Function	MathWorks Function	MathWorks Product	Notes
xms	`modifyblock`	`set_param`	Simulink built-in	Use `set_param` command to modify blocks.
xms	`modifybubble`	NONE	Stateflow	
xms	`modifyconnection`	`delete_line`, `add_line`	Simulink built-in	
xms	`modifystd`	NONE	Stateflow	
xms	`modifysuperblock`	`set_param`	Simulink built-in	
xms	`modifytransition`	NONE	Stateflow	
xms	`modifyusertype`	NONE	NONE	
xms	`pictodsn`	NONE	NONE	
xms	`printmodel`	Choose `Print` from the **File** menu in your Simulink Model	Simulink	
xms	`psets`	Simulink takes inputs from MATLAB Workspace	Simulink	
xms	`psets_AddToList`	Simulink takes inputs from MATLAB Workspace	Simulink	
xms	`psets_Load`	Simulink takes inputs from MATLAB Workspace	Simulink	

Xmath Module	Xmath Function	MathWorks Function	MathWorks Product	Notes
xms	psets_Save	Simulink takes inputs from MATLAB Workspace	Simulink	
xms	queryblock	find_system and get_param	Simulink built-in	
xms	queryblockoptions	get_param	Simulink built-in	
xms	querybubble	NONE	Stateflow	
xms	querybubbleoptions	NONE	Stateflow	
xms	querycatalog	NONE	Simulink	Simulink does not have an equivalent catalog browser.
xms	queryconnection	find_system and get_param	Simulink built-in	Use the findall option.
xms	queryconnection-options	get_param	Simulink built-in	
xms	querystd	NONE	Stateflow	
xms	querystdoptions	NONE	Stateflow	
xms	querysuperblock	find_system and get_param	Simulink built-in	
xms	querysuperblock-options	get_param	Simulink built-in	
xms	querytransition	NONE	Stateflow	
xms	querytransitionsoptions	NONE	Stateflow	
xms	read_rawfile	NONE	NONE	
xms	realsim_autoplot	NONE	NONE	

Xmath Module	Xmath Function	MathWorks Function	MathWorks Product	Notes
xms	renamedatastore	Data Store Read and Data Store Write blocks		You must use these blocks to create data stores. To modify these blocks, at the command line, enter set_param.
xms	renamestd	NONE	Stateflow	
xms	renamesuperblock	set_param	Simulink built-in	
xms	rve_get	NONE	NONE	
xms	rve_info	NONE	NONE	
xms	rve_put	NONE	NONE	
xms	rve_quit	NONE	NONE	
xms	rve_reset	NONE	NONE	
xms	rve_start	NONE	NONE	
xms	rve_stop	NONE	NONE	
xms	rve_update	NONE	NONE	
xms	setsbdefault	NONE	Simulink	Simulink does not allow you to change the default preferences.
xms	showsbdefault	NONE	Simulink	Simulink does not allow you to change the default preferences.

Xmath Module	Xmath Function	MathWorks Function	MathWorks Product	Notes
xms	`sim`	sim	Simulink built-in	
xms	`simout`	`[sizes,x0,xord] =` Simulink_Model_Name	Simulink	
xms	`sysbldevent`	`set_param(gcb, 'openfcn', 'enter_code_here')`	Simulink built-in	`openfcn` is one of many model callbacks.
xms	`sysbldrelease`	NONE	Simulink	

Transition SystemBuild Blocks to Simulink

A

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
SystemBuild	ABM integration	NONE	NONE	Integration method — comparison list in Help.
Piece-wise Linear	AbsoluteValue Block	Abs	Math Operations	
Trigonometric	Acos Block	Trigonometric Function	Math Operations	
Algebraic	Algebraic-Expression Block	AlgExpression	LIBSB2SL/ALG	
SuperBlocks	Altia Block	NONE	NONE	Use Gauges Blockset.

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
SystemBuild	analyze	NONE	NONE	See `find_system` and `get_param`, and the Model Info block.
Trigonometric	Arccosine	Trigonometric Function	Math Operations	
Trigonometric	Arcsine	Trigonometric Function	Math Operations	
Trigonometric	Arctangent	Trigonometric Function	Math Operations	
Artificial Intelligence	Artificial Intelligence	Fuzzy Logic Controller	Fuzzy Logic Toolbox™	
Trigonometric	Asin	Trigonometric Function	Math Operations	
Trigonometric	Atan2	Trigonometric Function	Math Operations	
Coordinate Transformation	AxisInverse	NONE	NONE	
Coordinate Transformation	AxisRotation	DAxisRotation or IAxisRotation	LIBSB2SL/TRN	

B

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Dynamic	Backlash	Relay	Nonlinear	
Interpolation	BiCubicInterp	BiCubicInterp	LIBSB2SL/NTP	
Interpolation	BiLinearInterp	BiLinearInterp	LIBSB2SL/NTP	
User Programmed	BlockScript	BlockScript or ZIBlockScript	LIBSB2SL/USR	

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Software Constructs	Break	Stateflow	Stateflow	
Archive	BreakPoints	BreakPoints	LIBSB2SL/ARC	

C

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Coordinate Transforms	Cartesian2Polar	Cart2Polar	LIBSB2SL/TRN	
Coordinate Transforms	Cartesian2Spherical	Cart2Sph	LIBSB2SL/TRN	
Dynamic	ComplexPoleZero	CGainDamps-Freqs or DGainDamps-Freqs	LIBSB2SL/DYN	
	ComponentReference	NONE	NONE	
Logical	Condition	ConditionBlock	LIBSB2SL/SUP	
MATRIXx® Equations	Constant	Constant	Sources	
Interpolation	ConstantInterp	ConstantInterp	LIBSB2SL/NTP	
Power Exponential Logarithmic	ConstantPowerU	Constant**u	LIBSB2SL/PEL	
Software Constructs	Continue	Chart	Stateflow	
Trigonometric	CosAsin	CosAsin	LIBSB2SL/TRG	
Trigonometric	CosAtan2	CosAtan2	LIBSB2SL/TRG	
Trigonometric	Cosine	Trigonometric Function	Math Operations	

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Algebraic	CrossProduct	CrossProd	LIBSB2SL/ALG	
Interpolation	CubicSplineInterp	CubicInterp	LIBSB2SL/NTP	

D

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Logical	DataPathSwitch	Switch	LIBSB2SL/LOG	
SuperBlocks	DataStore	DataStoreRW or DataStoreW	LIBSB2SL/SUP	
Piece-wise Linear	DeadBand	Dead Zone	Discontinuities	
Logical	Decoder	Decoder	LIBSB2SL/LOG	
Algebraic	DotProduct	DotProduct	LIBSB2SL/ALG	

E

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Algebraic	Element-by-Element Division	Product	Math Operations	Specify division.
Algebraic	Element-by-Element Product	Product	Math Operations	
SuperBlock	Enabled SuperBlocks	Enabled Subsytem	Ports & Subsystems	
Logical	Encoder	Encoder	LIBSB2SL/LOG	
Power Exponential Logarithmic	Exponential	Math Function	Math Operations	

F

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Artificial Intelligence	Fuzzy Logic	Fuzzy Logic Controller	Fuzzy Logic Toolbox	

G

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Algebraic	Gain	Gain	Math Operations	
Logical	GainScheduler	GainScheduler	LIBSB2SL/LOG	
SystemBuild	Gear's method integration	NONE	NONE	Integration method — comparison list in Help.

