Improving Embedded Software Productivity with Auto Code Generation in Model Based Development approach

By

Nithinraj Rajendrakumar & Ganesh Parthiban

Embedded Systems
Renault Nissan Technology Business Centre India
INDEX

- RNTBCI – Brief Introduction
- Engine Management System - MBD approach
- Preparing for Auto Code Generation (ACG)
- Model Migration
- ACG – Impact on Productivity
- Pilot Project
- Lessons learnt from Pilot project
- Results - Auto code generation on live project
- Conclusion & Next steps
## RNTBCI – Brief introduction

<table>
<thead>
<tr>
<th>Headquarters</th>
<th>France</th>
<th>Japan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date of Establishment</td>
<td>1898 (114 years back)</td>
<td>1933 (79 years back)</td>
</tr>
<tr>
<td>Global presence</td>
<td>118 countries</td>
<td>160 countries</td>
</tr>
<tr>
<td>No. of employees</td>
<td>128,332</td>
<td>155,099 (apprx.)</td>
</tr>
<tr>
<td>Brands</td>
<td>Renault, Dacia &amp; Renault Samsung</td>
<td>Nissan and Infiniti</td>
</tr>
<tr>
<td>Design Centres</td>
<td>5 countries</td>
<td>11 countries</td>
</tr>
<tr>
<td>Manufacturing facility</td>
<td>17 countries</td>
<td>20 countries</td>
</tr>
<tr>
<td>2011 Group sales volume</td>
<td>2.7 Million Vehicles</td>
<td>4.7 Million Vehicles</td>
</tr>
<tr>
<td>2011 Group turnover</td>
<td>52,059 Million USD (apprx.)</td>
<td>118,950 Million USD (apprx.)</td>
</tr>
<tr>
<td>2011 Group net income</td>
<td>2,555 Million USD (apprx.)</td>
<td>4,320 Million USD (apprx.)</td>
</tr>
</tbody>
</table>

Renault Nissan Technology and Business Centre India Private Limited, is a 50:50 Joint Venture established under the Renault Nissan Alliance.

We cater to local & global needs of Nissan and Renault in various research and automotive technologies.

We have an in-house Software Development Centre.
Legacy EMS Architecture is already modular – No Auto code generation
MBD - Responsibilities

RENAULT FRANCE

System requirements

ECU requirements

Software architecture

Modélisation Of SWC

Validation System plan

Validation ECU plan

Validation SWC Plan

Validations MIL SWC

Validations SIL SWC

Validations HIL ECU

Validations MIL ECU

Validations System

ECU Supplier

RENAULT NISSAN TECHNOLOGY & BUSINESS CENTRE INDIA PRIVATE LIMITED
MBD – Need for model migration

- **Rules for MBD Strategies**
  - **Customized rules**
    - ~ 5% of rules adaptation to Renault specificities
  - **Optimized rules**
    - ~ 20% of rules optimized for automatic code generation
  - **Global Standard Rules**
    - (MAAB, Misra...)
  - ~ 75% of rules coming from Global standards

- **Rules for Coding Customization**
  - **Renault specific architecture**
  - **Multi Target Support (32 Bit)**
  - **Re-Use Library API**
  - **Arithmetic APIs**
  - **Multi configuration support**
  - **Fixed point code generation**

**MBD Tools**

**Suppliers:**
- Continental
- Valeo
- Bosch
MBD – Preparation for ACG

• Rules for MBD Strategies

  - Global Standard Rules (MAAB, MISRA…)
  - ~ 5% of rules adaptation to Renault specificities
  - ~ 75% of rules coming from Global standards
  - Optimized rules: ~ 20% of rules optimized for automatic code generation

• Renault specific architecture
• Multi Target Support
• 32 Bit Controller
• Re-Use Library API
• Arithmetic APIs
• Multi configuration support
• Fixed point code generation

Refining MBD Rule

• Specifications are compliant with MIL & SIL environments
• Specifications are compliant with automatic coding tools (RTW-EC)
• Specifications are still compliant with manual coding
Model Migration (EMS2010 ➔ EMS2012)

