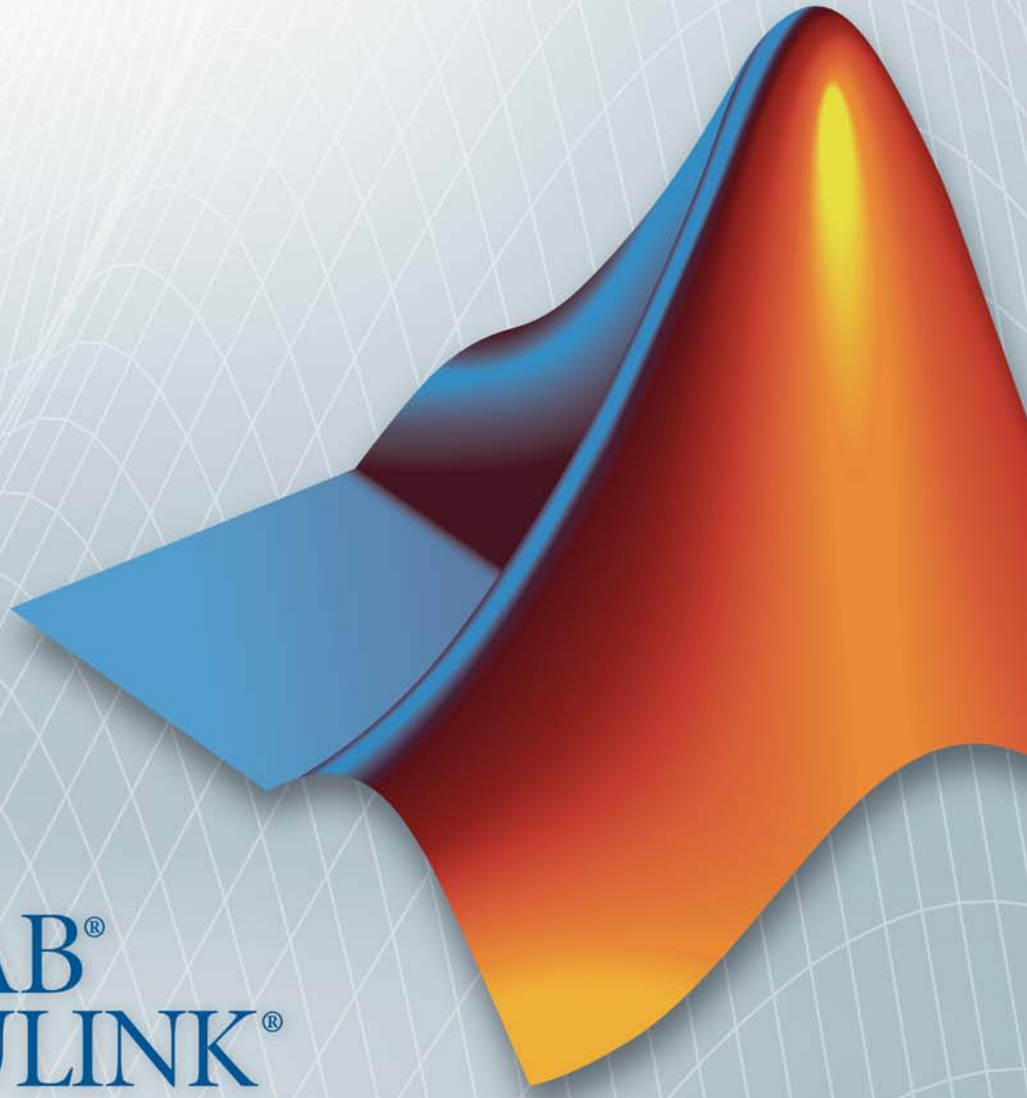
SB2SL 2
User's Guide



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

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

Patents

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

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+)

Contents

1 Converting SystemBuild SuperBlocks to Simulink Models

1

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). 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. However, due to modeling differences between the two environments, you might want to perform further model optimizations to achieve top simulation performance. MathWorks strongly recommends that you validate all models after translation.

Software Requirements

Version 2.7.7 of SB2SL requires MATLAB Version 7.12 and Simulink Version 7.7. 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 UNIX® 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 1-34.

Installation

The SB2SL software is available only through Web download. Follow the installation instructions from the Web download page for the version that you want to download. For further instructions on how to install the UNIX version of the SB2SL software, see the installation documentation.

Using SB2SL

In this section...

Prerequisites

Before translating a SystemBuild model, you must save it in ASCII format (usually with a file extension `.xmd` or `.sbd`).

