

# **Simulink<sup>®</sup> Coder<sup>™</sup> Release Notes**

---

## How to Contact MathWorks



[www.mathworks.com](http://www.mathworks.com) Web  
[comp.soft-sys.matlab](mailto:comp.soft-sys.matlab) Newsgroup  
[www.mathworks.com/contact\\_TS.html](http://www.mathworks.com/contact_TS.html) Technical Support



[suggest@mathworks.com](mailto:suggest@mathworks.com) Product enhancement suggestions  
[bugs@mathworks.com](mailto:bugs@mathworks.com) Bug reports  
[doc@mathworks.com](mailto:doc@mathworks.com) Documentation error reports  
[service@mathworks.com](mailto:service@mathworks.com) Order status, license renewals, passcodes  
[info@mathworks.com](mailto: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.

*Simulink® Coder™ Release Notes*

© COPYRIGHT 2011 by The MathWorks, Inc.

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

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

### Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://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](http://www.mathworks.com/patents) for more information.

<b>Bug Reports</b> .....	<b>1</b>
<b>Summary by Version</b> .....	<b>2</b>
<b>Version 8.1 (R2011b) Simulink® Coder Software</b> .....	<b>4</b>
<b>Version 8.0 (R2011a) Simulink® Coder Software</b> .....	<b>12</b>
<b>Compatibility Summary for Simulink® Coder Software</b> .....	<b>29</b>



## Bug Reports

Software is inherently complex and is not free of errors. The output of a code generator might contain bugs, some of which are not detected by a compiler. MathWorks reports critical known bugs brought to its attention on its Bug Report system at <http://www.mathworks.com/support/bugreports/>. Use the **Saved Searches and Watched Bugs** tool with the search phrase “Incorrect Code Generation” to obtain a report of known bugs that produce code that might compile and execute, but still produce wrong answers.

The bug reports are an integral part of the documentation for each release. Examine periodically all bug reports for a release, as such reports may identify inconsistencies between the actual behavior of a release you are using and the behavior described in this documentation.

In addition to reviewing bug reports, you should implement a verification and validation strategy to identify potential bugs in your design, code, and tools.

## Summary by Version

This table provides quick access to what's new in each version. For clarification, see “Using Release Notes” on page 2.

<b>Version (Release)</b>	<b>New Features and Changes</b>	<b>Version Compatibility Considerations</b>	<b>Fixed Bugs and Known Problems</b>
<b>Latest Version V8.1 (R2011b)</b>	Yes Details	Yes Summary	Bug Reports Includes fixes
V8.0 (R2011a)	Yes Details	Yes Summary	Bug Reports Includes fixes

### Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks® products required for this product (for example, MATLAB® or Simulink®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

If you are upgrading from a software version other than the most recent one, review the current release notes and all interim versions. For example, when you upgrade from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

### What Is in the Release Notes

#### New Features and Changes

- New functionality
- Changes to existing functionality

## **Version Compatibility Considerations**

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product release appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

## **Fixed Bugs and Known Problems**

MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. Bug Reports include provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

## **Documentation on the MathWorks Web Site**

Related documentation is available on [mathworks.com](http://mathworks.com) for the latest release and for previous releases:

- Latest product documentation
- Archived documentation

## Version 8.1 (R2011b) Simulink Coder Software

This table summarizes what's new in V8.1 (R2011b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes — Details labeled as Compatibility Considerations, below. See also Summary	Bug Reports Includes fixes

