

# Embedded Coder

---

## Generate C and C++ code optimized for embedded systems

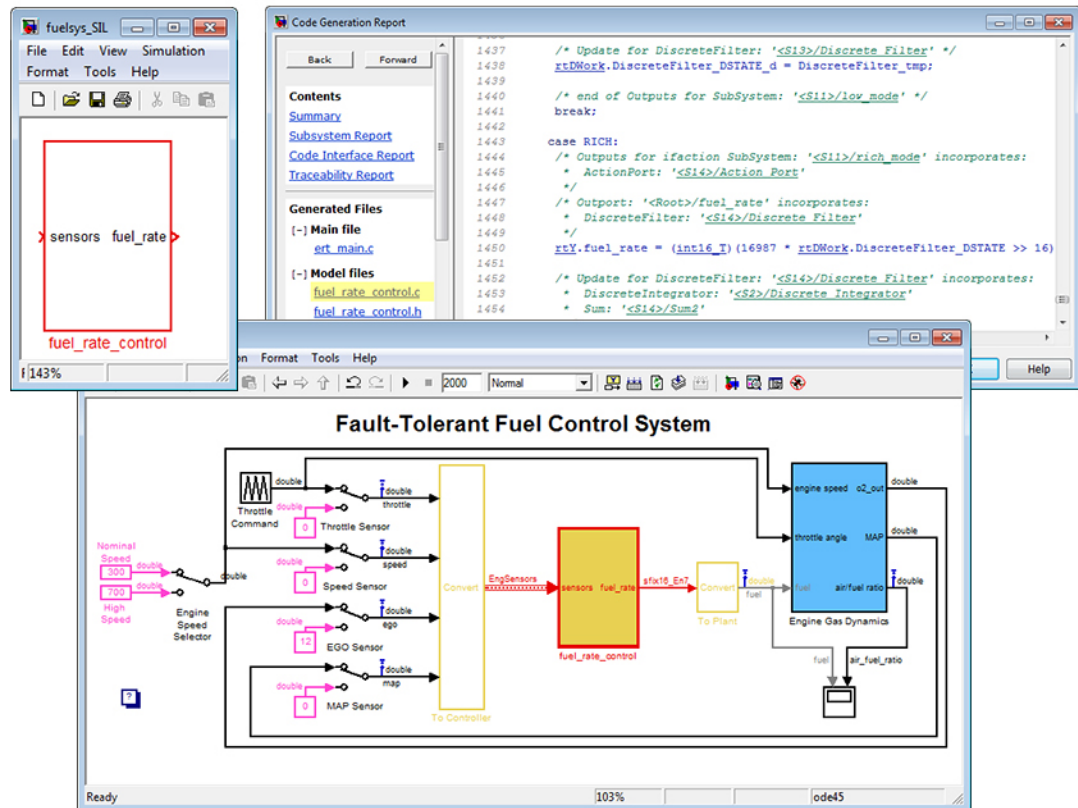
Embedded Coder™ generates readable, compact, and fast C and C++ code for use on embedded processors, on-target rapid prototyping boards, and microprocessors used in mass production. Embedded Coder enables additional [MATLAB Coder™](#) and [Simulink Coder™](#) configuration options and advanced optimizations for fine-grain control of the generated code's functions, files, and data. These optimizations improve code efficiency and facilitate integration with legacy code, data types, and calibration parameters used in production. You can incorporate a third-party development environment into the build process to produce an executable for turnkey deployment on your embedded system.

Embedded Coder offers built-in support for AUTOSAR and ASAP2 software standards. It also provides traceability reports, code interface documentation, and automated software verification to support DO-178, IEC 61508, and ISO 26262 software development.

Learn more about MathWorks support for certification in [automotive](#), [aerospace](#), and [industrial automation](#) applications.

