Simulink Coder

Generate C and C++ code from Simulink and Stateflow models

Simulink Coder™ (formerly Real-Time Workshop®) generates and executes C and C++ code from Simulink® diagrams, Stateflow® charts, and MATLAB® functions. The generated source code can be used for real-time and nonreal-time applications, including simulation acceleration, rapid prototyping, and hardware-in-the-loop testing. You can tune and monitor the generated code using Simulink or run and interact with the code outside MATLAB and Simulink.

Key Features
▪ ANSI/ISO C and C++ code and executables for discrete, continuous, or hybrid Simulink and Stateflow models
▪ Incremental code generation for large models
▪ Integer, floating-point, and fixed-point data type support
▪ Code generation for single-rate, multirate, and asynchronous models
▪ Single-task, multitask, and multicore code execution with or without an RTOS
▪ External mode simulation for parameter tuning and signal monitoring

Selecting and Working with Targets

The Simulink Model Explorer provides a single user interface in Simulink to configure all code generation settings. From the Model Explorer you can:
• Select a code generation target
• Configure the target for code generation
• Generate code for models or subsystems

The Code Generation Advisor checks your model and code generation settings, and then offers advice on how to improve your configuration based on your needs. Once you have identified the optimal settings for your project, you can save them as a configuration set to reuse with other models.

Large aircraft system modeled using Simulink. Simulink Coder supports advanced Simulink features for large-scale modeling, including Model Blocks, variant subsystems, and arrays of buses.

Selecting Targets

Simulink Coder uses system target files to translate your models into source code and executables. Target files specify the environment on which your generated code will run. Simulink Coder includes target files for several ready-to-run configurations, and supports third-party and custom targets as well. Built-in targets include:

Generic Real-Time Target — Generates code for interactive tuning of model parameters, logs and displays real time simulation results, and allocates data statically for efficient real-time execution

Generic Real-Time Malloc Target — Uses dynamic memory allocation in the generated code, enabling you to include multiple instances of your model or multiple models in one executable

Rapid Simulation Target (RSim) — Provides a fast, flexible test platform for performing batch or Monte Carlo simulation studies using fixed-step or variable-step solvers, and lets you save data for each run to unique output files.
You can extend these targets to create customized run-time interfaces and device driver files, which enables you to target your specific environment and access Simulink Coder execution and debug capabilities in it.

Compiling and Executing Generated Code

Simulink Coder offers built-in support for compilation and execution of generated code with popular desktop environments, including:

- Eclipse™ IDE
- Microsoft® Visual C++® IDE
- Linux® operating system
- Windows® operating system

You can generate and deploy optimized C and C++ code in real-time embedded systems using Embedded Coder.

Defining and Controlling Data

Simulink Coder enables you to define and control how the model data appears in the generated code. You can also manage your data by:

- Declaring data types as integer, floating-point, and fixed-point
- Specifying storage to tune and calibrate parameters or constants
- Specifying storage to monitor and log signal data
- Reusing storage to minimize locally scoped data

Simulink Coder generates code from data stored in your Simulink diagram or in a data dictionary accessed via the Simulink Model Explorer. By employing different data dictionary sets, you can redeploy code from a single model to multiple targets.

Simulink Model Explorer showing various data types and storage classes used in a model. Note that custom storage classes require Embedded Coder.
Generating Code

Simulink Coder offers comprehensive code generation support for Simulink and Stateflow features and components, including:

- Model blocks for generating code incrementally
- Atomic subsystems and atomic subcharts for reusing code via reentrant C functions
- Bus objects and arrays of buses for generating structures in your code
- S-functions for simulating and interfacing with legacy code
- MATLAB function blocks in Simulink and MATLAB functions in Stateflow for generating code from models with MATLAB components

Simulink Coder also supports the Simulink product family, including fixed-point modeling, physical modeling, signal processing and communications modeling, and rapid prototyping and hardware-in-the-loop (HIL) testing products.

