ECU Development at Renault

Renault’s development process integrates rapid prototyping and model-based design

By Thomas Omwara, Renault SA

At the Renault Powertrain Control and Calibration department (DIMDCMAP) in Lardy, France, ECU software developers have integrated model-based design and prototyping into their development process. ETAS tools play an important supporting role throughout ECU software development at Renault.

Our mission at the Renault Test Center is to design and develop embedded software for powertrain units. This mission also involves developing new control strategies that continue to improve the performance of powertrain units and using efficient methods to shorten development cycles. To be able to fulfill all aspects of that mission, developers at the Powertrain Control and Calibration Department (DIMDCMAP) introduced rapid prototyping and running model-based software design into their development process. The two methodologies enable us to gain better control of our ECU software design in the least amount of time and with higher software quality.

Prototyping at Renault

Prototyping systems enable our developers to test and validate new control strategies designed for Renault engines and automatic gearboxes. An example is the suppression of a sensor signal that updates a variable that in turn is determined by a new strategy (algorithm for calculation). Another example is a new after-treatment strategy such as an NOx trap, or a particulate filter (CSF). In order to make a new control strategy work with the overall engine control, we use a rapid prototyping system. Rapid prototyping systems offer a suitable hardware and software environment for intensive modifications, updates, and corrections. Such systems allow developers to begin in-vehicle testing early in development where it is necessary to be able to quickly switch back and forth from the development phase to the execution phase.

The development phase involving the prototyping of strategies for engine control is located at the base of the small V in the process shown in Figure 1. We use the prototyping of strategies to do an early validation loop. This gives us a good foundation for managing the remaining phases of a development cycle.

During this phase, we use two kinds of prototyping methods: rapid prototyping and “mock up.” Which of the two methods we use depends on whether or not we have read access to the C source code.

However, if we develop new strategies that we need to test, we use an open interface tool to communicate with the controller. ETAS ES1000/1135 supports testing new control strategies, while the existing embedded software runs on the serial microcontroller. This method allows our developers to validate functional aspects of their work.

Model-based design avoids the redundancy built into the specification-based design process and saves time. In addition, based on this approach, tools offering simulation, verification, validation, and documentation options can be used throughout development. Indeed, every piece of information is translated through the model (algorithms and the data dictionary) and the data dictionary (calibrations and data management). The two items together are the base for our development process at Renault. All phases of the development process—from simulation to functional validation—depend on the pair “Model and Data Dictionary.” However, the effective use of model-based design and rapid prototyping throughout the development process also depends on tools with open interfaces.

Renault as software architect

At Renault, we design our software with reusability in mind. Model-based design is an excellent method for developing modular control strategies. Two examples: the model of an engine start-up strategy was developed in ASCET/M and its software prototype was tested on an M9R diesel engine. The model of an idle-control strategy was developed in MATLAB/Simulink® and its software prototype was tested on a K9K diesel engine. Although the control strategies were created in two different modeling tools, either software prototype can be run independently on either of the two diesel engines (Figure 2). And open interface tools such as the integration platform INTECRIO give us great flexibility in terms of how we work with our prototyping systems and thus our powertrain control systems.

Introducing model-based design and open interface tools into the development process gives us overall control of our command strategies. All these measures support Renault to position itself as the architect of its software and facilitate the cooperation between automotive manufacturers and suppliers.