

MATLAB EXPO 2018

Generating Industry Standards
Production C Code Using
Embedded Coder

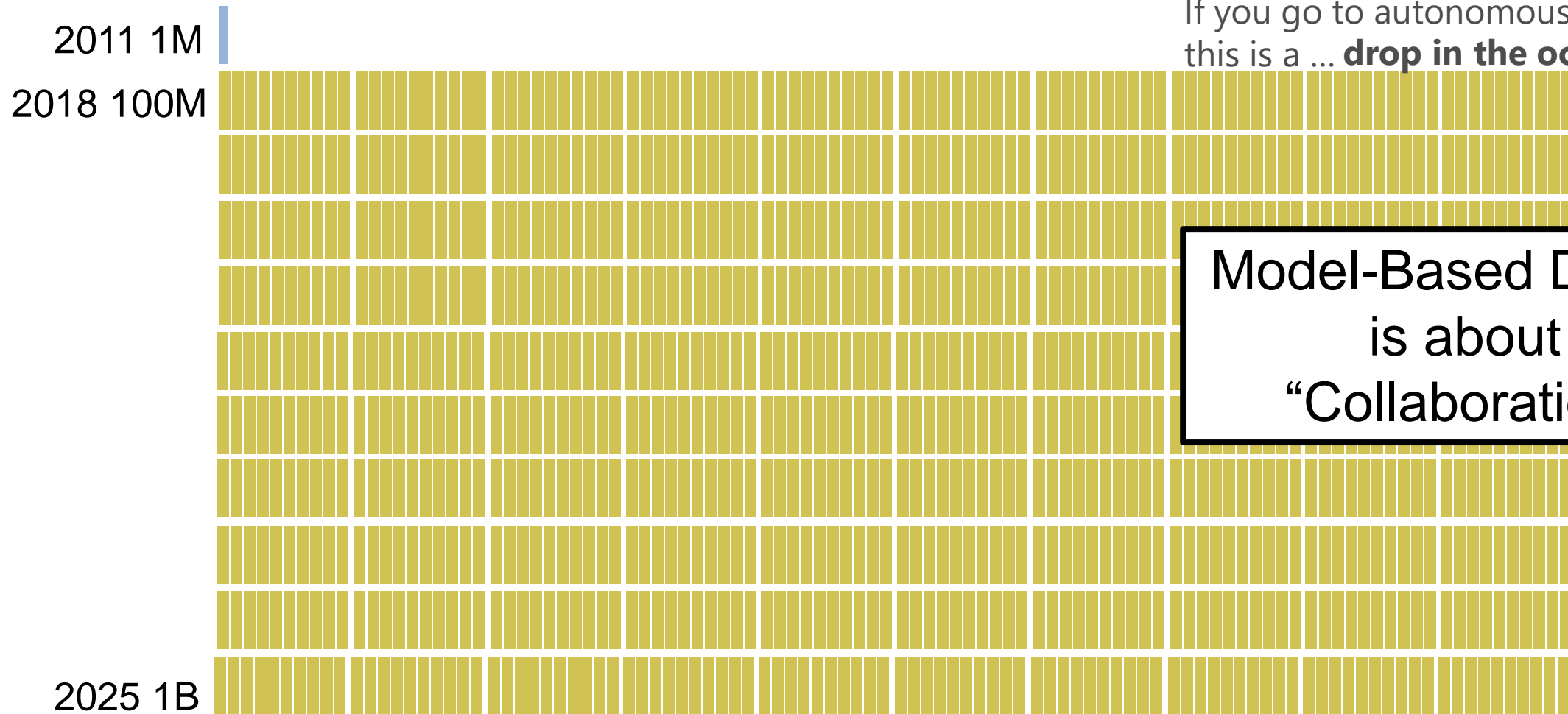
Rajat Arora
Durvesh Kulkarni



Lines of Code (LOC) is exploding

"We've got 100 million lines of code in the 7 Series.

If you go to autonomous driving this is a ... **drop in the ocean.**"



Model-Based Design
is about
"Collaboration"

Join Hands & Develop

...”standardization by **customizing** the Simulink® development environment”...

A software architect helps

- multiple engineers
- multiple projects

generate code that **conforms**
to the organization standard

Software
Architect

Software
Engineer

A software engineer directly
interacts with and generates
code from production models.

...deploys changes

requests changes...

Join Hands & Develop

Software
Architect

Software
Engineer

...Collaborative Workflow...

Quick Study

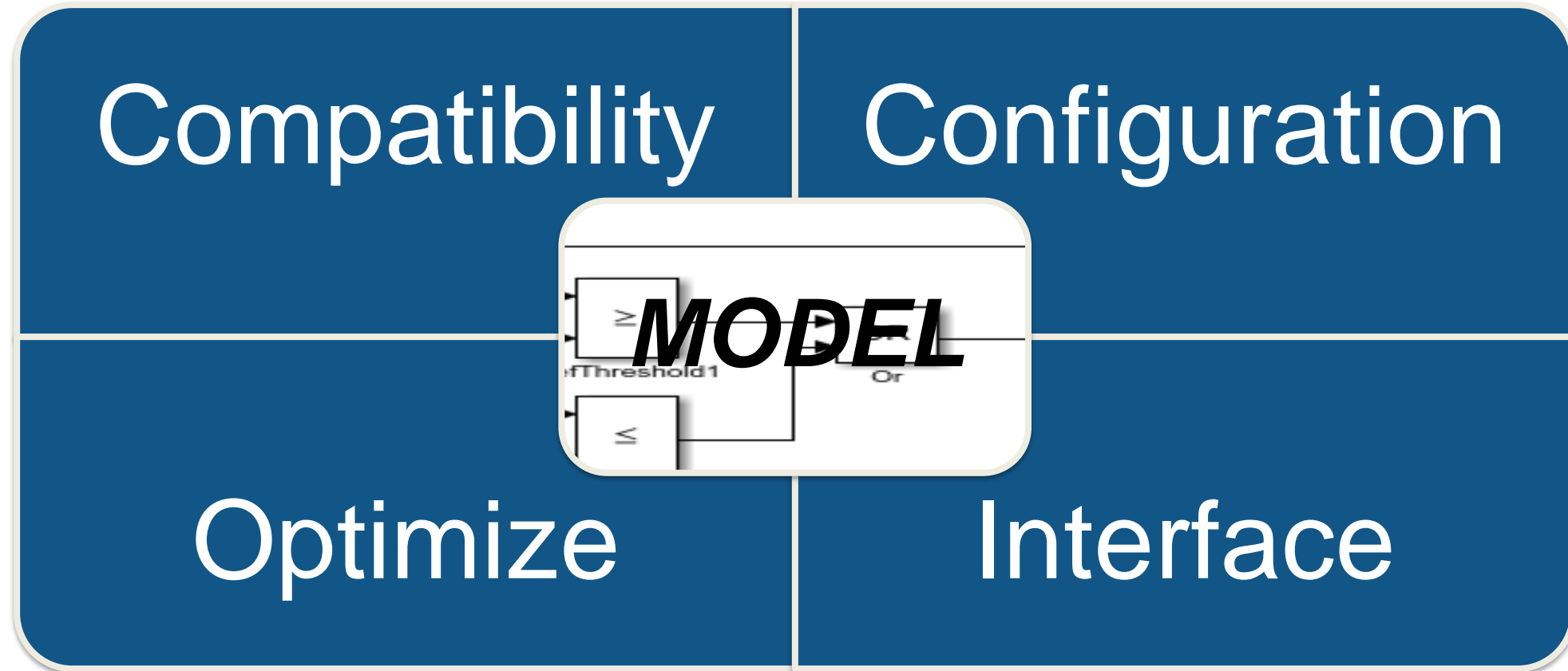
Standardize Code Architecture

Production Code Details

Improve Code Efficiency

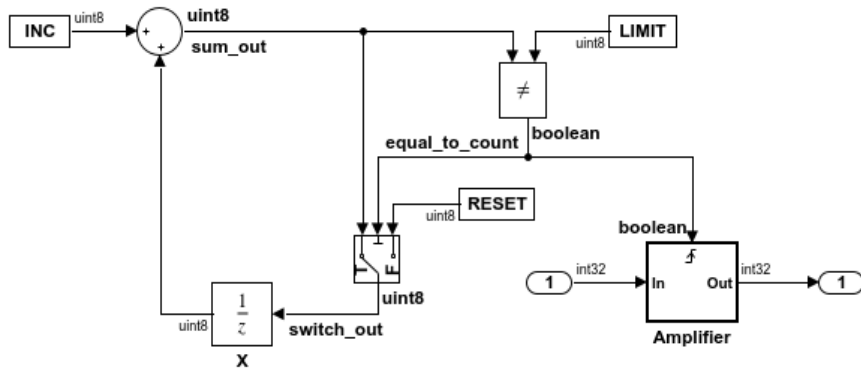
Standards & Certification

Where Should I Start From ?



Grid Locked Situation !!!

Fast Code Generation Using Quick Start



Copyright 1994-2017 The MathWorks, Inc.

SIMULINK MODEL

Contents

- [Summary](#)
- [Subsystem Report](#)
- [Code Interface Report](#)
- [Traceability Report](#)
- [Static Code Metrics Report](#)
- [Code Replacements Report](#)

Generated Code

[-] Main file

[ert_main.c](#)

[-] Model files

[Amplifier0.c](#)

[Amplifier0.h](#)

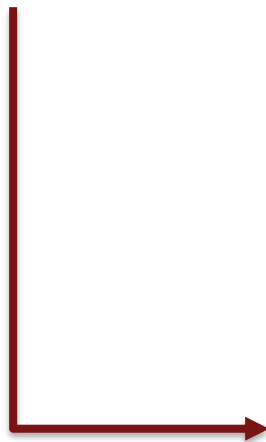
[+] Shared files (2)

```

14  */
15
16  #include "Amplifier0.h"
17
18  /* Previous zero-crossings (trigger) states */
19  PrevZCX rtPrevZCX;
20
21  /* Real-time model */
22  RT_MODEL rtM;
23  RT_MODEL *const rtM = &rtM;
24
25  /* Model step function */
26  void Amplifier0_custom(const int32_T arg_In, boolean_T arg_Trigger, int32_T
27  *arg_Out)
28  {
29  /* Outputs for Triggered SubSystem: '<Root>/Amplifier' incorporates:
30  * TriggerPort: '<S1>/Trigger'
31  */
32  /* Inport: '<Root>/Trigger' */
33  if (arg_Trigger && (rtPrevZCX.Amplifier_Trig_ZCE != POS_ZCSIG)) {
34  /* Output: '<Root>/Out' incorporates:
35  * Gain: '<S1>/Gain'
36  * Inport: '<Root>/In'
37  */
38  *arg_Out = arg_In << 1;
39  }
40
41  rtPrevZCX.Amplifier_Trig_ZCE = arg_Trigger;
42
43  /* End of Inport: '<Root>/Trigger' */

```

GENERATED CODE



Embedded Coder Quick Start: rtwdemo_counter

Welcome > System > Output > Deployment > Word Size > Optimization > Generate Code

Embedded Coder® Quick Start helps you generate production code for a Simulink model.

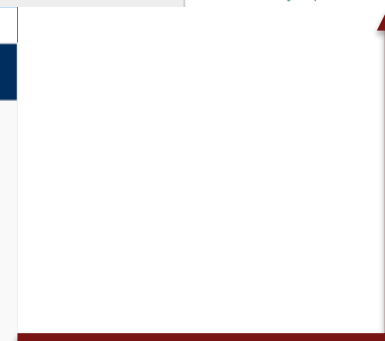
The Quick Start tool:

- Asks a few questions about your code generation goals and your target hardware.
- Validates your model against your selections.
- Shows you the recommended configuration changes.
- Applies the configuration changes and generates code.

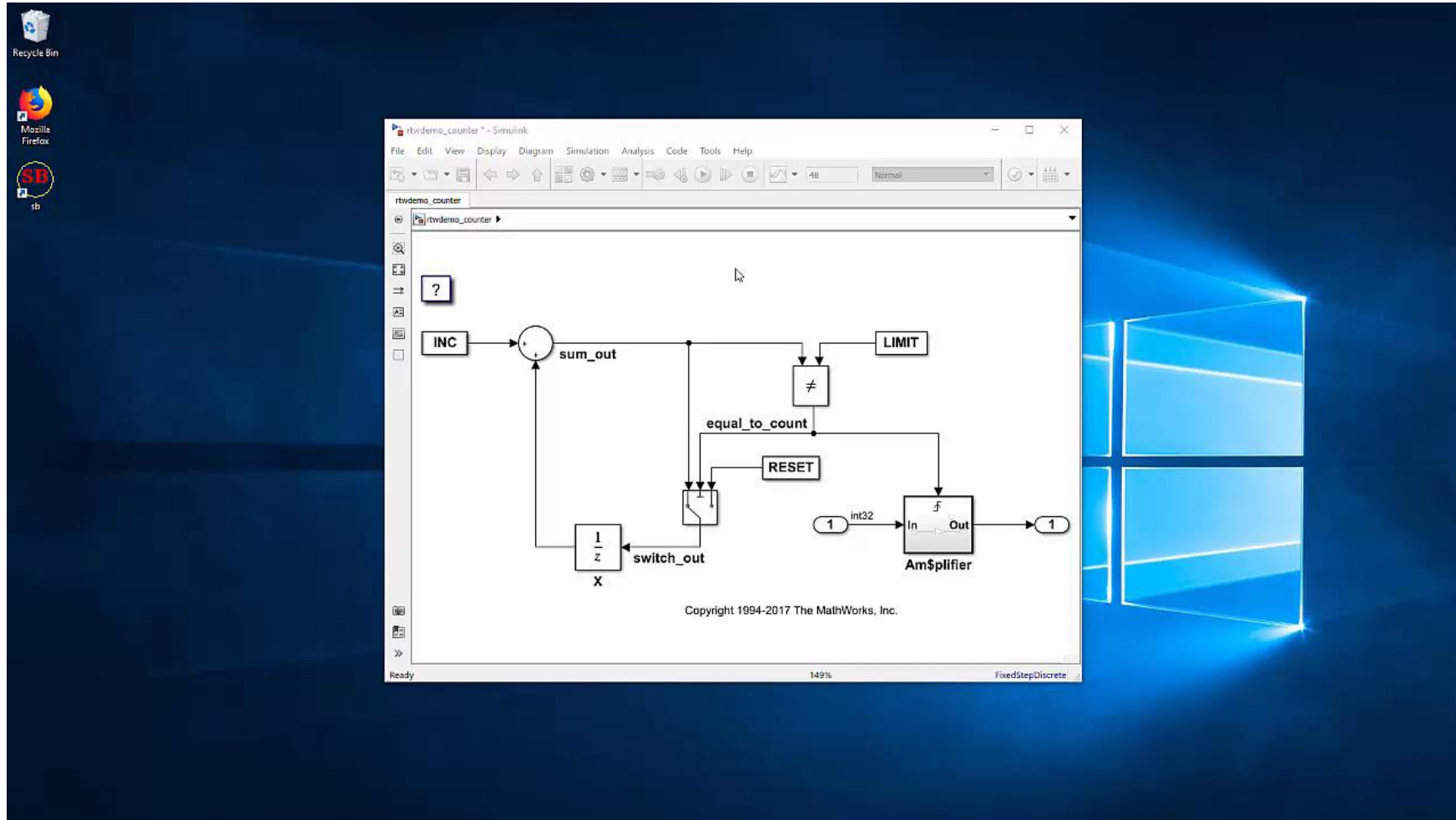
No changes are made to your configuration until you choose to generate code. After successful code generation, the Quick Start tool presents possible next steps.