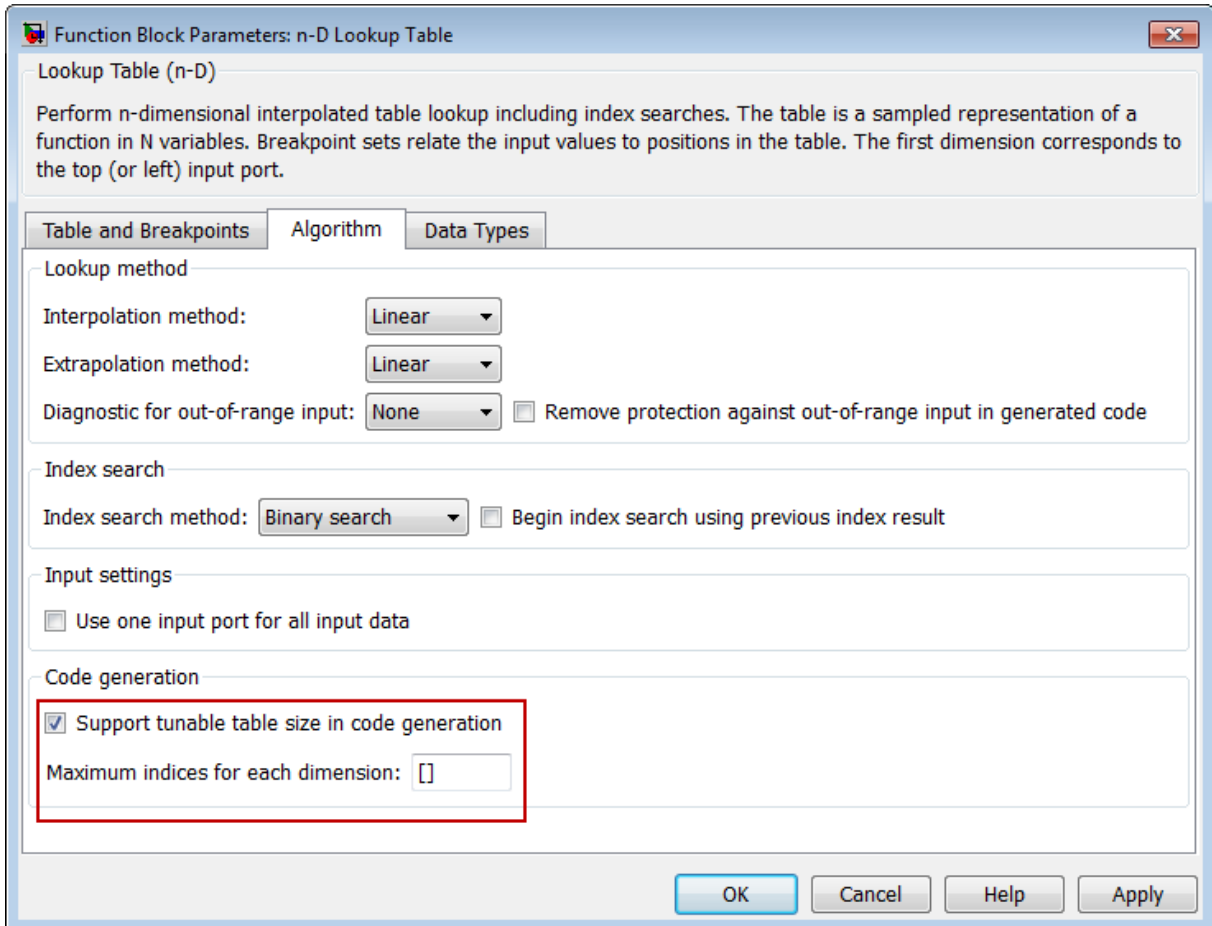
New features and changes introduced in this version are

- “n-D Lookup Table Block Supports Tunable Table Size” on page 5
- “Complex Output Support in Generated Code for the Trigonometric Function Block” on page 6
- “Code Optimizations for the Combinatorial Logic Block” on page 6
- “Code Optimizations for the Product Block” on page 6
- “Enhanced MISRA C Code Generation Support for Stateflow Charts” on page 7
- “Change for Constant Sample Time Signals in Generated Code” on page 8
- “New Code Generation Advisor Objective for GRT Targets” on page 8
- “Improved Integer and Fixed-Point Saturating Cast” on page 8
- “Generate Multitasking Code for Concurrent Execution on Multicore Processors” on page 9
- “Changes for Desktop IDEs and Desktop Targets” on page 9
- “Reserved Keyword UNUSED\_PARAMETER” on page 10
- “Target API for Verifying MATLAB® Distributed Computing Server™ (MDCS) Worker Configuration for Parallel Builds” on page 10
- “License Names Not Yet Updated for Coder Product Restructuring” on page 11
- “New and Enhanced Demos” on page 11



## n-D Lookup Table Block Supports Tunable Table Size

The n-D Lookup Table block provides new parameters for specifying a tunable table size in the generated code.



This enhancement enables you to change the size and values of your lookup table and breakpoint data without regenerating or recompiling the code.

## **Complex Output Support in Generated Code for the Trigonometric Function Block**

In previous releases, the imaginary part of a complex output signal was always zero in the generated code for the Trigonometric Function block. In R2011b, this limitation no longer exists. Code that you generate for any function in this block now supports complex outputs.

## **Code Optimizations for the Combinatorial Logic Block**

The Simulink® Coder™ build process uses a new technique to provide more efficient code for the Combinatorial Logic block.

Benefits include:

- Reuse of variables
- Dead code elimination
- Constant folding
- Expression folding

For example, in previous releases, temporary buffers were created to carry concatenated signals for this block. In R2011b, the build process eliminates unnecessary temporary buffers and writes the concatenated signal to the downstream global buffer directly. This enhancement reduces the stack size and improves code execution speed.

## **Code Optimizations for the Product Block**

The Simulink Coder build process provides more efficient code for matrix inverse and division operations in the Product block. The following summary describes the benefits and when each benefit is available:

<b>Benefit</b>	<b>Small matrices (2-by-2 to 5-by-5)</b>	<b>Medium matrices (6-by-6 to 20-by-20)</b>	<b>Large matrices (larger than 20-by-20)</b>
Faster code execution time	Yes, much faster	No, slightly slower	Yes, faster
Reduced ROM and RAM usage	Yes, for real values	Yes, for real values	Yes, for real values
Reuse of variables	Yes	Yes	Yes
Dead code elimination	Yes	Yes	Yes
Constant folding	Yes	Yes	Yes
Expression folding	Yes	Yes	Yes
Consistency with MATLAB® Coder™	Yes	Yes	Yes

### Compatibility Considerations

In the following cases, the generated code might regress from previous releases:

- The ROM and RAM usage increase for complex input data types.
- For blocks configured with 3 or more inputs of different dimensions, the code might include an extra buffer to store temporary variables for intermediate results.