### Key Features

- Optimization and code configuration options that extend MATLAB Coder and Simulink Coder
- Storage class, type, and alias definition using [Simulink®](#) data dictionary capabilities
- Processor-specific code optimization
- Multirate, multitask, and multicore code execution with or without an RTOS
- Code verification, including SIL and PIL testing, custom comments, and code reports with tracing of models to and from code and requirements
- Integration of Texas Instruments' Code Composer Studio™, Analog Devices™ VisualDSP++®, and [other third-party embedded development environments](#)
- Standards support, including ASAP2, AUTOSAR, DO-178, IEC 61508, ISO 26262, and MISRA C® in Simulink



A fixed-point model with generated code and its simulation mode set for SIL execution. Embedded Coder lets you quickly generate, document, and test code for production embedded systems.

## Configuring and Working with Targets

To configure code generation settings for Embedded Coder, you use the [MATLAB Coder](#) project user interface or the Simulink Model Explorer. You can also configure each setting directly using [MATLAB](#) commands and scripts.

From the MATLAB Coder project user interface, you can:

- Generate code for your MATLAB files and functions
- Opt to use Embedded Coder features
- Configure the project settings for code generation
- Create, load, and reuse multiple projects

From the Simulink Model Explorer, you can:

- Generate code for your [Simulink](#) models and subsystems
- Select an Embedded Coder target
- Configure the target for code generation
- Create, load, and reuse multiple configuration sets

## Selecting Targets

Embedded Coder uses configuration objects and system target files to translate your [MATLAB](#) code and [Simulink](#) models into production-quality source code and executables.

For a MATLAB configuration object, you specify one of the following output targets:

- MEX-file
- C/C++ static library
- C/C++ executable

For a Simulink system target file, you specify the real-time environment on which your generated code will run. Embedded Coder includes target files for several ready-to-run configurations, and supports third-party and custom targets as well. Built-in targets include:

**Embedded Real-Time Target** — Generates ANSI/ISO C, C++, and encapsulated C++ code with floating-point and fixed-point data for efficient real-time execution on virtually any production processor

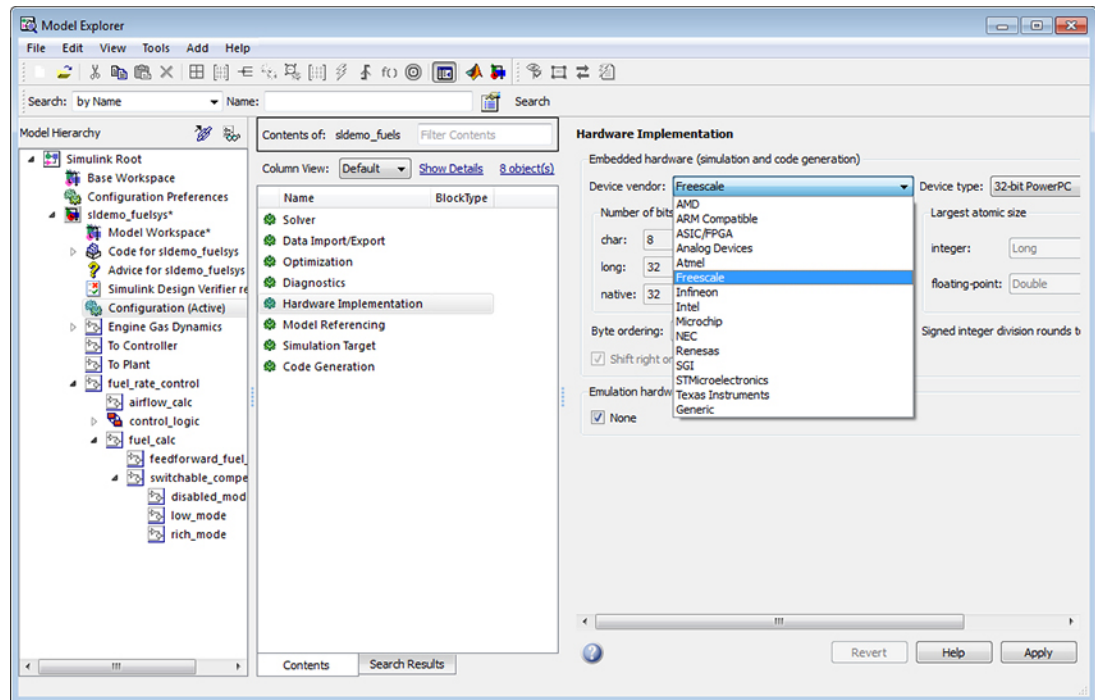
**AUTOSAR Target** — Generates C code and run-time interfaces that support development of [AUTOSAR](#) software components