QUICK START – 7 Simple Steps

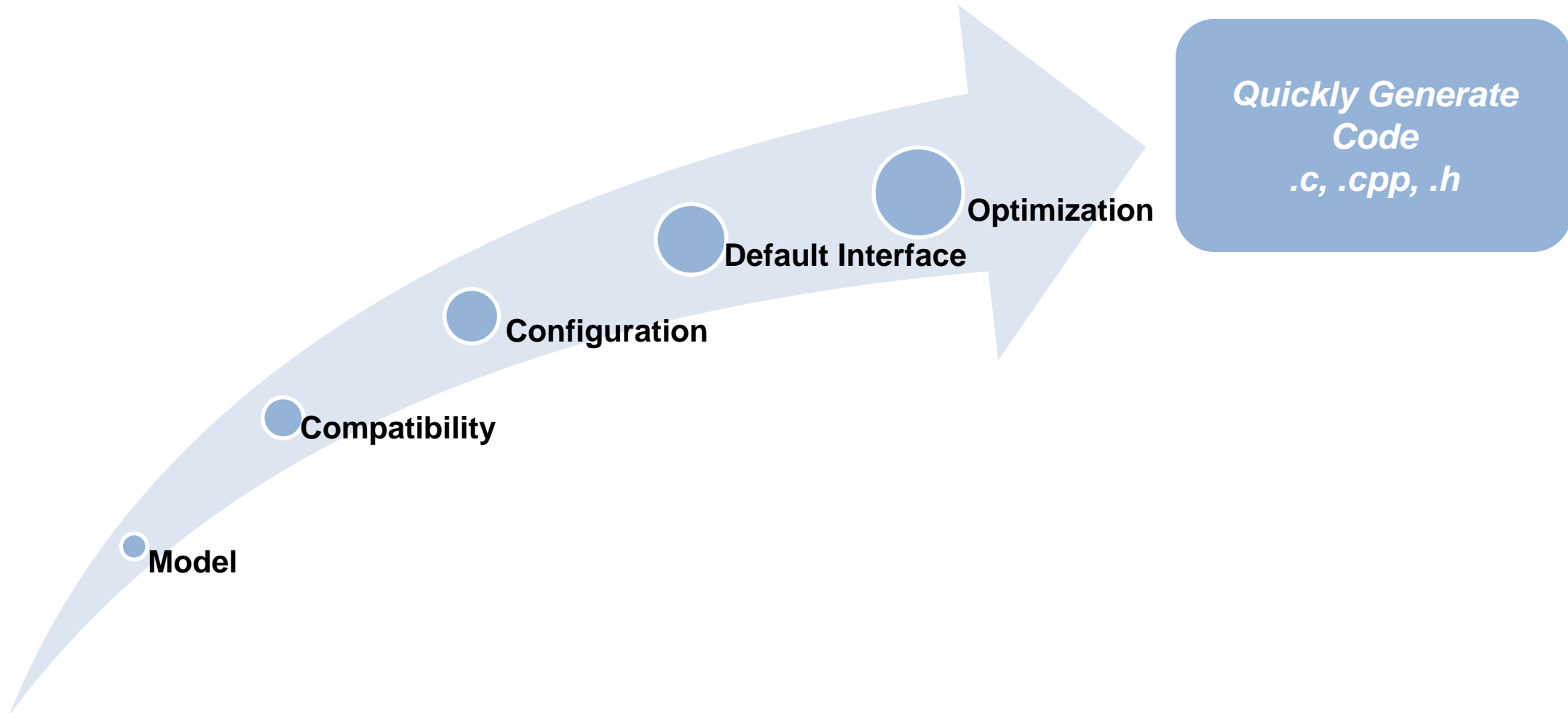
Next



7 Simple Steps for Generating Quick Code



Quick Start Wizard = True; Grid Solved = Quick Start Wizard;



Join Hands & Develop

...Collaborative Workflow...

Quick Study

Standardize Code Architecture

Production Code Details

Improve Code Efficiency

Standards & Certification

System
Architect

Software
Engineer

How to go about quickly
architecting my code ?

The “Coder Dictionary” For Your Code Architecting Needs

1. Create, Customize, Your Storage Classes

The screenshot shows the Embedded Coder Dictionary interface. The main window displays a table of storage classes with columns for Name, Storage Type, Data Scope, Header File, Definition File, Data Initialization, Memory Section, and Source. The 'MyParamStruct' entry is highlighted in blue. To the right, the 'PROPERTY INSPECTOR' panel shows the configuration for 'MyParamStruct', including Name, Description, Source, Storage Type (Structured), Data Scope (Exported), Header File (SN.h), Definition File (SN.c), Data Initialization (Static), and Memory Section (None).

Name	Storage Type	Data Scope	Header File	Definition File	Data Initializ...	Memory Section	Source
ExportedGlobal	Unstructured	Exported	---	---	Auto	None	Built-in
ImportedExtern	Unstructured	Imported	---	---	Auto	None	Built-in
ImportedExternPointer	Unstructured	Imported	---	---	Auto	None	Built-in
BitField	FlatStructure	Exported	---	---	Auto	None	Simulink package
Const	Unstructured	Exported	<Instance specific>	<Instance specific>	Auto	MemConst	Simulink package
ConstVolatile	Unstructured	Exported	<Instance specific>	<Instance specific>	Auto	MemVolatile	Simulink package
Define	Unstructured	Exported	<Instance specific>	---	Macro	None	Simulink package
ImportedDefine	Unstructured	Imported	<Instance specific>	---	Macro	None	Simulink package
ExportToFile	Unstructured	Exported	<Instance specific>	<Instance specific>	Auto	None	Simulink package
ImportFromFile	Unstructured	Imported	<Instance specific>	---	Auto	None	Simulink package
FileScope	Unstructured	File	---	---	Auto	None	Simulink package
AutoScope	Unstructured	Auto	---	---	Auto	None	Simulink package
Struct	FlatStructure	Exported	---	---	Auto	None	Simulink package
GetSet	AccessFunction	Imported	<Instance specific>	---	Auto	None	Simulink package
CompilerFlag	Unstructured	Imported	---	---	Macro	None	Simulink package
Reusable	Unstructured	<Instance specific>	<Instance specific>	<Instance specific>	Dynamic	None	Simulink package
MYStruct	Structured	Exported	SN.h	SN.c	Dynamic	None	untitled
MyParamStruct	Structured	Exported	SN.h	SN.c	Static	None	untitled

```

22 /* ConstVolatile memory section */
23 /* Definition for custom storage class: ConstVolatile */
24 const volatile real32_T TEST = 50.0F;
25

```

The “Coder Dictionary” For Your Code Architecting Needs

The screenshot displays the Embedded Coder Dictionary interface. The main window is titled "Embedded Coder Dictionary: untitled". It features a toolbar with icons for Add, Duplicate, Remove, Manage Package, and View Model. Below the toolbar are tabs for Storage Classes, Function Customization Templates, and Memory Sections. A search bar is located above a table of function templates. The table has columns for Name, Function Name, Memory Section, and Source. The row for "myTemplate" is selected, and an arrow points to the "Function Customization Templates" tab. To the right, the PROPERTY INSPECTOR shows fields for Name (myTemplate), Description (Description), Source (untitled), Function Name (MyCustom_SRSN), and Memory Section (MemorySection1). Below the table, the text "2. Customize Functions" is written in red. A code snippet is shown in a box:

```
#pragma (FAST_begin)

extern void MyCustom_untitled_step(void);

#pragma (FAST_end)
```

The “Coder Dictionary” For Your Code Architecting Needs

The screenshot shows the 'Embedded Coder Dictionary: untitled' window. The main area contains a table with columns: Name, Comment, Pre Statement, Post Statement, Statements Surround, and Source. The 'MemorySection1' row is selected. To the right, the 'PROPERTY INSPECTOR' shows the configuration for 'MemorySection1', with a callout box highlighting the 'Pre Statement', 'Post Statement', and 'Statements Surround' fields.

Name	Comment	Pre Statement	Post Statement	Statements Surround	Source
MemConst	/* Const memory section */				Simulink package
MemVolatile	/* Volatile memory section */				Simulink package
MemConstVolatile	/* ConstVolatile memory section */				Simulink package
MemorySection1		#pragma (FAST_begin)	#pragma (FAST_end)	Each variable	untitled

PROPERTY INSPECTOR

Name: MemorySection1

Description: Description

Source: untitled

Comment:

Pre Statement: #pragma (FAST_begin)

Post Statement: #pragma (FAST_end)

Statements Surround: Each variable

**3. Create, Customize
Memory Sections**

Easily configure your **Model** with Code Mapping Editor

The screenshot shows the Simulink Embedded Coder environment. The main window displays a block diagram of a counter model with blocks for 'INC', 'sum_out', 'LIMIT', 'equal_to_count', 'RESET', 'switch_out', and 'Amplifier'. A red circle highlights the 'Code Mappings - C' window, which is open to the 'Data Defaults' tab. This window contains a table for defining storage classes for model elements.

Model Element Category	Storage Class
Inports	Default
Outports	Default
Global parameters	Default
Local parameters	Default
Shared local data stores	Default
Global data stores	Default
Internal data	Default
Constants	Default



```
typedef struct {
    uint8_T X_Delay;
} DW;

typedef struct {
    ZCSigState Amplifier_Trig_ZCE;
} PrevZCX;

typedef struct {
    int32_T Input;
} ExtU;

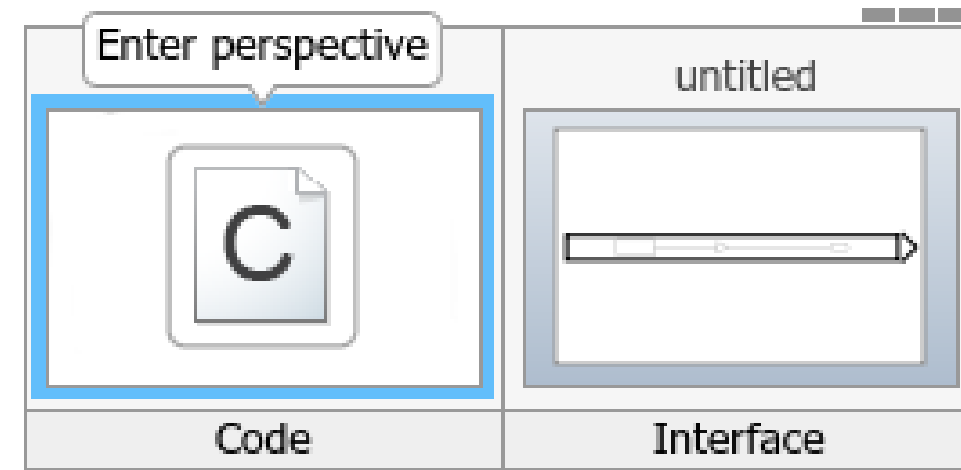
typedef struct {
    int32_T Output;
} ExtY;
```

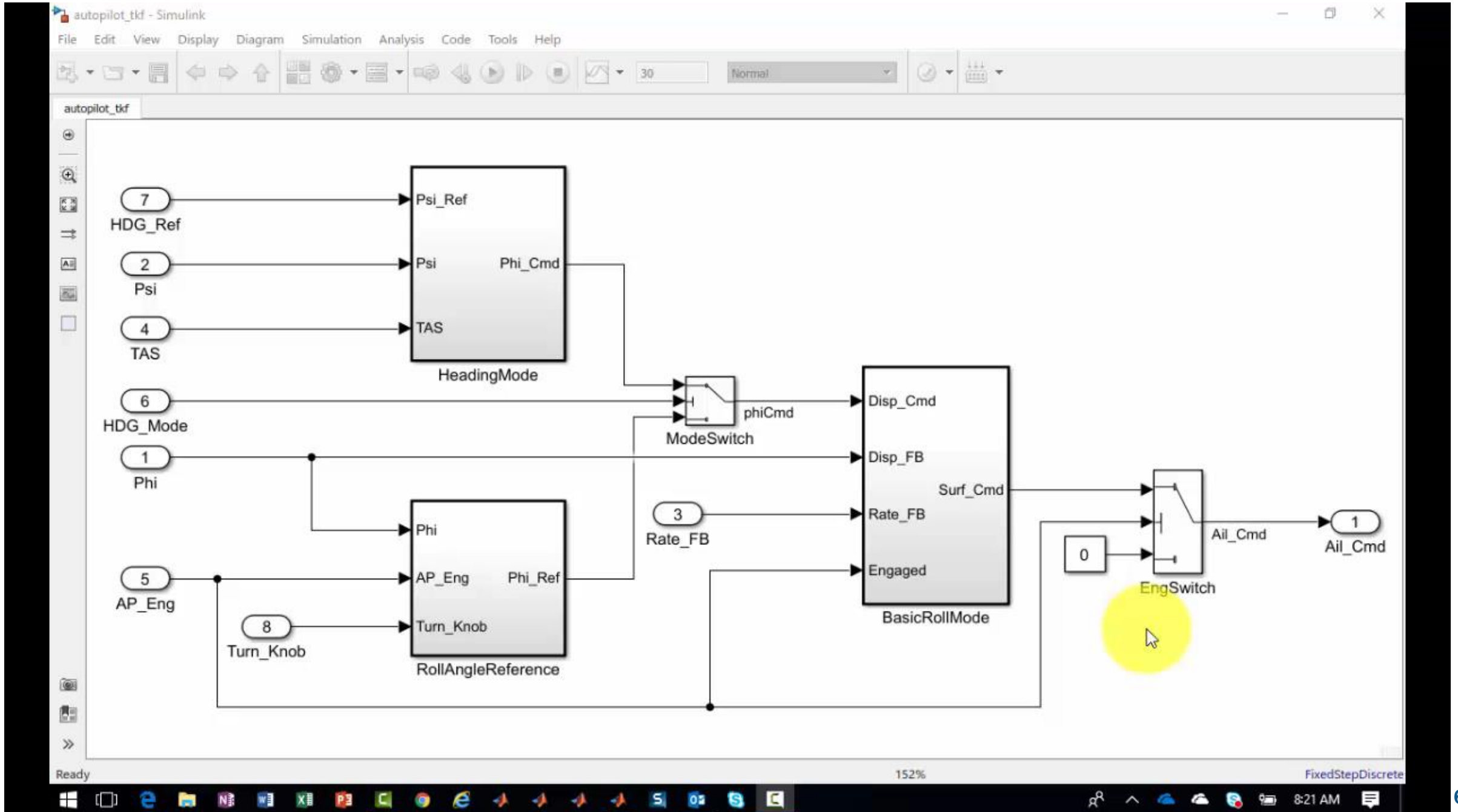
rtwdemo_counter.h

```
DW rtDW;
PrevZCX rtPrevZCX;
ExtU rtU;
ExtY rtY;
```

rtwdemo_counter.c

CoderPerspective
=
{'QuickStart', 'CoderDictionary', 'CodeMapping'}





Join Hands & Develop

...”standardization *by **customizing** the Simulink® development environment*”...