### Enhanced MISRA C Code Generation Support for Stateflow Charts

In previous releases, the code generated to check whether or not a state in a Stateflow® chart was active included a line that looked something like this:

```
if (mdl_state_check_er_DWork.is_active_c1_mdl_state_c == 0)
```

In R2011b, that line has been modified to:

```
if (mdl_state_check_er_DWork.is_active_c1_mdl_state_c == 0U)
```

This enhancement supports MISRA C® 2004, rule 10.1.

## Change for Constant Sample Time Signals in Generated Code

In previous releases, constant sample time signals were initialized even if the **Data Initialization** field of their custom storage class was set to None.

In R2011b, constant sample time signals using a custom storage class for which the **Data Initialization** field is set to None will not be initialized for non-conditionally executed systems in generated code.

### Compatibility Considerations

If you use such constant time signals, you will notice that they are not initialized in the generated code in R2011b. To enable their initialization, change the setting of the **Data Initialization** field of their custom storage class from None to another value.

## New Code Generation Advisor Objective for GRT Targets

In R2011b, Execution efficiency is now available as a Code Generation Advisor objective for models with generic real-time (GRT) targets. You can use this objective to achieve faster code execution times for your models. For more information, see “Application Objectives”.

## Improved Integer and Fixed-Point Saturating Cast

Simulink Coder software now eliminates more dead branches in both integer and fixed-point saturation code.

## **Generate Multitasking Code for Concurrent Execution on Multicore Processors**

The Simulink Coder product extends the concurrent execution modeling capability of the Simulink product. With Simulink Coder, you can generate multitasking code that uses POSIX threads (Pthreads) or Windows threads for concurrent execution on multicore processors running Linux®, Mac® OS X, or Windows®.

See “Configuring Models for Targets with Multicore Processors”.

## **Changes for Desktop IDEs and Desktop Targets**

- “New Target Function Library for Intel IPP/SSE (GNU)” on page 9
- “Support Added for Single Instruction Multiple Data (SIMD) with Intel Processors” on page 9

### **New Target Function Library for Intel IPP/SSE (GNU)**

This release adds a new Target Function Library (TFL), Intel IPP/SSE (GNU), for the GCC compiler. This library includes the Intel Performance Primitives (IPP) and Streaming SIMD Extensions (SSE) code replacements.

### **Support Added for Single Instruction Multiple Data (SIMD) with Intel Processors**

This release adds support for the SIMD capabilities of the Intel® processors. The use of SIMD instructions increases throughput compared to traditional Single Instruction Single Data (SISD) processing.

The Intel IPP/SSE (GNU) TFL (code replacement library) optimizes generated code for SIMD.

The performance of the SIMD-enabled executable depends on several factors, including:

- Processor architecture of the target
- Optimized library support for the target

- The type and number of TFL replacements in the generated algorithmic code

Evaluate the performance of your application before and after using the TFL.

To use the SIMD capabilities with GCC and Intel processors, enable the Intel IPP/SSE (GNU) TFL. See “Target Function Library (TFL) and Embedded TargetsDesktop Targets”.

### **Reserved Keyword `UNUSED_PARAMETER`**

The Simulink Coder software adds the `UNUSED_PARAMETER` macro to the reserved keywords list for code generation. To view the complete list, see “Reserved Keywords”. In R2011b, code generation now defines `UNUSED_PARAMETER` in `rt_defines.h`. Previously, it was defined in `model_private.h`.

### **Target API for Verifying MATLAB® Distributed Computing Server™ (MDCS) Worker Configuration for Parallel Builds**

R2010b added the ability to use remote workers in MDCS configurations for parallel builds of model reference hierarchies. This introduced the possibility that parallel workers might have different configurations, some of which might not be compatible with a specific Simulink Coder target build. For example, the appropriate compiler might not be installed on a worker system.

R2011b provides a programming interface that target authors can use to automatically check the configuration of parallel workers and, if the parallel workers are not set up properly, take appropriate action, such as throwing an error or reverting to sequential builds. For more information, see “Supporting Model Referencing” in the Simulink Coder documentation.

For more information about parallel builds, see “Reducing Build Time for Referenced Models” in the Simulink Coder documentation.

## License Names Not Yet Updated for Coder Product Restructuring

The Simulink Coder and Embedded Coder™ license name strings stored in `license.dat` and returned by the `license('inuse')` function have not yet been updated for the R2011a coder product restructuring. Specifically, the `license('inuse')` function continues to return `'real-time_workshop'` for Simulink Coder and `'rtw_embedded_coder'` for Embedded Coder, as shown below:

```
>> license('inuse')
matlab
matlab_coder
real-time_workshop
rtw_embedded_coder
simulink
>>
```

The license name strings intentionally were not changed, in order to avoid license management complications in situations where Release 2011a or higher is used alongside a preR2011a release in a common operating environment. MathWorks plans to address this issue in a future release.

