Automatic Code Generation for Embedded Control Systems

Tiffany Liang
Application Engineer
MathWorks
Embedded Control Systems

Automotive

Aero/Def

Construction Machine

Power Electronics

Industrial Equipment

Robot
Model-Based Design “From Concept to Code”

- **Modeling & Simulation**
- **Control Design**
- **Physical Modeling**
- **Real-Time Test**
- **Code Generation**
- **Verification & Validation**
Quick MBD Overview
Brushless DC Motor Control

Model

Code

Code Generation

if (g_f4_iq_ref > MTR_LIMIT_IQ) {
  g_f4_iq_ref = MTR_LIMIT_IQ;
} else {
  if (g_f4_iq_ref < -MTR_LIMIT_IQ) {
    g_f4_iq_ref = -MTR_LIMIT_IQ;
  }
}

TI F28069ISO + DRV8312-C2-KIT used
モーター制御システム外観
Example on YouTube MATLAB Channel

https://www.youtube.com/watch?v=wxYTLbYfBP0
How to Embed Your Model? Coder Solution!

**MATLAB Coder®**
- MEX
- C
- MATLAB

**Simulink Coder®**
- C
- RCP/HILS

**HDL Coder™**
- HDL
- FPGA/ASIC

**Embedded Coder®**
- C/C++
- MCU/DSP

**Simulink PLC Coder™**
- ST
- PLC
C/C++ Code Development with Embedded Coder

MATLAB Program
Simulink/Stateflow Model

Code Generation

- Used with existing code
- Used for implementation
- Standalone executable/dll
What is Your Merit?

Modeling & Simulation

- Graphical modeling
  - Readable, easy understanding
- Early verification & validation
  - Concept test without hardware

Automatic Code Generation

- Coding time reduction
- Error reduction from model to code conversion
- Automatic synchronization between model and code
User Story

Model-based design was a key enabler to Volt’s rapid development.

“We have a single source for how a particular function should behave. Automatic code generation using The MathWorks’ Real-Time Workshop Embedded Coder was vital to meeting Volt’s aggressive program timing.”

Greg Hubbard
Senior Manager

Nearly 100% of the software for many of Volt’s modules was generated automatically.
boolean_T reset;
const volatile int8_T k = 2;

void rst_cntr_step(void)
{
    if (reset) {
        y = 0;
    } else {
        y += (int16_T)(k * u);
    }
}

- Equivalent Code as Model
- High Readability & Efficiency
Data Attributes

- You can define data attributes in your model
  - Signal line
  - States (Unit Delay etc.)
  - Block parameter
  - Stateflow data

```c
boolean_T reset;
const volatile int8_T k = 2;

void rst_cntr_step(void)
{
    if (reset) {
        y = 0;
    } else {
        y += (int16_T)(k * u);
    }
}

void rst_cntr(
    boolean_T reset,
    int16_T u,
    int16_T *y)
```
Code Generation Report

Code Metrics (lines, memory size, etc.)

Generated Code

Traceability between Model and Code

Simulink Report Generator required to generate model view
How about Efficiency of Generated Code?

### Table 2: ROM and RAM comparison between a floating-point hand code and auto code.

<table>
<thead>
<tr>
<th></th>
<th>Hand Code</th>
<th>Auto Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>6408</td>
<td>6192</td>
</tr>
<tr>
<td>RAM</td>
<td>132</td>
<td>112</td>
</tr>
</tbody>
</table>

### 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 QD0 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>4.62</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>
How about the Standard Compliance?

**MISRA C**
- Most rules satisfied
- You can download MISRA compliance document
  http://www.mathworks.com/matlabcentral/answers/102532-misra-c

**ISO 26262 / IEC 61508**
- Tool Qualified by TÜV SÜD
  (ISO26262 ASIL A-D)

**AUTOSAR**
- You can download AUTOSAR Support Package via Simulink menu
How to Design Fixed-Point Controller?
Fixed-Point Designer

Fixed-Point Data Type Settings

Conversion between Float and Fixed-Point

Useful for Quantization Error Test

Fixed-Point Simulation

Efficient Fixed-Point Scaling

Reduce Quantization Error

Efficient Code Generation

int16_T k = 13U;

Fixed-Point Value

y = y + k * u >> 3;

Scaling Adjustment Calculation

Fixed-Point Code Generation
One More Example: Digital DC/DC Converter Control System

Vin = 9V

Vout = 2V Controlled by TI C2000 MCU

TI C2000 DPS Workshop Kit (CPU: F28035)
Simulation Model
Use for circuit/controller design, parameter tunings

- Making an analog circuit with Simscape components
- Fixed-point PI controller + Rate Limiter (Soft Start)
Code Generation Model to F28035
Turnkey to download the controller model into MCU

Using EC TI C2000 support package to implement I/O and ISR

1 kHz periodic task

PWM ISR 200 kHz task

100Hz periodic task
## CLR/IQmath Blocks and PIL Profiling

C2000 Support Package provides code optimization opportunity and execution time estimation.

<table>
<thead>
<tr>
<th>Model</th>
<th>Generated Code</th>
<th>PIL Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI-C (No Optim)</td>
<td>rtb_Gain * Pgain</td>
<td>6.15 us (cannot run at 200 kHz)</td>
</tr>
<tr>
<td>TI C28x CLR Only</td>
<td>c28x_mul_s32_s32_s32_s32_sr(Pgain, rtb_Gain, 24L)</td>
<td>3.23 us</td>
</tr>
<tr>
<td>TI C28x CLR + IQmath</td>
<td>_IQ24mpy(rtb_Gain, Pgain)</td>
<td>1.8 us</td>
</tr>
</tbody>
</table>
Summary

You can achieve highly efficient development cycle

Simulink/Stateflow

Model
Desktop Simulation

Concept
Specification

Focus on Improvement

Coder Products

MCU/DSP
Experiment

Reduce Coding

Code