Configure default settings for model.

Apply Embedded Coder Dictionary

code definitions with:

- **Code Mapping Editor**
- **Model Data Editor**
- **Function Prototype Control**

Creates code definition for Data
& Functions with:

- **Embedded Coder Dictionary**
- **Custom Storage Class Designer**

**Software
Architect**

**Software
Engineer**

Join Hands & Develop

Software
Architect

Software
Engineer

...Collaborative Workflow...

Quick Study

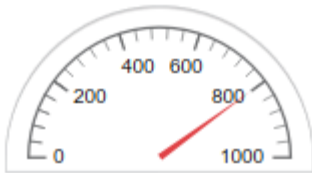
Standardize Code Architecture

Production Code Details

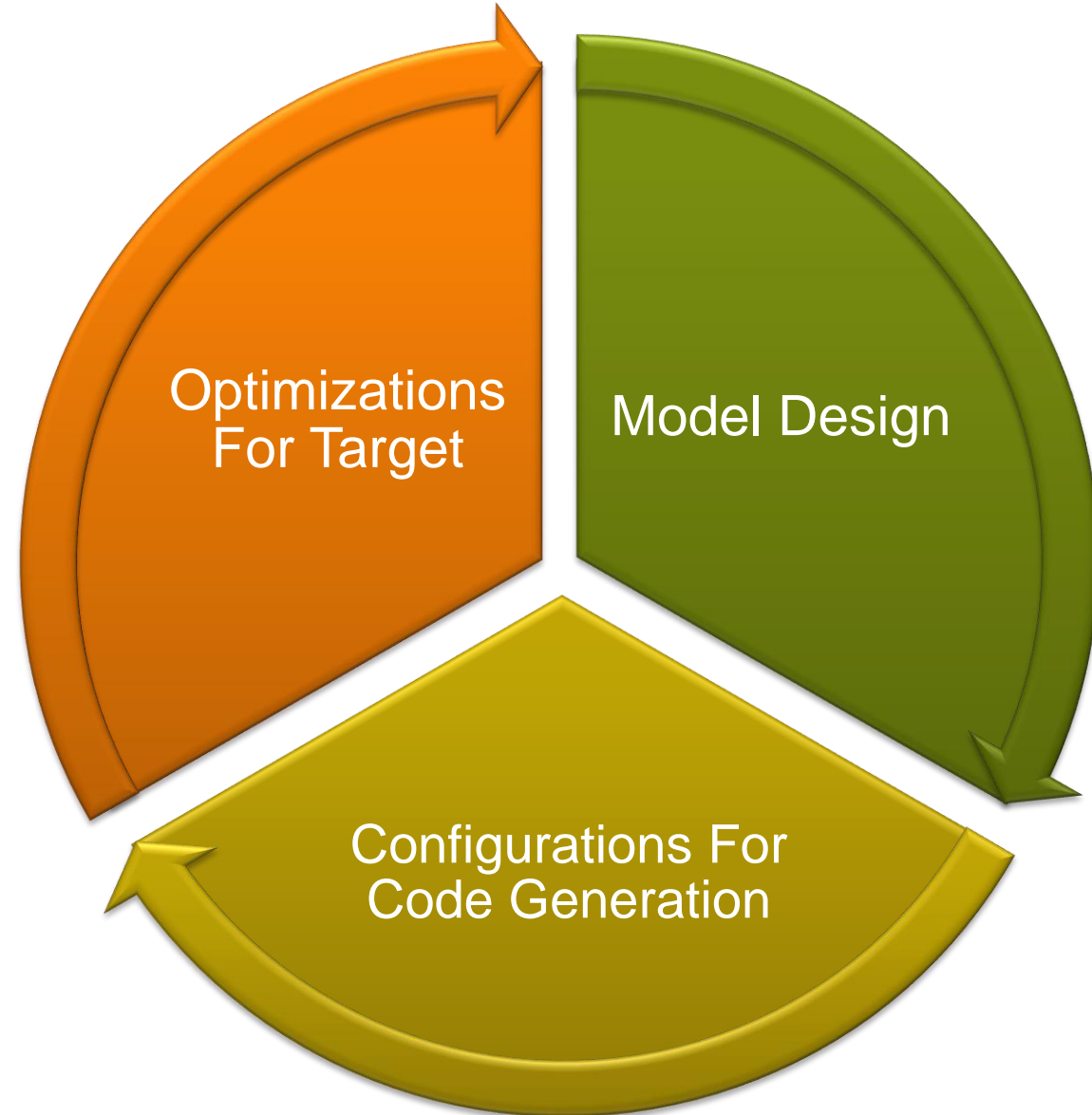
Improve Code Efficiency

Standards & Certification

How and What Should I Optimize?



Execution Speed



Achieve Optimized Code Via Design

Design using requirements & simulate the model to observe behavior

2 Functional Requirements

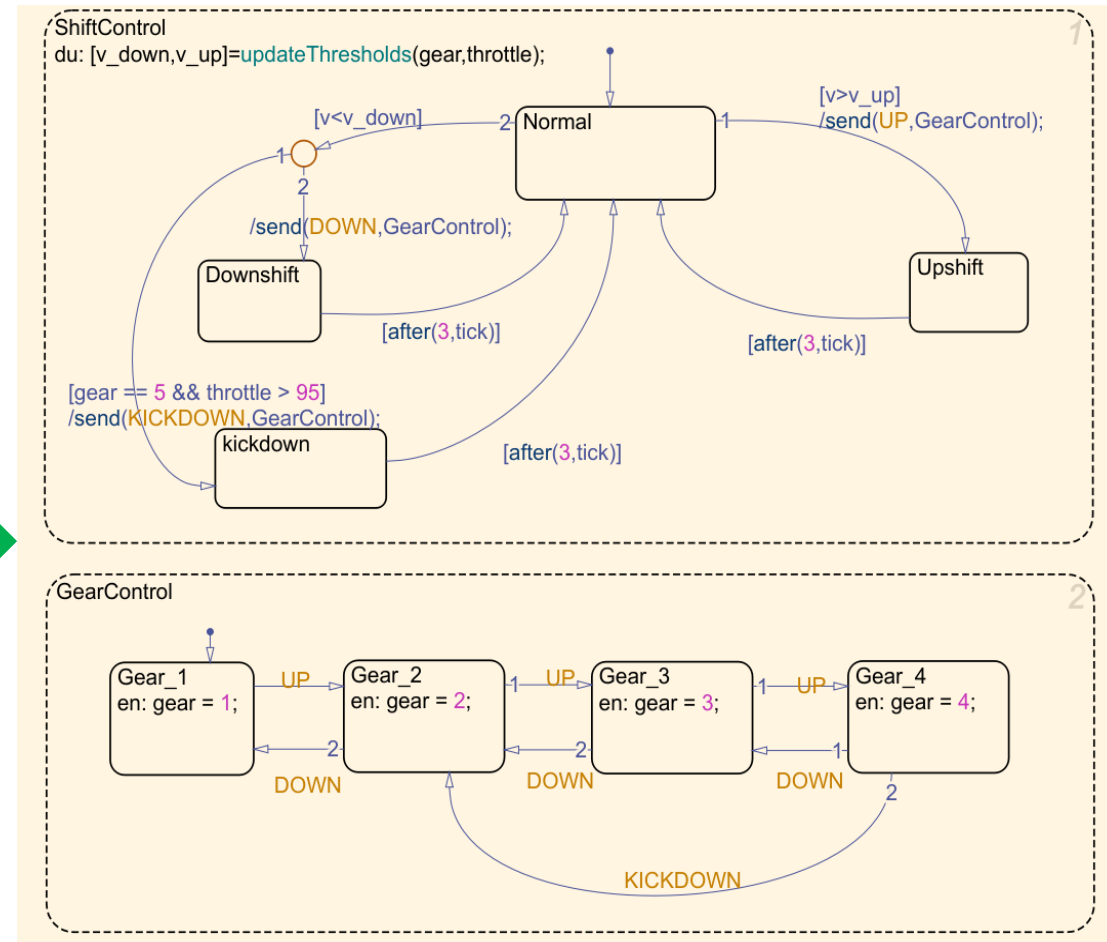
2.1 Function Description
Automatic gear selection algorithm calculates transmission gear based on vehicle speed and throttle pedal sensor input value.

2.2 Gear Shift Algorithm
Gear shift algorithm obtains upper threshold and down threshold values from the threshold calculation algorithm and computes desired gear number.

2.2.1 Gear State
This state contains and outputs desired gear number. Four gears are available and four different states represent them.

2.2.2 Selection State
Based on the threshold value and current gear number gear's up-shift, down-shift and steady state decision is made.

2.3 Threshold Calculation Algorithm
Two threshold values are calculated based on current gear value and throttle value. Upper threshold (up_th) is calculated via a interpUp-table with pre-calibrated data. Down threshold (down_th) is also calculated via interpDn-table with pre-calibrated data.



Achieve Optimized Code Via Design

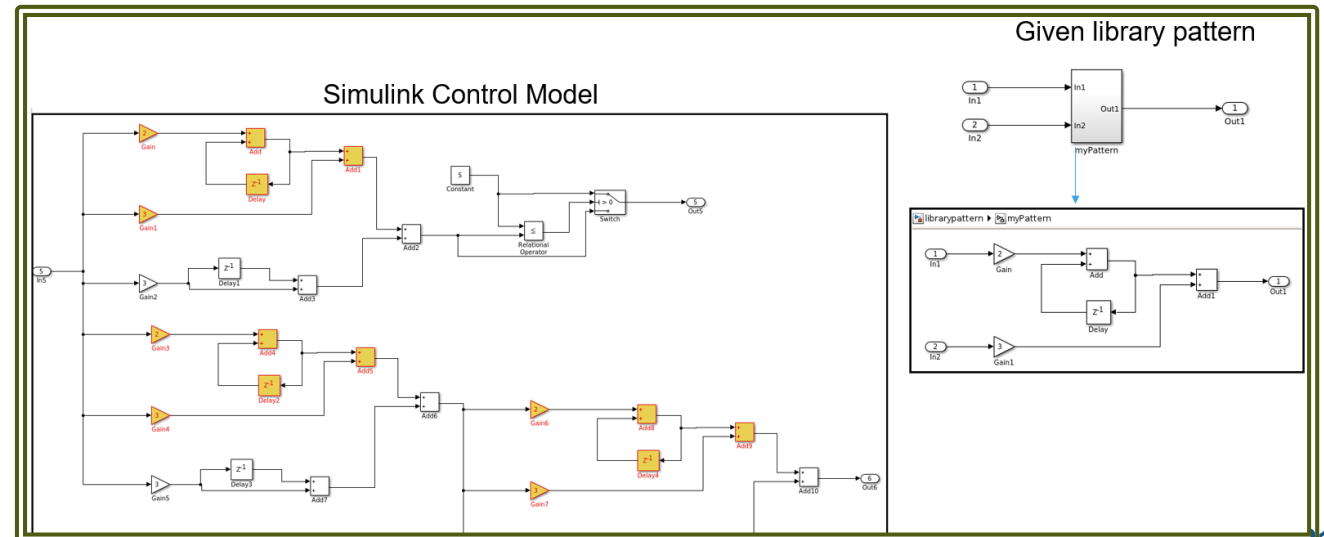
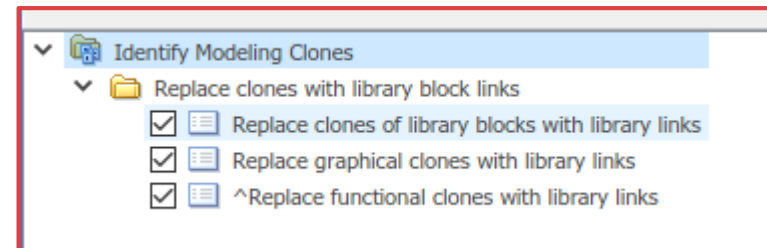
Use Model Advisor to apply and establish best Modeling Practices

- MAAB, MISRA, ISO/IEC/DO guidelines, etc.
- Simulink and Stateflow Guidelines
- Clone Detection