For more information about using the function, see the `license` documentation.

## New and Enhanced Demos

The following demos have been enhanced in R2011b:

Demo...	Now...
<code>rtwdemo_pmsmfoc_script</code>	Shows how you can perform system-level simulation and algorithmic code generation using Field-Oriented Control for a Permanent Magnet Synchronous Machine

## Version 8.0 (R2011a) Simulink Coder Software

This table summarizes what's new in V8.0 (R2011a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Yes Details below	Yes—Details labeled as <b>Compatibility Considerations</b> , below. See also Summary	Bug Reports Includes fixes

New features and changes introduced in this version are

- “Coder Product Restructuring” on page 13
- “Changes for Desktop IDEs and Desktop Targets” on page 18
- “Code Optimizations for Discrete State-Space Block, Product Block, and MinMax Block” on page 21
- “Ability to Share User-Defined Data Types Across Models” on page 21
- “C API Provides Access to Root-Level Inputs and Outputs” on page 22
- “ASAP2 File Generation Supports Standard Axis Format for Lookup Tables” on page 22
- “ASAP2 File Generation Enhancements for Computation Methods” on page 23
- “Lookup Table Block Option to Remove Input Range Checks in Generated Code” on page 23
- “Reentrant Code Generation for Stateflow Charts That Use Events” on page 24
- “Redundant Check Code Removed for Stateflow Charts That Use Temporal Operators” on page 25
- “Support for Multiple Asynchronous Function Calls Into a Model Block” on page 25
- “Changes to ver Function Product Arguments” on page 26



- “Updates to Target Language Compiler (TLC) Semantics and Diagnostic Information” on page 27
- “Change to Terminate Function for a Target Language Compiler (TLC) Block Implementation” on page 27
- “New and Enhanced Demos” on page 27

## **Coder Product Restructuring**

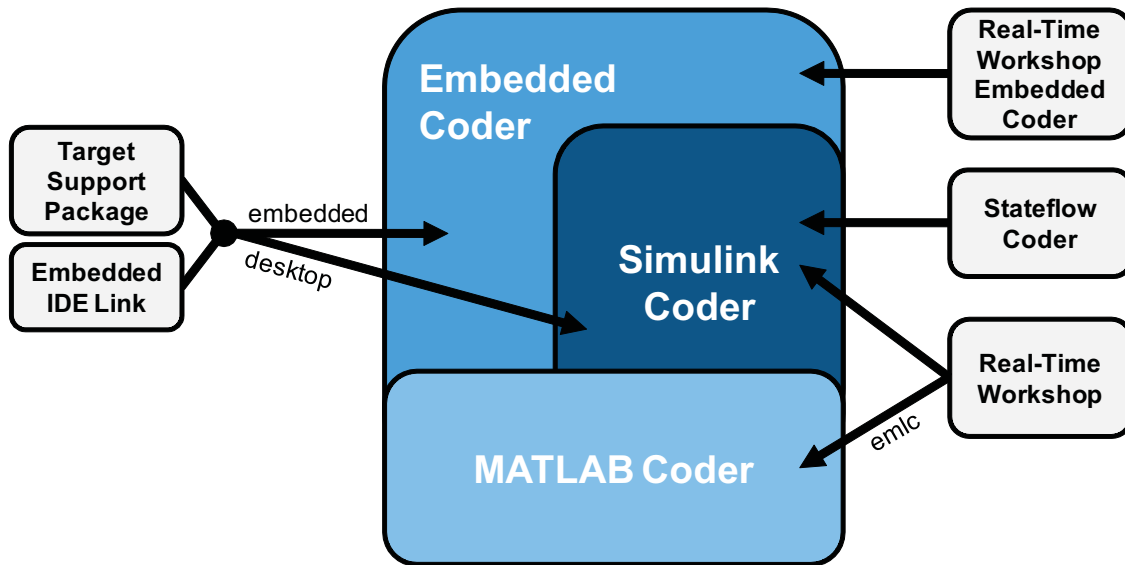
- “Product Restructuring Overview” on page 13
- “Resources for Upgrading from Real-Time Workshop or Stateflow® Coder” on page 14
- “Migration of Embedded MATLAB Coder Features to MATLAB® Coder” on page 15
- “Migration of Embedded IDE Link and Target Support Package Features to Simulink® Coder and Embedded Coder” on page 15
- “User Interface Changes Related to Product Restructuring” on page 16
- “Simulink Graphical User Interface Changes” on page 17

## **Product Restructuring Overview**

In R2011a, the Simulink Coder product combines and replaces the Real-Time Workshop® and Stateflow® Coder™ products. Additionally,

- The Real-Time Workshop facility for converting MATLAB code to C/C++ code, formerly referred to as Embedded MATLAB® Coder, has migrated to the new MATLAB Coder product.
- The previously existing products Embedded IDE Link™ and Target Support Package™ have been integrated into the new Simulink Coder and Embedded Coder products.

The following figure shows the R2011a transitions for C/C++ code generation related products, from the R2010b products to the new MATLAB Coder, Simulink Coder, and Embedded Coder products.