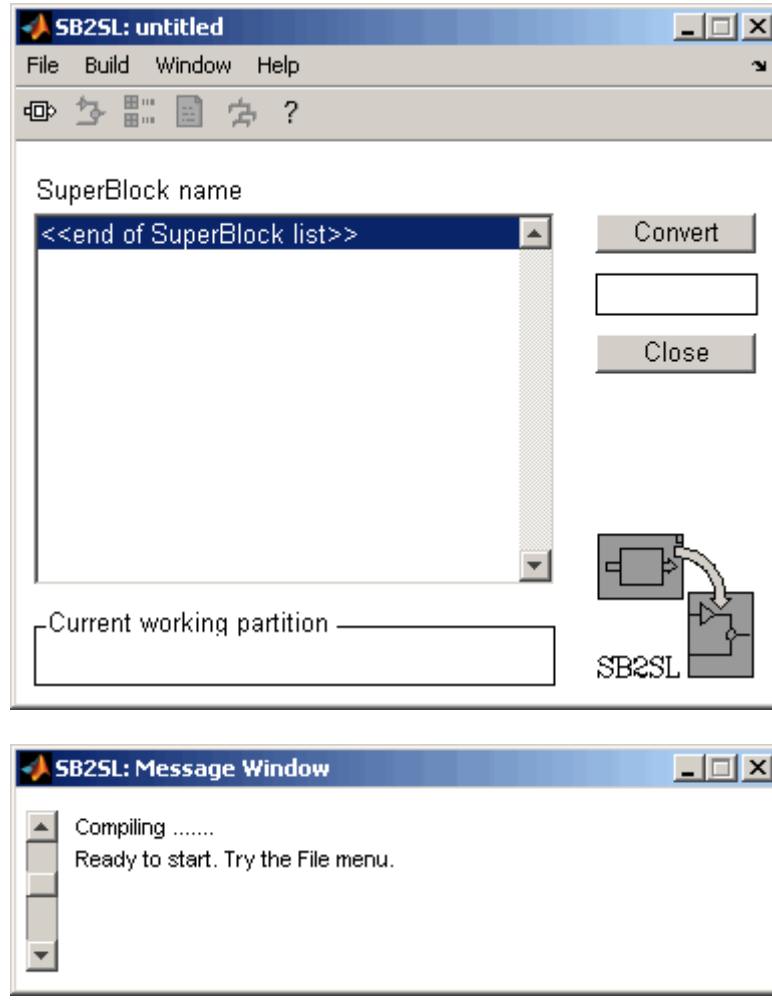
To make possible the transfer of parameterized variables (`%vars`) from SB2SL software, you must also make sure the variables are declared and resident in the Xmath workspace. Then save the SystemBuild model with the **Xmath Variables** option set to **Save All**.

Starting 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.



Main SB2SL GUI and Message Window

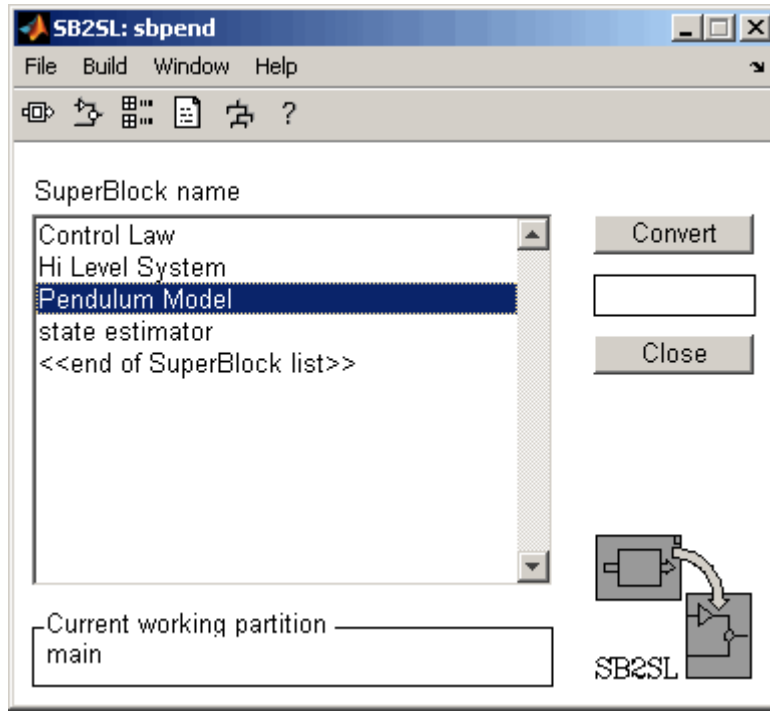
Loading a SystemBuild Model into SB2SL

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.



