PC-Based Validation of ECU Software

Sean Wyatt - ETAS, Inc.

Venue: Oakland University School of Engineering and Computer Science, Rochester, MI
Date: October 13, 2012
What you will learn

- Motivation for PC-Based Development
- How to create MiL and SiL platforms using ETAS INTECRIO software;
- How to calibrate C-code using production tools like ETAS INCA;
- How to improve productivity, reduce cost by leveraging the PC platform;
Agenda

- Why PC-based Development
- Definitions
- What is INTECRIIO?
- Model-in-the-loop with INTECRIIO
- Software-in-the-loop with INTECRIIO
- Other ETAS Solutions
- Summary
Agenda

- Why PC-based Development
  - Definitions
  - What is INTECRIO?
  - Model-in-the-loop with INTECRIO
  - Software-in-the-loop with INTECRIO
  - Other ETAS Solutions
  - Summary
Why PC-Based Development?

Consumption and emission requirements

Increased inter domain connectivity

New layered standard architecture

Engine control complexity

0% 100%


Effort

Number of calibrations

ECU

New functions and variants

Engine Management
Transmission Management
Battery Management
Vehicle Motion Management
Brake Management

Consumption and emission requirements
Why PC-Based Development?

- Simulation: Today – 5% to MATH, 2015 – 20%
- Benches: Today – 25% to LAB, 2015 – 40%
- Vehicles: Today – 70% from ROAD, 2015 – 40%
Why PC-Based Development?

“Road to lab to math is basically the idea that you want to be as advanced on the evolutionary scale of engineering as possible. Math is the next logical step in the process over testing on the road and in the lab. Math is much more cost effective because you don’t have to build preproduction vehicles and then waste them. We’ve got to get out in front of the technology so it doesn’t leave us behind. We have to live and breathe math. When we do that, we can pass the savings on to the consumer.”
- A senior auto executive

Summary: Development hardware (ECUs, vehicles etc.) is expensive and PCs are more powerful and inexpensive than ever.
Why PC-Based Development?

Automotive industry is already leveraging PC-based development for a range of activities, e.g.:
- software functional testing and V&V
- legacy c-code simulation on the PC
- target-like timing analysis (RTOS)
- calibration development
- system development using plant models
- complete ECU hardware emulation

→ But more can be done.
Several studies...

Electronic virtualization for the US auto industry: Is survival in the balance?

Virtualization Tools Aims to Help Automotive Developers Manage Software Complexity

Virtual prototypes will enable automotive developers to tackle growing software intricacies and speed their time to market.

The Evolution of GM’s Revolution — How Math Made it Happen

The recent changes in vehicle design are incredible. The benefits of math-based tools become clearer when you take a look at the way it began.
Agenda

- Why PC-based Validation
- Definitions
- What is INTECRIO?
- Model-in-the-loop with INTECRIO
- Software-in-the-loop with INTECRIO
- Other ETAS Solutions
- Summary
Definitions

Virtualization
From Wikipedia, the free encyclopedia

In computing, virtualization (or virtualisation) is the creation of a virtual (rather than actual) version of something, such as a hardware platform, operating system, storage device, or network resources.[1]

Our Definition: A virtual ECU development environment is characterized by the absence of physical hardware, i.e. PC-based representation of all components, with or without real-time execution
Evolution of Virtualization

ECU Application Models, Software

Driver, Vehicle and Environment Models

Virtual Environment

Real ECU

Real Vehicle

Real Environment
What’s included for today?

Included:

- ECU algorithm models or embedded code
- ECU real-time operating system (RTOS) and other basic software components
- Physical systems (e.g. engine, transmission, body, chassis, driver, environment models)

Not included:

- Virtualization of the ECU hardware (e.g. microcontroller, peripherals)
- Virtualization of the physical buses (e.g. CAN, FlexRay)
Steps to Create a Virtual Environment

The main ingredients:
- Physical Models or stimuli
- ECU Algorithm Models
- ECU Embedded Code

The glue and tools necessary:
- Integration and Build Environment
- Execution Platform
- Instrumentation

The development tasks you are interested in:
- Verification and Validation
- Calibration and Measurement
- Automatic Testing
Model-in-the-Loop (MiL) Simulation