The following sections address topics related to the product restructuring.

### Resources for Upgrading from Real-Time Workshop or Stateflow Coder

If you are upgrading to Simulink Coder from Real-Time Workshop or Stateflow Coder, review information about compatibility and upgrade issues at the following locations:

- “Compatibility Summary for Simulink® Coder Software” on page 29 (latest release)
- In the Archived documentation on the MathWorks web site, select R2010b, and view the following tables, which are provided in the release notes for Real-Time Workshop and Stateflow Coder:
  - *Compatibility Summary for Real-Time Workshop Software*
  - *Compatibility Summary for Stateflow and Stateflow Coder Software*

These tables provide compatibility information for releases up through R2010b.

- If you use the Embedded IDE Link or Target Support Package capabilities that now are integrated into Simulink Coder and Embedded Coder, go to the Archived documentation and view the corresponding tables for Embedded IDE Link or Target Support Package:
  - *Compatibility Summary for Embedded IDE Link* (R2010b)
  - *Compatibility Summary for Target Support Package* (R2010b)

You can also refer to the rest of the archived documentation, including release notes, for the Real-Time Workshop, Stateflow Coder, Embedded IDE Link, and Target Support Package products.

### **Migration of Embedded MATLAB Coder Features to MATLAB Coder**

In R2011a, the MATLAB Coder function `codegen` replaces the Real-Time Workshop function `emlc`. The `emlc` function still works in R2011a but generates a warning, and will be removed in a future release. For more information, see “Migrating from Real-Time Workshop `emlc` Function” in the MATLAB Coder release notes.

### **Migration of Embedded IDE Link and Target Support Package Features to Simulink Coder and Embedded Coder**

In R2011a, the capabilities formerly provided by the Embedded IDE Link and Target Support Package products have been integrated into Simulink Coder and Embedded Coder. The follow table summarizes the transition of the Embedded IDE Link and Target Support Package hardware and software support into coder products.

Former Product	Supported Hardware and Software	Simulink Coder	Embedded Coder
Embedded IDE Link	Altium® TASKING		x
	Analog Devices™ VisualDSP++®		x
	Eclipse™ IDE	x	x
	Green Hills® MULTI®		x
	Texas Instruments' Code Composer Studio™		x
Target Support Package	Analog Devices™ Blackfin®		x
	ARM®		x
	Freescale™ MPC5xx		x
	Infineon® C166®		x
	Texas Instruments™ C2000™		x
	Texas Instruments C5000™		x
	Texas Instruments C6000™		x
	Linux OS	x	x
	Windows OS	x	
	VxWorks® RTOS		x

### User Interface Changes Related to Product Restructuring

Some user interface changes were made as part of merging the Real-Time Workshop and Stateflow Coder products into Simulink Coder. They include:

- Changes to code generation related elements in the Simulink Configuration Parameters dialog box

- Changes to code generation related elements in Simulink menus
- Changes to code generation related elements in Simulink blocks, including block parameters and dialog boxes, and block libraries
- References to Real-Time Workshop and Stateflow Coder and related terms in error messages, tool tips, demos, and product documentation replaced with references to the latest software

## Simulink Graphical User Interface Changes

Where...	Previously...	Now...
Configuration Parameters dialog box	<b>Real-Time Workshop</b> pane	<b>Code Generation</b> pane
Model diagram window	<b>Tools &gt; Real-Time Workshop</b>	<b>Tools &gt; Code Generation</b>
Subsystem context menu	<b>Real-Time Workshop</b>	<b>Code Generation</b>
Subsystem Parameters dialog box	Following parameters on main pane: <ul style="list-style-type: none"> <li>• <b>Real-Time Workshop system code</b></li> <li>• <b>Real-Time Workshop function name options</b></li> <li>• <b>Real-Time Workshop function name</b></li> <li>• <b>Real-Time Workshop file name options</b></li> <li>• <b>Real-Time Workshop</b></li> </ul>	On new <b>Code Generation</b> pane and renamed: <ul style="list-style-type: none"> <li>• <b>Function packaging</b></li> <li>• <b>Function name options</b></li> <li>• <b>Function name</b></li> <li>• <b>File name options</b></li> <li>• <b>File name (no extension)</b></li> </ul>

Where...	Previously...	Now...
	file name (no extension)	

## Changes for Desktop IDEs and Desktop Targets

- “Feature Support for Desktop IDEs and Desktop Targets” on page 18
- “Location of Blocks for Desktop Targets” on page 18
- “Location of Demos for Desktop IDEs and Desktop Targets” on page 19
- “Multicore Deployment with Rate Based Multithreading” on page 20

### Feature Support for Desktop IDEs and Desktop Targets

The Simulink Coder software provides the following features as implemented in the former Target Support Package and Embedded IDE Link products:

- Automation Interface
- External Mode
- Multicore Deployment with Rate Based Multithreading
- Makefile Generation (XMakefile)

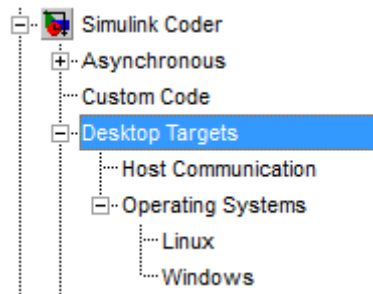
---

**Note** You can only use these features in the 32-bit version of your MathWorks products. To use these features on 64-bit hardware, install and run the 32-bit versions of your MathWorks products.