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.

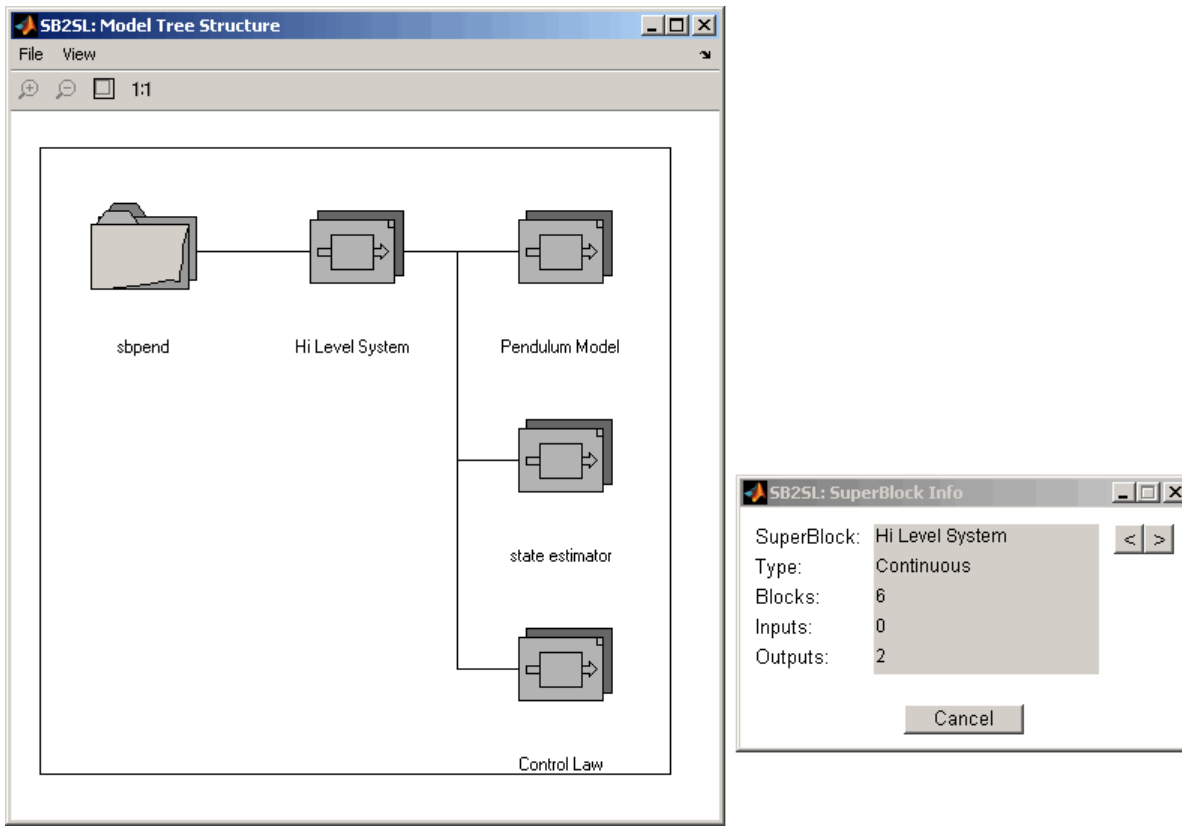
Selecting SystemBuild SuperBlocks

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.

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.).



Model Tree Structure and SuperBlock Information Windows

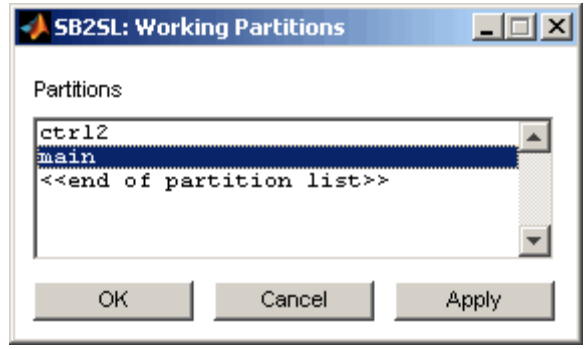
Selecting a SuperBlock Partition for Conversion

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**.

