Virtual ECUs Used to Develop Renault's Engine Management Software

© QTronic

In 2016, Renault started to use virtual Electronic Control Units to aid the development of engine management software. Developers of the OEM can now simulate and calibrate the entire engine control on a PC even before real ECU hardware with production C code becomes available. Renault and QTronic describe how this changes the development process.

MOTIVATION

Renault develops most of the Engine Management Software (EMS) for its diesel and gasoline engines in-house. To this end, a model-based development process based on Matlab/Simulink has been established since 2010 [2-4]. This process is shown in **FIGURE 1**.

In 2014, a critical assessment of the process revealed a weak point: When function developers edit one of the 200 modules of the EMS, they must wait for weeks until they receive the results of ECU tests, **FIGURE 1** (blue line). Until then, only the testing of single modules (green line) on PC is possible. However, module tests take only few interactions of the module under test with other modules of the EMS into account. Consequently, integration problems are often discovered late, when ECU test results become available. It would be much better to test a module in full system context immediately after a change of the module and before the module is released to the next process step. Renault then decided to integrate virtual ECUs (vECU) into the development process to enable early validation and test of modules in system context, **FIGURE 1** (red line).

To implement the idea, Renault invited leading providers of tools for the virtualization of ECUs to participate in a benchmark. Participants were asked to use their tool to setup an incremental process for building a virtual ECU from 200 modules (Simulink models) of the EMS. Main selection criteria were the time needed to configure an initial setup, the time needed for the incremental rebuild of a vECU and the execution speed of the resulting vECU. In this process, the vECU tool Silver (by QTronic [1]) was selected to implement the desired improvement.

VIRTUALIZATION OF THE RENAULT EMS

Today, Model-based Development (MBD) on PC is the established method of choice for developing control software for automotive powertrains. Surprisingly, the dominance of MBD does not mean that developers are typically able to sim-

AUTHORS

Yohan Jordan
is Development Engineer for Powertrain Control Software Tools at Renault S.A. in Paris (France).

Dirk von Wissel
is Expert in Powertrain Control Functional Architecture at Renault S.A. in Paris (France).

Adrian Dolha
is Engineer at QTronic Software S.R.L in Cluj-Napoca (Romania).

Dr. Jakob Mauss
is Managing Director at QTronic GmbH in Berlin (Germany).

ulate their ECU on PC, even if they have full access to all models of the ECU. For example, loading and initializing all 200 modules of a typical Renault EMS into 64-bit Simulink takes more than 10 h. Interactive simulation of a module in full ECU context is clearly out of reach then. The only way to achieve reasonable execution times is to apply compilation techniques, either within the Simulink environment, or (as done here), in a Simulink external integration environment, based on exported C code. As the case study presented here shows, such a compiled environment is not straight forward to set up. In our case, it took many months. It seems that developers have therefore widely accepted the fact, that they cannot run the ECU model inside their model-based IDE. Instead, they must wait until the C code generated by the models has been integrated with the target ECU hardware, and the ECU software can be executed on an HiL system. Of course, this practice fundamentally contradicts the idea of MBD which is about executable models whose behaviour can be explored and assessed during design.

Renault has now implemented an incremental process to build virtual ECUs. Incremental means to build a vECU from given 200 modules by generating C code only for those modules that have been edited since the previous build. This way, an vECU can be built within a few minutes.

The build process is shown in **FIGURE 2**. For each of the 200 modules, a Simulink wrapper model is generated automatically that fixes the datatype of each input and output to a configured datatype and provides a signal interface to the Operating system (Os). The wrapper model and the wrapped module is then translated into C code. For this step, Simulink Coder is used, not Embedded coder which is more expensive and hence not available for function developers. Finally, the resulting C code is compiled for Windows PC to be executed by the vECU using Silver's Os. The Silver Os supports periodic and event-triggered tasks, such as initial and crankshaft-synchronous ones.