H

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Dynamic	Hysteresis	Cbacklash or Dbacklash	LIBSB2SL/DYN	

I

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Software Constructs	IfThenElse	NONE	NONE	Use Stateflow to implement.
Implicit	Implicit blocks	NONE	NONE	
Implicit	ImplicitConstraint	NONE	NONE	
Implicit	ImplicitOutput	NONE	NONE	

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Implicit	ImplicitUserCode	NONE	NONE	
Implicit	ImplicitVariable	NONE	NONE	
Dynamic	Integrator	CIntegrator or Dintegrator	LIBSB2SL/DYN	
Interpolation	Interpolation blocks	Lookup table blocks, Interpolation using Prelookup	Lookup Tables	
User Programmed	UCB	S-functions	User-Defined Functions	UCB uses C or Fortran code. S-functions use C/C++, M, or Fortran code.

J

No SystemBuild blocks begin with J.

K

No SystemBuild blocks begin with K.

L

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Matrix Equations	LeftMultiply	Product	Math Operations	Choose `Matrix(*)` for the **Multiplication** parameter in the Simulink Product block.
Dynamic	LimitedIntegrator	CLimInt or DLimInt	LIBSB2SL/DYN	

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Piece-wise Linear	Limiter	Saturation	Discontinuities	
Archived	Linear Interpolation Table	Interp Table	LIBSB2SL/ARC	
Interpolation	LinearInterp	LinearInterp	LIBSB2SL/NTP	
Power Exponential Logarithmic	Logarithm	Math Functions	Math Operations	
Logical	LogicalExpression	LogExpression or ZILogExpression	LIBSB2SL/LOG	
Logical	LogicalOperator	Logical Operator or NOT	Math Operations or LIBSB2SL/LOG	

M

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
User Programmed	MathScriptBlock	Fcn	User-Defined Functions	
Matrix Equations	MatLeftDivide	Fcn	User-Defined Functions	
Matrix Equations	MatRightDivide	Fcn	User-Defined Functions	
Matrix Equations	MatrixInverse	Fcn	User-Defined Functions	
Matrix Equations	MatrixMultiply	Product	Math Operations	
Matrix Equations	MatrixTranspose	Fcn	User-Defined Functions	
Interpolation	MultilinearInterp	MultilinearInterp	LIBSB2SL/NTP	

N

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Signal Generator	NormalRandom	CNormalRandom or DNormalRandom	LIBSB2SL/SNG	
Dynamic	Nth Order Integrator	Integrator	Continuous	
Dynamic	NumDen	CNumDenCoeffs or DNumDenCoeffs	LIBSB2SL/DYN	

O

No SystemBuild blocks begin with O.

P

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Dynamic	PIDController	CPIDControlLaw or DPIDControlLaw	LIBSB2SL/DYN	
Coordinate Transformation	Polar2Cartesian	Polar2Cart	LIBSB2SL/TRN	
Dynamic	PoleZero	CGainZerosPoles or DGainZerosPoles	LIBSB2SL/DYN	
Algebraic	Polynomial	1VarPoly	LIBSB2SL/ALG	
Piece-wise linear	Preload	Preload	LIBSB2SL/PWL	
Signal Generator	PulseTrain	Pulse Train	LIBSB2SL/SNG	

Q

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Piece-wise Linear	Quantization	Quantization	LIBSB2SL/PWL	

R

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Signal Generator	Ramp	LimRamp	LIBSB2SL/SNG	
SuperBlocks	ReadVariable	From Workspace	Sources	The ReadVariable and From Workspace blocks are different concepts, but have the same main functionality.
Logical	RelationalOperator	RelationalOperator, NEQV, or EQV	Math Operations or LIBSB2SL/LOG	
Dynamic	Reset Integrator	CResetIntegrator or DResetIntegrator	LIBSB2SL/DYN	
Matrix Equations	RightMultiply	Fcn	S-Functions and Lookup	
SystemBuild	Runge-Kutta integration	NONE	NONE	Integration method — comparison list in Help.

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
SystemBuild	variable-step Adams-Moulton integration	NONE	NONE	Integration method — comparison list in Help.
SystemBuild	variable-step Kutta-Merson integration	NONE	NONE	Integration method — comparison list in Help.

S

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Piece-wise Linear	Saturation	Saturation	Discontinuities	
Matrix Equations	ScalarGain	Gain	Math Operations	Gain can be used as both a scalar gain and an element gain.
Software Construct	Sequencer Bar	NONE	Simulink	Use the block priorities to force order of operation.
Algebraic	ShiftRegister (Type Conversion)	ShiftRegister	LIBSB2SL/LOG	
Signal Generator	Signal Generator Palette	Sources Library	Sources	
Power Exponential Logarithmic	SignedSquareRoot	SignedSqrt	LIBSB2SL/PEL	
Trigonometric	SinAtan2	SinAtan2	LIBSB2SL/TRG	

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Trigonometric	Sine	Trigonometric Function	Math Operations	
Signal Generator	SinWave	SinusoidGen	LIBSB2SL/SNG	
Software Construct	Software Construct Palette	NONE	Stateflow	Use Stateflow to get these software constructs into your Simulink model.
Coordinate Transforms	Spherical2Cartesian	Sph2Cart	LIBSB2SL/TRN	
Dynamic	SpringMassDamper	CSpringMassDamper or DSpringMassDamper	LIBSB2SL/DYN	
Power Exponential Logarithmic	SquareRoot	Math Function	Math Operations	
Signal Generator	SquareWave	SquareWave	LIBSB2SL/SNG	
Dynamic	StateSpace	CStateSpace or DStateSpace	LIBSB2SL/DYN	
Superblocks	STD	Stateflow	Stateflow	
Signal Generator	Step	StepFcn	LIBSB2SL/SNG	
Logical	Stop Simulation	Stop Simulation	Sinks	
Algebraic	Summer	Sum	Math Operations	
Superblocks	SuperBlock	Subsystem	Ports & Subsystems	
Superblocks	SuperBlocks Palete	Signal Routing library	Signal Routing	

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Superblocks	SuperBlock — Enabled	Enable Subsystem	Ports & Subsystems	Place an Enable block in a subsystem to make an enabled subsystem.
Superblocks	SuperBlock — Triggered	Trigger	Ports & Subsystems	Place a Trigger block in a subsystem to make a triggered subsystem.

T

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
SuperBlocks	Text	NONE	NONE	In Simulink, double click the white space to create text.
Coordinate Transform	Three-Axis Inverse Rotation	NONE	NONE	
Coordinate Transform	Three-Axis Rotation	NONE	NONE	
Piece-wise Linear	Threshold (DeadBand)	Dead Zone	Discontinuities	
Dynamic	TimeDelay	DTimeDelay	LIBSB2SL/DYN	
Dynamic	Transport Lag	Unit Delay or Transport/Variable Transport Delay	Discrete, Continuous	
Algebraic	TypeConversion	Type Conversion	S-functions and Lookup Tables	

U

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Power Exponential Logarithmic	UPowerConstant	u**Constant	LIBSB2SL/PEL	
User Programmed	UCB	S-Functions	S-Functions and Lookup Tables	UCB uses C or Fortran code. S-function uses C/C++, M, Ada, or Fortran code.
Signal Generator	UniformRandom	CUniformRandom or DUniformRandom	LIBSB2SL/SNG	
System Build	Usertype	NONE	NONE	Simulink Fixed Point allows user data types to be defined.

V

No SystemBuild blocks begin with V.

W

SystemBuild Palette	SystemBuild Block	Simulink Block	Simulink Block Library	Notes
Signal Generator	Waveform	GenWaveform	LIBSB2SL/SNG	
Software Constructs	While	NONE	Stateflow	
SuperBlocks	WriteVariable	Goto/From blocks	Signal Routing	

X

No SystemBuild blocks begin with X.

Y

No functions begin with Y.

Z

No SystemBuild blocks begin with Z.

