Production Code Generation Using Model-Based Design: An AC Motor Control Case Study

Chris Fillyaw, Application Engineering
Model your system, design your controller, and generate code using Simulink
Simulation vs. Hardware Measurements

Velocity Calculation (Sim vs. HW)

- **Velocity (Rad/Sec)**
- **Time (Sec)**

- **HW Velocity**
- **Sim Velocity**
What are we going to see in this case study?

- Getting started with production code generation
- Managing the data in your design
- Solving a field issue in simulation and generating code with the solution.
- Verifying code execution time of the generated code
- Scaling up the design and modeling more components of your system
What are we going to see in this case study?

- **Getting started with production code generation**
  - Managing the data in your design
  - Solving a field issue in simulation and generating code with the solution.
  - Verifying code execution time of the generated code
  - Scaling up the design and modeling more components of your system
Where do you start when adopting Model-Based Design?
Start with a core controller component

PMSM Controller Application Testbench

Velocity Control

Outer velocity control with inner loop current control.
Let’s explore this component...
Simulate to Understand Behavior
Prep Model for Code Generation with “Code Generation Objectives”
Control Interface Elements

Function Prototype:

```c
void Controller(real32_T arg_command, real32_T arg_measured,
real32_T arg_phase_currents[2], real32_T arg_electrical_position,
real32_T arg_phase_voltages[3])
```

Calibration Parameters:

- **Value:** 10
- **Data type:** single
- **Dimensions:** [1 1]
- **Complexity:** real
- **Minimum:** 0
- **Maximum:** 100
- **Units:** Volt/Ampere

Measurement Signals:

- **Storage class:** ExportedGlobal
Package up the generated code and distribute

Enable and specify filename:

Zip file created with all required source files:
What are we going to see in this case study?

- Getting started with production code generation

- Managing the data in your design

- Solving a field issue in simulation and generating code with the solution.

- Verifying code execution time of the generated code

- Scaling up the design and modeling more components of your system
How do I manage data for my design?
Data Management with Simulink

Define

Share

Edit

Simulink Project
Model Explorer
Simulink Data Objects
Data Management with Simulink

Define → Edit → Store → Share → Define

- Simulink Data Dictionary
- Simulink Project
- Model Explorer
- Simulink Data Objects
Data Dictionaries link to your Simulink models
See differences, revert, and save changes to the repository
Change tracking
Simulink Project integration
Enumerations scoped to data dictionary

```matlab
classdef controllerModeEnum < Simulink.IntEnumType

    % Copyright 2013 The MathWorks, Inc.

    enumeration
        Error (0)
        StandBy (1)
        VelocityControl (2)
        OpenLoopVelocityControl (3)
    end

end
```

**Enumerated Type: controllerModeEnum**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>0</td>
<td>Current Mode: Error has occurred</td>
</tr>
<tr>
<td>StandBy</td>
<td>1</td>
<td>Current Mode: Motor OK, control disabled</td>
</tr>
<tr>
<td>VelocityControl</td>
<td>2</td>
<td>Current Mode: Velocity closed-loop control is active</td>
</tr>
<tr>
<td>OpenLoopVelocityControl</td>
<td>3</td>
<td>Current Mode: Velocity open-loop control is active</td>
</tr>
</tbody>
</table>

Default: Error
What are we going to see in this case study?

- Getting started with production code generation
- Managing the data in your design
- **Solving a field issue in simulation and generating code with the solution.**
- Verifying code execution time of the generated code
- Scaling up the design and modeling more components of your system
Are we done?

What if something breaks?
Jeff,

I have an issue with the software that you sent to me earlier this week. See the screen shot below. With the motor under speed control, I inserted a couple torque pulses on to the output shaft. A strange overshoot occurs under this condition and I can't make it go away by adjusting cals. I also noticed that the max current is limited to 1.3A. This limitation is hindering my calibration efforts, so I'd like to request a tuning parameter for this limit if possible.