---

### Location of Blocks for Desktop Targets

Blocks from the former Target Support Package product and Embedded IDE Link product are now located in Simulink Coder under Desktop Targets.

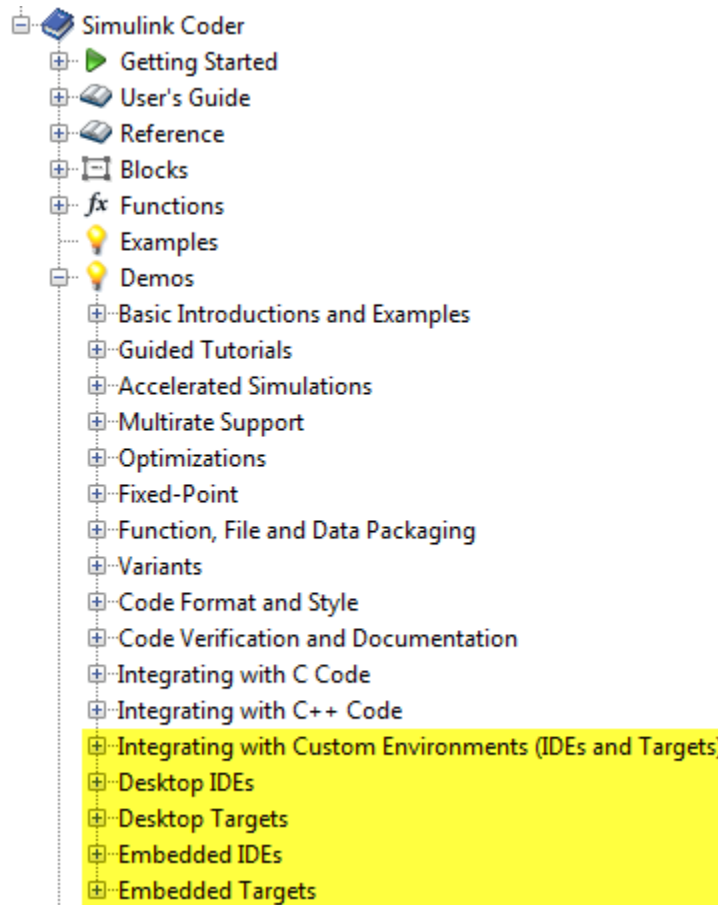


Desktop Targets includes the following types of blocks:

- Host Communication
- Operating Systems
  - Linux
  - Windows

### **Location of Demos for Desktop IDEs and Desktop Targets**

Demos from the former Target Support Package product and Embedded IDE Link product now reside under Simulink Coder product help. Click the expandable links, as shown.



## Multicore Deployment with Rate Based Multithreading

You can deploy rate-based multithreading applications to multicore processors running Windows and Linux. This feature improves performance by taking advantage of multicore hardware resources.

Also see the “Running Target Applications on Multicore Processors” user’s guide topic.



## Code Optimizations for Discrete State-Space Block, Product Block, and MinMax Block

The Simulink Coder build process uses a new technique to provide more efficient code for the following blocks:

- Discrete State-Space
- Product (element-wise matrix operations)

Benefits include:

- Reuse of variables
- Dead code elimination
- Constant folding
- Expression folding

For example, in previous releases, temporary buffers were created to carry concatenated signals for these blocks. In R2011a, the build process eliminates unnecessary temporary buffers and writes the concatenated signal to the downstream global buffer directly. This enhancement reduces the stack size and improves code execution speed.

The build process also provides more efficient code for the MinMax block. In R2011a, expression folding is enhanced with several local optimizations that enable more aggressive folding. This enhancement improves code efficiency for foldable matrix operations.

## Ability to Share User-Defined Data Types Across Models

In previous releases, user-defined data types that were shared among multiple models generated duplicate type definitions in the `model_types.h` file for each model. R2011a provides the ability to generate user-defined data type definitions into a header file that can be shared across multiple models, removing the need for duplicate copies of the data type definitions. User-defined data types that you can set up in a shared header file include:

- Simulink data type objects that you instantiate from the classes `Simulink.AliasType`, `Simulink.Bus`, `Simulink.NumericType`, and `Simulink.StructType`
- Enumeration types that you define in MATLAB code

For more information, see “Sharing User-Defined Data Types Across Models” in the Simulink Coder documentation.

## **C API Provides Access to Root-Level Inputs and Outputs**

The C API now provides programmatic access to root-level inputs and outputs. This allows you to log and monitor the root-level inputs and outputs of a model while you run the code generated for the model. To generate C API code for accessing root-level inputs and outputs at run time, select the model option **Generate C API for: root-level I/O**.