The resulting vECU can be executed with different calibration data sets. Calibration data is read by the vECU from the file system at runtime. This enables pre-calibration of all EMS modules on PC. In Silver, the vECU typically runs in closed-loop with an engine model, **FIGURE 3**. Renault currently uses models

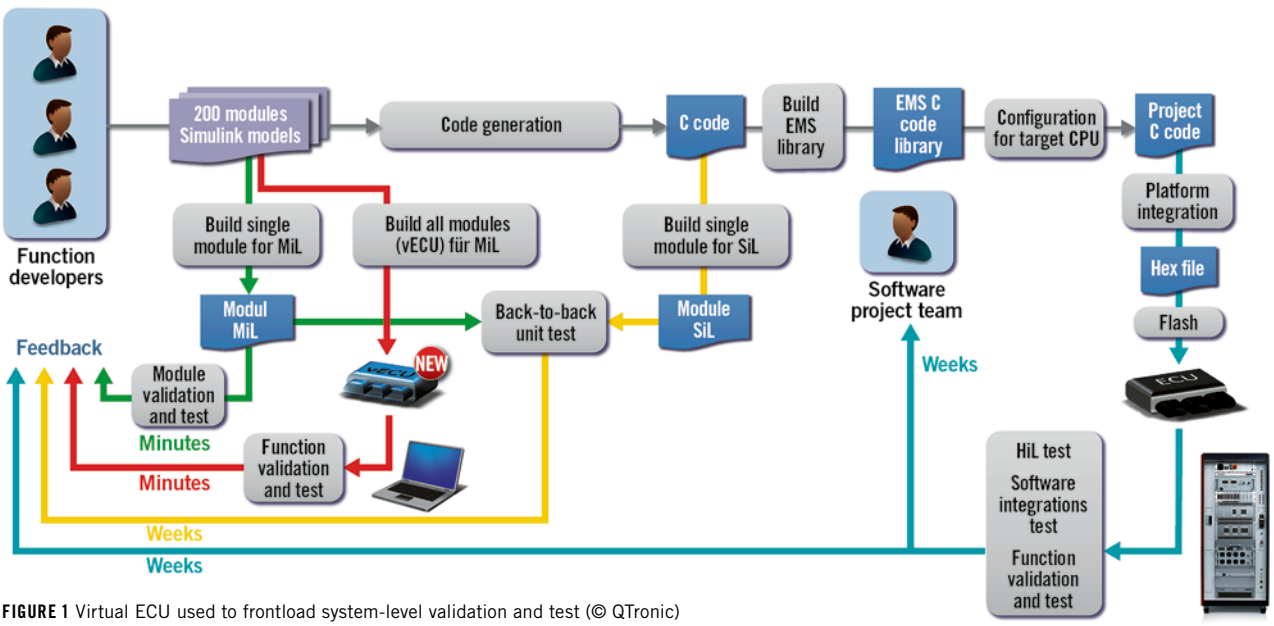

FIGURE 1 Virtual ECU used to frontload system-level validation and test (© QTronic)

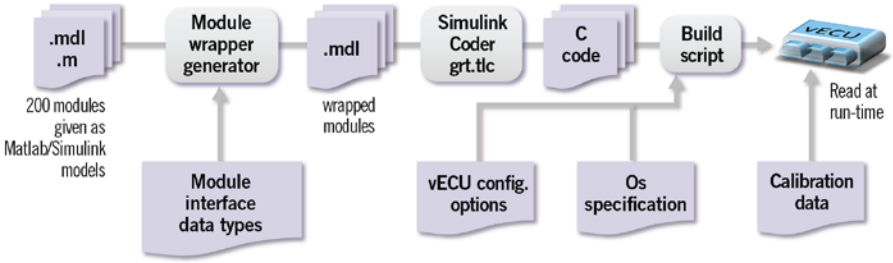

FIGURE 2 Automated process to build a virtual ECU (© QTronic)

developed with LMS Amesim (Siemens) for use on HiL test systems. Such models also run in Silver. This way, control loops of the EMS can be analysed and tuned on PC.

Silver loads and initializes a vECU of the Renault EMS in less than 5 s. Execution time depends mostly on the number of variables measured during simulation. When measuring just 170 variables per ms, the vECU runs four times faster than real time on a typical PC. When measuring 20000 variables per ms, execution time is three times slower than real time.

DIFFERENCES BETWEEN REAL AND VIRTUAL ECU

A vECU is a model of the real ECU. Consequently, not all properties of the real ECU are visible in the virtual ECU. This means that not all tests that are relevant can also be moved to the PC. For some tests, access to real ECU hardware is needed. The most crucial differences between real and virtual ECU are:

- Timing: The virtual ECU behaves like a device with unlimited computing power. Silver does not attempt to predict the time that a task would take to run on the real ECU hardware. Instead, Silver assumes that the task runs entirely within a single point in time, not a time interval. Consequently, a task running on a vECU cannot be interrupted by another task. Instead, the vECU runs all tasks exactly at the configured time points, either periodically or at specified events. A Silver vECU can therefore not be used to check whether the real ECU provides sufficient computing power to meet given real time requirements.
- Basic software: In the way it is employed to interact with sensors, actuators and buses, basic software is not hosted by the vECUs currently used by Renault for engine development. Consequently, basic software cannot currently be tested using vECUs. This is by no means a restriction of vECUs as such but rather a consequence of the work flow shown in **FIGURE 1**: A vECU is derived here early from given Simulink models, while basic software becomes only available weeks later during platform integration.
- Production code: A vECU as used here runs C code generated with Simulink Coder, while the real ECU runs C code generated with Embedded Coder. Both program codes differ and might hence show different behaviour at runtime. Again, this is not a restriction of vECUs, but a consequence of the build process depicted in **FIGURE 2**.