Transition Xmath to MATLAB

Transition Xmath Functions to MATLAB

A

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`abcd`	`tf2ss`	MATLAB	
Intrinsic	`abort`	`dbquit`	MATLAB built-in	
Intrinsic	`abs`	`abs`	MATLAB built-in	
Intrinsic	`acos`	`acos`	MATLAB built-in	Defined from -1 to 1 in Xmath; MATLAB returns complex value.
Intrinsic	`acosh`	`acosh`	MATLAB built-in	
Signal Analysis Module	`Adconversion`	NONE	NONE	`c2d` in Control System Toolbox™ converts continuous time transfer functions to discrete time transfer functions.

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Control Design Module	afeedback	NONE	NONE	
Xmath	aiginterp	NONE	NONE	
Xmath	alias	NONE	NONE	
Intrinsic	all	all	MATLAB built-in	
Control Design Module	append	append	Control System Toolbox	See also the daug function in Robust Control Toolbox™.
Xmath	argn	nargin, varargin, varargout	MATLAB built-in	
MATRIXxD	arma	ar	System Identification Toolbox™	
MATRIXxD	armax	armax	System Identification Toolbox	
MATRIXxD	arma2ss	NONE	NONE	No arma objects in MATLAB.
Intrinsic	ascii	ascii	MATLAB built-in	
Intrinsic	asin	asin	MATLAB built-in	Defined from -1 to 1 in Xmath; MATLAB returns complex value.
Intrinsic	asinh	asinh	MATLAB built-in	
Intrinsic	atan	atan	MATLAB built-in	
Intrinsic	atanh	atanh	MATLAB built-in	
Intrinsic	atan2	atan2	MATLAB built-in	
SystemBuild	autocode	rtwbuild	Simulink Coder	

B

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Signal Analysis Module	`backdiff`	`c2d`	Control System Toolbox	
Model Reduction Module	`balance`	`ssbal`	Control System Toolbox	
Model Reduction Module	`balmoore`	`balreal`	Control System Toolbox	
Signal Analysis Module	`bandpass`	`fdatool`	Signal Processing Toolbox™	Use MATLAB GUI for general filter design and select `bandpass`.
Signal Analysis Module	`bandstop`	`fdatool`	Signal Processing Toolbox	Use MATLAB GUI for general filter design and select `bandstop`.
Intrinsic	`beep`	`beep`	MATLAB	MATLAB `beep` does not display a message.
MATRIXxD	`bj`	`bj`	System Identification Toolbox	
Xμ Module	`blkbal`	`balance`	MATLAB	
Xμ Module	`blknorm`	`norm`	MATLAB	
Control Design Module	`bode`	`bode`	Control System Toolbox	
MATRIXxD	`bpm2inn`	`ssdata`	System Identification Toolbox	
Model Reduction Module	`bst`	`bstmr`	Robust Control Toolbox	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`build`	Simulink	Simulink	
Signal Analysis Module	`buttconstr`	`butter`	Signal Processing Toolbox	
Signal Analysis Module	`butterworth`	`butter`	Signal Processing Toolbox	

C

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Signal Analysis Module	`cancel`	NONE	NONE	
MATRIXxD	`canform`	`canform`	System Identification Toolbox	
SystemBuild	`catalog` (main)	NONE	Simulink	Simulink does not have an equivalent catalog browser.
Signal Analysis Module	`ccepstrum`	`cceps`	MATLAB built-in	
Intrinsic	`char`	`char`	MATLAB built-in	
Signal Analysis Module	`chebconstr`	`cheb1ord andcheby1` or `cheb2ord` and `cheby2`	Signal Processing Toolbox	Use `chebxord` to get the lowest filter order and natural frequency, and then use `chebyX` to generate a filter.
Signal Analysis Module	`chebyshev`	`cheby1`, `cheby2`	Signal Processing Toolbox	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`check`	`exist`	MATLAB built-in	
Basic	chop	`a = [.1,.5,.9]; a(abs(a)<tol) = 0`		No built-in function. Use example and enter tolerance (tol).
Signal Analysis Module	`circonv`	`cconv`	Signal Processing Toolbox	
Signal Analysis Module	`circorr`	`cconv`	Signal Processing Toolbox	
Intrinsic	`clock`	`clock`, or `tic` and `toc`	MATLAB built-in	Use `clock` to return time and date. Use `tic` and `toc` to calculate elapsed time.
Robust Control Module	`clsys`	`des2ss`	Robust Control Toolbox	
Signal Analysis Module	`coherence`	`cohere`	Signal Processing Toolbox	
Basic	`colorind`	`cdata` — variable input to `fcn`	MATLAB Variable	
Signal Analysis Module	`combinePF`	`residue`	MATLAB	Use three input arguments form.
Xmath	Command (*.`mfc`)	Function MATLAB file (*.`m`)	MATLAB	Function MATLAB files create their own workspace.
Intrinsic	`comment`	`%`	MATLAB built-in	Everything after `%` is treated as a comment.

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`commentof`	NONE	NONE	You cannot place a comment on a variable or a workspace in MATLAB.
Intrinsic	`#`	`%`	MATLAB built-in	Everything after `%` is treated as a comment.
Intrinsic	Complex number — *jay*	`i` or `j`	MATLAB built-in	
Intrinsic	`condition`	`cond`	MATLAB	
Intrinsic	`conj`	`conj`	conj	
Control Design Module	`connect`	`connect`	Control System Toolbox	
Xμ Module	`conpdm`	`struct`	MATLAB built-in	Structures are the closest thing to a PDM in MATLAB.
Xμ Module	`consys`	`ss([],[],[], [1 2;3 4])`	Control System Toolbox	
Model Reduction Module	`controllable`	`ctrb` and `ctrbf`	Control System Toolbox	`ctrb` and `ctrbf` return how controllable the system is.
Intrinsic	`colvolve`	`conv`	MATLAB	
Basic	`copyfile`	`!copy` *path_to_orig_file path_to_new_location*	MATLAB built-in	Use `!` (bang) to access the operating system and copy a file.
Signal Analysis Module	`correlate`	`xcorr`	Signal Processing Toolbox	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`cos`	`cos`	MATLAB built-in	
Intrinsic	`cosh`	`cosh`	MATLAB built-in	
Intrinsic	`cosm`	NONE	MATLAB	
Intrinsic	`cot`	`cot`	MATLAB	
Intrinsic	`coth`	`coth`	MATLAB	
Basic	`covariance`	`cov`	MATLAB	
Intrinsic	`csc`	`csc`	MATLAB	
Intrinsic	`csch`	`csch`	MATLAB	
Basic	`csum`	`cumsum`	MATLAB built-in	
MATRIXxD	`ctrcf`	`idss`	System Identification Toolbox	
Xµ Module	`ctrlplot`	`bode`, `nichols`, `nyquist`, `sigma`	Control System Toolbox	

D

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Xµ Module	`daug`	`daug`	Robust Control Toolbox	
Signal Analysis Module	`dbtolin`	NONE	NONE	MATLAB does not have db units. You can create db units using MATLAB objects.
Intrinsic	`debug`	`dbstop`	MATLAB	Debugger is also built into the MATLAB editor.