Setting 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 1-9)
- Generating reports from the translation (“Report Generation Options” on page 1-13)
- Converting the reports to various text formats (“Report Formatting Options” on page 1-15)

- Changing GUI font sizes for the translation option dialog boxes (“Window Preferences” on page 1-16)

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

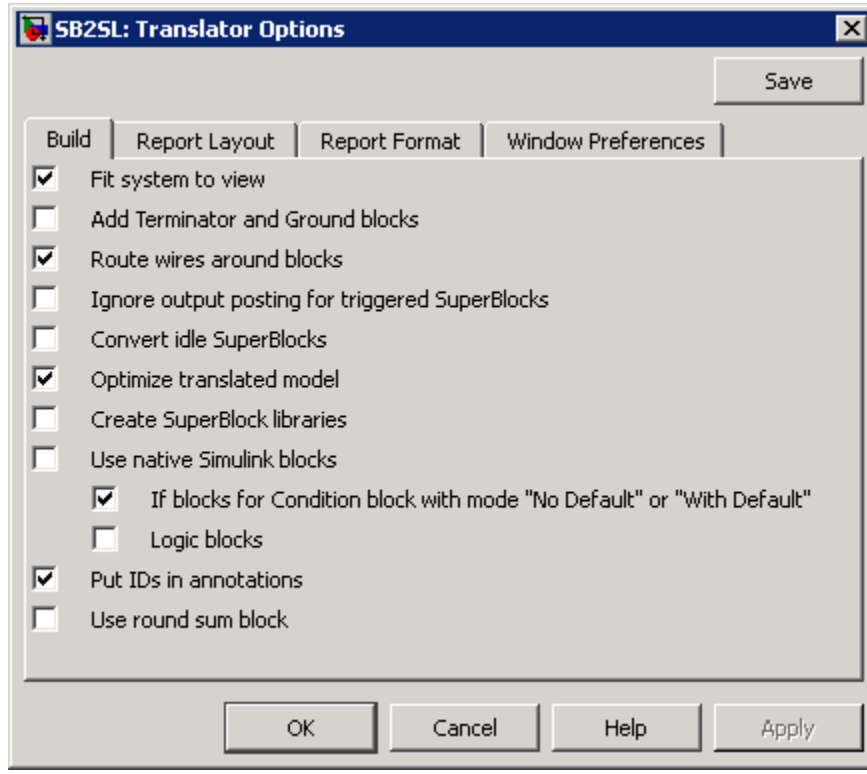
To reset default option settings, in the MATLAB Command Window, type the following:

```
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.



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.

Option	Description
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
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.

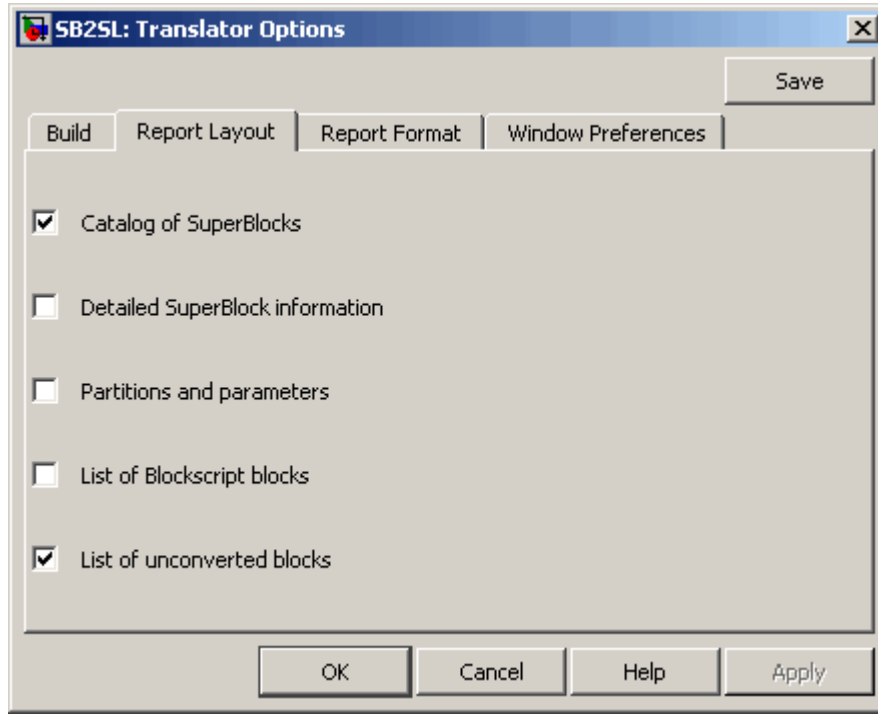
Option	Description
	Note If you have a library from a previous conversion, SB2SL will use that library. If you want to convert a model that reuses names from an earlier model conversion, you should convert the new model into an empty directory. Converting this model into the same directory as the earlier 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.



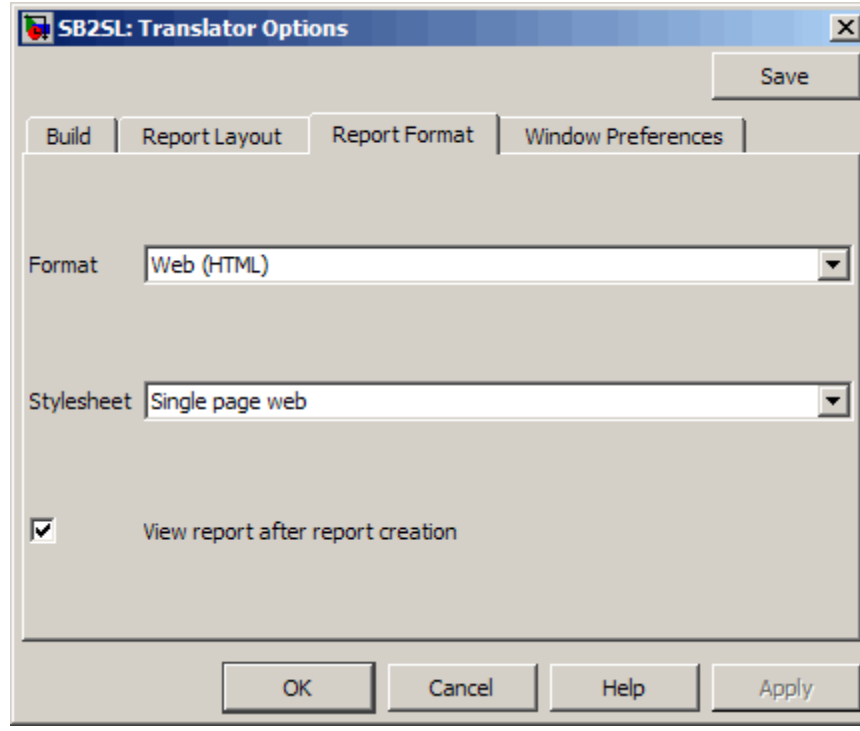
The following report options are available for inclusion in the build report:

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

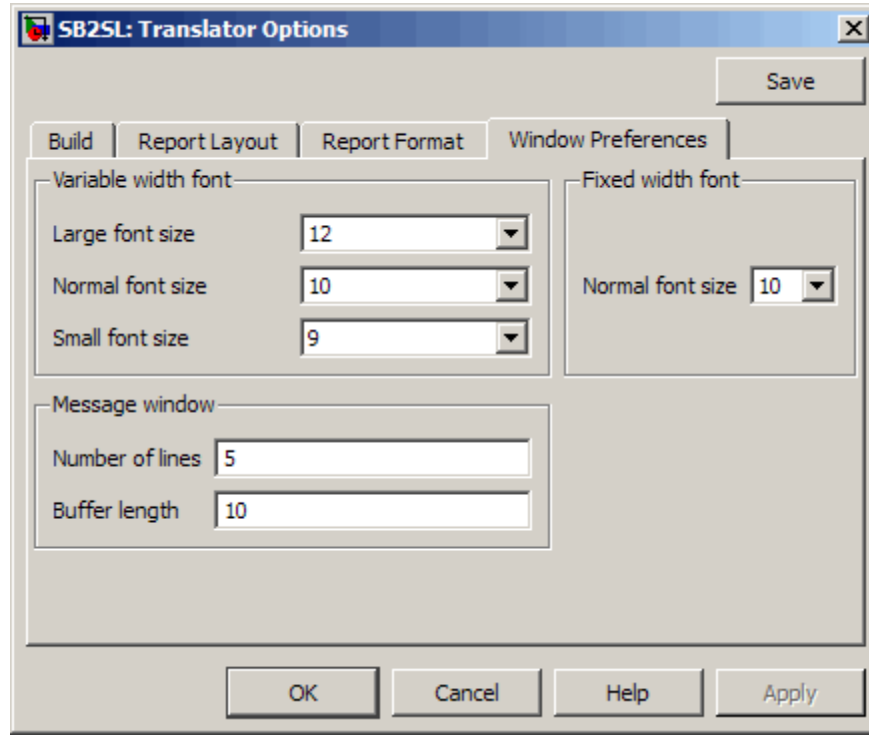
You have the following options for specifying the format of generated reports. Click the **Report Format** tab in the Translator Options window to access these options. Click the **Report Format** tab in the Translator Options window to access these options.



