Simulink® Coder™ Release Notes

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.

Simulink® Coder™ Release Notes

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.

Contents

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.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems
Latest Version 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` for the latest release and for previous releases:

- Latest product documentation
- Archived documentation

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 **Compatibility Considerations**, below. See also Summary	Bug Reports Includes fixes

New features and changes introduced in this version are

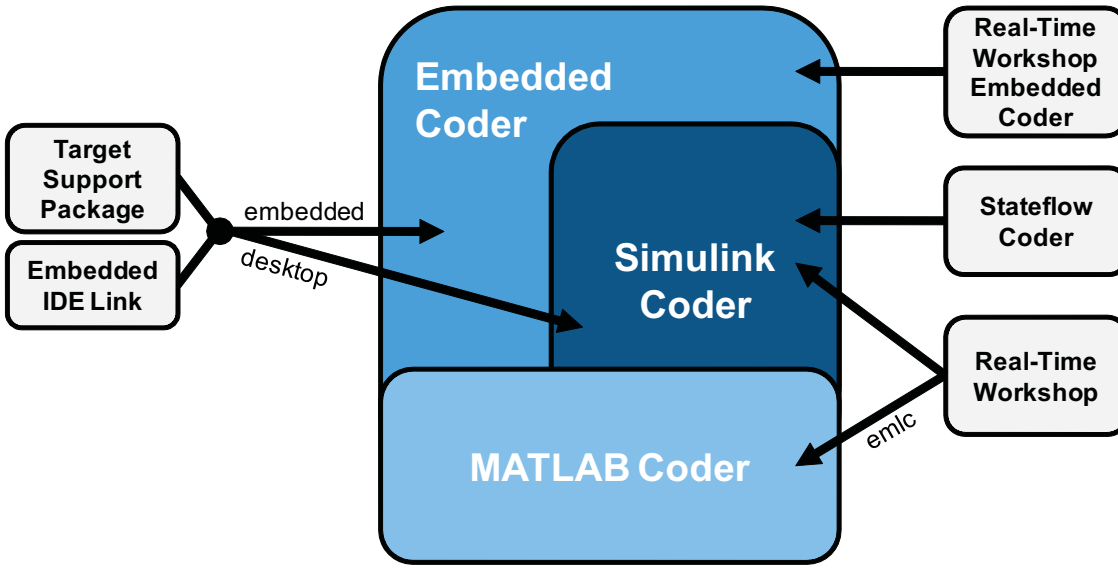
Coder Product Restructuring

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 21 (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	**Real-Time Workshop** pane	**Code Generation** pane
Model diagram window	**Tools > Real-Time Workshop**	**Tools > Code Generation**
Subsystem context menu	**Real-Time Workshop**	**Code Generation**
Subsystem Parameters dialog box	Following parameters on main pane: • **Real-Time Workshop system code** • **Real-Time Workshop function name options** • **Real-Time Workshop function name** • **Real-Time Workshop file name options** • **Real-Time Workshop**	On new **Code Generation** pane and renamed: • **Function packaging** • **Function name options** • **Function name** • **File name options** • **File name (no extension)**

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

Changes for Desktop IDEs and Desktop Targets

Feature Support for Desktop IDEs and Desktop Targets

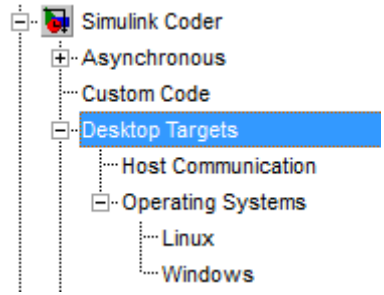
The Simulink Coder software provides the following features as implemented in the former Target Support Package and Embedded IDE Link products:

- 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.