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Signal Analysis Module	decimate	decimate	Signal Processing Toolbox	
Intrinsic	default	NONE	NONE	
Control Design Module	deffreqrange	NONE	Signal Processing Toolbox	
Intrinsic	define	NONE	NONE	Placing a function in a toolbox directory will have it parsed only when MATLAB starts.
Control Design Module	deftimerange	NONE	NONE	
Basic	delaunay	delaunay	MATLAB	
Signal Analysis Module	delay	NONE	NONE	
Intrinsic	delete	clear, clear all, and so forth	MATLAB built-in	
Xμ Module	delsubstr	strrep	MATLAB built-in	
Basic	demo	demo	MATLAB	
Intrinsic	det	det	MATLAB built-in	
Signal Analysis Module	detrend	detrend	MATLAB	
Signal Analysis Module	dht	gallery	MATLAB	gallery has an option (k=5) that specifies Hartley transform.
Intrinsic	diagonal	diag	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Basic	`directory`	`whos`	MATLAB built-in	Type `x = whos` to get the variables in the current workspace.
Control Design Module	`discretize`	`c2d`	Control System Toolbox	
Intrinsic	`display`	`display`, `disp`	MATLAB built-in	
Signal Analysis Module	`divide`	NONE	NONE	MATLAB does not have polynomial objects. You can use `tf` to create a transfer function object.
Intrinsic	`domain`	NONE	NONE	PDMs do not exist in MATLAB.
Basic	`dsearch`	`dsearch`	MATLAB	

E

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`echo`	`echo`	MATLAB built-in	
Intrinsic	`eig`	`eig`	MATLAB built-in	
Signal Analysis Module	`ellipconstr`	`ellip`	Signal Processing Toolbox	
Signal Analysis Module	`elliptic`	`ellip`	Signal Processing Toolbox	
Intrinsic	`erase`	`clc`, `clear`, `clf`, `cla`	MATLAB	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`error`	`error`	MATLAB built-in	
Control Design Module	`estimator`	`estim`	MATLAB built-in	
MATRIXxD	`etfe`	`etfe`	System Identification Toolbox	
Intrinsic	`execute`	`eval`	MATLAB built-in	
Intrinsic	`exist`	`exist`	MATLAB built-in	
Intrinsic	`exit`	`break`	MATLAB built-in	Entering `exit` in MATLAB will end your session.
Intrinsic	`exp`	`exp`	MATLAB built-in	
Intrinsic	`expm`	`expm`	MATLAB built-in	
Intrinsic	`eye`	`eye`	MATLAB built-in	

F

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`fft`	`fft`, `fft2`	MATLAB	
Signal Analysis Module	`fftpdm`	`fft`		
Signal Analysis Module	`filter`	`filter`	Signal Processing Toolbox	
Intrinsic	`find`	`find`		

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Signal Analysis Module	`firparks`	`remez`	Signal Processing Toolbox	`remez` designs a linear-phase FIR filter using the Parks-McClellan algorithm. The Parks-McClellan algorithm uses the Remez exchange algorithm and Chebyshev approximation theory.
Signal Analysis Module	`firremez`	`remez`	Signal Processing Toolbox	`remez` designs a linear-phase FIR filter using the Parks-McClellan algorithm. The Parks-McClellan algorithm uses the Remez exchange algorithm and Chebyshev approximation theory.
Signal Analysis Module	`firwind`	`fir1`	Signal Processing Toolbox	
Xμ Module	`fitsys`	`for`	Robust Control Toolbox	
Intrinsic	`for`	`for`	MATLAB built-in	
Signal Analysis Module	`forwdiff`	`c2d`	Control System Toolbox	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`fprintf`	`fprintf`	MATLAB built-in	

Basic	`frac`	`a=rand(3);`
		`b=mod(a,1)`
			MATLAB	

Intrinsic	`freq`	`freqresp`	Control System Toolbox	
Signal Analysis Module	`freqcircle`		MATLAB	
Signal Analysis Module	`freqcont`	`freqresp`	Control System Toolbox	

G

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
GUI	`gdmessage`	`questdlg`	MATLAB	
Intrinsic	`get`	`get`	MATLAB built-in	Comparison of MATRIXx of the environment variables is difficult.
Intrinsic	`getchoice`	NONE	MATLAB built-in	Create a GUI using `uicontrol`.
LNX function	`getenviron`	NONE	NONE	Use ! (bang) with system commands.
Basic	`getfile`	`uigetfile`	MATLAB built-in	
Intrinsic	`getline`	`inputdlg`	MATLAB	
Intrinsic	`go`	`dbcont`	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	goto	NONE	NONE	
Signal Analysis Module	gqam	amod	Communications System Toolbox™	qam is one of the options to be passed to amod.
Basic	griddata	griddata	MATLAB	MATLAB contains additional methods.
Signal Analysis Module	gsin	NONE	NONE	Construct matrix or structure with basic commands.
Xμ Module	gstep	NONE	NONE	Construct matrix or structure with basic commands.

H

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Model Reduction Module	hankelsv	hank2sys	Communications System Toolbox	The hank2sys function does not plot a bar graph.
Basic	hardcopy	print	MATLAB	
Intrinsic	help	help	MATLAB built-in	
Intrinsic	hessenberg	hess	MATLAB built-in	
Signal Analysis Module	highpass	NONE	NONE	Design a new filter (fir1, fir2, sptool, and so forth).
Basic	hilbert	hilb	MATLAB	
Robust Control Module	hinfcontr	ncfsyn	Robust Control Toolbox	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Xμ Module	hinfnorm	hinfnorm	Robust Control Toolbox	
Xμ Module	hinfsyn	hinfsyn	Robust Control Toolbox	
Basic	histogram	hist	MATLAB	
Xμ Module	h2norm	h2norm	Robust Control Toolbox	
Xμ Module	h2syn	h2syn	Robust Control Toolbox	

I

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
MATRIXxD	idfreq	freqresp	System Identification Toolbox	
MATRIXxD	idimpulse	impulse	System Identification Toolbox	Use impulse from System Identification Toolbox on an idmodel or iddata object.
MATRIXxD	idsim	sim	System Identification Toolbox	Use sim from System Identification Toolbox on an idmodel or iddata object.
Intrinsic	if	if	MATLAB built-in	
Intrinsic	ifft	ifft	MATLAB	
Intrinsic	imag	imag	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`impinvar`	`impinvar`	Signal Processing Toolbox	
Control Design Module	`impulse`	`impulse`	Control System Toolbox	
Intrinsic	`index`	`findstr`	MATLAB built-in	`findstr` finds all instancse of the string not only the first instance.
Intrinsic	`indexlist`	`cell`	MATLAB built-in	MATLAB does not have indexed lists. Use cell arrays instead.
Intrinsic	`initial`	`initial`	Control System Toolbox	
MATRIXxD	`initmodel`	`pem`	System Identification Toolbox	`InitialState` property of the `pem` function.
MATRIXxD	`initx0`	`pem`	System Identification Toolbox	Value of `'Estimate'` for `InitialState` of the `pem` function.
MATRIXxD	`inn2pe`	`pe`	System Identification Toolbox	
Intrinsic	`int`	`fix`	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Basic	`integrate`	`quad`, `quadl`	MATLAB	MATLAB computes the integral over a given region. There is symbolic integration using `int` from the Symbolic Math Toolbox™.
Signal Analysis Module	`impplot`	`impz`	Signal Processing Toolbox	
Xμ Module	`interp`	`trsp,dtrsp`	Robust Control Toolbox	
Basic	`interpolate`	`interp1`, `interp2`, `interp3`, `interpn`	MATLAB	Xmath function maps to a PDM. Determine what dimension interpolation you want to implement.
Intrinsic	`inv`	`inv`	MATLAB built-in	
Basic	`ipcwc`	`invhilb`	MATLAB	
Intrinsic	`ipcwc`	NONE	NONE	The MATLAB engine or MATLAB MEX-files run C or Fortran code with MATLAB functionality. These functions run independently of the contents of the MATLAB

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
				Command Window.
MATRIXxD	irea	imp2ss	Robust Control Toolbox	
Intrinsic	is	is*	MATLAB built-in	Forms include iscell, iscellstr, ischar, isempty, isequal, isfield, isfinite, isnan, and so forth.

J

No functions begin with J.

K

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	keep	NONE	NONE	
Intrinsic	kronecker	kron	MATLAB	

L

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	length	prod(size(*x*))	MATLAB built-in	
Basic	licensecheckout	NONE	NONE	MATLAB automatically checks out license for the toolbox you are using.

