Workflow for Control System Design and Implementation

- Dhirendra Singh, Application Engineer
- Shobhit Shanker, Application Engineer
Agenda

- Industry Trends and Challenges
- Design PID Controller & Feedback Compensator
- Adding Control logic
- Implementing control algorithm
Challenge - Increasing Software Content

Software content increasing 3x – 5x per year
How to develop and manage such large amount of software?
Growth of Automotive Electronics and Software

- **1970**
  - Motorola 6800
    - 8 bit / 1 MHz / 4 kBytes

- **1980**
  - Freescale MPC5674F
    - 32 bit / 200+ MHz / 4000 kBytes

- **1990**
  - C
  - Microcode/Assembly

- **2000**
  - Modeling

- **2010**
  - Abstraction

- **2020**
  - Processor

- **Systems**
  - Ignition
  - Fuel Injection
  - ETC
  - ABS
  - ESC
  - HVAC
  - AFS
  - PCS
  - AEP
  - HUD
  - HCCI
Model-Based Design - Controller Design

- **Research**
- **Requirements**
- **Design**
  - Environmental Models
  - Mechanical
  - Electrical
  - Control Algorithms
  - Supervisory Logic
- **Implementation**
  - C, C++
  - VHDL, Verilog
  - Structured Text
  - MCU, DSP, FPGA, ASIC, PLC
- **Integration**
- **Test & Verification**
What is Our End Goal?

Real-Time or Embedded Processor or ECU Code Generation

Embedded Processor or ECU
Problem Formulation

Higher Performance:

1. **Faster trajectory**  
   Vmax: 150 → 250 rad/s  
   Amax: 2000 → 5000 rad/s²

2. **Decreased error at standstill**  
   error < 1 rad
Agenda

- Industry Trends and Challenges
- **Design PID Controller & Feedback Compensator**
- Adding Control logic
- Implementing control algorithm
Control Design: *solving your design problem*

- Tuning a PID controller
- Specifying the controller structure in Simulink
- Setting up your design environment
- Analyzing and tuning the closed loop
- Discretizing controller
- Optimizing design parameters
- Designing fault detection logic

Demo
PID Controller Block with Automatic Tuning
Agenda

- Design PID Controller & Feedback Compensator
- Adding Control logic
- Implementing control algorithm
Designing Fault Detection Logic
Model-Based Design - Implementation

RESEARCH

REQUIREMENTS

DESIGN

Environmental Models

Mechanical

Electrical

Control Algorithms

Supervisory Logic

IMPLEMENTATION

C, C++

VHDL, Verilog

Structured Text

MCU

DSP

FPGA

ASIC

PLC

INTEGRATION

TEST & VERIFICATION
Agenda

- Design PID Controller & Feedback Compensator
- Adding Control logic
- Implementing control algorithm
Workflow for Control System Design - Implementation

Shobhit Shanker
Senior Application Engineer- Code Generation and Verification
Implement Model as Software On-target

Controller Model

Plant Model

Code Generation

Embedded Processor or ECU
Demo – Code Generation
How Did We Get Here?
## Workflow For Embedded Code Generation

<table>
<thead>
<tr>
<th>Setting up Code Generation Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Modeling Guidelines Checking</td>
</tr>
<tr>
<td>- Code Generation Advisor Settings e.g. RAM/ROM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Add Software Design Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Algorithm Partitioning e.g. reusable libraries, Model reference</td>
</tr>
<tr>
<td>- Data Typing, Scoping of Variables, Fixed Point Details, Sw-DD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generate Code &amp; Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Peer review, Code Walkthrough</td>
</tr>
<tr>
<td>- Standards Compliance e.g. MISRA-AC-AGC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verification and Integration of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Software-in-loop testing(SIL)</td>
</tr>
<tr>
<td>- Processor-in-Loop(PIL) for Target Testing</td>
</tr>
<tr>
<td>- Platform Integration</td>
</tr>
</tbody>
</table>
Setting up Code-Generation Environment

- Discretizing the model
  - Variable → Fixed-Step
- Code Advisor
- Float-Fixed point conversion
- Modeling guidelines checking MAAB, DO178B/C, ISO26262
Add Software Design Details

- Naming of Variable
- Storage class (Scope)
- Datatype
- Comments
- Algorithm Partitioning
  - Functions
  - Reusable Functions
  - Inline
  - #Pragmas
Generate Code and Review

- Traceability
- Static Code Metrics
  - RAM
  - Call Tree
  - SLOC
- Comments
Continuous Test and Verification (SIL/PIL)
Model Testing

- Develop test inputs
  - Types of inputs:
    - Functional requirement
    - High level use case
    - Sensitivity analysis
    - “Edge Case” testing
    - Robustness testing
    - …etc

- Apply the test inputs to model
- Analyze the results
  - Expected output
- Measure coverage
Log Signals of Interest

Identify numerical differences

Supports:
- Normal, SIL, PIL
- Logged signals
- Imported signals
- HTML reports
Software-in-the-Loop (SIL) Testing:
Verify Production Controller with Software-in-the-loop

Execution
- Host/Host
- Nonreal-time

Compiled C Code
S-Function
(Windows DLL)
Processor-in-the-Loop (PIL) Testing:
Verify Production Controller with Processor-in-the-loop

Execution
• Host/Target
• Nonreal-time
Platform Integration

Controller Model

Communication Interfaces
- Comm Drivers
- Generated Algorithm Code
- Output Drivers
- Input Drivers
- Optional Target Optimized Code
- Special Device Drivers
- Scheduler/Operating System and Support Utilities

Sensors
- Sensors

Actuators
- Actuators

Special Interfaces
- Special Interfaces

Tuning
- Tuning
Summary

- Design, Tune and Test Controllers in Simulink
- Convert Controller Models to Embedded Code
- Test Generated Code on Target
What’s Next?

- Explore `>>rtwdemos`
- Attend Embedded Code Generation class [www.mathworks.com/services/training/courses/SLEC_1.html](http://www.mathworks.com/services/training/courses/SLEC_1.html)
- Request Advisory Services: Developing Embedded Targets, Model-Based Design for ISO 26262 or DO-178 [www.mathworks.com/services/consulting/areas/design.html](http://www.mathworks.com/services/consulting/areas/design.html)

Thank You!