- Defined as the ECU algorithm model “in-the-loop” with the plant models
  - May be run within the modeling environment, or

  **Physical Models or stimuli**
  **ECU Algorithm Models**

  **Connect models in Simulink**

  **Run in Simulink under Windows**
Model-in-the-Loop (MiL) Simulation

- Defined as the ECU algorithm model "in-the-loop" with the plant models
  - May be run within the modeling environment, or
  - Run as an executable under Windows - from model-generated code
Software-in-the-Loop (SiL) Simulation

• Defined as the ECU embedded software “in-the-loop” with the plant models
  - May be run within the modeling environment, or
Software-in-the-Loop (SiL) Simulation

- Defined as the ECU embedded software “in-the-loop” with the plant models
  - May be run within the modeling environment, or
  - Run as an executable under Windows

1. Physical Models or stimuli
2. ECU Algorithm Models
3. ECU Embedded C-Code

- Integrate model-code with ECU code and compile
- Run executable under Windows
Agenda

- Why PC-based Validation
- Definitions
- What is INTECRIO?
  - Model-in-the-loop with INTECRIO
  - Software-in-the-loop with INTECRIO
- Other ETAS Solutions
- Summary
Virtual Setup for ECU Development

Environment (Road, Wind, etc.)

Driver

Subsystem (e.g. Engine)

Subsystem (e.g. Transmission)

Subsystem (e.g. Chassis)

Vehicle Network (e.g. CAN)
Virtual Setup for ECU Development

- **Environment** (Road, Wind, etc.)
- **Vehicle**
  - Subsystem (e.g. Engine)
  - Subsystem (e.g. Transmission)
  - Subsystem (e.g. Chassis)
- **Driver**
  - Set-Point (Sensor)
- **Virtual Sensor/Actuator Signal Bus**
- **Vehicle Network (e.g. CAN)**

**Diagram**
Virtual Setup for ECU Development

Environment
(Road, Wind, etc.)

Vehicle Subsystem Models
(e.g. Engine, Transmission, Chassis)

Driver
Set-Point (Sensor)

Virtual Sensor/Actuator Signal Bus

Vehicle Network (e.g. CAN)
Virtual Setup for ECU Development

Environment
(Road, Wind, etc.)

Vehicle Subsystem Models
(e.g. Engine, Transmission, Chassis)

Driver

Set-Point (Sensor)

Virtual Sensor/Actuator and Network Signal Bus

function1

function 2

function 3
What is INTECRIO?

- Driver Model
- Environment Model

Vehicle Subsystem Models
(e.g. Engine, Transmission, Chassis)

Virtual Sensor/Actuator and Network Signal Bus
What is INTECRIO?

Plant Models
Driver, Vehicle, Environment

Application Software
Design and Specification

INTECRIO
RTA-OSEK for PC Target

HEX and A2L files
Key Takeaways

■ PC-based development of ECU software is becoming more widespread due to cost and performance advantages

■ INTECRIO is a versatile platform for PC-based verification and validation (e.g. in MiL and SiL phases)
Agenda

- Why PC-based Validation
- Definitions
- What is INTECRIO?
- **Model-in-the-loop with INTECRIO**
- Software-in-the-loop with INTECRIO
- Other ETAS Solutions
- Summary
INTECRIO for MiL

Simulink

Embedded Coder

R2007a

Simulink

R2011a

Simulink Coder

Hex and A2L

Virtual Prototyping
INTECRIO for MiL
INTECRIIO for MiL: Demo Setup

- **Goal:** Simulate a Simulink-based plant (throttle) + Simulink throttle controller specification.
Agenda

- Why PC-based Validation
- Definitions
- What is INTECRIO?
- Model-in-the-loop with INTECRIO
- Software-in-the-loop with INTECRIO
- Other ETAS Solutions
- Summary
INTECRIO for SiL – 2 steps

Description files (ASAM-MCD, AUTOSAR, OSEK-VDX)