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Basic	licensefile	NONE	NONE	
Basic	licenseinfo	NONE	NONE	
Basic	licenseuser	NONE	NONE	
SystemBuild	lin	linmod	Simulink	
Signal Analysis Module	linearfm	chirp	Signal Processing Toolbox	chirp is similar but not exactly the same as linearfm .
Robust Control Module	linfnorm	norm	Control System Toolbox	
SystemBuild	linksim	NONE	NONE	Before running model (use mex function), compile MEX-file S-function into *.dll .
Signal Analysis Module	lintodb	NONE	Signal Processing Toolbox	
Intrinsic	list	cell	MATLAB built-in	
SystemBuild	listusertype	NONE	NONE	No user-defined variable types in MATLAB.
Intrinsic	load	load	MATLAB built-in	
Intrinsic	lock	NONE	MATLAB	Use persistent and mlock together to get similar results to lock.
Intrinsic	log	log	MATLAB built-in	
Intrinsic	logm	logm	MATLAB	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Signal Analysis Module	lognormal	lognrnd	Statistics	See also logncdf, logninv, lognpdf, and lognstat.
Intrinsic	logspace	logspace	MATLAB	
Intrinsic	log10	log10	MATLAB built-in	
Signal Analysis Module	lowpass	fir1, fir2	Signal Processing Toolbox	See also sptool, fdatool, and so forth.
Optimization Module	lpopt	fmincon	Optimization Toolbox™	See also linprog and lsqlin.
Control Design Module	lqgcomp	reg	Control System Toolbox	
Robust Control Module	lqgltr	ltru	Robust Control Toolbox	ltru is similar but not the same as lqgltr .
Intrinsic	lu	lu	MATLAB built-in	
Control Design Module	lyapunov	lyap, dlyap	Control System Toolbox	

M

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Control Design Module	makecontinuous	d2c		
Intrinsic	makematrix	str2num and str2mat		

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`makepoly`	NONE	NONE	Polynomials in MATLAB are represented as vectors.
Control Design Module	`margin`	`margin`	Control System Toolbox	
Signal Analysis Module	`markoff`	NONE	NONE	
Signal Analysis Module	`matchedpz`	`c2d`	Control System Toolbox	
Intrinsic	`max`	`max`	MATLAB built-in	For matrices, use `max(`*`mymatrix`*`(:))`.
MATRIXxD	`maxlike`	NONE	NONE	
Basic	`mean`	`mean`	MATLAB	For matrices, use `max(`*`mymatrix`*`(:))`.
Xμ Module	`mergeseg`	NONE	NONE	
Intrinsic	`min`	`min`	-	For matrices, use `min(`*`mymatrix`*`(:))`.
Model Reduction Module	`minimal`	NONE	NONE	
Xμ Module	`mkpert`	`dypert`	Robust Control Toolbox	
Xμ Module	`mkphase`	`genphase`	Robust Control Toolbox	
Intrinsic	`mod`	`mod` `rem`	MATLAB	Use `mod` to get modulus. Use `rem` to get remainder.
Model Reduction Module	`modal`	NONE	NONE	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Xμ Module	`modalstate`	NONE	NONE	
Signal Analysis Module	`modcarrier`	`amod`	Communications System Toolbox	
Model Reduction Module	`mreduce`	NONE	NONE	
MATRIXxD	`mtxplt`	`subplot`	MATLAB built-in	
Xμ Module	`mu`	`mu`	Robust Control Toolbox	
Model Reduction Module	`mulhank`	`hankmr`	Robust Control Toolbox	
Xμ Module	`musynfit`	`musynfit`	Robust Control Toolbox	

N

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`names`	`fieldnames`	MATLAB	
Intrinsic	`new`	MATLAB and Functions workspaces	MATLAB	
Intrinsic	`next`	Visual debugging	MATLAB	
Control Design Module	`nichols`	`nichols`	Control System Toolbox	
Intrinsic	`none`	`all`	MATLAB built-in	
Intrinsic	`norm`	`norm`	MATLAB built-in	
Intrinsic	`numden`	`nyquist`	Control System Toolbox	
Control Design Module	`nyquist`	`nyquist`	Control System Toolbox	

O

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Model Reduction Module	`observable`	`obsv`	Control System Toolbox	
SystemBuild	ODASSL Integration	Stiff Solver — ode15s, ode23s, ode23t, ode23tb	Simulink	
MATRIXxD	`oe`	`oe`	System Identification Toolbox	
Intrinsic	`ones`	`ones`	MATLAB built-in	`ones` in MATLAB with only one input makes a square matrix of ones.
Intrinsic	Operators (and PDMs)	Operators (and Structs)	MATLAB built-in	Cannot perform math operations on structures in MATLAB; you must index into them.
Intrinsic	operators (Xmath)	See Operator Variable Conversion mapping	MATLAB built-in	
Model Reduction Module	`ophank`	`ohklmr`	Robust Control Toolbox	
Optimization Module	`optimize`	`lsqnonlin`	Optimization Toolbox	See also `linprog` and `fsolve`.
Signal Analysis Module	`orderfilt`		Signal Processing Toolbox	See `medfilt1`, `medfilt2`, `mean`, `Median`, `Minimum`, `Maximum`.

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Xμ Module	orderstate	NONE	Control System Toolbox	
Basic	orth	orth	MATLAB	
Intrinsic	oscmd	! or dos orunix	MATLAB built-in	
Robust Control Module	osscale	osborne	Robust Control Toolbox	
SystemBuild	Overconstrained DASSL	Stiff Solver — ode15s, ode23s, ode23t, ode23tb	Simulink	Except for constrained DAE problems, integration results from ODASSL and DASSL are the same.

P

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Signal Analysis Module	padcrop	NONE	MATLAB	Index into vector, or concatenate zeros onto the ends.
Signal Analysis Module	partialsum	cumsum	MATLAB	
Intrinsic	pause	pause	MATLAB	
Intrinsic	pdm	struct	MATLAB built-in	
Basic	pdmplot	NONE	MATLAB built-in	
Signal Analysis Module	pdmslice	*b*=*a*(:,:,*n*) %Index into a 3–D array	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
MATRIXxD	pem	rpem	System Identification Toolbox	
Robust Control Module	pfscale	ssv	Robust Control Toolbox	
Xmath	pgui	guide	MATLAB	
Signal Analysis Module	phaseshift	NONE	NONE	
Constant	pi	pi	MATLAB built-in	
Intrinsic	pinv	pinv	MATLAB	
Basic	plot	plot, plot3	MATLAB	
Basic	aliases	Write a script	MATLAB	
Basic	animation	getframe, movie		
Basic	Bar plots	bar, barh	MATLAB	
Basic	erase	clf, cla	MATLAB	
Basic	Graph object	handle	MATLAB	
Basic	Grids and graph lines	grid on, grid off	MATLAB	
Basic	Holding plot options	Use Handle Graphics to set default stye you want	MATLAB	
Basic	HPGL output	print -dhpgl	MATLAB	
Basic	keeping	hold on, hold off	MATLAB	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Basic	labelling	title, xlabel, ylabel, zlabel, legend, xticklable, yticklabel, zticklabel	MATLAB	
Basic	Light source	light	MATLAB	
Basic	Line styles	Use handle graphics to set line properties	MATLAB	
Basic	logarithmic	loglog, semilogx, semilogy	MATLAB	
Basic	Marker styles	Specify CLM in the plot command; plot(1:10,1:10,'k-*')	MATLAB	
Basic	Multiple plots	subplot	MATLAB	
Basic	Positioning objects	Use handle graphics; set(gca, 'position', [.5 2 4 1]);	MATLAB	
Basic	projection	camproj	MATLAB	
Basic	Resetting defaults	Set defaults using Handle Graphics	MATLAB	
Basic	rotate and angle	rotate	MATLAB	
Basic	scale	Use Handle Graphics to set DataAspectRatio	MATLAB	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Basic	Scaling axes	`xlim`, `ylim`, `zlim`	MATLAB	
Basic	Setting colors	Handle Graphics	MATLAB	