I know it's late on a Friday afternoon, but in order to meet our prototype delivery on Tuesday we will need a software release with both of these issues fixed ASAP. I'm willing to come in over the weekend if you think I can be of any help. Please give me a call when you get this message.
Summary of the issues

Pretty Good...

Why?

Make this a Cal
Plan of attack

1. Duplicate the issue in Simulation
2. Add tuning parameter for current limit
3. Investigate issue and fix overshoot in the model
Duplicating the issue in Simulation
Adding a tuning parameter

Specify data object in model explorer:

Use data object for constant value:
Export data objects to data dictionary script

Run this script on model startup:

```matlab
% MATLAB file generated by Simulink.saveVars on 24-Sep...
% MATLAB version: 8.1.0.604 (R2013a)

paramAdcZeroOffsetDriverUnits = Simulink.Parameter;
paramAdcZeroOffsetDriverUnits.Value = 2048;
paramAdcZeroOffsetDriverUnits.CoderInfo.StorageClass = 'paramAdcZeroOffsetDriverUnits.CoderInfo.Alias = '';
paramAdcZeroOffsetDriverUnits.CoderInfo.Alignment = -1;
paramAdcZeroOffsetDriverUnits.CoderInfo.CustomStorageClass = 'paramAdcZeroOffsetDriverUnits.Description = ...
    'Offset to calibrate 0 current from ADC measurement';
paramAdcZeroOffsetDriverUnits.DataType = 'int16';
paramAdcZeroOffsetDriverUnits.Min = 0;
paramAdcZeroOffsetDriverUnits.Max = 4095;
paramAdcZeroOffsetDriverUnits.DocUnits = 'Raw driver unit

paramAmpPerDriverUnit = Simulink.Parameter;
paramAmpPerDriverUnit.Value = -0.004147766020629882;
paramAmpPerDriverUnit.CoderInfo.StorageClass = 'Exp...
paramAmpPerDriverUnit.CoderInfo.Alias = '';
paramAmpPerDriverUnit.CoderInfo.Alignment = -1;
paramAmpPerDriverUnit.CoderInfo.CustomStorageClass = 'Exp...
paramAmpPerDriverUnit.Description = ...
```
Final solution
What are we going to see in this case study?

- Getting started with production code generation
- Managing the data in your design
- Solving a field issue in simulation and generating code with the solution.
- **Verifying code execution time of the generated code**
- Scaling up the design and modeling more components of your system
How do I know if the generated code is efficient enough?
Publications showing efficiency of generated code

Application of Auto-Coding for Rapid and Efficient Motor Control Development

James Walters, Cahya Harianto, Edward Kelly, and Tanto Sugiarto
Delphi Automotive

ABSTRACT
In hybrid and electric vehicles, the control of the electric motor is a critical component of vehicle functions such as motoring, generating, engine-starting and braking. The efficient and accurate control of motor torque is performed by the motor controller. It is a complex system incorporating sensor sampling, data processing, controls, diagnostics, and 3-phase Pulse Width Modulation (PWM) generation which are executed in sub-100 µsec periods. Due to the fast execution rates, care must be taken in the software coding phase to ensure the algorithms will not exceed the target processor's throughput capability.

Production motor control development often still follows the path of customer requirements, component requirements, simulation, hand-code, and verification test due to the concern for processor throughput. In the case of vehicle system controls, typically executed no faster than 5-10 msec periods, auto-coding tools are used for algorithm development as well as testing. The advantages of auto-coding to greatly speed the development process by linking the tools for simulation, code generation and testing early in the development process as well as to more easily investigate performance issues late in the process are well known. It is not uncommon, however, to lose coding efficiency with this approach. While the loss of efficiency may be tolerable for slow periods, it is not acceptable at faster periods used in motor controls as it will preclude the algorithms from executing or drive unnecessarily expensive solutions.