Macros for accessing C API generated structures are located in `matlabroot/rtw/c/src/rtw_capi.h` and `matlabroot/rtw/c/src/rtw_modelmap.h`, where `matlabroot` represents your MATLAB installation root.

For more information, see “Generate C API for: root-level I/O” and “Data Interchange Using the C API” in the Simulink Coder documentation.

## **ASAP2 File Generation Supports Standard Axis Format for Lookup Tables**

In previous releases, ASAP2 file generation for lookup table blocks supported the Fix Axis and Common Axis formats, but not the Standard Axis format, a format in which axis points are global in code but not shared among tables. R2011a adds support for Standard Axis format.

For more information, see “Defining ASAP2 Information for Lookup Tables” in the Simulink Coder documentation.

## ASAP2 File Generation Enhancements for Computation Methods

### Custom Names for Computation Methods

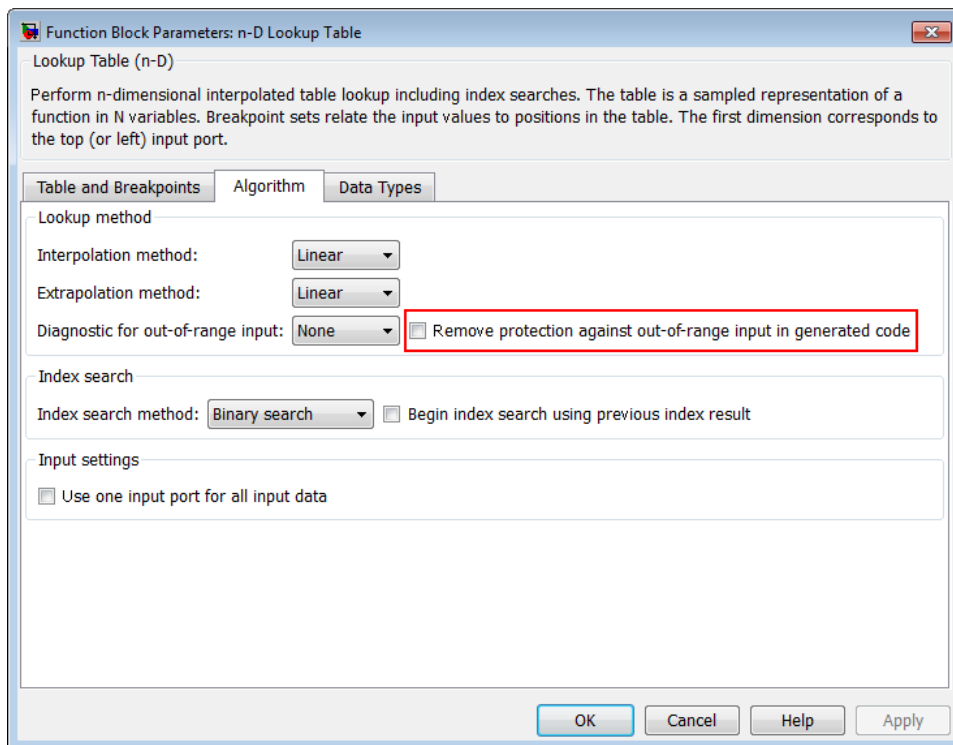
In generated ASAP2 files, computation methods translate the electronic control unit (ECU) internal representation of measurement and calibration quantities into a physical model oriented representation. R2011a adds the MATLAB function `getCompuMethodName`, which you can use to customize the names of computation methods. You can provide names that are more intuitive, enhancing ASAP2 file readability, or names that meet organizational requirements. For more information, see “Customizing Computation Method Names” in the Simulink Coder documentation.

### Ability to Suppress Computation Methods for FIX\_AXIS When Not Required

Versions 1.51 and later of the ASAP2 specification state that for certain cases of lookup table axis descriptions (integer data type and no doc units), a computation method is not required and the Conversion Method parameter must be set to the value `NO_COMPU_METHOD`. Beginning in R2011a, you can control whether or not computation methods are suppressed when not required, using the Target Language Compiler (TLC) option `ASAP2GenNoCompuMethod`. For more information, see “Suppressing Computation Methods for FIX\_AXIS” in the Simulink Coder documentation.

### Lookup Table Block Option to Remove Input Range Checks in Generated Code

When the breakpoint input to a Prelookup, 1-D Lookup Table, 2-D Lookup Table, or n-D Lookup Table block always falls within the range of valid breakpoint values, you can disable range checking in the generated code. By selecting **Remove protection against out-of-range input in generated code** on the block dialog box, your code can be more efficient.



## Reentrant Code Generation for Stateflow Charts That Use Events

When you generate code for Stateflow charts that use events, the code does not use a global variable to keep track of the currently active event. Elimination of this global variable enables the code to be reentrant, which allows you to:

- Deploy your code in multithreading environments
- Share the same algorithm with different persistent data
- Compile code that uses function variables that are too large to fit on the stack

In previous releases, reentrant code generation was not possible for charts that used events.