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

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.



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.

Converting SuperBlocks to Simulink Models

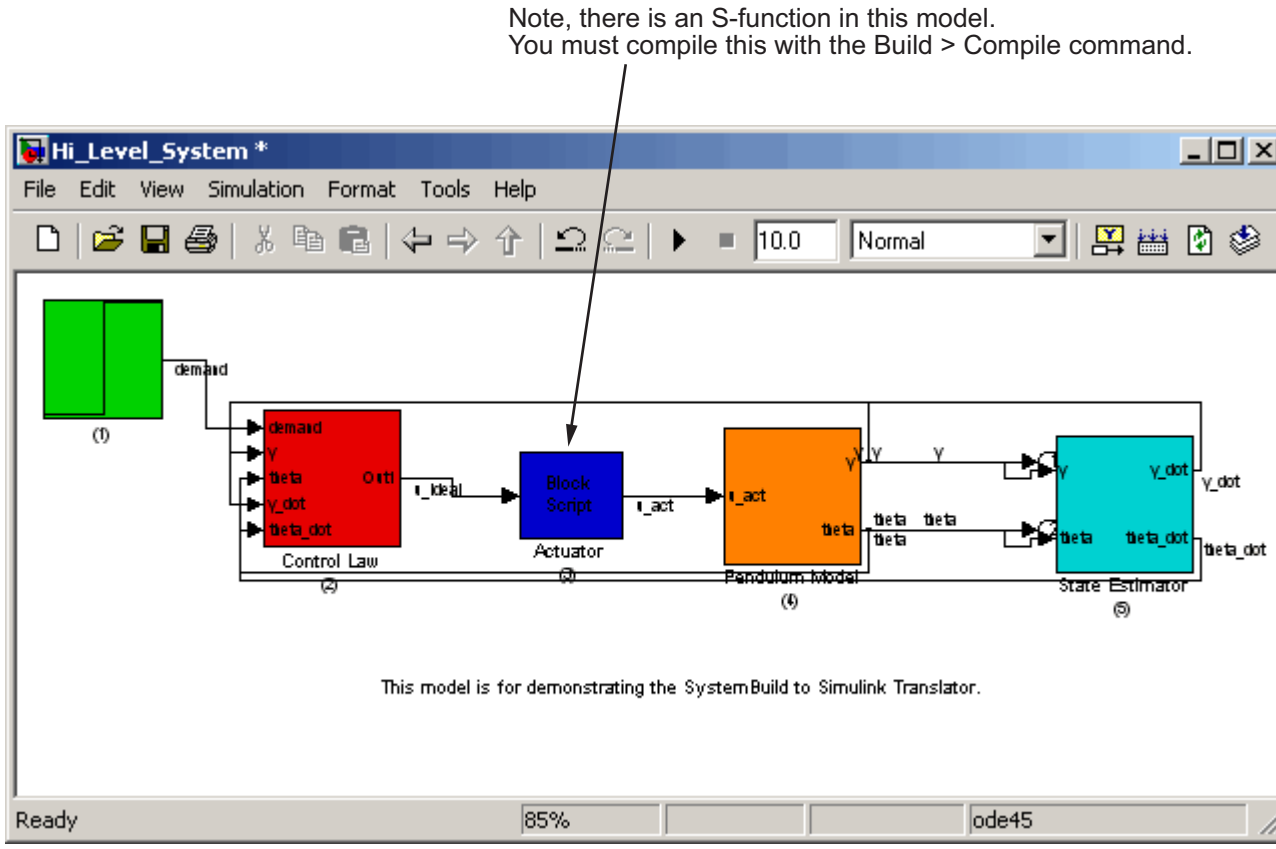
Before converting your SuperBlock to a Simulink model, you can set options for building the model and recording the translation. See "Setting Translation Options" on page 1-8 for more information.

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 "Setting Translation Options" on page 1-8)

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.



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 1-35 and "Suggestions for Handling Unconverted Blocks" on page 1-36 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 parameter settings:

Configuration Parameter	Value	Command–Line Parameter	Value
Optimization pane: **Inline parameters**	`On`	`'InlineParams'`	`'on'`
Solver pane: **Type**	`Variable-step`	`'SolverType'`	`'Variable-step'`
Solver pane: **Solver**	`ode45`	`'SolverName'`	`'ode45'`
Connectivity tab: **Mux blocks used to create bus signals**	`none`	`'StrictBusMsg'`	`'ErrorLevel1'`
Diagnostics pane: Data Validity: **Signal resolution**	`Explicity only`	`'SignalResolution-Control'`	`'UseLocalSettings'`
Model Referencing pane: **Rebuild options**	`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

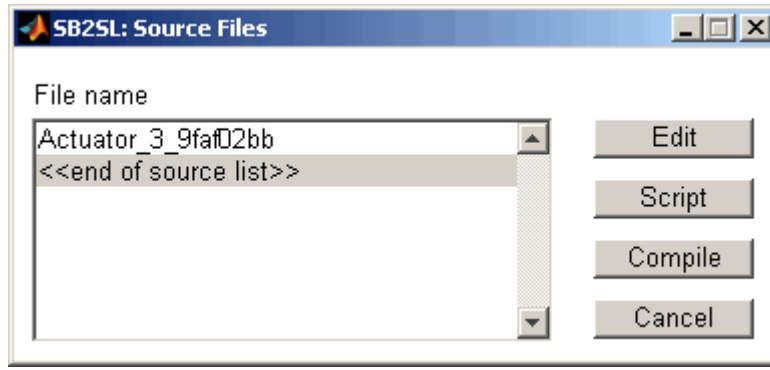