Clone Detection

Model Advisor Guidelines

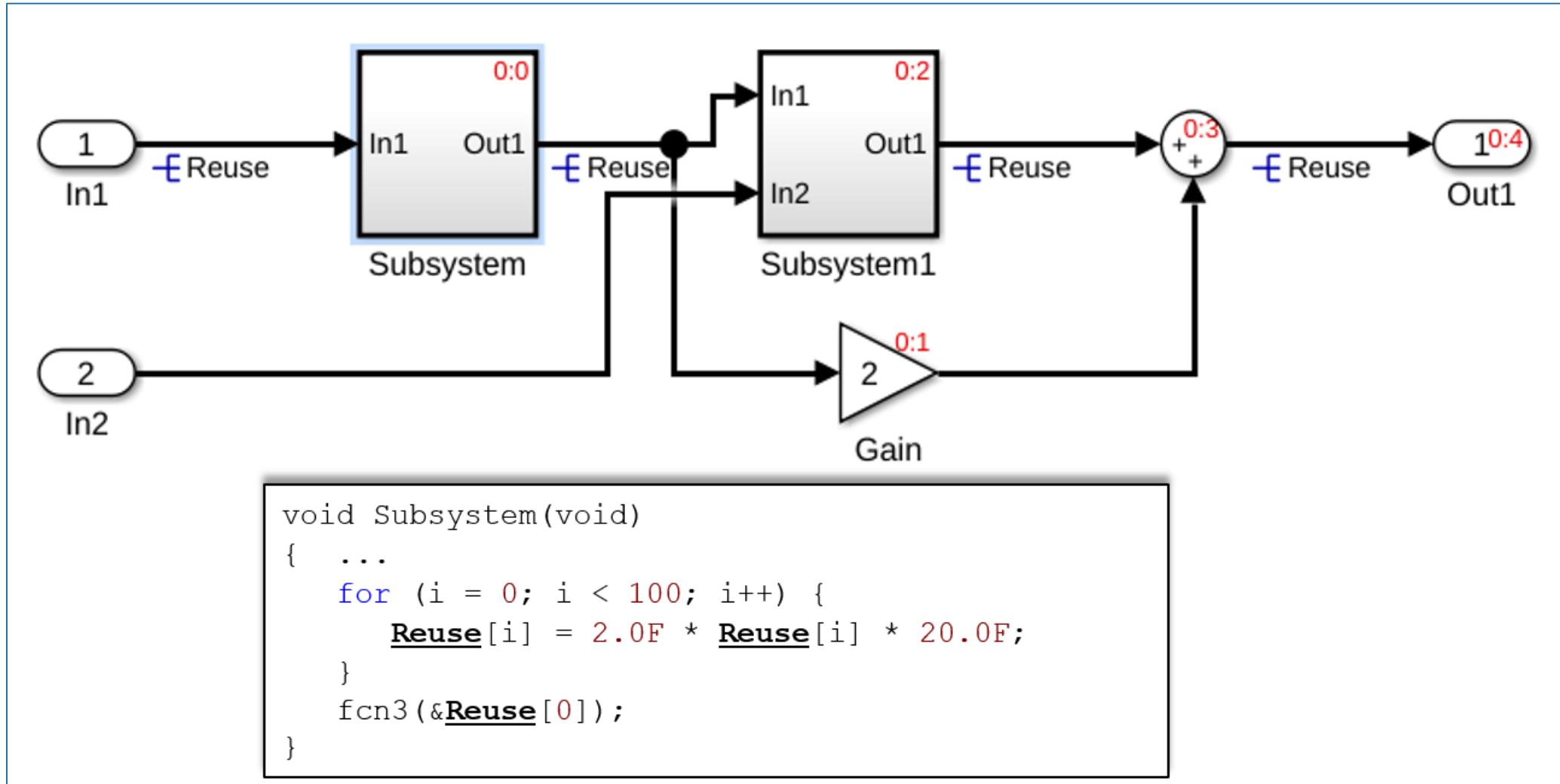
- > Modeling Standards for DO-178C/DO-331
- > Modeling Standards for EN 50128
- > Modeling Standards for IEC 61508
- > Modeling Standards for IEC 62304
- > Modeling Standards for ISO 26262
- > Modeling Standards for MAAB
- > Modeling Standards for JMAAB
- > Modeling Standards for MISRA C:2012



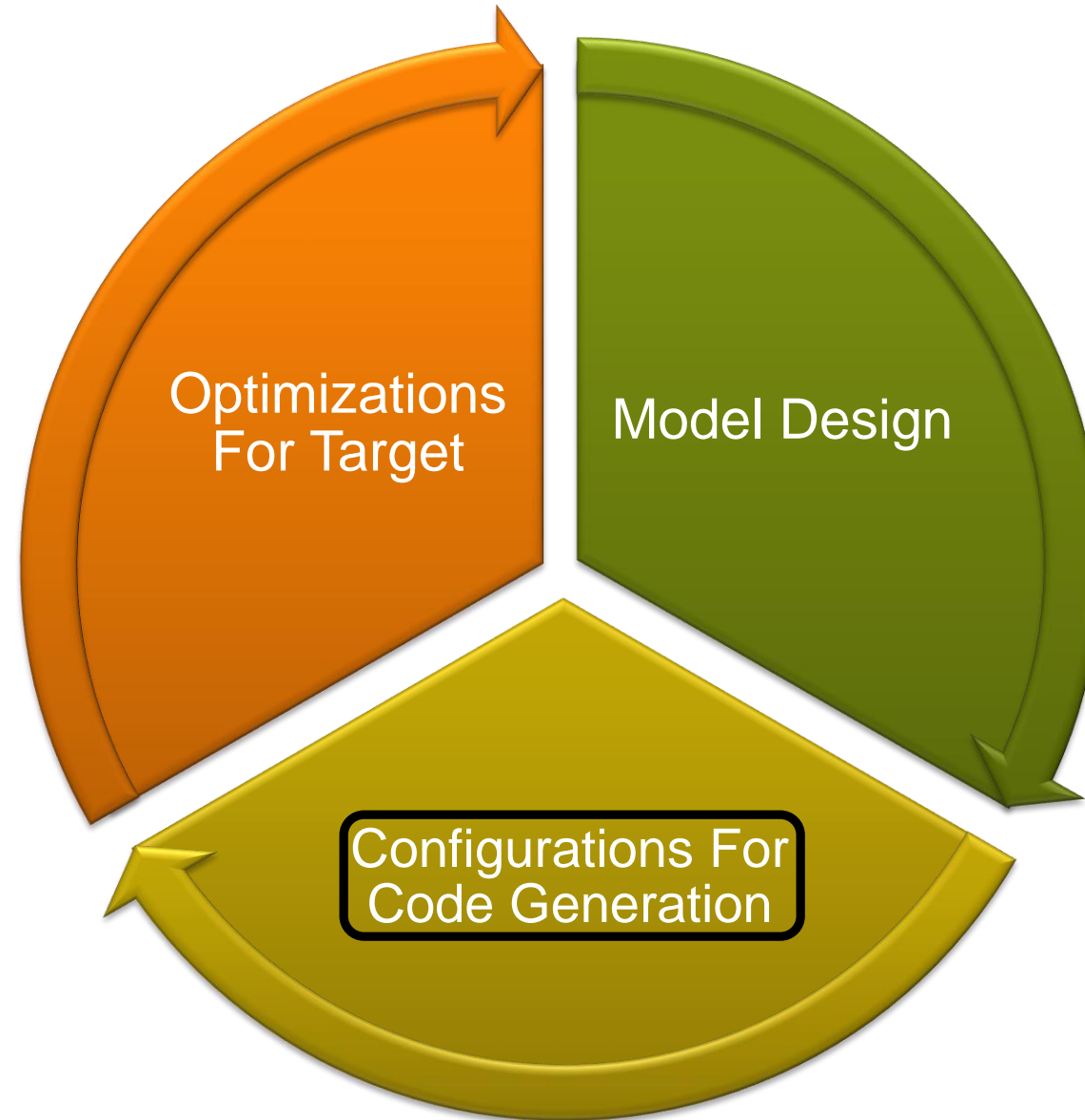
Achieve Optimized Code Via Design

Use Simulink Data Objects and Custom Storage Classes to optimize

- E.g. Reusable (Data copy reduction), Localizable (Force local vs global use)



How and What Should I Optimize?



Easy Optimization Levels and Priorities

Optimization levels

Level: Maximum ▼ Priority: Balance RAM and speed ▼

Specify custom optimizations

▶ Details

- Levels:

- Minimum
- Balanced with readability
- Maximum

- Priorities

- Balance RAM and speed
- Maximize execution speed
- Minimize RAM

Advanced Customization

- Select individual optimizations as desired
- Preserves existing setting from previous versions

Optimization levels

Level: Priority:

Specify custom optimizations

▼ Details

Use memcpy for vector assignment Memcpy threshold (bytes):

Loop unrolling threshold:

Enable local block outputs

Reuse local block outputs

Eliminate superfluous local variables (expression folding)

Reuse global block outputs

Perform inplace updates for Bus Assignment blocks

Reuse buffers for Data Store Read and Data Store Write blocks

Simplify array indexing

Pack Boolean data into bitfields

Reuse buffers of different sizes and dimensions

Optimize global data access:

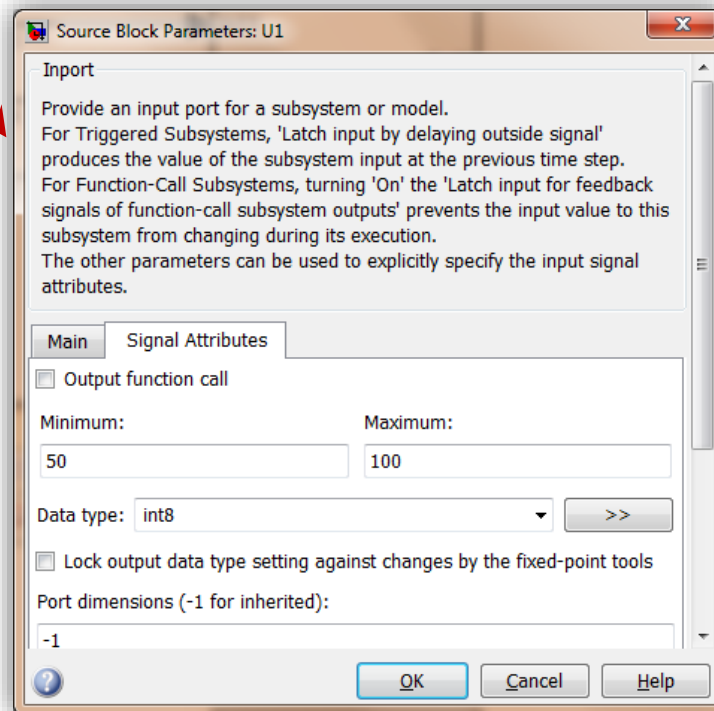
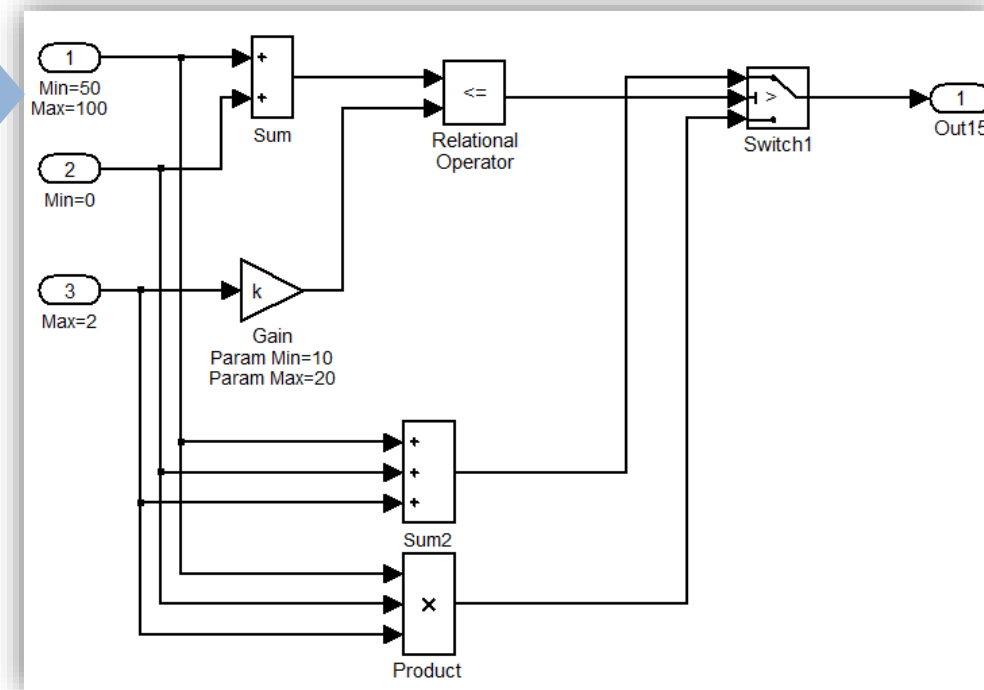
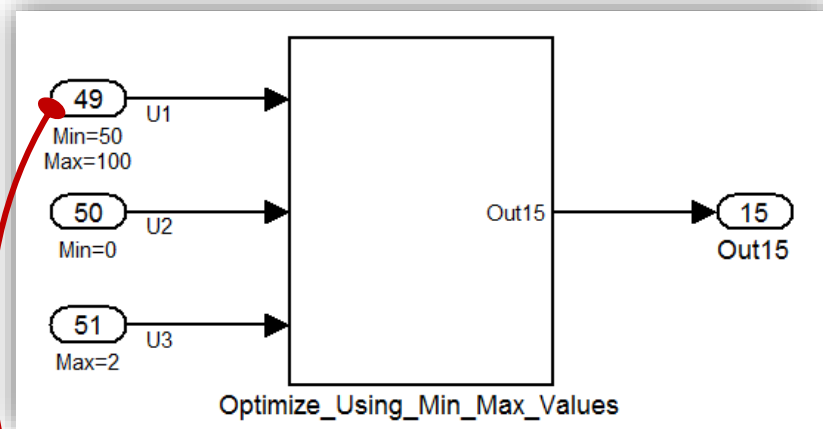
Optimize block operation order in the generated code:

Stateflow

Use bitsets for storing state configuration

Use bitsets for storing Boolean data

Optimize Using Min & Max Values



Optimize Using Min & Max Values

Code generation

Optimize using the specified minimum and maximum values

```

rtb_Sum = U1 + U2;

/* Gain: '<S8>/Gain' incorporates:
 * Inport: '<Root>/U3'
 */
rtb_Sum2 = Code_Optimization_P.Gain_Gain * U3;

/* RelationalOperator: '<S8>/Relational Operator' */
rtb_RelationalOperator = (rtb_Sum <= rtb_Sum2);

/* Switch: '<S8>/Switch1' */
if (rtb_RelationalOperator) {
  /* Sum: '<S8>/Sum2' incorporates:
   * Inport: '<Root>/U1'
   * Inport: '<Root>/U2'
   * Inport: '<Root>/U3'
   */
  rtb_Sum2 = (U1 + U2) + U3;
} else {
  /* Product: '<S8>/Product' incorporates:
   * Inport: '<Root>/U1'
   * Inport: '<Root>/U2'
   * Inport: '<Root>/U3'
   */
  rtb_Sum2 = U1 * U2 * U3;
}
  