**Shared Library Target** — Generates a shared library version of your code for host platform execution, either as a Windows® dynamic link library (.dll) file or a UNIX® shared object (.so) file

**IDE Link Target** — Generates code for compilation and deployment using a supported third-party integrated development environment (IDE) such as Texas Instruments' Code Composer Studio

### Defining Embedded Hardware Characteristics

For MATLAB or Simulink code generation, you select the deployment processor from a predefined list or use generic target settings. You can also extend the predefined list for your custom environment.



Simulink Model Explorer, which provides access to a predefined list or generic settings for specifying the microprocessor for code deployment. Embedded Coder generates code for any microprocessor or DSP, including 8-bit, 16-bit, and 32-bit.

### Defining and Controlling Custom Data

Embedded Coder enables you to define and control how the model data appears in the generated code. To facilitate software integration, you can specify class, size, and complexity of [MATLAB](#) data with the [MATLAB Coder](#) project user interface for entry point functions and global data.

For MATLAB code, Embedded Coder supports all MATLAB Coder data definitions including fixed-point objects.

For [Simulink](#) models, Embedded Coder supports the following data specification and data dictionary capabilities for generating code:

**Simulink data object** — Provides predefined storage classes, including constant, volatile, exported global, imported global, define directive, structure, bit field (including bit-packed structure), and get and set access methods

**Module packaging data object** — Provides preconfigured attributes for advanced data objects typically used in mass production, such as memory segments to calibrate or tune lookup tables

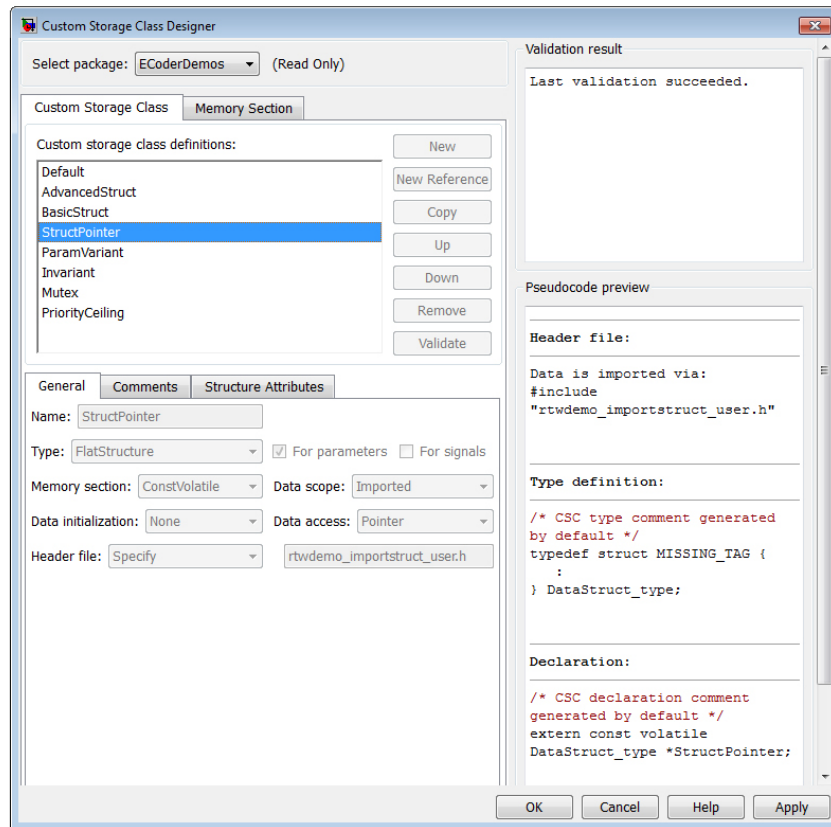
**User data type** — Lets you create abstract types for complex data so you can precisely control how model data appears in the generated code, interface with any legacy data, and augment or replace Simulink built-in types