APPLICATIONS OF VIRTUAL ECUS AT RENAULT ENGINE DEVELOPMENT

In 2016, Renault created the first fully functional virtual EMS using the process described above. The process has been repeated for about six releases (updates and different platforms) of the EMS software since then. The following applications of virtual ECUs have been implemented so far.

MODULE DEVELOPMENT IN SYSTEM CONTEXT

A virtual ECU can be configured at compile time for co-simulation with Simulink. At runtime, Simulink run then the module selected at compile time, bypassing the corresponding module in the vECU, while all other 200 modules are executed by the vECU. The module developer can edit his module in Simulink and run the resulting virtual ECU immediately, without rebuilding it. This way, the effect of an edit on behavior can immediately be seen by the developer in the context of all modules of the ECU. Rebuild of the vECU is only required when the signal interface for the edited module changes, or to get more recent implementations of other modules into the validation loop. Thanks to incremental build, such a rebuild takes only minutes.

PRE-CALIBRATION

About 50 % of the development time for engine controllers is spent for engine calibration, i.e. tuning of thousands of parameter, characteristic curves and maps of the engine control software. With virtual ECUs, Renault started to frontload calibration related activities as well. This is called pre-calibration. The objective is to develop better start values for calibration to gain more time for fine tuning in later phases of development, when calibration is performed using real hardware. A Silver virtual ECU loads calibration data at runtime from a human readable text file. A calibration engineer can then vary parameters dynamically during closed-loop simulation using sliders provided by Silver (online tuning).

VIRTUAL INTEGRATION OF MODULES INTO THE ECU

Virtualization enables all 200 modules to be integrated into a single virtual ECU and the entire EMS to be tested in closed loop with an engine simulation even before generating production C code. This way, integration problems and errors in modules can be detected weeks earlier. To exploit this idea, simulation results recorded on an HiL test system are currently compared with simulation results computed on PC using a virtual ECU for sufficiently similar versions of EMS software and calibration data. Unexpected differences quickly point to potential problems.

