Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Model-Based Implementation Design of Automotive Controllers

Shigeru Oho*, Makoto Ishikawa*, George Saikalis**

**Central Research Laboratory, Hitachi, Ltd., Kokubunji, Tokyo*
Japan (e-mail: shigeru.oho.ku@ Hitachi.com).
***Hitachi America, Ltd., Farmington Hills, MI48335*

Abstract: Model-based development was introduced to implementation design of automotive electronic controllers. The goals and issues of implementation design were discussed, and a model-based approach of processor-in-the-loop simulation was proposed as a method to meet the design goals. An application example of electronic throttle control of automotive engines was demonstrated with the processor-model.

1. INTRODUCTION

Model-based development (MBD) applies to a whole spectrum of development activities. In basic design, where control system concepts are proposed and proved, there are number of successful MBD practices (e.g., Mutz et al., 2002). MBD is also useful for implementation designs, where automotive controllers are actually designed in complete detail; control software is generated and validated, microcontrollers are chosen to execute the control codes, electronic circuits are developed and examined for sensor interfaces, actuator drivers, communication busses, and so forth. A better part of these implementation designs has been, however, carried out though conventional non-MBD processes due presumably to the lack of practical modeling means of microcontrollers and control software. The advance in processor modeling technology now allows us to apply MBD to implementation designs (Saikalis et al., 2006, Ishikawa et. al., 2007). In this paper we first point out the design issues in the implementation process of automotive controllers, and then discuss how the MBD method resolves them. To demonstrate the implementation MBD, a whole system of an electronic throttle control of automotive engines was modeled with a virtual microcontroller, and its system behavior, control signals and software run were analyzed.

2. IMPLIMENTATION DESIGN

2.1 Development process

Development process of control systems begins with a basic (upstream) design for a given set of system requirements, then proceeds to implementation design (downstream). Basic design is responsible for establishing a control system architecture and control strategies to meet the system requirements. On the other hand, the goal of implementation design is not to improve control performance but to realize specified performance in a real world. Implementation design also ensures reliable system operation and achieves the lowest cost.

In a modern MBD approach control designs are formulated into executable specifications in the form of signal-flow diagrams and/or state-transition charts. Once a control design is proved in basic design, then it is implemented into an electronic control unit (ECU) for mass-production. An automatically generated object code bridges basic and implementation designs. The control code embedded in the microcontroller in the ECU interacts with the control plant and achieves the specified control functions.

2.2 Implementation issues

Ideal implementation design should not change the control system behavior at all, and the system performance must be maintained as achieved in the basic design. The most important goal of the implementation design is to ensure correct and accurate execution of the control algorithms that are established in the basic design. Control code execution in the ECU is not always identical to the one on the PC used for the basic design, however. The embedded microcontroller in the ECU often uses coarser data expression and runs at slower speed in computing, and thus can yield a subtle difference in control code execution.

Implementation design is also responsible for meeting reliability requirements of the control systems. ECUs have such potential causes of system malfunctioning and total system down as circuit component failure, loss of contact in wiring harnesses, timing jitter in control signals, and so on. Cutting the cost of an ECU is also very important goal in implementation design, too. Due to cost pressure ECU engineers may cling to an out-of-date microcontroller that could be easily overloaded with ever-inflating modern control software.

3. PROCESSOR-IN-THE-LOOP SIMULATION

The implementation issues raised above can be better approached if a whole control system, including an ECU and its microcontroller, is modeled into a virtual system. Virtualization of control systems brings us an obvious benefit of visualization; it allows us to examine every nook and cranny of a control system as closely as needed. Visualization is extremely useful, in particular, for execution analysis of control codes because today's single-chip microcontrollers permit very limited access to trace data for debugging. In-circuit emulation to track microcontroller operations, a conventional hardware-based system development method, is becoming less useful as microcontrollers integrate more functions and run at higher clock speed. Once an ECU is integrated into an electro-mechanical system, its internal operation becomes very difficult to monitor.

Control engineers have been using numerous tools to generate executable specifications of control strategies, to simulate electronic circuits, and to model sensors, actuators and control plants. The advance in simulation technology has finally linked the missing connection between these hardware models and the control software; a virtual processor simulating a microcontroller executes the object code of control software, and the software precisely interacts with the control hardware as they do in the real world. In Fig. 1, we built a virtual embedded mechatronic system of automotive engine throttle control. In this demonstration, the control plant was modeled with Saber (Synopsis), and the microcontroller was virtualized with CoMET (VaST). These simulation tools were interfaced with each other to realize a co-simulation environment. Fig. 2 and 3 show the observed signals in the co-simulation.