Alternatively, if the subsystem is atomic and the subsystem contents meet the criteria for model reference (see "Referencing a Model" in *Simulink User's Guide*), you can convert the subsystem to a referenced model. See the Converting Subsystems to Model Reference demo for an example of this.

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 1-9). 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.

Compiling 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.

For more information on the MATLAB MEX-file capability and Simulink S-functions, see the *MATLAB External Interfaces* or the *Developing S-Functions* documentation.

Saving 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 **Save > Model** to save the Simulink model to a file so that it can be reloaded directly from the MATLAB and Simulink environment.
- Select **Save > Data** 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. See "Using Callback Functions" in *Simulink User's Guide* for details on model callbacks.

Generating 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 1-13 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 "Referencing Configuration Sets") 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.

Improving 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:

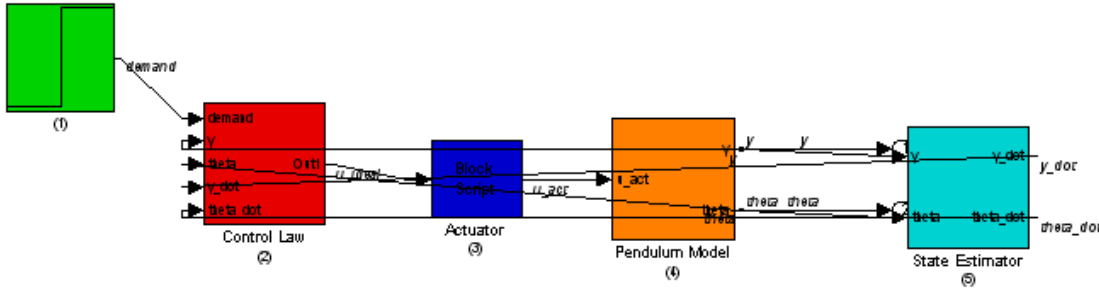
- Manually cleaning up the wiring using the tips in "Wiring Cleanup Tips" on page 1-26
- Converting block and system interfaces to native Simulink modeling styles: vectorization, matrix signals, and buses using the guidelines in "Migrating to a Native Simulink Modeling Style" on page 1-29

Wiring Cleanup Tips

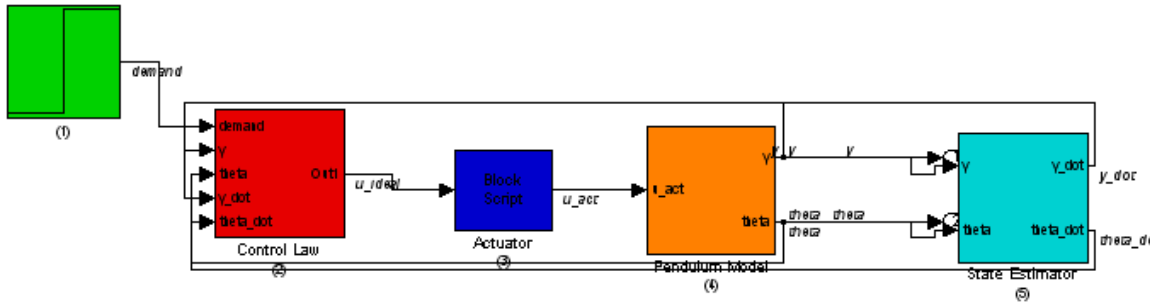