```

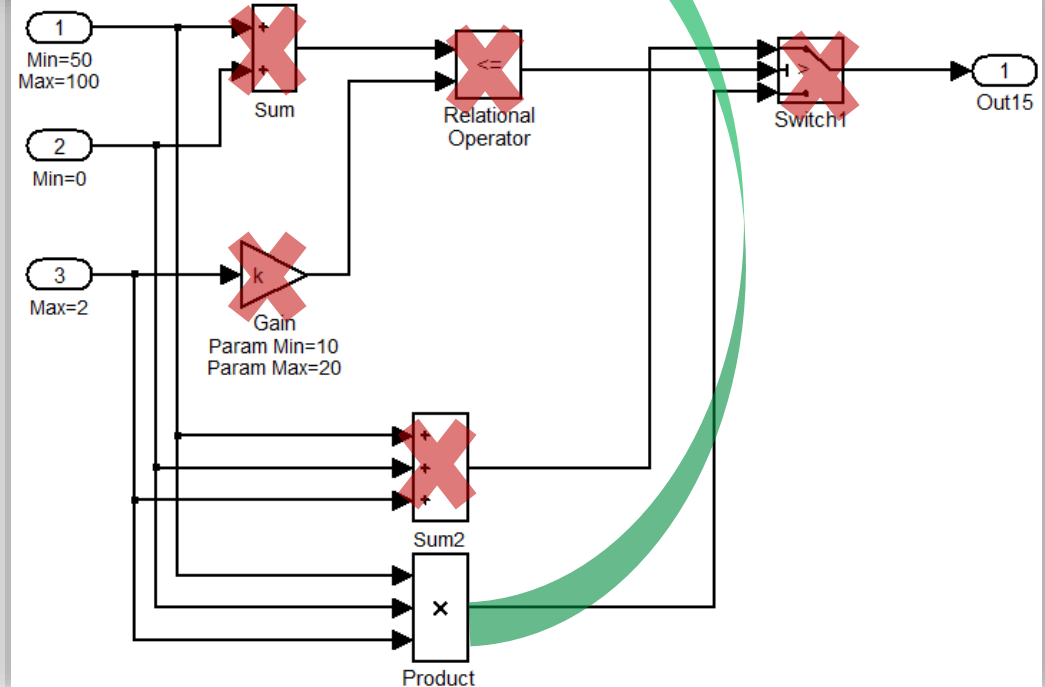
Code generation

Optimize using the specified minimum and maximum values

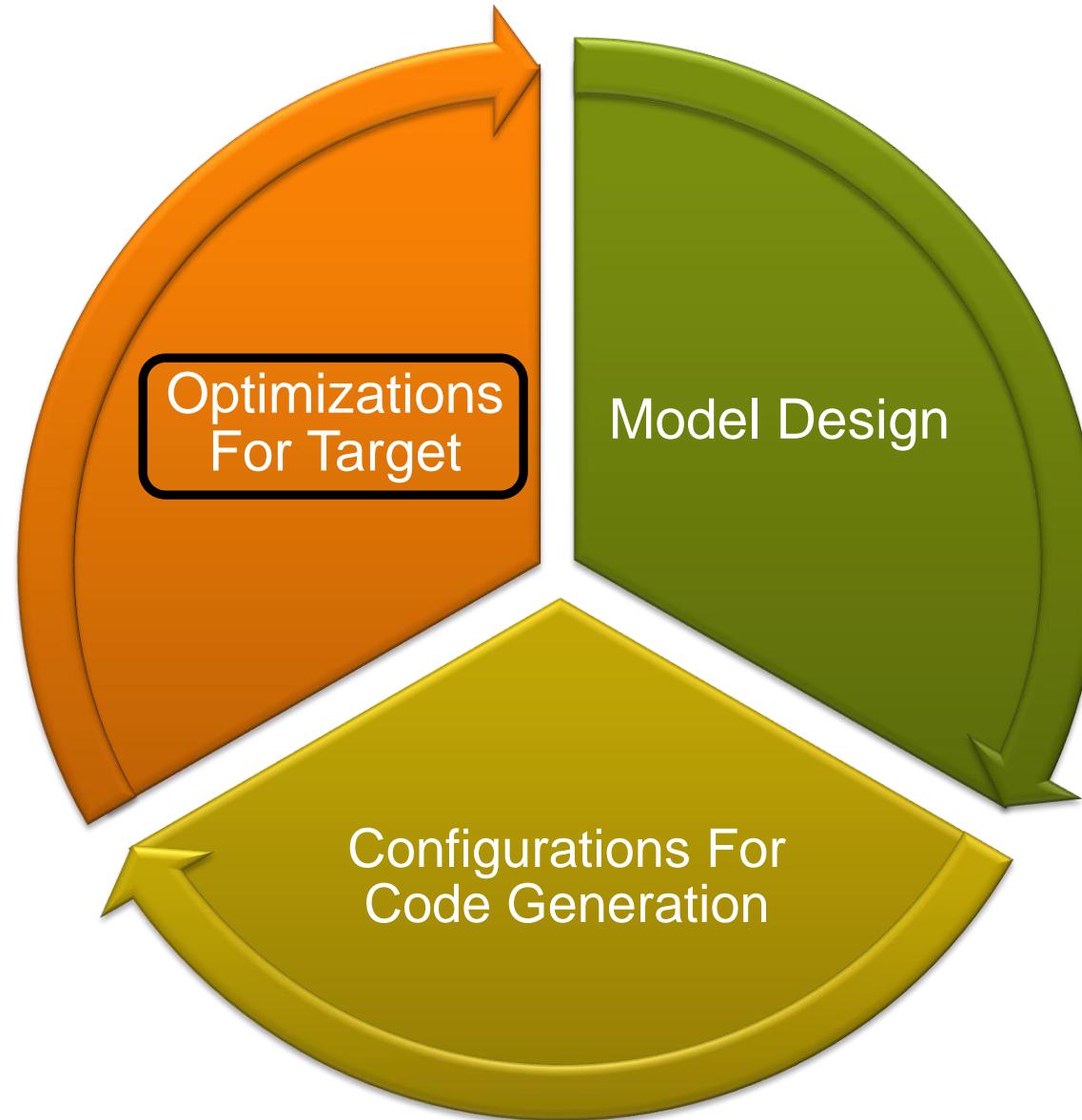
```

/* Product: '<S8>/Product' incorporates:
 * Inport: '<Root>/U1'
 * Inport: '<Root>/U2'
 * Inport: '<Root>/U3'
 * Switch: '<S8>/Switch1'
 */
rtb_Sum2 = U1 * U2 * U3;

/* Outport: '<Root>/Out15' */
Code_Optimization_Y.Out15 = rtb_Sum2;
  
```



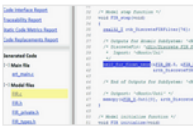
How and What Should I Optimize?




Target Based Optimizations

- Hardware Support Packages
- Code Replacement Libraries for Custom libraries eg.
 - ARM Cortex A Ne10
 - Intel SSE, AVX
 - ARM Cortex M CMSIS
- S-Functions for legacy code
- Organization wide Custom Libraries via Code Replacement Libraries

Results 1 - 18 of 18 >





ARM Cortex A Ne10 Library Support from DSP System Toolbox 

Optimized C code generation from MATLAB or Simulink for ARM

Vendors: ARM

Tags: Support Package Installer Enabled, C/C++ Code Generation, MathWorks Supported





ARM Cortex A Support from Embedded Coder 

Generate code optimized for Cortex A processors.

Vendors: ARM

Tags: Support Package Installer Enabled, C/C++ Code Generation, MathWorks Supported





ARM Cortex-M CMSIS Library Support from DSP System Toolbox 

Optimized C code generation from MATLAB or Simulink for ARM

Vendors: STMicroelectronics, ARM

Tags: Support Package Installer Enabled, C/C++ Code Generation, MathWorks Supported





ARM Cortex-M Support from Embedded Coder 

Generate code optimized for Cortex-M processors.

Vendors: STMicroelectronics, ARM

Tags: Support Package Installer Enabled, C/C++ Code Generation, MathWorks Supported



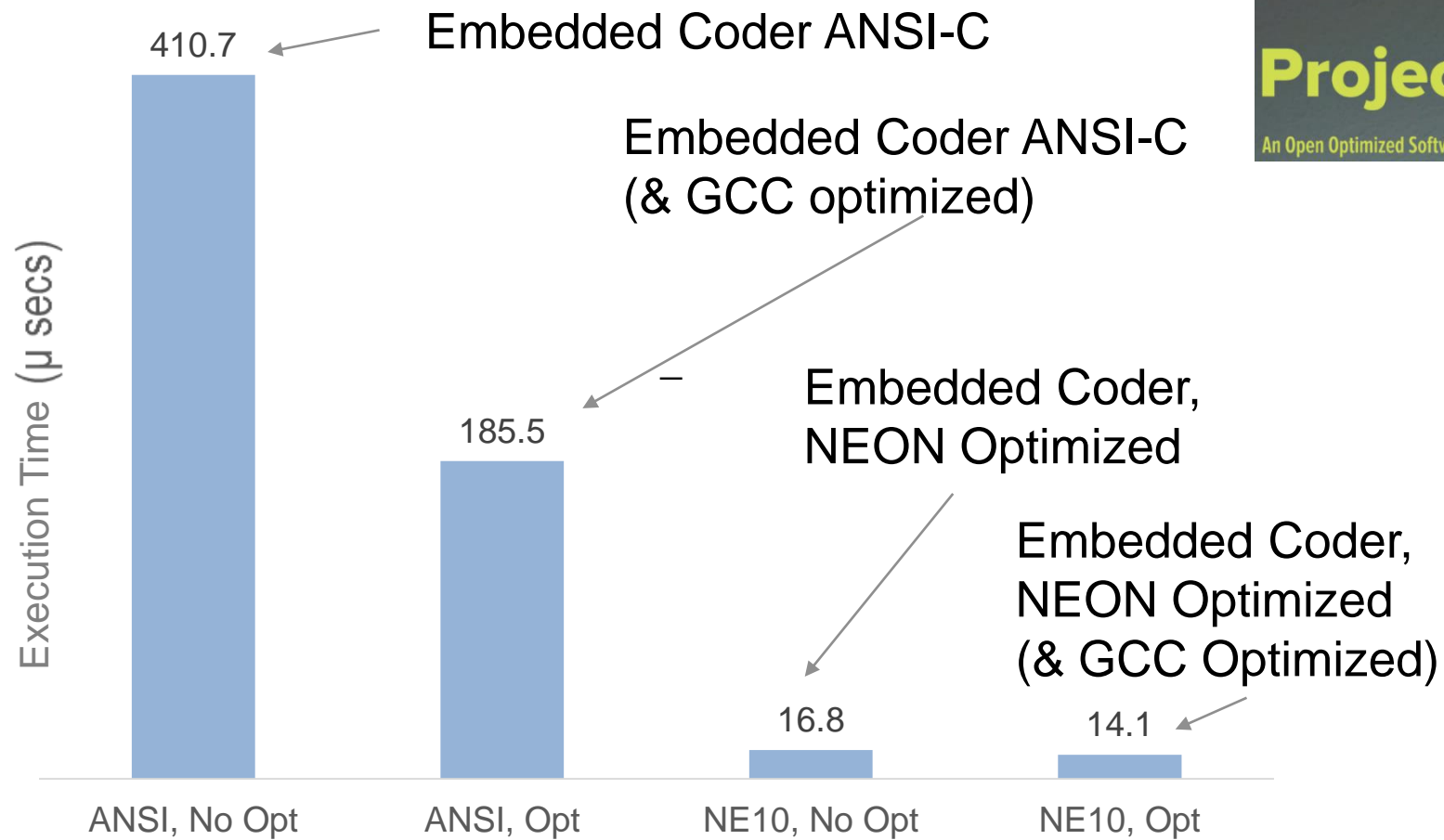
ARM Cortex-R Support from Embedded Coder 

Generate code optimized for Arm Cortex-R processors

Vendors: TTI, ARM

Tags: Support Package Installer Enabled, C/C++ Code Generation, MathWorks Supported