Basic	`strip`	`a=rand(8,4);`
		`for i=1:4;`
		`subplot(4,1,i);`
		`plot(a(:,i));`
		`end`
			MATLAB	

Basic	`templates`	Use Handle Graphics to set defaults	MATLAB	
Basic	Text position or style	`text`, `gtext`	MATLAB	
Basic	Tic marks	Tic properties of Handle Graphics	MATLAB	
Basic	`zooming`	`zoom`	MATLAB	
Signal Analysis Module	`pmdemod`	`demod`	Signal Processing Toolbox	
Signal Analysis Module	`poisson`	`poissrnd` or `poisscdf`	Statistics	Functions are very similar to each other.
Basic	Polar Plots	`polar`	MATLAB	
Control Design Module	`poleplace`	`place`	Control System Toolbox	
Control Design Module	`poles`	`pole`	Control System Toolbox	
MATRIXxD	`polezero`	`pzmap`	Control System Toolbox	
MATRIXxD	`poltrend`	`detrend`	System Identification Toolbox	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Basic	polyfit	polyfit	MATLAB	
Intrinsic	polynomial	MATLAB represents polynomials as row vectors	MATLAB	
Basic	polyval	polyval	MATLAB	
Basic	polyvalm	polyvalm	MATLAB	
MATRIXxD	prbs	idinput	System Identification Toolbox	
Intrinsic	print	save	MATLAB built-in	
Intrinsic	product	prod(*x*(:))	MATLAB built-in	

Q

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
GUI	qplot	plot, set, and get	MATLAB	No direct function map. See plot, set, and get.
Optimization Module	qpopt	quadprog	Optimization Toolbox	
Intrinsic	qr	qr	MATLAB built-in	
SystemBuild	QuickSim integration	NONE	NONE	Use ode45 for systems such as this system.
Intrinsic	quit	quit or exit	MATLAB built-in	
Intrinsic	qz	qz	MATLAB built-in	

R

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Signal Analysis Module	`rampinvar`	`c2d`	Control System Toolbox	
Intrinsic	`random`	`rand`, `randn`	MATLAB built-in	
Xµ Module	`randpdm`	NONE	NONE	PDMs do not exist in MATLAB.
Xµ Module	`randpert`	`randel`	Robust Control Toolbox	
Xµ Module	`randsys`	`sysrand`	Robust Control Toolbox	
Intrinsic	`rank`	`rank`	MATLAB	
Signal Analysis Module	`rcepstrum`	`rceps`	Signal Processing Toolbox	
Intrinsic	`rcond`	`rcond`	MATLAB built-in	MATRIXx uses `LINPACK`; MATLAB uses `LAPACK`.
Signal Analysis Module	`rdintegrate`	`quad`	MATLAB	
Intrinsic	`read`	`load`	MATLAB built-in	See also `textread`, `dlmread`, or `fscanf`.
xms	`read_rawfile`	NONE	NONE	
LNX function	`read_sv`	`textread`	MATLAB	
Intrinsic	`real`	`real`	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Model Reduction Module	redschur	bstschmr, bstschml, schmr	Robust Control Toolbox	Use bstmr to put in Schur balanced form, and then one of the reduction fcns in Robust Control Toolbox.
MATRIXxD	reflect	polystab	Signal Processing Toolbox	
Xmath	refnum	License Number	MATLAB	
Control Design Module	regulator	reg	Control System Toolbox	
Intrinsic	remove	NONE	NONE	
Signal Analysis Module	residue	residue	MATLAB	
Intrinsic	return	return	MATLAB built-in	
Control Design Module	riccati	are	Control System Toolbox	are is an obsolete function.
Xμ Module	riccati_eig	ric_eig	Robust Control Toolbox	
Xμ Module	riccati_schur	ric_schr	Robust Control Toolbox	
Signal Analysis Module	ricean	NONE	NONE	
Xμ Module	rifd	NONE	NONE	
Intrinsic	rlinfo	NONE	NONE	
Control Design Module	rlocus	rlocus	Control System Toolbox	
Control Design Module	rms	NONE	NONE	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`rootlocus`	`rlocus`	Control System Toolbox	
Intrinsic	`roots`	`roots`	MATLAB	Different output order from each other.
Intrinsic	`round`	`round`, `ceil`, and `fix`	MATLAB built-in	
Basic	`rref`	`rref`	MATLAB	

S

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`save`	`save`	MATLAB built-in	
Intrinsic	`schur`	`schur`	MATLAB built-in	
Xmath	Script files (*`.ms`)	Script file (*.m)	MATLAB built-in	Script executes as if you typed commands at the MATLAB command prompt. Uses MATLAB workspace.
MATRIXxD	`sdf`	`psd`	Signal Processing Toolbox	
MATRIXxD	`sds`	`n4sid`	System Identification Toolbox	
Xμ Module	`sdtrsp`	`sdtrsp`	Robust Control Toolbox	
Intrinsic	`sec`	`sec`	MATLAB	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`sech`	`sech`	MATLAB	
Xmath	Selecting Graphic Objects	`get`	MATLAB built-in	Use `get` and the objects handle to select the object. Use `set` to modify the object.
Intrinsic	`set autocompile`	Toolbox caching	MATLAB built-in	Set capability in the MATLAB preferences.
Intrinsic	`set break`	`dbstop` in MATLAB file at line number	MATLAB built-in	
Intrinsic	`set buffering`	NONE	MATLAB	Once an output is available, it is displayed.
Intrinsic	`set build`	NONE	Simulink	When you type the model name, Simulink opens.
Intrinsic	`set commanddiary`	`diary`	MATLAB built-in	`diary on` and `diary off` will begin and end a diary session.
Intrinsic	`set debugonerror`	`dbstop` if error	MATLAB built-in	
Intrinsic	`set directory`	`cd` *`path_to_directory`*	MATLAB built-in	`cd` will change directories in MATLAB in the same way that `cd` works in DOS or Linux operating systems.

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`set display`	NONE	MATLAB	Once an output is available, it is displayed, unless a semicolon is placed at the end of the line of code.
Intrinsic	`set distribution`	`rand` — uniform distribution `randn` — normal distribution	MATLAB built-in	
Intrinsic	`set echo`	`echo`	MATLAB built-in	Explicitly turn `echo on` and `echo off` in your MATLAB files.
Intrinsic	`set format`	`format format_option`	MATLAB built-in	Format options — short, long, short e, long e, short g, long g, hex, +, bank, rat, compact, loose.
Intrinsic	`set logarea`	NONE	MATLAB	
Intrinsic	`set path`	`addpath`, `rmpath`, `path`	MATLAB	
Intrinsic	`set partition`	`dbup`, `dbdown`	MATLAB built-in	You can use `db*` functions while debugging.
Intrinsic	`set pause`	NONE	MATLAB	All `pause` commands will be executed.
Intrinsic	`set precMATRIXxon`	`format` `format_option`	MATLAB built-in	Format options — short, long, short e, long e, short g, long g, hex, +, bank, rat, compact, loose.