- Migration process is to apply all MBD rules and EMS software architecture on APL modules to be ACG compliance

- Customized “Model Examiner Tool” used to apply and verify model with respect to rules retaining the exact functionality
ACG – Impact on Productivity

- Impact on productivity
  - HIGH
  - MEDIUM
  - LOW
  - NO CHANGE
ACG – Productivity target

- Impact on productivity:
  - HIGH
  - MEDIUM
  - LOW
  - NO CHANGE

**Expectation**
Over all ~50% Productivity increase compared to manual coding

Input EMS2012 module

Requirement/Impact Analysis

Fixed point data typing

Conventional method data typing

Renault coding rules customization tool

Renault specific tools to check coding architecture compliance

1. Verifying all inputs
2. Tool compliance version

Test Case Generation using Reactis / SLDV

Functional Test Vectors

Test Results

Delivery of Code and SIL report
Pilot Project

- Process duration*

* Not to the scale
## Lessons learnt on Pilot Project

### CHALLENGES:

<table>
<thead>
<tr>
<th>✓ Floating point Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Restriction on block usage (ex gain block) due to Floating point to Fixed point conversion</td>
</tr>
<tr>
<td>- 64 bit data typing</td>
</tr>
<tr>
<td>- Data loss manipulation due to magic numbers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>✓ RAM/ROM Demands</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Scope for Model Designs optimization</td>
</tr>
<tr>
<td>- Data Typing in slope and Bias (high precision resolution and range)</td>
</tr>
<tr>
<td>- Creation of RTW inline function while code generations</td>
</tr>
<tr>
<td>- Re-Use library block for code generation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>✓ Adherence to Legacy development practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Fixed point tool box for data typing due to slope and bias</td>
</tr>
<tr>
<td>- Specification range saturation</td>
</tr>
<tr>
<td>- Adaptation of Renault coding architecture</td>
</tr>
</tbody>
</table>

### WORK AROUND:

<table>
<thead>
<tr>
<th>✓ Floating point Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Short term Certification process introduced</td>
</tr>
<tr>
<td>- Limiting Data type to 32 bit and manually code is implemented based for 64 bit data typing</td>
</tr>
<tr>
<td>- Data typing for magic number also to minimize data loss</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>✓ RAM/ROM Demands</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Short term Certification process introduced</td>
</tr>
<tr>
<td>- Solution under investigation</td>
</tr>
<tr>
<td>- Introduced TFL for all arithmetic calculations, but still for few arithmetic combinations RTW inline function is created.</td>
</tr>
<tr>
<td>- S-Function block is replaced while code generations for better reuse of library API</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>✓ Adherence to Legacy development practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Manual data typing being performed</td>
</tr>
<tr>
<td>- RTW was not saturating code based on specification requirement but with respect to data typing, so saturation block was added.</td>
</tr>
<tr>
<td>- Customization in RTW-EC code generation tool.</td>
</tr>
</tbody>
</table>
Lesson 1: Floating point algorithm

- Restriction on Gain block usage
- Possible data loss during Floating point to Fixed point conversion.
- There is no standard rule to define the data type for the parameter value.
Lesson 2: RAM/ROM Demands

- RAM/ROM consumption had an average increase of 35% to 40% in ACG compared to manual code
- Targeting to bring down memory consumption by
  - Optimizing the model design
  - Refining rules for optimized code generation

Example: Scope for Model Designs optimization

11 lines of code

3 lines of code
Lesson 3: Reworking on models

The generated code does not limit the data to max value of 200 as mentioned in the signal attributes.

So, Manual addition of saturation block is necessary to comply with Auto Code Generation.
ACG implementation on Live Project

CM was chosen as a first module for deploying ACG process
Results – Auto code generation on Live Project

Average Productivity improvement achieved for **coding & Unit Testing** is 59% for 10 modules on live project
Conclusion & Next steps

Overall experience with ACG process resulted in satisfactory productivity improvement and brought out other areas of further improvements.

Next planned improvements:
- Introduction of SLDV
- Using Fixed point tool box
- Extension of ACG process to other ECU sw development
Thank You !