MATLAB EXPO 2018

Generating Industry Standards Production C Code Using Embedded Coder

Rajat Arora
Durvesh Kulkarni
Lines of Code (LOC) is exploding

2011 1M

2018 100M

2025 1B

“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

A software architect helps
- multiple engineers
- multiple projects

generate code that conforms to the organization standard

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

…”standardization by customizing the Simulink® development environment”…

…deploy 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?

Compatibility

Configuration

Optimize

Interface

Grid Locked Situation !!!
Fast Code Generation Using Quick Start

SIMULINK MODEL


GENERATED CODE

QUICK START – 7 Simple Steps
7 Simple Steps for Generating Quick Code
Quick Start Wizard = True; Grid Solved = Quick Start Wizard;

Quickly Generate Code
.c, .cpp, .h
Join Hands & Develop

System Architect

Software Engineer

…Collaborative Workflow…

Quick Study

Standardize Code Architecture
Production Code Details
Improve Code Efficiency
Standards & Certification
How to go about quickly architecting my code?
1. Create, Customize, Your Storage Classes

22 /* ConstVolatile memory section */
23 /* Definition for custom storage class: ConstVolatile */
24 const volatile real32_T TEST - 50.0F;
The “Coder Dictionary” For Your Code Architecting Needs

2. Customize Functions

```c
#pragma (FAST_begin)
extern void MyCustom_untitled_step(void);
#pragma (FAST_end)
```
The “Coder Dictionary” For Your Code Architecting Needs

3. Create, Customize Memory Sections
Easily configure your **Model** with Code Mapping Editor

```c
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?

RAM

ROM

Execution Speed

Optimizations For Target

Model Design

Configurations For Code Generation
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_db) is calculated via interpolant-table with pre-calibrated data. Down threshold (down_db) is also calculated via interpolant-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

Model Advisor Guidelines

Modeling Standards for DC-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 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)

```c
void Subsystem(void)
{
    ...  
    for (i = 0; i < 100; i++) {
        Reuse[i] = 2.0F * Reuse[i] * 20.0F;
    }
    fcn3(&Reuse[0]);
}
```
How and What Should I Optimize?

- Model Design
- Configurations For Code Generation
- Optimizations For Target
Easy Optimization Levels and Priorities

- **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
Optimize Using Min & Max Values

![Diagram of a system with blocks labeled U1, U2, and U3, and input and output nodes. The block parameters window is also shown.]
Optimize Using Min & Max Values

```plaintext
rtb_Sum = U1 + U2;

/* Gain: '<S8>/Gain' incorporates:
 * Import: '<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>/Sum4' incorporates:
     * Import: '<Root>/U1'
     * Import: '<Root>/U2'
     * Import: '<Root>/U3'
     */
    rtb_Sum2 = (U1 + U2) + U3;
} else {
    /* Product: '<S8>/Product' incorporates:
     * Import: '<Root>/U1'
     * Import: '<Root>/U2'
     * Import: '<Root>/U3'
     */
    rtb_Sum2 = U1 * U2 * U3;
}
```
How and What Should I Optimize?

- Optimizations For Target
- Model Design
- Configurations For Code Generation
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
Auto Code Performance ARM Cortex-A

Execution Time (μ secs)

410.7

185.5

16.8

14.1

ANSI, No Opt
ANSI, Opt
NE10, No Opt
NE10, Opt

Embedded Coder ANSI-C
Embedded Coder ANSI-C (& GCC optimized)
Embedded Coder, NEON Optimized
Embedded Coder, NEON Optimized (& GCC Optimized)

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

SISD - Single Instruction Single Data

SIMD - Single Instruction Multiple Data

DATA POOL → INSTRUCTION POOL → PROCESSING UNIT → DATA POOL

DATA POOL → INSTRUCTION POOL → PROCESSING UNIT → DATA POOL
ARM Cortex-A Optimized Code
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)
{
    __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);
    }
    for (i = 0; i < size %4 ; i++)
    {
        C[i+k1] = A[i+k1]*B[i+k1];
    }
}

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_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
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

…“standardization by customizing the Simulink® development environment”…
Join Hands & Develop

...Collaborative Workflow...

Software Architect

Software Engineer

Quick Study

Standardize Code Architecture

Production Code Details

Improve Code Efficiency

Standards & Certification
MathWorks approach for AUTOSAR

Authoring Tools ➔ Software Architecture Definition

Run Time Environment (RTE)

Application Layer

Basic Software

ECU Abstraction Layer

Microcontroller Abstraction Layer

Complex Device Drivers

ECU Hardware

Modeling & Code Generation

Modeling and Simulation

BSW Configuration & RTE Generation Tools

Authors: SW Providers

NVRAM Manager

Diagnostics Event Manager

......
AUTOSAR Workflow

**Top down**

1. Export ARXML
   - AUTOSAR SW-C
   - SW-C Description

2. Import/Update
   - AUTOSAR SW-C
   - SW-C Description

3. Simulink

4. C Code

**Round Tripping**

5. AUTOSAR Authoring Tool
   - dSPACE
   - ETAS
   - VECTOR
   - Mentor Graphics

6. Bottom Up

7. Export ARXML
   - AUTOSAR SW-C
   - SW-C Description

8. Import/Update
   - AUTOSAR SW-C
   - SW-C Description
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

Model Advisor

Embedded Coder

Modify or justify

*c / *.h

Polyspace Bug Finder*

* Supports all MISRA-C 2012 Rules

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

Categories
- Mandatory
- Required

☑️ + documented Deviations
How to achieve secure coding guideline conformance

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

Design Verifier

Polyspace Bug Finder / Code Prover

Polyspace Bug Finder
- Find defects
  - ISO-17961
  - CERT-rules
  - CERT-all
  - Static
  - Dynamic
  - CWE
  - Dataflow
  - Resources
  - Custom

Polyspace options (for Embedded Coder generated code)
- Polyspace
- Product mode: Bug Finder
- Settings from: MISRA C 2012 checking for generated code
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

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

…”standardization by customizing the Simulink® development environment”…
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
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

“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

Link to technical article
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
Thank You ☺ !!!