The following tools help you design and manage project data in Simulink:

**Custom Storage Class Designer** — Lets you graphically create custom definitions and declarations to import data structures into the generated code, export data, conserve memory, or automatically generate data compliant with exchange standards, such as ASAM or ASAP2

**Simulink Model Explorer** — Displays all data used by Simulink models and [Stateflow](#)<sup>®</sup> charts and provides customizable views so you can tailor the information in a data dictionary format

Embedded Coder gives you access to ASAP2 data exchange files in Simulink, enabling you to export model data with complex data definitions using the ASAP2 standard. You can modify the built-in capabilities to produce other data exchange mechanisms.



A custom storage class created with the Customer Storage Class Designer, which lets you design, view, and validate complex data types using an intuitive graphical interface.

## Optimizing and Packaging Code

Using Embedded Coder, you can control function boundaries, preserve expressions, and apply optimizations on multiple blocks to further reduce code size. Data is exchanged with the generated code via global variables or function arguments. You can trace the generated code to blocks and signals in your model.

Embedded Coder options for generating code from [MATLAB](#) code and [Simulink](#) models enable you to:

- Generate processor-specific code for math functions and operators
- Reuse code for exporting to legacy or external environments
- Eliminate unnecessary initialization, termination, logging, and error-handling code
- Remove floating-point code from integer-only applications

Additional Embedded Coder optimization and configuration options are available for Simulink models, enabling you to:

- Generate code variants using macros for preprocessor compilation from models
- Store Boolean data and [Stateflow](#) states in bitsets
- Control the format of each generated file
- Determine how global data is defined and referenced
- Specify the contents and placement of comments

The image displays a MATLAB/Simulink workspace with three main windows:

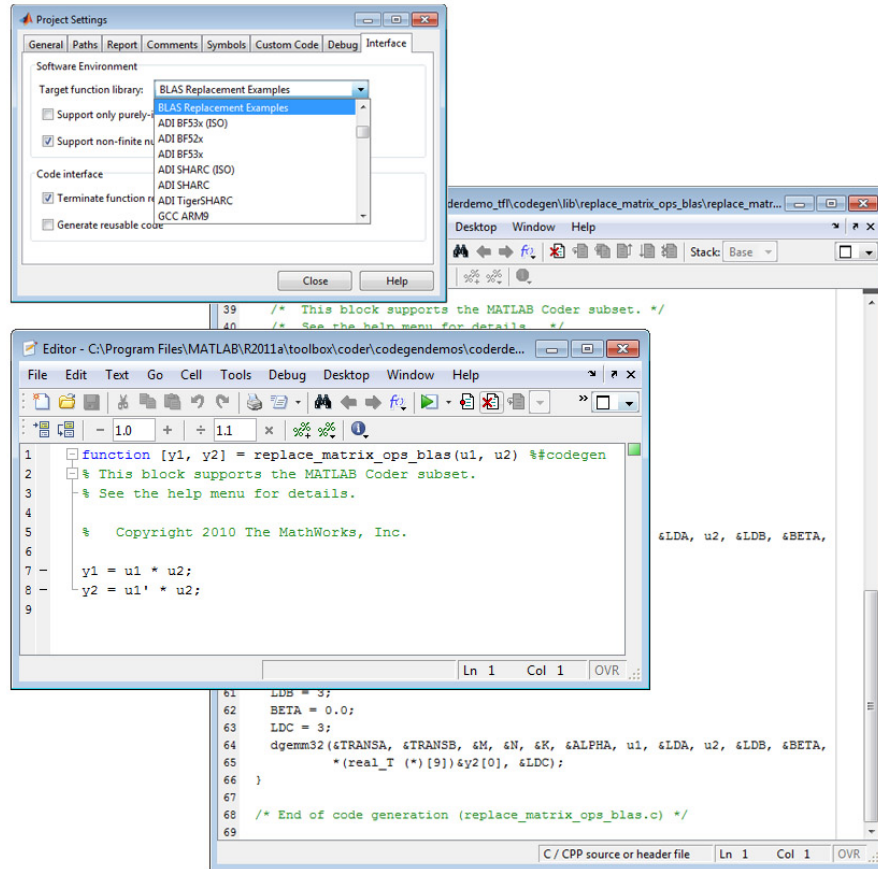
- Simulink Model:** A block diagram with inputs In1, In2, In3, In4, In9, and In10. It contains blocks for Matrix Multiply, DGEMM32, DGEMM32\_2, Hermitian Function, and ZGEMM32. The model is running at 33% simulation speed.
- Configuration Parameters (Active):** A dialog box for 'rvdemo\_tfblas/Configuration'. The 'Software environment' section is expanded, showing 'Target function library' set to 'Matrix Multiplication to BLAS Examples'. Other options include 'Utility code generation', 'Support' (floating-point, absolute, variable), 'Multiversion type definition', 'Code interface', and 'Data exchange'.
- MATLAB Function Block Editor:** A window showing the code for the 'DGEMM32\_2' block. The code includes comments and MATLAB syntax for block configuration and data flow.

```

/* MATLAB Function Block: 'cRoots/DGEMM32_2' incorporates:
 * Input: 'cRoots/In1'
 * Input: 'cRoots/In2'
 * Output: 'cRoots/Out2'
 */

/* MATLAB Function 'DGEMM32_2': 'cSP>1' */
73
74 /* This block supports MATLAB for code generation. */
75 /* See the help menu for details. */
76 /* 'cSP>1:1' */
77 TRANSB = 'N';
78 TRANS = 'N';
79 M = 10;
80 N = 10;
81 K = 20;
82 ALPHA = 1.0;
83 LDB = 20;
84 BETA = 0.0;
85 LDC = 10;
86 dgemm32(TRANSB, 4TRANSB, 4M, 4N, 4K, 4ALPHA, xIn1, 4LDA, xIn2, 4LDB,
87 4BETA, ((real_T) [100]) 4xOut7[0], 4LDC);
88
89 /* Product: 'cRoots/DGEMM32' incorporates:
90 * Input: 'cRoots/In3'
91 * Input: 'cRoots/In4'
92 * Output: 'cRoots/Out3'
93 */