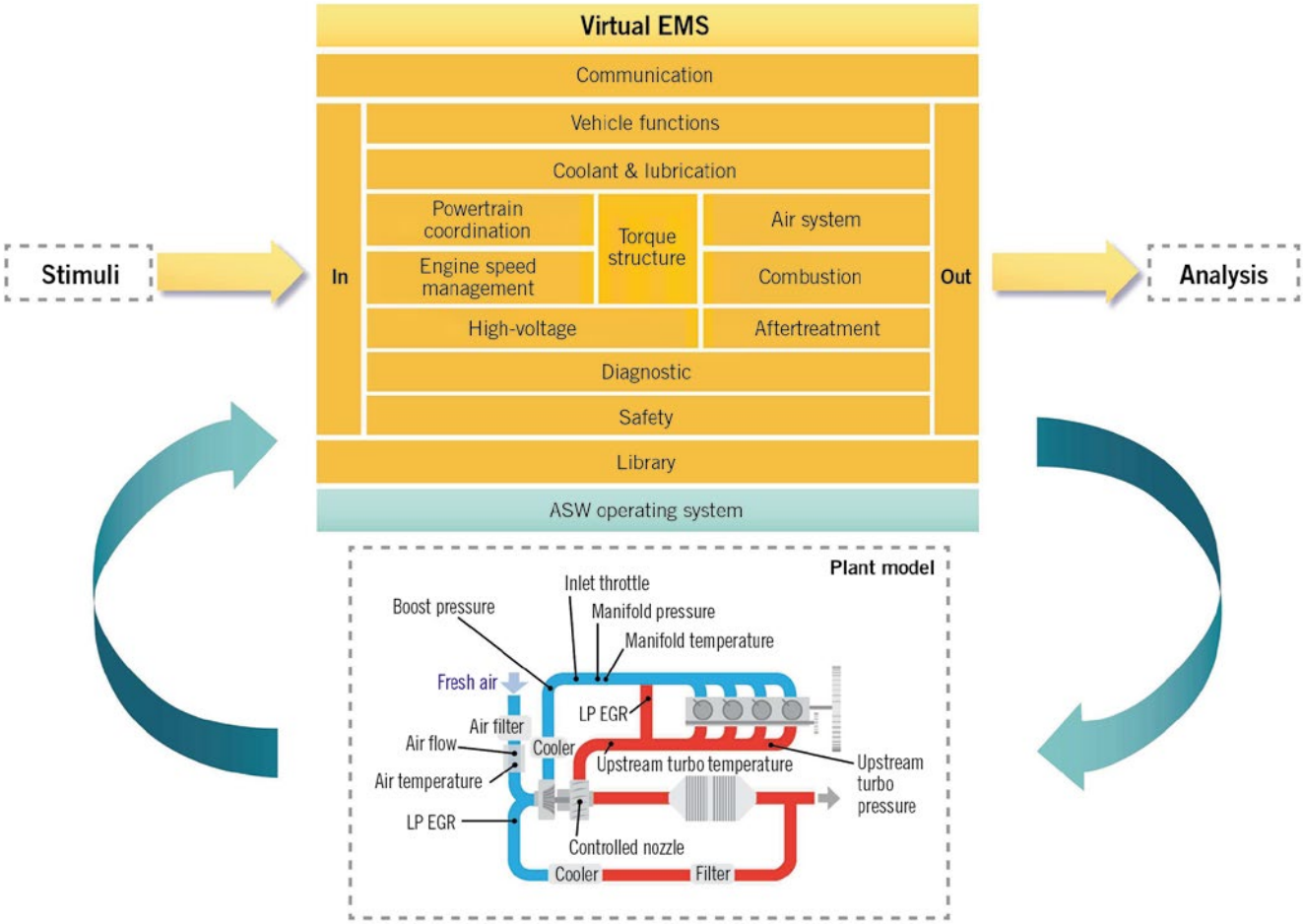

FIGURE 3 Virtual ECU of Renault EMS connected to engine model (© QTronic)

SUPPORT JOINT DEVELOPMENT WITHIN THE RENAULT-NISSAN PARTNERSHIP

Renault and Nissan jointly develop certain EMS modules. The development of a shared set of reusable EMS modules requires rework of interfaces and methodologies on both sides. In this context, Silver virtual ECUs are currently used as a platform for virtual integration of modules. This helps to discover and eliminate integration problems earlier.

VEHICLE-LEVEL SIMULATION

Renault and Nissan use the Digital Electronic Integration PlatForm (D-EIPF [5]) to validate networked ECUs in vehicle context. D-EIPF has been developed in-house since 2010. With D-EIPF, the communication behavior of networked ECUs of an entire vehicle can be simulated without the need to access real ECU or vehicle hardware. Silver has been integrated into the D-EIPF environment. This way, much more realistic engine models become available in D-EIPF, which increases the scope of tests that can be executed on D-EIPF.

A vECU could also be used to validate and quantify system requirements for projected engine systems. However, this is currently not a use case at Renault.

CONCLUSION

Renault started to use virtual ECUs to frontload test and calibration related activities during the development of engine management software. This helps to detect problems earlier, increases the quality of the models and shortens development cycles.
In a next step, the existing MiL-based virtual ECUs will be complemented by SiL-based virtual ECUs. The latter are based on production C code and will contain basic software as well, which further minimizes the behavioral gap between real and virtual ECUs. This will make it possible to move even more development steps to the virtual platform.

REFERENCES

[1] Linssen, R.; Uphaus, F.; Mauss, J.: Simulation of Networked ECUs for Drivability Calibration. In: ATZelektronik worldwide (2011), No. 4, pp. 16-21
[2] von Wissel, D.; Moreno Lahore, P.: Renault Model-Based Design – Powertrain control development process. 23rd International AVL Conference Engine & Environment, Graz, Austria, September 8 to 9, 2011
[3] Dressler, J. M.: A Walk through EMS 2010 Modular Software Development. 4th European Congress ERTS, Toulouse, 2008
[4] von Wissel, D. ; Quelin, J-M.: Industrial use of HIL Engine Management System validation. 9th Symposium Automotive Powertrain Control Systems, Berlin, September 20 to 21, 2012
[5] Watanabe, A.; Sotome, A.: Functional Development Methodology for On-Board Distributed ECU Systems for Production Vehicle Application. In: SAE Int. J. Passeng. Cars – Electron. Electr. Syst. 5(2):492-500, 2012, https://doi.org/10.4271/2012-01-0929