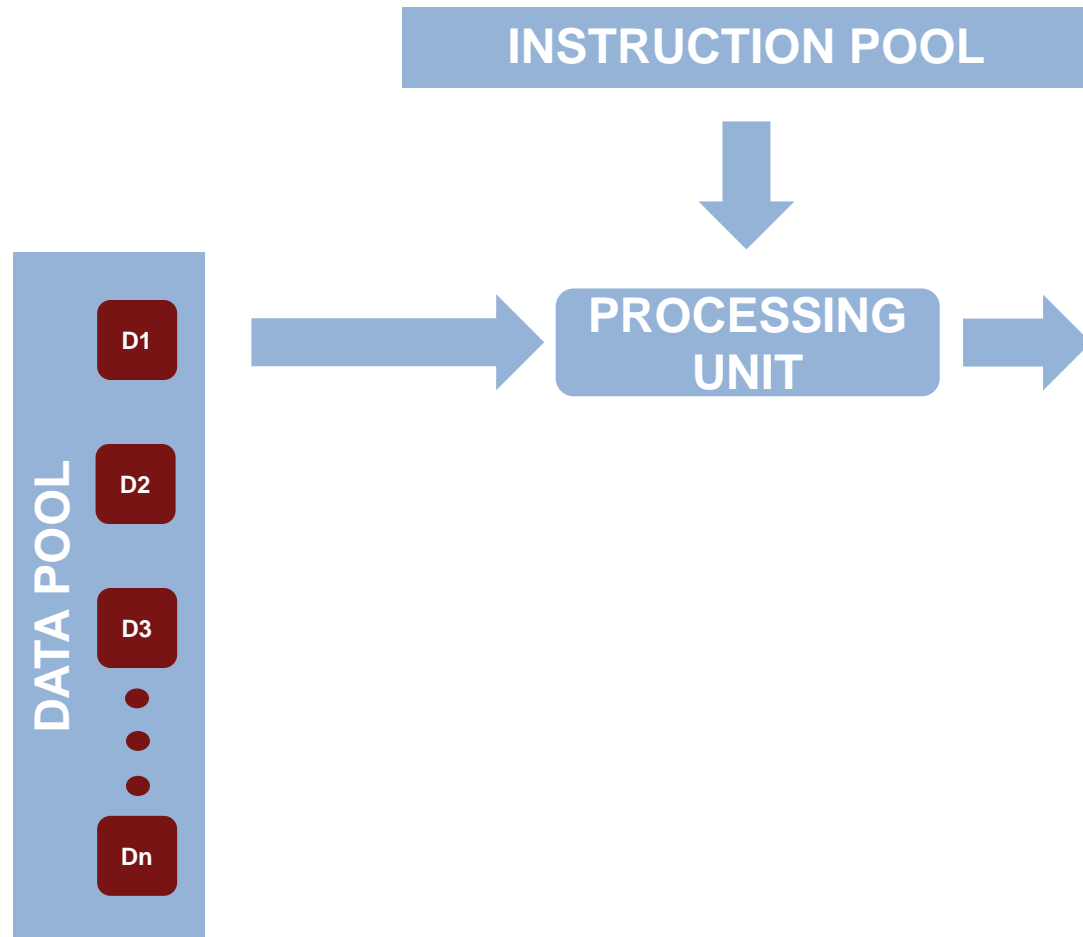
Auto Code Performance ARM Cortex-A



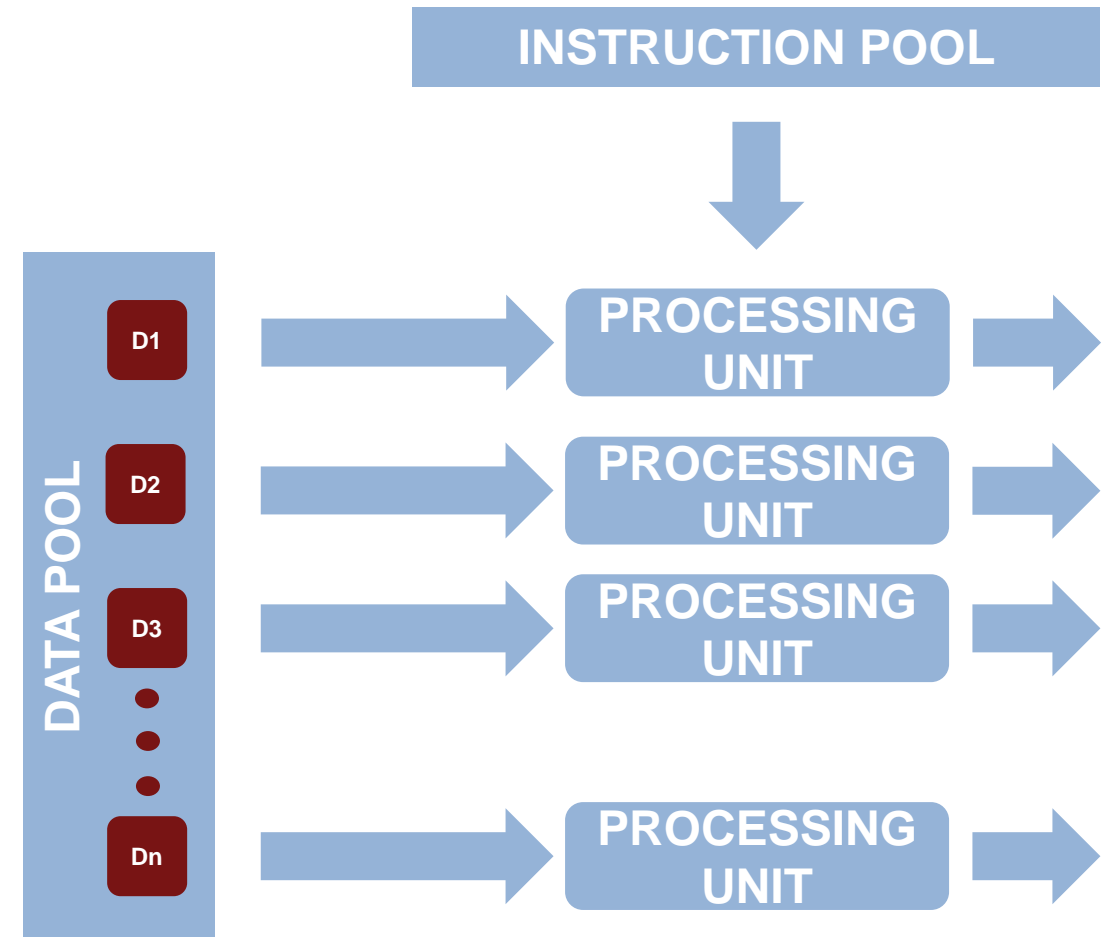
Run Format: [ANSI or Ne10], [gcc no opt or gcc -O2], ARM 1Ghz Cortex A8

What is SIMD?

SISD - **S**ingle **I**nstruction **S**ingle **D**ata



SIMD - **S**ingle **I**nstruction **M**ultiple **D**ata



ARM Cortex-A Optimized Code

Software environment

Code replacement library: ARM Cortex-A

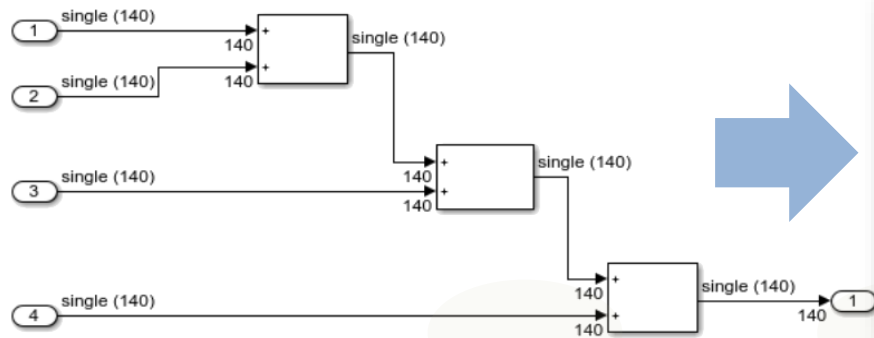
non-finite numbers
 continuous time

able function ▾

```

198 /* DiscreteFir: '<S1>/Discrete FIR Filter' */
199 ne10_fir_float_neon(&ex_fir_ne10_tut_DW.S, &rt
  
```


SIMD for Intel SSE/AVX using Code Replacement Library



Software environment

Code replacement library: Intel SSE (Linux)

Shared code placement: None

Support non-finite num

Code interface

Code interface packaging: Intel AVX (Linux), XPC_BLAS

- None
- GNU C99 extensions
- AUTOSAR 4.0
- Intel SSE (Windows)
- Intel AVX (Windows)
- Intel SSE (Linux)**
- Intel AVX (Linux)
- XPC_BLAS

```
void simd_model_step(void)
{
    __attribute__((aligned(16))) real32_T rtb_Add[140];
    __attribute__((aligned(16))) real32_T rtb_Add1[140];
    mw_gcc_sse_mm_add_f32x4(simd_model_U.In1, 140, 1, simd_model_U.In2, rtb_Add);
    mw_gcc_sse_mm_add_f32x4(rtb_Add, 140, 1, simd_model_U.In3, rtb_Add1);
    mw_gcc_sse_mm_add_f32x4(rtb_Add1, 140, 1, simd_model_U.In4, simd_model_Y.Out1);
}
```

```
void mw_gcc_sse_mm_add_f32x4(const float * A, int Row, int Col, const float * B, float * C)
```

Wrapper function

```

{
    __m128 sse_a, sse_b, sse_c;
    int size = Row*Col;
    int i;
    int k=0;
    for (i = 0; i < size ; i+=4)
    {
        sse_a = _mm_load_ps(A+i);
        sse_b = _mm_load_ps(B+i);
        sse_c = _mm_add_ps(sse_a, sse_b);
        _mm_store_ps(C+i, sse_c);
    }
    k=i+4;
    for (i = 0; i < size%4 ; i++)
    {
        C[k+i] = A[k+i]+B[k+i];
    }
}
```

Redundant Loops

```
void simd_model_step(void)
{
    int32_T idx;
    __m128 tmp;
    __m128 tmp_0;
    for (idx = 0; idx <= 136; idx += 4) {
        tmp = _mm_load_ps(&simd_model_U.In1[idx]);
        tmp_0 = _mm_load_ps(&simd_model_U.In2[idx]);
        tmp = _mm_add_ps(tmp, tmp_0);
        tmp_0 = _mm_load_ps(&simd_model_U.In3[idx]);
        tmp_0 = _mm_add_ps(tmp, tmp_0);
        tmp = _mm_load_ps(&simd_model_U.In4[idx]);
        tmp = _mm_add_ps(tmp_0, tmp);
        _mm_store_ps(&simd_model_Y.Out1[idx], tmp);
    }
}
```

- Built on enhanced Code Replacement infrastructure
- No wrapper functions, fused loops, minimized load and stores

Join Hands & Develop

...”standardization *by **customizing** the Simulink® development environment*”...

Defines Architectural Details and Usage of Standard Libraries via:

- **Embedded Coder Dictionary**
- **Custom Storage Class Designer**
- **Modeling and Code Generation Best Practices**
- **Custom Replacement Libraries**
- **Target Libraries**

Software Architect

Software Engineer

Configure default settings for model. Applies Software definitions, optimizations and performs verification using:

- **Code Mapping Editor**
- **Model Data Editor**
- **Function Prototype Control**
- **Model Advisor**
- **Model Optimization Configurations**
- **Simulink Data Object & Storage Classes**

Join Hands & Develop

Software
Architect

Software
Engineer

...Collaborative Workflow...