The following guidelines describe how you can visually clean a Simulink model that results from a SystemBuild model translation:

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 **Format > 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 (**Build > Options**, 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.

Silencing 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 (**Build > Options**, 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 (**Build > Options**, 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.

Migrating 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 "Using Composite Signals" in *Simulink User's Guide* for further information.

- 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 (see "Using the MATLAB Function Block" in *Simulink User's Guide*.

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.mdl`.

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.

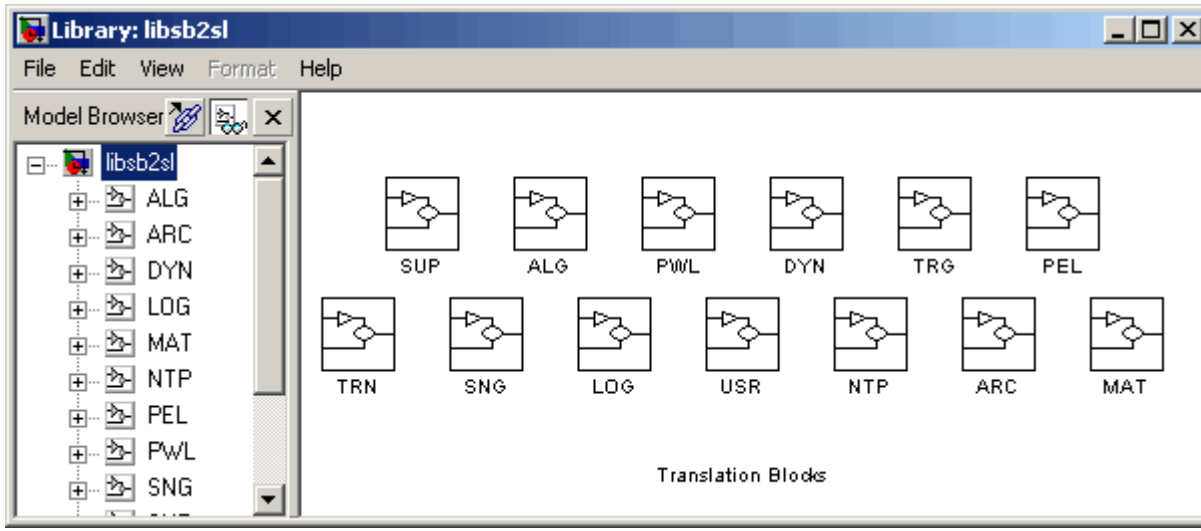
SB2SL Simulink Library

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.mdl`. 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

Using Simulink Coder Software with 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. See “About Referenced Model Simulation Modes” in *Simulink User’s Guide* for further details.

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 the job of converting 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.

File Formats

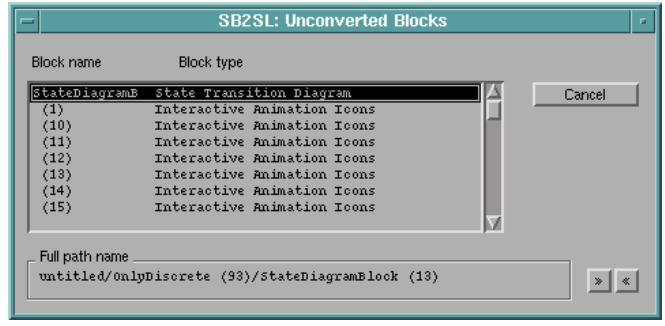
SB2SL cannot read SystemBuild files stored in the binary file format.

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 1-37 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

The Simulink *Developing S-Functions* guide provides information that you can refer to when converting these blocks.

See...	For...
"How S-Functions Work"	General information on how S-functions work.
"Writing S-Functions in C"	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.
"S-Function Callback Methods — Alphabetical List"	Reference of S-function callback methods.
"About DWork Vectors"	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 "Using the S-Function Builder Block to Convert Simple UserCode Block Code" on page 1-38.

- If your UserCode block is more complex, consider manually converting your code to an S-function. See "Manually Converting More Complex UserCode Block Code" on page 1-38. If a UserCode block has `MONIT`, `EVENT`, and/or `LIN` modes, it is more complex.
- If your UserCode block contains Fortran code, see "Converting UserCode Block Fortran Code" on page 1-40.

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

Before you start, see "Building S-Functions Automatically" in the *Developing S-Functions* guide. 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 Converting 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`
`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.

Converting UserCode Block Fortran Code

If your UserCode block code contains Fortran code, see "Creating Level-2 Fortran S-Functions" in the *Developing S-Functions* guide. 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')
```

Index

S

T

W