.c, .h files

INCODIO

Software Component

Simulink Models

Software Component

ASAP2, OIL

INTECRIO

HEX and A2L files

Embedded System Workshop
INTECRIO for SiL – Step 1 with INCODIO

#include "throttle.h"

float Perc_Desired_Position;
float Perc_Actual_Position;
float Perc_Actual_Position_Unfiltered;

const float Pos_Offset = 45.9;
const float Pos_Gain = 13.513;
const float Rate = 0.01;

void Throttle_Controller_Init(void)
{
    P_term = 0.0;
    I_term = 0.0;
    D_term = 0.0;
    D_temp = 0.0;
}

void Throttle_Controller(void)
{
    float pos_error;
    pos_error = Perc_Desired_Position - Perc_Actual_Position;
    P_term = K_S * pos_error;
    I_term += DT * P_term / T_s;
    D_term = (I_d_S * P_term + D_temp * D_Gain) / T_v_S;
    D_temp += D_term * 0.01;
    PID_control = P_term + I_term + D_term;
}
INTECRIIO for SiL

Demo
INTECRIO for MiL + SiL: Demo Setup

- **Goal:** Verify that the C-code implementation of the Throttle Controller matches specification, reusing the stimulus and plant Simulink models.
Calibration of C-code with INCA
Key Takeaways

- **MiL**: INTECRIO integrates one or more Simulink models at the code level, adds scheduling information and compiles for PC target.

- **SiL**: INCODIO parses C-code and makes it visible to INTECRIO for integration with models or other C-code. INTECRIO adds scheduling information and compiles for PC target.

- **Calibration**: An industry standard tool like INCA can be used to pre-calibrate ECU software in the SiL environment.
Agenda

- Why PC-based Validation
- Definitions
- What is INTECRIO?
- Model-in-the-loop with INTECRIO
- Software-in-the-loop with INTECRIO
- Other ETAS Solutions
- Summary
INCA with INCA-SIP

Algorithm Engineer

Simulink®
Control & Plant Models
Legacy C-code

INCA-SIP

API

XCP on Ethernet

Windows
INCA with INCA-VLINK

Calibration Engineer

INCA-EIP

TCP/IP on Ethernet

INCA-VLINK

Simulink®
Control & Plant Models

Legacy C-code

compile

.dll

INCA

Windows

Embedded System Workshop
ISLOAR-EVE* (ETAS Virtual ECU)

*Product not yet released
Agenda

- Why PC-based Validation
- Definitions
- What is INTECRIO?
- Model-in-the-loop with INTECRIO
- Software-in-the-loop with INTECRIO
- Other ETAS Solutions

**Summary**
Summary

- Virtual techniques can substantially lower the cost of verification and validation of ECU control models and software.

- Calibrators can use the same tools in the virtual environment as in the vehicle.

- Re-use of test cases, models and calibration data across the development cycle further optimizes cost and effort.

- AUTOSAR architecture further enables validation of ECU software on the PC platform.
Where to find more information?

1. Virtualization in Automotive Embedded Systems
   http://www.slideshare.net/REALTIMEATWORK/rts10-virtualization

2. Automotive: The Next Frontier for Virtualization
   http://www.ok-labs.com/blog/entry/automotive-the-next-frontier-for-virtualization/

3. Virtualization Technologies for Cars
   http://marko-wolf.de/files/PeWoWo08_ECU_Virtualization_EN.pdf

4. Road-to-lab-to-math: a new path to improved product
   www.sae.org/automag/features/futurelook/05-2005/1-113-5-78.pdf

5. SAE papers:
   - 2010-01-0660: Maximizing Test Asset Re-use across MiL, SiL and HiL Development Platforms
   - 2010-01-0431: Advanced Techniques for Simulating ECU C-code on the PC
   - 2012-01-0961: Model-Based Verification and Validation of Electronic Engine Control
Skills and Networking Opportunities

1. Skills required to leverage PC-based development:
   - Controls development (e.g. MATLAB/Simulink®)
   - Software development (e.g. ASCET, C-code)
   - Plant modeling (i.e. physical systems)
   - Verification and validation (i.e. testing of embedded software)
   - Understanding of AUTOSAR architecture

2. Networking opportunities:
   - SAE World Congress (Detroit): April 16-18, 2013
Thank you!

감사합니다.

有難うございました

Merci

Vielen Dank