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`set seed`	`rand` — uniform distribution `randn` — normal distribution	MATLAB built-in	Enter seed in the function call.
Intrinsic	`set sessionDiary or set commandDiary`	`diary`	MATLAB built-in	`diary on` and `diary off` will begin and end a diary session.
Intrinsic	`set timeStamp`	NONE	MATLAB	MATLAB does not save a timestamp with its variables.
Intrinsic	`set uiupdate`	NONE	MATLAB	MATLAB does not have an equivalent option to stop updates.
Intrinsic	`set watch`	NONE	MATLAB	MATLAB does not have an equivalent option to monitor a variable during debugging.
xms	`setsbdefault`	NONE	Simulink	You cannot change the default preferences in Simulink.
Intrinsic	`show`	NONE		
Intrinsic	`show buffering`	NONE	MATLAB	Once an output is available, it is displayed.
Intrinsic	`show commanddiary`	`get(0, 'Diary')`	MATLAB	
Intrinsic	`show commands`	NONE	MATLAB	
Intrinsic	`show directory`	`cd`, `pwd`	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`show display`	NONE	MATLAB	Once an output is available, it is displayed, unless a semicolon is placed at the end of the line of code.
Intrinsic	`show distribution`	NONE	MATLAB built-in	Use different functions for different distributions — `rand` (uniform) and `randn` (normal).
Intrinsic	`show echo`	`get(0, 'Echo')`	MATLAB	
Intrinsic	`show format`	`get(0, 'Format')`	MATLAB	
Intrinsic	`show functions`	NONE	MATLAB	
Intrinsic	`show debug`	`dbstack`	MATLAB	
Intrinsic	`show debugonerror`	NONE	MATLAB	In the MATLAB Editor, from the **Breakpoints** menu, choose `Stop on Errors`.
Intrinsic	`show logarea`	NONE	MATLAB	
Intrinsic	`show partition`	`dbup`, `dbdown`, and `whos`	MATLAB built-in	You can use `db*` functions while debugging.
Intrinsic	`show partitions`	`dbup`, `dbdown`, and `whos`	MATLAB	You can `db*` functions can while debugging.
Intrinsic	`show path`	`path`	MATLAB	
Intrinsic	`show precMATRIXxon`	`get(0, 'Format')`	MATLAB	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	show seed	NONE	MATLAB	Seed is entered in the function calls.
Intrinsic	show sessiondiary	get(0, 'Diary')	MATLAB built-in	
Intrinsic	show variables	who, whos	MATLAB	
Intrinsic	show uiupdate	NONE	Simulink	
xms	showsbdefault	NONE	Simulink	You cannot change the default preferences in Simulink.
Intrinsic	sign	sign	MATLAB built-in	
xms	sim function	sim	MATLAB built-in	
xms	simout	[sizes,x0,xord] = Simulink_Model_Name	Simulink	
Xμ Module	simtransform	ss2ss	Control System Toolbox	See also ss in Control System Toolbox.
Intrinsic	sin	sin	MATLAB built-in	
Robust Control Module	singriccati	aresolv	Robust Control Toolbox	
Intrinsic	sinh	sinh	MATLAB built-in	
Intrinsic	sinm	NONE	MATLAB	
Intrinsic	size	size	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Robust Control Module	smargin	ncfsyn	Robust Control Toolbox	
Basic	sns2sys	ss	Control System Toolbox	
Intrinsic	sort	sort	MATLAB built-in	
Xμ Module	spectrad	vrho	Robust Control Toolbox	
Signal Analysis Module	spectrum	pwelch	Signal Processing Toolbox	
Basic	spline	spline	MATLAB	
Intrinsic	sprintf	sprintf	MATLAB built-in	
Intrinsic	sqrt	sqrt	MATLAB built-in	
Intrinsic	sqrtm	sqrtm	MATLAB	
Robust Control Module	ssv	ssv	Robust Control Toolbox	
MATRIXxD	ss2arma	NONE	NONE	MATLAB does not have arma objects.
Basic	stable	NONE	Control System Toolbox	
Intrinsic	stair	NONE	Control System Toolbox	
Xμ Module	starp	starp	Robust Control Toolbox	
Xmath	Startup file	Startup file	MATLAB	User-created file that executes on startup.

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Control Design Module	`step`	`step`	Control System Toolbox	
Signal Analysis Module	`stepinvar`	`c2d`	Control System Toolbox	
Intrinsic	`stop`	`!kill -9` (Linux only)	MATLAB built-in	Use `!` (bang) to access the operating system.
Intrinsic	`string`	`num2str`, `int2str`, and `mat2str`	MATLAB	
Intrinsic	`stringex`	`strrep`	MATLAB	
Xµ Module	`substr`	`strtok`	MATLAB	
Signal Analysis Module	`subsys`	`c2d`	Control System Toolbox	
SystemBuild	`subsystems`	NONE	Simulink	Simulink subsystems are analogous to SuperBlocks.
Intrinsic	`sum`	`sum`	MATLAB	
Intrinsic	`svd`	`svd`	MATLAB built-in	
Model Reduction Module	`svplot`	NONE	Control System Toolbox	
MATRIXxD	`sweep`	`chirp`	Signal Processing Toolbox	
Signal Analysis Module	`symbolmap`	`base2dec`, `dec2base`	MATLAB	See also `hex2num`, `hex2dec`, `dec2hex`, `bin2dec`, and `dec2bin`.
Xµ Module	`sysic`	`sysic`	Robust Control Toolbox	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	system	ss, tf, or zpk	Control System Toolbox	
SystemBuild	SystemBuild	Simulink	Simulink	
Basic	sys2sns	NONE	Control System Toolbox	

T

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	tan	tan	MATLAB built-in	
Intrinsic	tanh	tanh	MATLAB built-in	
MATRIXxD	taper		Signal Processing Toolbox	Use one of the following: bartlett, blackman, rectwin, chebwin, hamming, hann, kaiser and triang.
Intrinsic	toeplitz	toeplitz	MATLAB	
Intrinsic	trace	trace	MATLAB	
Intrinsic	tril	tril	MATLAB built-in	
SystemBuild	trim	trim	Simulink	
Intrinsic	triu	triu	MATLAB built-in	
Xµ Module	trsp	trsp	Robust Control Toolbox	
Model Reduction Module	truncate	sysls and hankmr	Robust Control Toolbox	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Basic	`tsearch`	`tsearch`	MATLAB	
Signal Analysis Module	`tustin`	`c2d`	Control System Toolbox	

U

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`uiButton`	Button style in `uicontrol`	MATLAB built-in	
Intrinsic	`uiComboBox`	Popup style in `uicontrol`	MATLAB built-in	
Intrinsic	`uiDestroy`	`close`	MATLAB built-in	`close` closes current figure. `close('`*`name`*`')` closes named figure. `close(`*`h`*`)` closes figure with handle *h*.
Intrinsic	`uiExist`	`findobj`	MATLAB built-in	
Intrinsic	`uiFileSelection`	`uigetfile`	MATLAB built-in	
Intrinsic	`uiFlush`	`drawnow`	MATLAB built-in	
Intrinsic	`uiGetValue`	`get`	MATLAB built-in	
Intrinsic	`uiHandle`	`get`	MATLAB built-in	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	uiHide	set	MATLAB built-in	
Intrinsic	uiLabel	Text style in uicontrol	MATLAB	
Intrinsic	uiList	List box style in uicontrol	MATLAB	
Intrinsic	uiMenu	uimenu	MATLAB built-in	
Intrinsic	uiMenuItem	uimenu	MATLAB built-in	
Intrinsic	uiMessage	msgbox	MATLAB	
Intrinsic	uiPanel	Frame style in uicontrol	MATLAB built-in	
Intrinsic	uiPlot	plot and subplot	MATLAB built-in	
Intrinsic	uiPlotArea	plot and subplot	MATLAB	
Intrinsic	uiPlotGet	ginput	MATLAB	
Intrinsic	uiPrompt	inputdlg	MATLAB	
Intrinsic	uiSeparator	NONE	NONE	
Intrinsic	uiSetValue	set	MATLAB built-in	
Intrinsic	uiShow	set	MATLAB built-in	
Intrinsic	uiSlider	slider style in uicontrol	MATLAB built-in	
Intrinsic	uiTab	tabdlg	MATLAB	
Intrinsic	uiTable	NONE	NONE	