Quick Study

Standardize Code Architecture

Production Code Details

Improve Code Efficiency

Standards & Certification

MathWorks approach for AUTOSAR

Authoring Tools



Software Architecture Definition

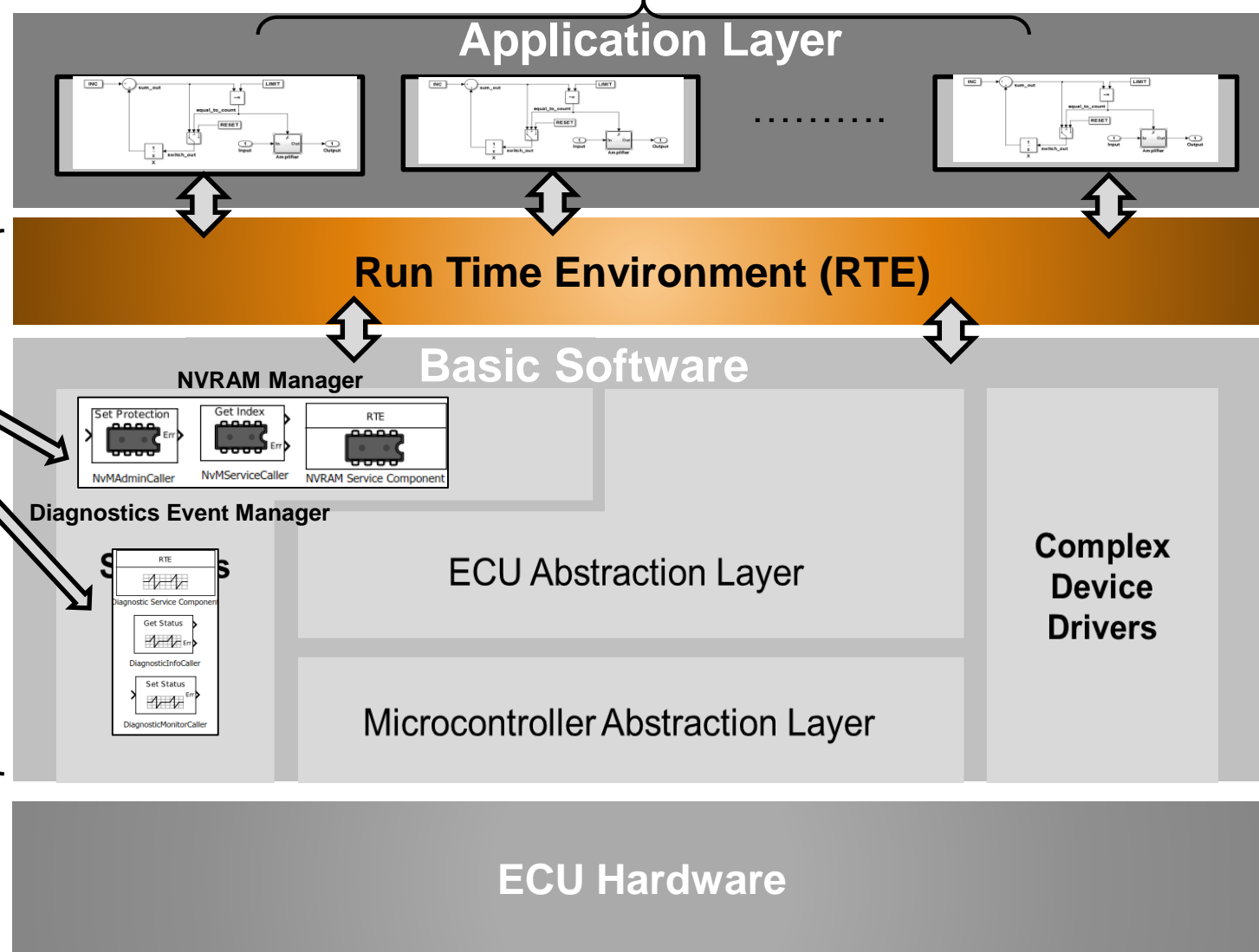


Modeling & Code Generation



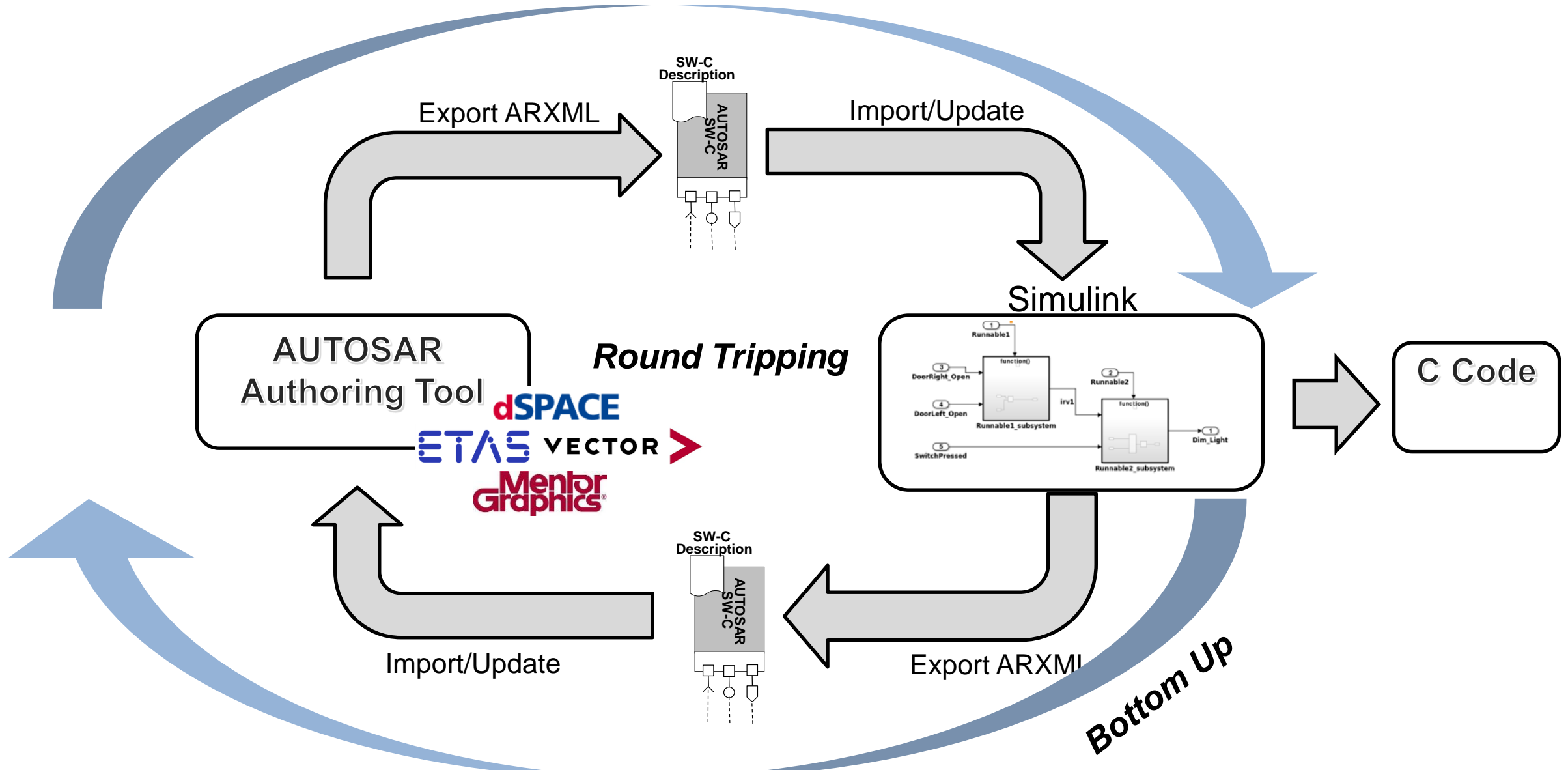
Modeling and Simulation

BSW Configuration & RTE Generation Tools

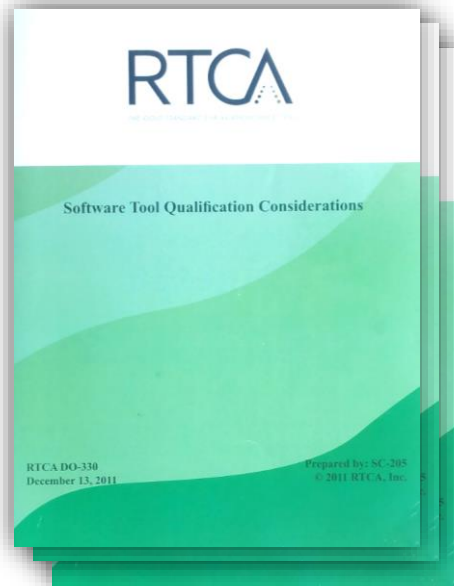


AUTOSAR Workflow

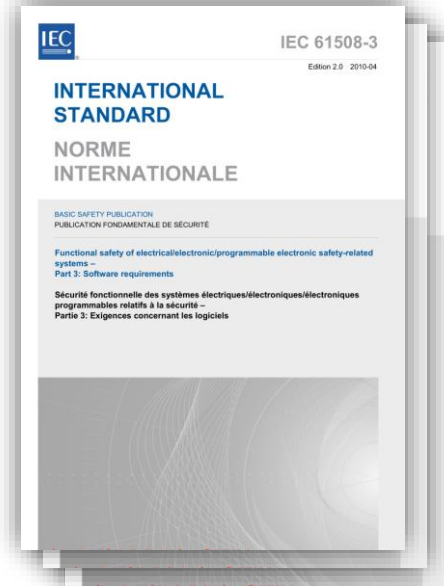
Top down



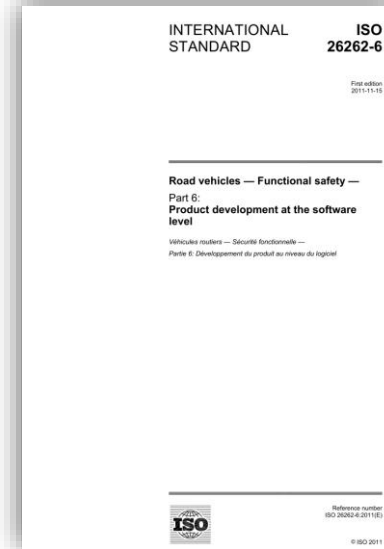
What Certification Standards Do We Support?



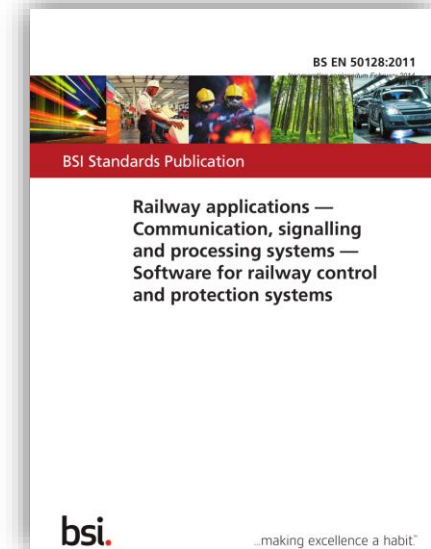
Aero:
DO-178C with
Supplements
and DO-254



Auto, Rail, Medical and others:
ISO 26262, EN 50128,
IEC 61508, 62304, 61511, 61131

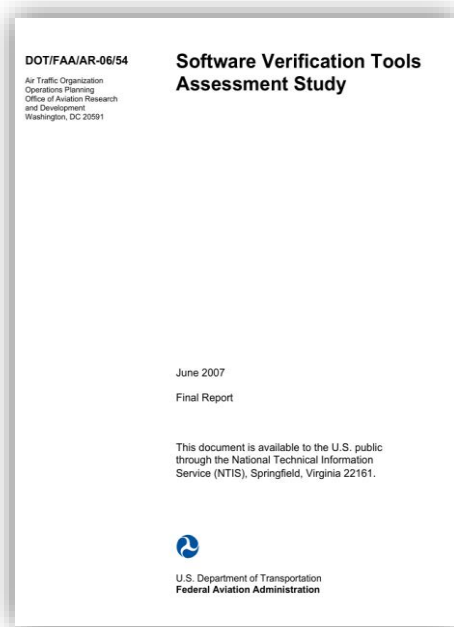


Model Based Design & Automotive SPICE

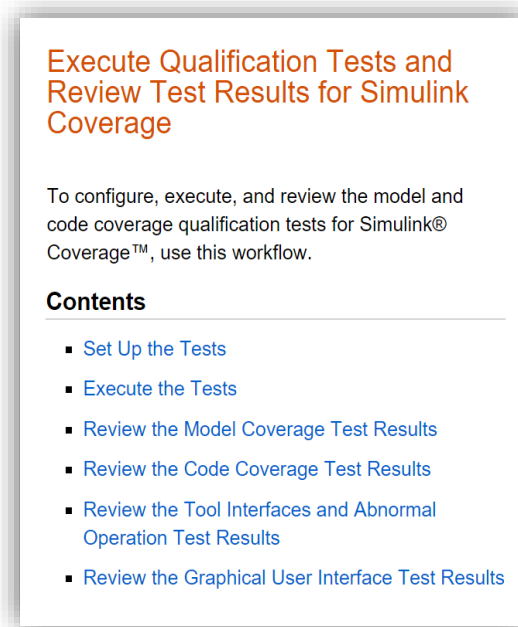


Qual and Cert Kits

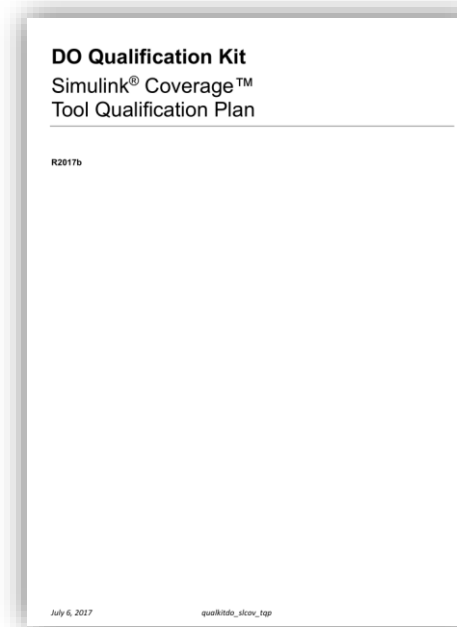
Everything Ready for Tool Qualification



Certification Basis



Interactive Test Procedures

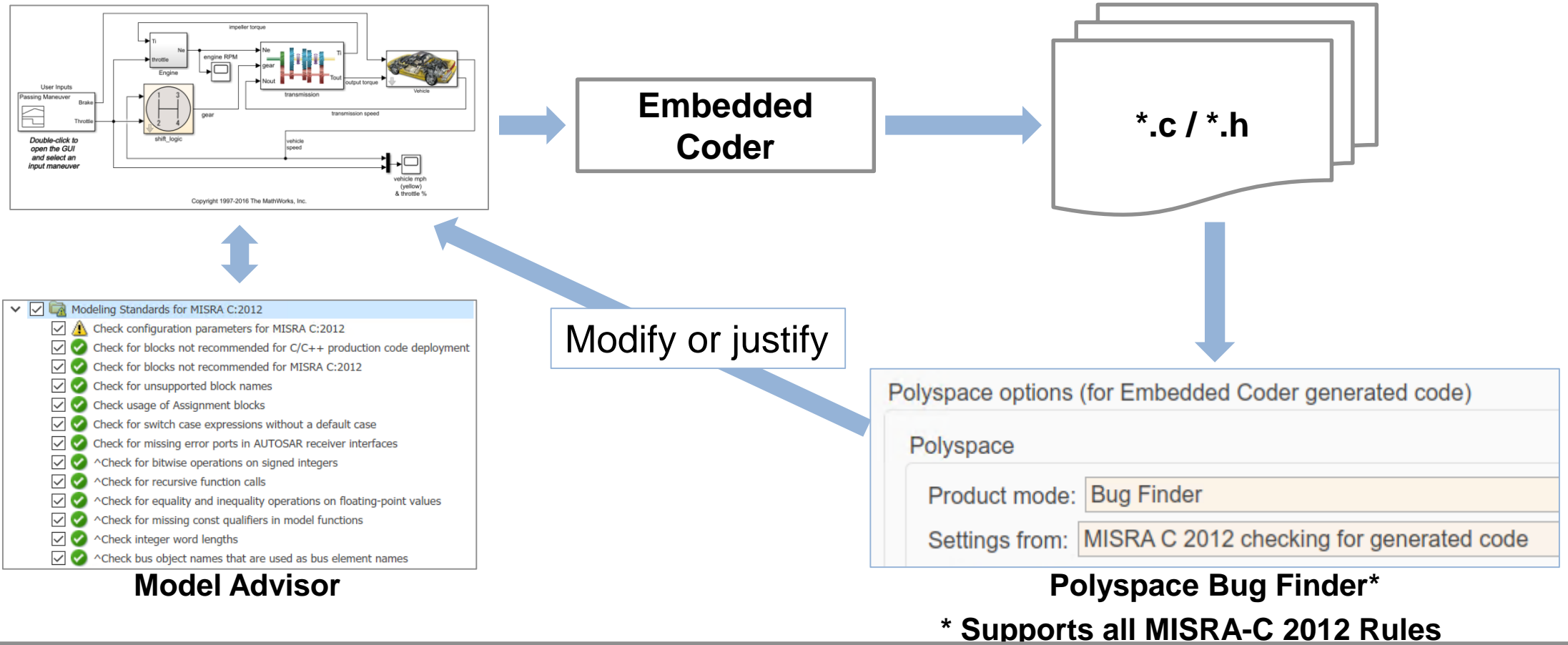


Artifacts and Templates



Pre-approval Certificates

How to achieve MISRA compliance



Help

MISRA C:2012 Compliance and Deviations for Code Generated using Embedded Coder



Categories

- Mandatory
- Required



+ documented Deviations

How to achieve secure coding guideline conformance

MISRA

CERT | Software Engineering Institute | Carnegie Mellon University

ISO/IEC TS 17961:2013
Information technology -- Programming languages, their environments and system software interfaces -- C secure coding rules

CWE Common Weakness Enumeration
A Community-Developed Dictionary of Software Weakness Types

Model Advisor

- > Modeling Standards for MISRA C:2012
- > Modeling Standards for Secure Coding (CERT C, CWE, ISO/IEC TS 17961)

Design Verifier

Simulink Design Verifier Results

Back to summary - Close results

sldvdemo_cruise_control_fxp_fixed/Fixed-Point Controller/Sum1

Overflow **ERROR** - View test case

Derived Ranges:
Output 1: [-128..128]

Block diagram showing: ufix16_En8, Target speed, sfx16_En8, error, PI Controller, sfx16_En13, 0, uat16.

Polyspace Bug Finder / Code Prover

Polyspace options (for Embedded Coder generated code)

Polyspace

Product mode: Bug Finder

Settings from: MISRA C 2012 checking for generated code

Polyspace Bug Finder

Find defects ISO-17961

- Defects default
- Name CERT-rules
- Static CERT-all
- Dynar ISO-17961
- Data CWE
- Resol all
- custom

Join Hands & Develop

...”standardization *by **customizing** the Simulink® development environment*”...