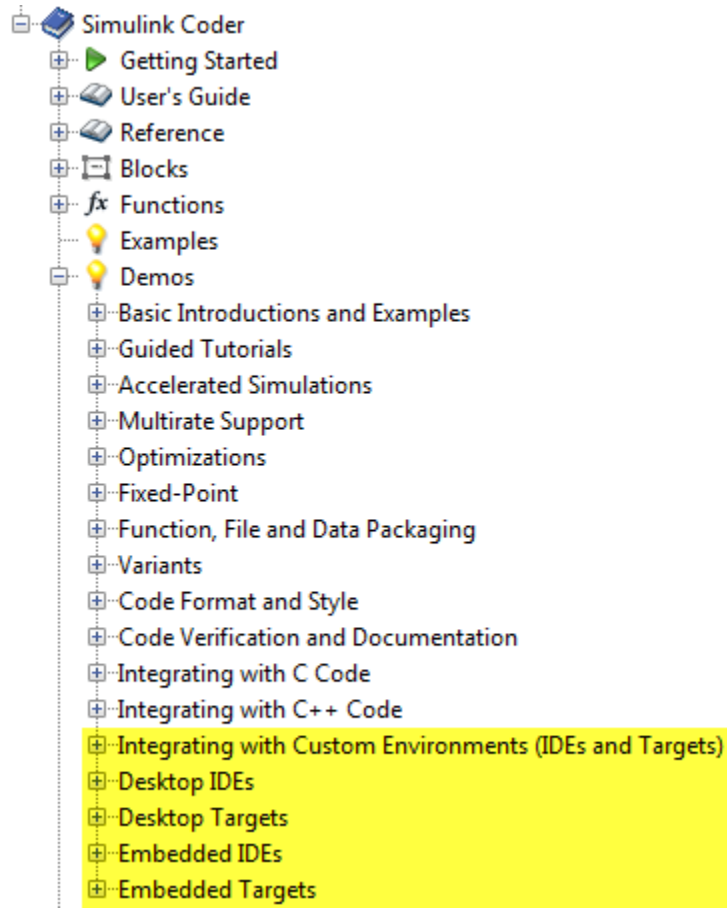


Desktop Targets includes the following types of blocks:

- 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.

Capture Video Input from USB Cameras on Linux

You can use the new Video Capture block to develop and prototype video applications for Linux running on your host computer. This block uses the Linux V4L2 API device driver framework, which supports most USB cameras.

Video Capture block can synthesize several types of video outputs. On Linux host computers, the block can also generate a live video output during model simulations. For more information, see the Linux Video Capture block reference topic. See the Video Stabilization demo in the product help for Simulink Coder under “Demos”.

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 "Interacting with Target Application Data 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

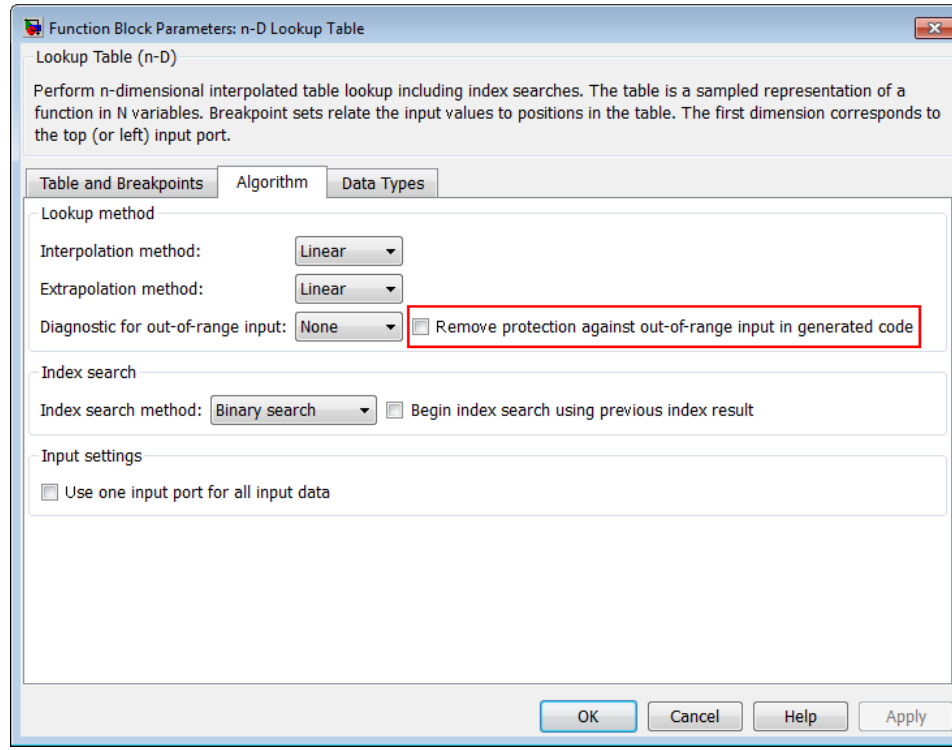
In generated ASAP2 files, computation methods translate the electronic control unit (ECU) internal representation of measurement and calibration quantities into a physical model oriented representation. R2011a adds the MATLAB function `getCompuMethodName`, which you can use to customize the names of 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 The demo model `rtwdemo_async_mdlreftop` 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...
`rtwdemo_async_mdlreftop`	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:

Demo...	Now...
`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
Latest Version V8.0 (R2011a)	See the **Compatibility Considerations** subheading for each of these new features or changes: • "Changes to ver Function Product Arguments" on page 18

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*