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	uiText	Edit style in uicontrol	MATLAB built-in	
Intrinsic	uiTimer	NONE	NONE	
Intrinsic	uiToolCreate	guide	MATLAB	
Intrinsic	uiVarChoice	uicontrol	MATLAB built-in	
Intrinsic	uiVarEdit	set	MATLAB built-in	
Intrinsic	uiVarView	set	MATLAB built-in	
Intrinsic	uiWindow	guide	MATLAB	
Intrinsic	uiWindowDeiconify	NONE	NONE	
Intrinsic	uiWindowIconify	NONE	NONE	
Intrinsic	uiWindowLower	NONE	NONE	
Intrinsic	uiWindowRaise	NONE	NONE	
Intrinsic	unalias	NONE	NONE	
Intrinsic	undefine	NONE	NONE	Remove function from the MATLAB path.
Intrinsic	unlock	NONE	NONE	

V

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
MATRIXxD	val	NONE	NONE	
Basic	variance	var	MATLAB	
Basic	version	var	MATLAB	

W

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`whatis`	`which`	MATLAB built-in	
Intrinsic	`while`	`while`	MATLAB built-in	
Intrinsic	`who`	`who`, `whos`	MATLAB built-in	
Signal Analysis Module	`window`	`fir1`, `fir2`	Signal Processing Toolbox	
Basic	`write_sv`	`dlmwrite`, `csvwrite`	MATLAB	
Model Reduction Module	`wtbalance`	`sfrwtbal`	Robust Control Toolbox	

X

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Signal Analysis Module	`xgraph`	`plot`	MATLAB built-in	
Xmath	`xmathCommand`	NONE	NONE	
Xmath	`XmathLoad`	NONE	NONE	
Xmath	`XmathSave`	NONE	NONE	

Y

No functions begin with Y.

Z

Xmath Module	MATRIXx Function	MathWorks Function	MathWorks Product	Notes
Intrinsic	`zeros`	`zeros`	MATLAB built-in	

Transition Xmath Operators and Variables to MATLAB Environment

Permanent Variables	Xmath	MATLAB	Notes
precision of machine	eps	eps	Both Xmath and MATLAB functions return the same value, 2.2204e-016.
global error status	err	N/A	
largest finite number	huge	realmax	REALMAX is four times larger than huge.
infinity	inf	inf, Inf	
sqrt(-1)	jay	i, j	
Not-A-Number	nan	NaN	
empty object	null	[]	
Pi	pi	pi	
smallest nonzero number	tiny	realmin	Both Xmath and MATLAB functions return the same value, 2.2251e-308.

<table>
<tr><th>Operators — Math and Logical</th><th>Xmath</th><th>MATLAB</th><th>Notes</th></tr>
<tr><td>and</td><td>&</td><td>&</td><td></td></tr>
<tr><td>or</td><td>|</td><td>|</td><td></td></tr>
<tr><td>not</td><td>!</td><td>~</td><td></td></tr>
<tr><td>addition</td><td>+</td><td>+</td><td></td></tr>
<tr><td>subtraction</td><td>-</td><td>-</td><td></td></tr>
<tr><td>multiplication</td><td>*</td><td>*</td><td></td></tr>
<tr><td>right division</td><td>/</td><td>/</td><td></td></tr>
</table>

Operators — Math and Logical	Xmath	MATLAB	Notes
left division	`\`	`\`	
transpose	`'`	`.'`	
elementwise multiplication	`.*`	`.*`	
Kronecker product	`.*.`	`kron`	
less than	`<`	`<`	
greater than	`>`	`>`	
less than or equal	`<=`	`<=`	
equality	`==`	`==`	
not equal	`<>`	`~=`	
power	`**, ^`	`^`	
complex conjugate transpose	`*'`	`'`	
decimal	`.`	`.`	
partition delimiter	`.`	NONE	MATLAB does not have partitions.
raise elements to a power	`.**`	`.^`	
raise elements to a power	`.^`	`.^`	
Hermitian (complex conjugate) transpose	`'*`	NONE	
indexing and precedence	`()`	`()`	Index into matrix or indicate order of operations.
matrix construction and concatenation	`[]`	`[]`	See also `strcat` and `strvcat` for string concatenation.

Operators — Math and Logical	Xmath	MATLAB	Notes
keyword delimeters for functions	`{}`	NONE	
indexing	`:`	`:`	
show output	`?`	NONE	MATLAB displays results if a trailing semicolon is not present.
separator	`,`	`,`	
separate rows, suppress output	`;`	`;`	
ellipsis	`...`		
equal	`=`	`=`	
capture error	`===`	`lasterr`, and `try, catch`	
delineate string	`&quot;`	`&quot;`	
insert double quote in string	`&quot;&quot;`	`&quot;`	MATLAB requires two single quotes (' ') to insert a single quote into a string.
comment	`&#35;`	`%`	

block comments	`&#35;{..}`
	`&#35;`
		NONE	

root Xmath directory	`$XMATH`	`$MATLAB`	
last command recall	`@`	Command History Window	
run command	`@num`	In Command History Window, double-click command.	

Operators — Math and Logical	Xmath	MATLAB	Notes
run commands	`@num:p`	Highlight and run commands from Command History	
execute the most recent command starting with &quot;str&quot;	`@str`	Type `str`, then up arrow	
list all commands starting with &quot;str&quot;	`@str:l`	Command History Window	
list the most recent command starting with &quot;str&quot;	`@str:p`	Command History Window	
list all commands	`@:l`	Command History Window	
execute the most recent command	`@@`	Up arrow, then **Enter**	
list the most recent command	`@@:p`	Up arrow, then **Enter**	

MATRIXx and MathWorks Product Table

MATRIXx	MathWorks	Notes
Control Design Module	Control System Toolbox	
Robust Control Module	Robust Control Toolbox	
Optimization Module	Optimization Toolbox	
Signal Analysis Module	Signal Processing Toolbox	

MATRIXx	MathWorks	Notes
Model Reduction Module	Robust Control Toolbox	Robust Control Toolbox provides model-reduction features.
Xμ Module	Robust Control Toolbox	
Interactive System Identification Module	System Identification Toolbox	System Identification Toolbox contains interactive features.
Interactive Control Design Module	Control System Toolbox	Control System Toolbox contains interactive features.

SystemBuild	Simulink	Fixed-point simulation capability is included with SystemBuild
State Transition Diagram Module	Stateflow	State Transition Diagram Module provides only basic state machine functionality. Stateflow provides state charts and flow charts.
Altia Design	Interface for Simulink available from Altia, Inc.	Linux and PC platforms supported.
	Gauges Blockset	PC only.
Altia Faceplate for SystemBuild	Interface for Simulink available from Altia, Inc.	Linux and PC platforms supported.
	Gauges Blockset	PC only.

SystemBuild	Simulink	Fixed-point simulation capability is included with SystemBuild
HyperBuild Module	Functionality is part of Simulink	HyperBuild Module provides only simulation acceleration features.
RT/Fuzzy Logic Module	Fuzzy Logic Toolbox	
Neural Networks Module	Neural Network Toolbox™	

BetterState with C Code Generator	Stateflow and Simulink Coder	Fixed-point simulation capability is included with SystemBuild
AutoCode C Single Processor	Simulink Coder	Code generation from Stateflow requires Simulink Coder. Embedded Coder® option is required for certain applications.
C Fixed Point Extension	Fixed-Point Toolbox™ and Simulink Fixed Point	
C Multiprocessor Extension	N/A	xPC Target™ supports shared memory I/O for multiprocessing applications.

Index