Defines Architectural Details and Usage of Standard Libraries via:

- **Embedded Coder Dictionary**
- **Custom Storage Class Designer**
- **Modeling and Code Generation Best Practices**
- **Custom Replacement Libraries**
- **Target Libraries**
- **Code Verification**



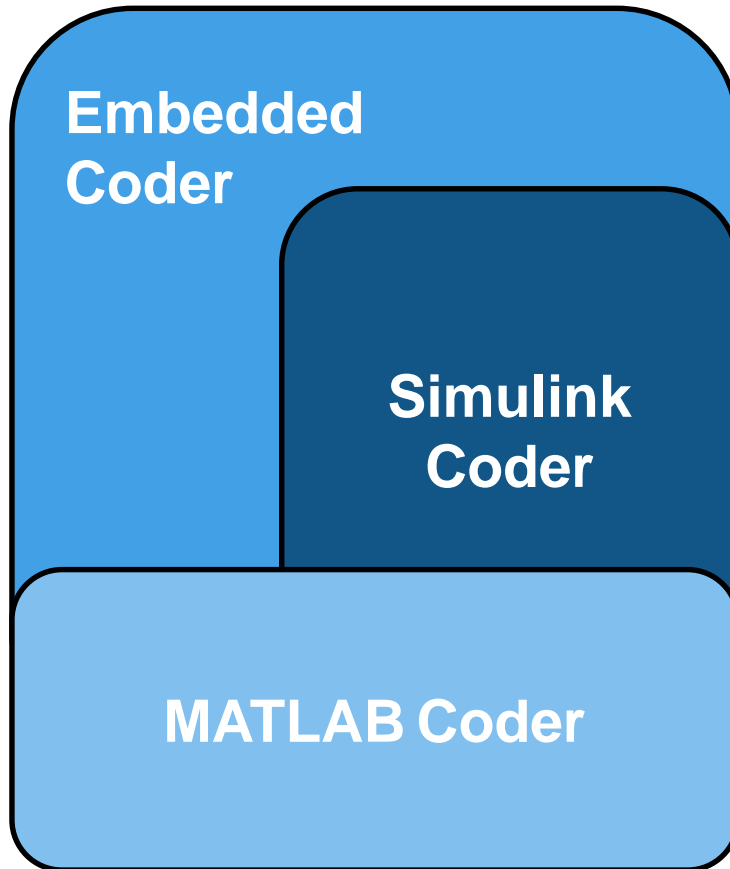
Software Architect

Software Engineer

Configure default settings for model. Applies Software definitions, optimizations and performs verification using:

- **Code Mapping Editor**
- **Model Data Editor**
- **Function Prototype Control**
- **Model Advisor**
- **Model Optimization Configurations**
- **Simulink Data Object & Storage Classes**
- **Model and Code Verification**

Embedded Coder For Production Code Generation



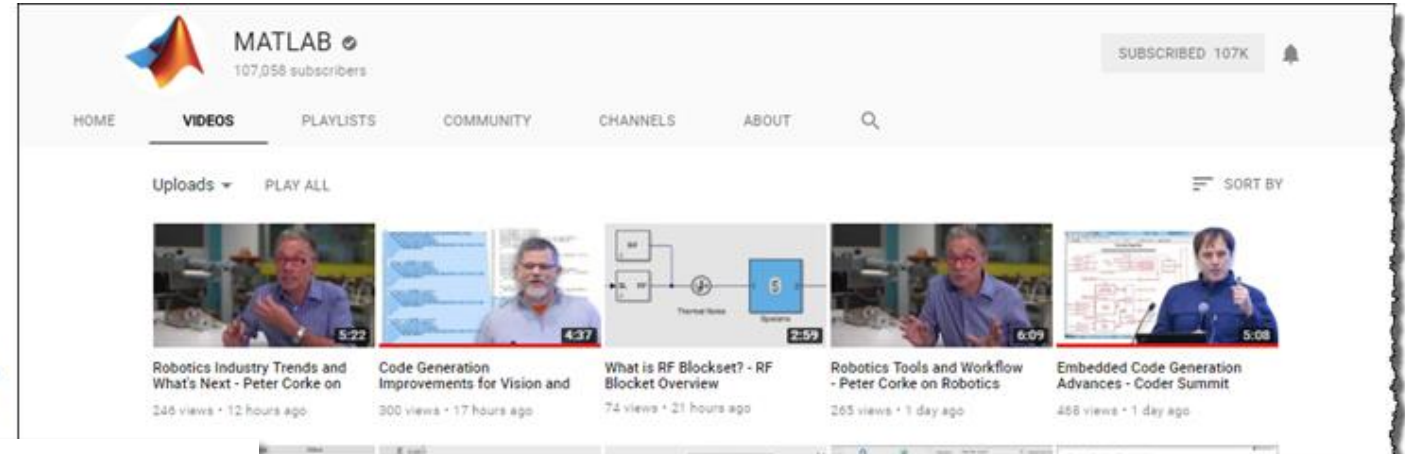
Production Code Generation with Embedded Coder

Generates C/C++ code optimized for embedded systems

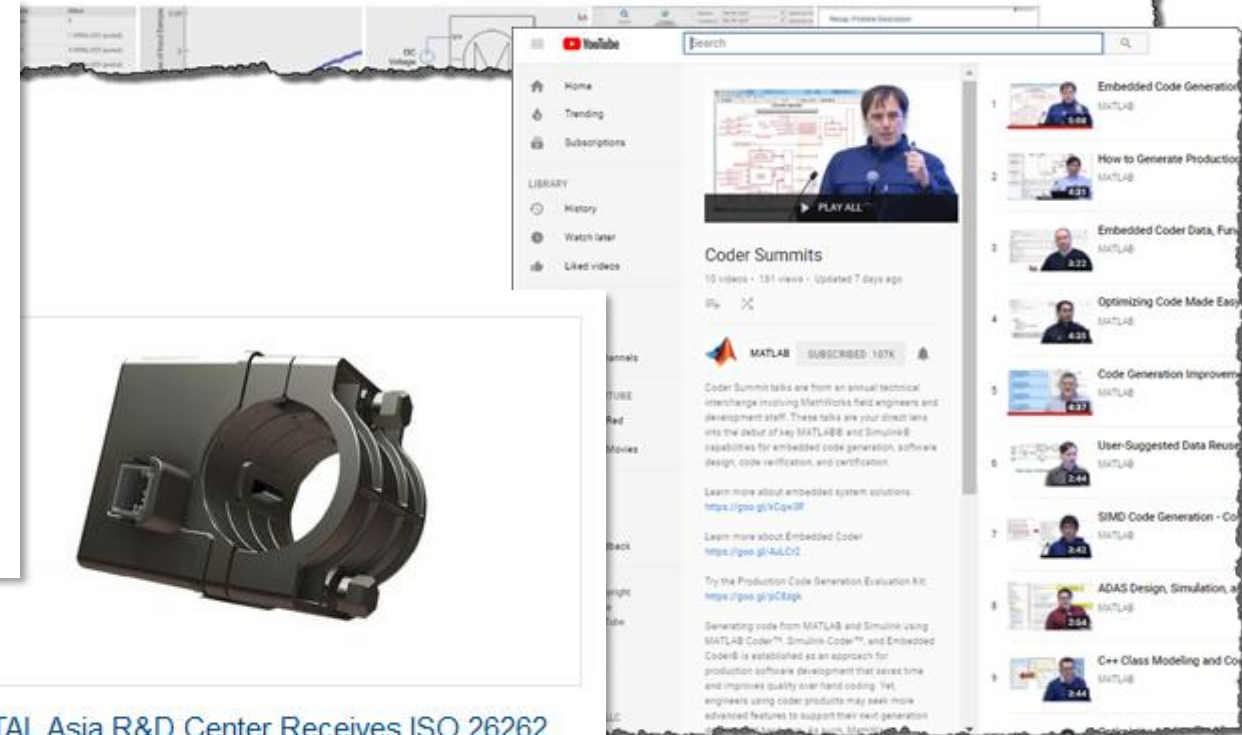
- ✓ Software-/ Processor –in-the-loop testing
- ✓ Easy integration of legacy code
- ✓ Advanced optimizations:
 - ✓ Maximize execution efficiency
 - ✓ Minimize RAM/ ROM usage
 - ✓ Target-specific function replacement
- ✓ File, Function and Interface control
- ✓ Data customization
- ✓ Support for ASAP2 and AUTOSAR
- ✓ Support for Industry Standards

Additional Resources

- [Coder Summit Videos](#)
- [AUTOSAR](#)
- [ISO 26262](#)
- [DO-178](#)
- [IEC 61508](#)
- [IEC 62304](#)
- [Embedded Code Generation](#)



BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design



KOSTAL Asia R&D Center Receives ISO 26262 ASIL D Certification for Automotive Software Developed with Model-Based Design

LG Chem Develops AUTOSAR - and ISO 26262 - Compliant Software for a Hybrid Vehicle Battery Management System

Challenge

Design and implement production battery management system (BMS) software for the Volvo XC90 plug-in hybrid

Solution

Use Model-Based Design with MATLAB and Simulink to model, simulate, verify, and generate production code for AUTOSAR application layer software components

Results

- Existing library of core components reused
- Software issues reduced by more than 50%
- ISO 26262 ASIL C certification achieved



The LG Chem battery management system.

“Model-Based Design with MATLAB and Simulink enables us to increase component reuse, reduce manual coding, improve communication with our customers, and ultimately deliver higher-quality BMS in less time.”

- Won Tae Joe, LG Chem

MathWorks Training Offerings

CODE GENERATION

Fundamentals of Code Generation for Embedded Applications

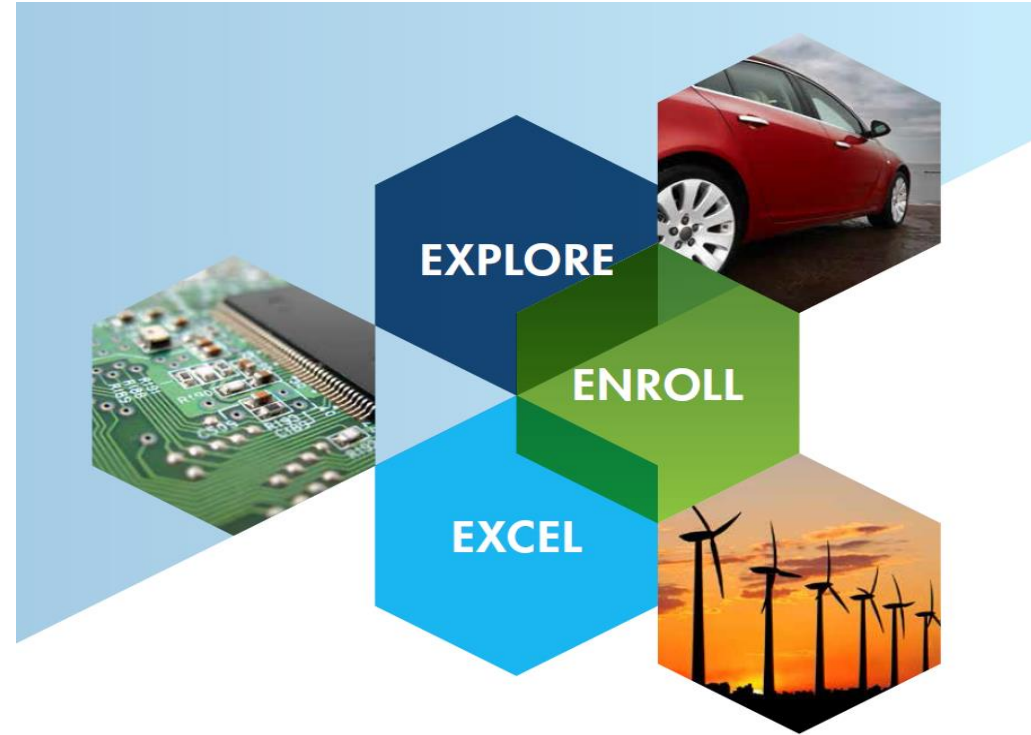
FUNDAMENTAL

- Simulation speedup with code generation
- Parameter tuning with external mode
- Code generation
- In-the-loop verification
- Code execution profiling

Embedded Coder for Production Code Generation

ADVANCED

- Generated code module and data structure
- Code generation options and optimizations
- Integrating generated code with external code
- Customizing data characteristics
- Advanced customization techniques
- Deploying embedded code



CURRICULUM PATHS

Code Generation/ Embedded Systems

Simulink for
System and
Algorithm Modeling



Fundamentals of
Code Generation for
Embedded Applications



Embedded Coder
for Production
Code Generation

<http://www.mathworks.com/services/training/>

Call To Action

Try or Buy

There are many ways to start using Embedded Coder. Download a free trial, or [explore pricing and licensing options](#).

Get a Free Trial

Test drive Embedded Coder.

Get a trial

Ready to Buy?

Purchase Embedded Coder and explore related products.

Contact sales

Pricing and licensing

Code from Simulink

Download an interactive tutorial that guides you through the implementation of a high-level PID throttle controller to a production executable with an accompanying test harness and code metrics report. Topics include data specification, legacy code integration, and build processes.

Try the evaluation kit



[Free Production Code Generation Evaluation Kit](#)

Search MathWorks.com

Examples Search Help

CONTENTS

Code Generation Workflow

```

%! step_function */
%!demo_POD_Eval_PI_step(wslid)

%! rfb_gain;
%! rfb_integralGainShape;
%! Discrete_Time_Integrator1;
%! Discrete_Time_Integrator1;

%! use: "S3/SimpleTable" incorporatees;
%! report: "S3/SimpleTable";
%! report: "S3/SimpleTable";

%! integralGainShape = stwemo_POD_Eval_PI_0_pos_0;
%! demo_POD_Eval_PI_0_rfb;
    
```

Generate C Code from a Control Algorithm for an Embedded System
Generate code for a control algorithm model, integrate the generated code with an existing system, and validate simulation and [Open Script](#)

Configure Data Interface in the Generated Code
Specify signals, states, and parameters for inclusion in generated code. [Open Script](#)

Partition Functions in the Generated Code
Associate subsystems in a model with function names and files. [Open Script](#)

Call External C Code from Model and Generated Code
Call existing, external functions from a simulation or from the generated code by using the Legacy Code Tool. [Open Script](#)

```

%!<S3>/Discrete Time Integrator1
%! in1
%! duct3
%! bleWrap): %!<S3>/SimpleTableWrap
%! or1 = I_Gain * SimpleTable;
%! (real_T*) ((I_OutMap(0))); 7)
%! me_Integrator1_DSTAT;
    
```

Thank You 😊 !!!