Code Generation Options

Simulink Coder provides user-selectable options to facilitate execution of generated code in desktop and simulation environments. These options include:

- Code reuse
- Expression folding
- Signal storage reuse
- Dead path elimination
- Parameter inlining
- Single-precision and prebuilt math libraries (including ISO C and GNU C)

Generated code using for-loop fusion and expression folding for a 10-element array model. Simulink Coder produces efficient and readable code for single and multidimensional signals.
Executing Code in a Real-Time Environment

Simulink Coder provides a complete framework for executing the generated code in real time and incorporating it into your execution environment. It generates single-rate or multirate code based on the periodic sample times you specify in the model.

Execution Modes

Code can be deployed with or without an RTOS, and in single-tasking, multitasking, or asynchronous mode.

**Single-Tasking Mode**

In single-tasking mode, a simple scheduler invokes the generated code in a single thread of execution, preventing preemption between rates.

**Multitasking Mode**

In multitasking mode, a deterministic, rate-monotonic scheduler invokes the generated code, enabling preemption between rates. In a bare-board environment, you preempt the code with nested interrupts. In an RTOS environment, you use task priorities and task preemption.

**Asynchronous Mode**

In asynchronous mode, nonperiodic or asynchronous rates are specified using Simulink S-functions. Simulink Coder translates these rates into the code necessary for your execution environment. You model and generate code for events, such as hardware interrupts, and trigger subsystems as independent tasks. You can use the included asynchronous block library as a template for creating a library appropriate for other environments.

**Real-Time Deployment**

Simulink and Simulink Coder provide a complete set of target-independent capabilities for real-time deployment. These include:

- Task priorities that you can specify for each rate in your model
- Production-quality counters and timers for computing absolute and elapsed time
- A Rate Transition block to specify data transfer mechanisms between rates (for example, semaphore, mutex, and double buffering), enabling you to make trade-offs among data integrity, determinism, and performance
- Overrun detection for incorporating error-handling logic for each rate

**Legacy Code Integration**

You can customize the generated code and its run-time environment by incorporating custom legacy C, Fortran, Ada, and C++ code for simulation and code generation. Customization tools in Simulink Coder include a Target Language Compiler (TLC), custom code blocks, template makefiles, build process APIs, and S-Function Builder. Using the Legacy Code Tool, you can rapidly integrate existing C and C++ functions into your Simulink models, which is particularly useful for integrating discrete-time code typically used in embedded systems.
Tuning Parameters and Logging Data

Simulink Coder helps you optimize your algorithm’s performance and improve code execution behavior by enabling you to interact with and analyze the generated executable.

Simulink Coder lets you monitor and tune block signals and parameters using the following data interfaces:

**Target-based C API** — Enables user-written code to access block outputs and parameters outside MATLAB and Simulink

**Host-based ASAP2 data exchange file** — Lets you use the ASAP2 standard data description for data measurement, calibration, and diagnostic systems outside MATLAB and Simulink

**Simulink External mode** — Lets you download new parameter values and upload signal values to view in Simulink or to record in the MATLAB workspace. (Opening a dialog box for a source block causes Simulink to pause. While Simulink is paused, you can edit the parameter values. You must close the dialog box to have the changes take effect and allow Simulink to continue.)

---

## Resources

<table>
<thead>
<tr>
<th>Product Details, Examples, and System Requirements</th>
<th>Online User Community</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Trial Software</th>
<th>Training Services</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.mathworks.com/trialrequest">www.mathworks.com/trialrequest</a></td>
<td><a href="http://www.mathworks.com/training">www.mathworks.com/training</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sales</th>
<th>Third-Party Products and Services</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Technical Support</th>
<th>Worldwide Contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.mathworks.com/support">www.mathworks.com/support</a></td>
<td><a href="http://www.mathworks.com/contact">www.mathworks.com/contact</a></td>
</tr>
</tbody>
</table>