Publications showing efficiency of generated code

Table 1. Throughput Comparison between Model-Generated Code and Hand-Code

<table>
<thead>
<tr>
<th>Task / Module</th>
<th>Throughput (uSec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model</td>
</tr>
<tr>
<td>Current Magnitude and Phase Process (2.2)</td>
<td>1.42</td>
</tr>
<tr>
<td>ABC to QDC Frame Transformation (2.3)</td>
<td>0.76</td>
</tr>
<tr>
<td>Resolver Harmonic Learn (1.12)</td>
<td>0.48</td>
</tr>
<tr>
<td>Angle Position Determination (2.1)</td>
<td>0.93</td>
</tr>
<tr>
<td>PI-Current Regulator (2.5)</td>
<td>7.62</td>
</tr>
<tr>
<td>Torque Mode (1.3)</td>
<td>4.82</td>
</tr>
<tr>
<td>DQ0 Rotating to Stationary Frame Transformation (2.7)</td>
<td>0.94</td>
</tr>
<tr>
<td>Complete 100 uSec Task</td>
<td>65.37</td>
</tr>
</tbody>
</table>

From Table 1, it can be observed that the overall model-generated code is within 1.5 uSec of the hand-code which represents a 2.4% penalty. It should also be noted that the hand-code used as a baseline comparison has been developed over many projects and has been optimized by the software team. In general, there is a small but measurable increase in the required time for the model-generated code. The benefits of directly using simulation models through the development process, allowing the domain experts to control the implementation, and more easily automating the verification testing are well worth the small penalty.

What did our example look like?

Final implementation: **12µs** out of **40µs** budget (30% CPU usage)
Why should I scale up the model to include more components?
Catch design issues earlier in development
What are we going to see in this case study?

- Getting started with production code generation
- Managing the data in your design
- Solving a field issue in simulation and generating code with the solution.
- Verifying code execution time of the generated code
- Scaling up the design and modeling more components of your system
How do I scale up the design?
Use function triggered subsystems for ISR boundaries
Manage complex systems with multiple model files
Use model referencing for multiple models

Control Loop Interrupt Service Routine

Copyright 2013 The MathWorks, Inc.
Can I manage algorithm variants without dirtying the model?
Parameterize algorithm variants
Parameterize algorithm variants

Data properties: hallElectricalVelocityVariant
Value: hallElectricalVelocityEnum.Division

Variant choices (list of child subsystems)

<table>
<thead>
<tr>
<th>Name (read-only)</th>
<th>Variant control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrical_Velocity_Division</td>
<td>hallElectricalVelocityVariant == hallElectricalVelocityEnum.Division</td>
</tr>
<tr>
<td>Electrical_Velocity_NR</td>
<td>hallElectricalVelocityVariant == hallElectricalVelocityEnum.NewtonRaphson</td>
</tr>
</tbody>
</table>

Electrical_Velocity_Division
- dir
- position_delta
- timer_delta

Electrical_Velocity_NR
- position_data
- velocity
- timer_data

Parameterize algorithm variants
What did we learn from this case study?

- **Getting started with production code generation**
  - Code generation advisor
  - Function prototype control
  - Package up code and artifacts

- **Managing the data in your design**
  - Store data in Data Dictionaries
  - Difference/save/revert working data from the dictionary
  - Simulink Project to manage all project files

- **Solving a field issue in simulation and generating code with the solution.**
  - Solve issues in simulation
  - Adding tuning parameters

- **Scaling up the design and modeling more components of your system**
  - Catch design issues earlier in the development process
  - Model referencing, subsystem variants, and reference configuration sets help architect your model and generated code.
Model your system, design your controller, and generate code using Simulink

System Model

Controller Model

Motor Model

Controller C Code

Motor Hardware

Embedded System
Where can I find more information?

- Explore a demo on generating C code from a controller and verifying its compiled behavior and execution time
  
  >> docsearch "Field-Oriented Control of Permanent Magnet Synchronous Machine"
  

- Watch recorded webinar: Embedded Code Generation for AC Motors
  
  http://www.mathworks.com/company/events/webinars/wbnr61549.html

- Watch 2012 MathWorks Automotive Conference recording of: Parameterizing and Verifying a Permanent Magnet Synchronous Motor Model
  