4. DISCUSSION

To examine control operations of the mechatronic system, we looked at the response of valve rotation in the throttle control demonstration. To study the behavior of a PID controller, we looked into the control commands that the processor model generated. To verify the control code execution, we checked the task switching by tracing the program counter value. To analyze the system reliability, we injected a failure to, e.g., the H-bridge driver and observed the consequence. Thus, the virtual approach allowed us to conduct numerous what-if scenarios to complete the implementation design.

The microcontroller shown in the Fig. 1 does not actually exist. It is a hybrid of two commercial chip designs (ARM processor and Renesas H8S peripherals). The virtual microcontroller demonstration suggests an idea of application-in-the-loop simulation approach to test new processor cores and/or peripheral functions in a given control system.

5. CONCLUSIONS

A processor-in-the-loop simulation was demonstrated as a method of model based development for implementation design of electronic controllers. The virtual approach should also be useful for microcontroller development as application-in-the-loop simulation.

REFERENCES

Mutz, M., Huhn, M., Goltz, U., and Kromke, C. (2002). Model Based Systems Development in Automotive. *SAE Paper* 03B-128.

Saikalis, G., Meyl, H., Oho, S., McCune, D.J, and Ishikawa, M. (2006). Virtual Embedded Mechatronics System. *SAE Paper* 2006-01-0861.

Ishikawa, M., McCune, D.J, Saikalis, G., and Oho, S. (2007). CPU Model-based Hardware/Software Co-design for Real-Time Embedded Control Systems. *SAE Paper* 2007-01-0776.

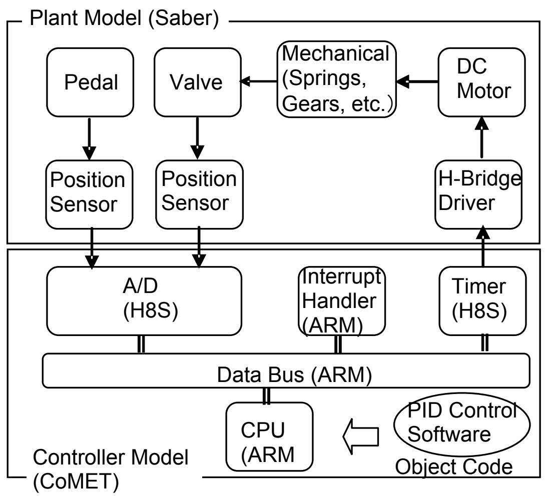

Fig. 1. Processor-in-the-loop simulation

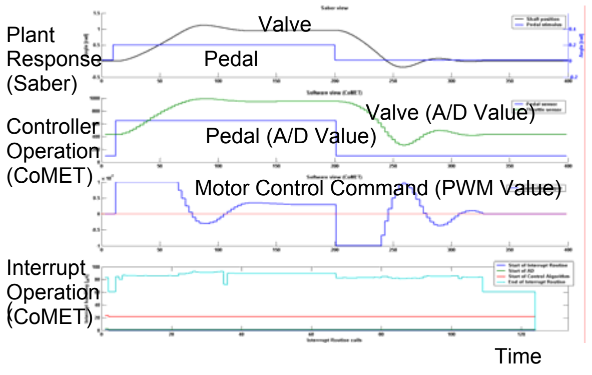

Fig. 2. Plant Response and Controller Operation

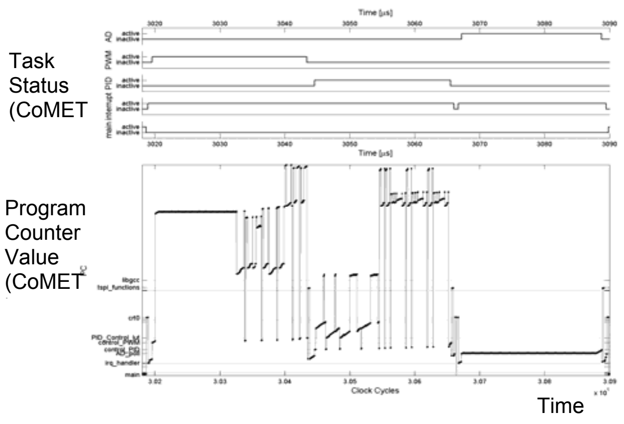

Fig. 3. Task Run Analysis