## Redundant Check Code Removed for Stateflow Charts That Use Temporal Operators

When you generate code for Stateflow charts that use temporal operators, the code excludes redundant checks for `tick` events and input events that are always true. This enhancement enables the code to be more efficient and applies to all temporal operators: `after`, `before`, `at`, `every`, and `temporalCount`.

In previous releases, the code generated for a temporal logic expression such as `after(x,tick)` would check for two conditions:

```
(event == tick) && (counter > x)
```

In R2011a, the code generated for `after(x,tick)` checks only for when the temporal counter exceeds `x`:

```
(counter > x)
```

This enhancement does not apply when a chart with multiple input events has super-step semantics enabled.

## Support for Multiple Asynchronous Function Calls Into a Model Block

Simulink and Simulink Coder software now support multiple asynchronous function calls into a Model block. This capability relies in part on the new Asynchronous Task Specification block.

The Asynchronous Task Specification block, in combination with a root-level Inport block, allows you to specify an asynchronous function-call input to a Model block. After placing this block at the output port of each root-level Inport block that outputs a function-call signal, select **Output function call** on the **Signal Attributes** pane of the Inport block. The Inport block then accepts function-call signals. You can use Asynchronous Task Specification blocks to specify parameters for the asynchronous task associated with the respective Inport blocks.

---

**Note** Use the new function call API, `LibBlockExecuteFcnCall`, to make function calls from an asynchronous source block to reference model destination blocks.

---

---

**Note** The demo model `rtwdemo_async_mdleftop` shows how you can simulate and generate code for asynchronous events on a real-time multitasking system, using asynchronous function calls as Model block inputs.

---

## Changes to ver Function Product Arguments

The following changes have been made to `ver` function arguments related to code generation products:

- The new argument `'simulinkcoder'` returns information about the installed version of the Simulink Coder product.
- The argument `'rtw'` works but now returns information about Simulink Coder instead of Real-Time Workshop. The software also displays the following message:

```
Warning: Support for ver('rtw') will be removed in a future release.  
Use ver('simulinkcoder') instead.
```

- The argument `'coder'`, which previously returned information about the installed version of the Stateflow Coder product, no longer works. The software displays a “not found” warning.

For more information about using the function, see the `ver` documentation.

## Compatibility Considerations

If a script calls the `ver` function with the `'rtw'` argument or the `'coder'` argument, update the script appropriately. For example, you can update the `ver` call to use the `'simulinkcoder'` argument, or remove the `ver` call.

## Updates to Target Language Compiler (TLC) Semantics and Diagnostic Information

Updates to TLC simplifies semantics and produces diagnostic information when using the scope resolution operator (`::`) and built-in function `EXISTS(::)`.

- If `var` can not be resolved in global scope, `::var` errors out
- If `var` can only be resolved in local scope, `EXISTS(::var)` returns false
- Diagnostic information highlights problematic TLC coding

For more information, see “Introducing the Target Language Compiler”.

## Change to Terminate Function for a Target Language Compiler (TLC) Block Implementation

Previously, the code generator attempted to execute the `Terminate` function from the TLC implementation of a block, even if the function did not exist. Now, the code generator only attempts to execute a `Terminate` function if it is defined in the TLC implementation of a block. In the case where the TLC implementation of a block includes a secondary TLC file, which includes a `Terminate` function, that `Terminate` function no longer executes.

## New and Enhanced Demos

The following demos have been added in R2011a:

Demo...	Shows How You Can...
<code>rtwdemo_async_mdleftop</code>	Simulate and generate code for asynchronous events on a real-time multitasking system, using asynchronous function calls as Model block inputs.

The following demos have been enhanced in R2011a:

<b>Demo...</b>	<b>Now...</b>
vipstabilize_fixpt_beagleboard videostabilization_host_templ	Use the new Video Capture block to simulate or capture a video input signal in the Video Stabilization demo.



## Compatibility Summary for Simulink Coder Software

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided in the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
<b>Latest Version V8.1 (R2011b)</b>	See the <b>Compatibility Considerations</b> subheading for each of these new features or changes: <ul style="list-style-type: none"> <li>• “Code Optimizations for the Product Block” on page 6</li> <li>• “Change for Constant Sample Time Signals in Generated Code” on page 8</li> </ul>
V8.0 (R2011a)	See the <b>Compatibility Considerations</b> subheading for each of these new features or changes: <ul style="list-style-type: none"> <li>• “Changes to ver Function Product Arguments” on page 26</li> </ul>

---

**Note** If you are upgrading to Simulink Coder from Real-Time Workshop or Stateflow Coder, you can review compatibility information for releases up through R2010b by going to the Archived documentation on the MathWorks web site. Select R2010b and view the following tables, which are provided in the release notes for each product:

- *Compatibility Summary for Real-Time Workshop Software*
- *Compatibility Summary for Stateflow and Stateflow Coder Software*

If you use the Embedded IDE Link or Target Support Package capabilities that now are integrated into Simulink Coder and Embedded Coder, go to the Archived documentation, select R2010b, and view the corresponding tables for each product:

- *Compatibility Summary for Embedded IDE Link*
  - *Compatibility Summary for Target Support Package*
-