```

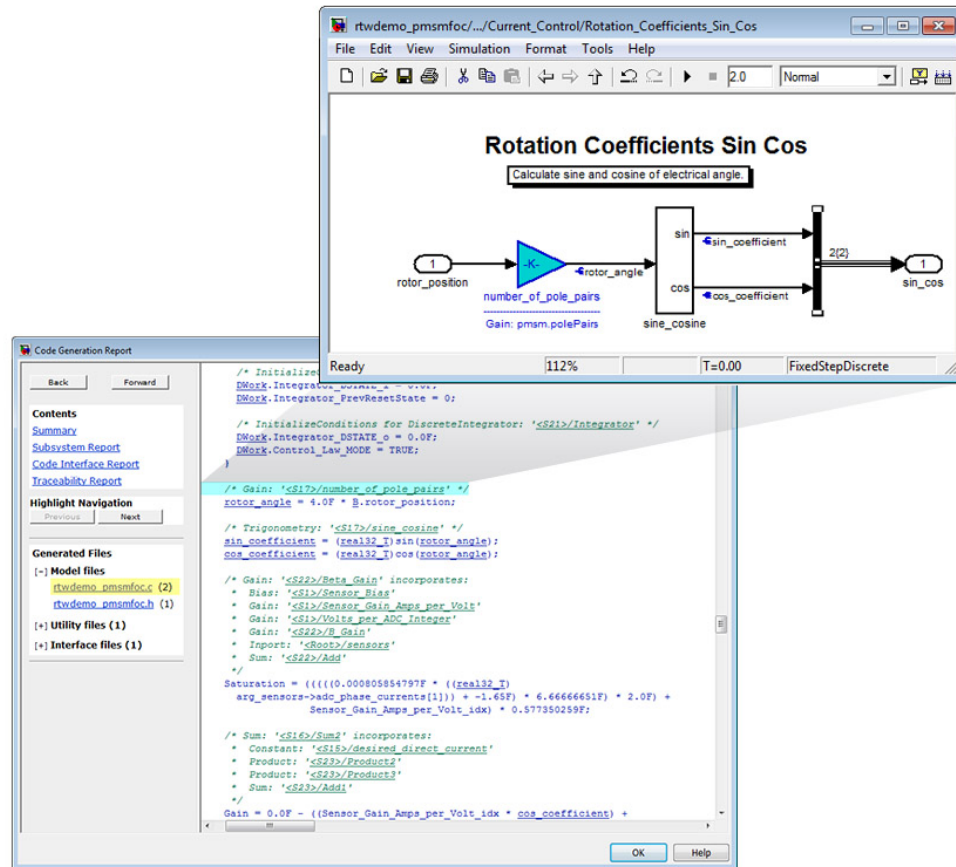


## Commenting, Tracing, and Documenting Code

Embedded Coder offers several capabilities for examining generated code for your [MATLAB](#) files and functions or your [Simulink](#) models and subsystems. Using these capabilities, you can:

- Generate a code report describing the modules and model configuration settings
- Control identifier formats for generated global data, data types, and functions
- Include MATLAB code as comments in generated code, including function help text

With Simulink, Embedded Coder also provides the ability to insert high-level requirements as code comments with links to the requirement source (requires [Simulink Verification and Validation™](#)). The code report for Simulink code generation also includes a code interface description, traceability report, and display of generated source files and code. Bidirectional links exist between the model and generated code, making it easy to navigate between every line of code and its corresponding Simulink model element, including subsystems, blocks, MATLAB functions and code, and [Stateflow](#) charts and transitions. You can click a link to highlight the corresponding model element or line of code, facilitating code reviews and debugging.



Simulink code generation report highlighting bidirectional traceability between algorithm and implementation.

## Executing and Verifying Code

Embedded Coder enables you to incorporate generated code into your code execution environment.

With [MATLAB](#), the code generated from Embedded Coder executes using the same execution framework as provided by MATLAB Coder.

With [Simulink](#), Embedded Coder significantly extends the real-time execution framework provided by [Simulink Coder](#). By default, the code can be executed with or without a real-time operating system (RTOS) and in single-tasking, multitasking, or asynchronous mode. You can also verify the code execution results using software-in-the-loop (SIL) and processor-in-the-loop (PIL) testing.

## Generating a Main Program

Embedded Coder generates an extensible main program based on information you provide for deploying the code in your real-time environment. This capability lets you generate and build a complete customized executable from your model.

## Grouping Rates

Embedded Coder generates single-rate or multirate code using periodic sample times specified in the model. For multirate, multitasking models, it employs a strategy called rate grouping that generates separate functions for the base rate task and for each sub-rate task in the model.

## Using Links and Targets

Automated deployment, integration, optimization, and execution of generated code is available for supported third-party IDEs, microprocessors, and RTOSs, including Wind River Systems® VxWorks®.

## Performing SIL and PIL Testing

Embedded Coder automates execution of generated code in Simulink for SIL testing or on the embedded target for PIL testing using Simulink simulation modes or S-function blocks. Code generation verification APIs help automate test execution and comparison of test results to simulation results from the original model. Integration with third-party tools enables structural code coverage analysis to measure test completeness.

### Resources

#### Product Details, Demos, and System Requirements

[www.mathworks.com/products/embedded-coder](http://www.mathworks.com/products/embedded-coder)

#### Trial Software

[www.mathworks.com/trialrequest](http://www.mathworks.com/trialrequest)

#### Sales

[www.mathworks.com/contactsales](http://www.mathworks.com/contactsales)

#### Technical Support

[www.mathworks.com/support](http://www.mathworks.com/support)

#### Online User Community

[www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)

#### Training Services

[www.mathworks.com/training](http://www.mathworks.com/training)

#### Third-Party Products and Services

[www.mathworks.com/connections](http://www.mathworks.com/connections)

#### Worldwide Contacts

[www.mathworks.com/contact](http://www.mathworks.com/contact)