

Mercedes-AMG Develops an On-Target Rapid Prototyping Platform

PC-based rapid prototyping within Model-Based Design is widely recognized as a quick, efficient way to test ideas and experimental solutions early in development, when they are easier and less costly to fix. Automatic embedded code generation takes rapid prototyping one step further: Once the design is refined and validated, engineers can automatically generate code from the model for real-time prototyping and deployment on the target system. When developing new designs intended for production use, engineers can make their ideas more comprehensible to suppliers, resulting in more effective, improved implementation.

Mercedes-AMG GmbH chose this approach when they modified a production electronic control unit (ECU) by adding sensors and actuators, and then rerouted signals through a third-party programmable controller during prototype development.

In this article, Sergej Kisin, a developer in the Vehicle Electrics/Electronics group at Mercedes, explains how his team replaced handwritten code with automatically generated code to implement controllers on hardware for on-target rapid prototyping.

Developing Requirements

Our requirements were very specific: We needed a single development platform and software tools that would be easy to use and quick to learn. We also required nonproprietary hardware as the interface for communication between the PC and the controller, and the target hardware had to be economical, compact, and suitable for use inside the vehicle.

Selecting Hardware

The design team chose the ESX programmable controller from Sensor-Technik Wiedemann (STW) GmbH. We had already used this product in a range of applications, and it satisfied all our hardware and safety-related requirements. The ESX controller works independently as a measuring, driving, or controlling device for sensor-actuator management. It is capable of executing a number of tasks in real time,

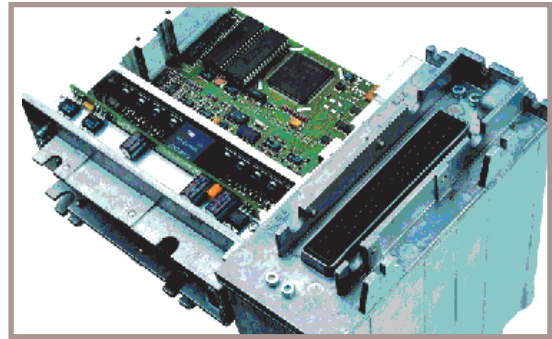


Figure 1. ESX hardware.

including motor management, fluid and inclination level control, and automatic gear-shift control.

Based on an Infineon 80C167CS processor, the ESX controller (Figure 1) provides a wide range of analog and digital inputs and outputs, as well as two CAN-C buses and a serial interface. The device possesses a robust operating system with an API that enabled us to access the hardware with C functions. Software can be compiled with the Altium TASKING compiler and downloaded via CAN-C or RS232.

Developing the Target

We selected MATLAB®, Simulink®, and Real-Time Workshop Embedded Coder as our development platform. There were convincing reasons for this choice: simplicity of use, an all-in-one solution, industry-wide acceptance, and short learning times.

We began by defining requirements: the ESX operating system was to remain unchanged, code generated for the target must be compatible with the operating system, and the ESX target blocks had to reflect the API functions. We then worked with MathWorks consultants to develop a customized Simulink ESX target block library (Figure 2).

A primary goal was ease of use and reduced complexity: The blockset had to possess a high level of abstraction and support only



Images courtesy Mercedes-AMG.

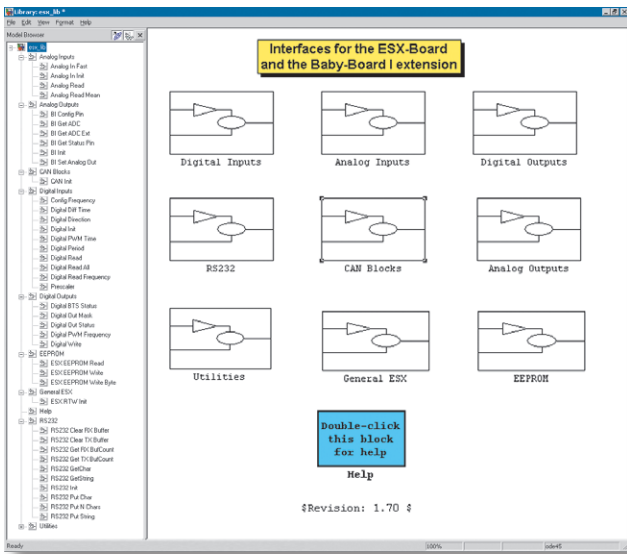


Figure 2. The Simulink ESX library reflects the functional hardware groups: analog inputs, analog outputs, digital inputs, digital outputs, CAN, and RS232. An ESX Init block executes a script that configures the model for code generation with the ESX target and sets the base sample time for the model.

those functions necessary for the control of the inputs and outputs. Certain blocks had to combine multiple functions to avoid excessive complexity.

Because we use xPC Target for functional rapid prototyping on high-performance microprocessors, we based the structure of the CAN blocks on xPC Target. This consistent approach made it easy to transition the Simulink model from rapid prototyping using xPC Target to on-target rapid prototyping using the ESX controller.

Testing

Once we had created the first test versions of the ESX target block set, we conducted random tests of the implemented functions in Simulink and compared the test results with handwritten production code to identify and eliminate errors.

The final test involved developing a control mechanism for an open-loop fuel system with variable pressure. The goal was to implement the new controller as a stand-alone solution in older vehicles that had previ-

ously used a conventional, closed-loop, constant-pressure fuel system. A similar controller was already a component of the motor control unit in production, making a comparison possible.

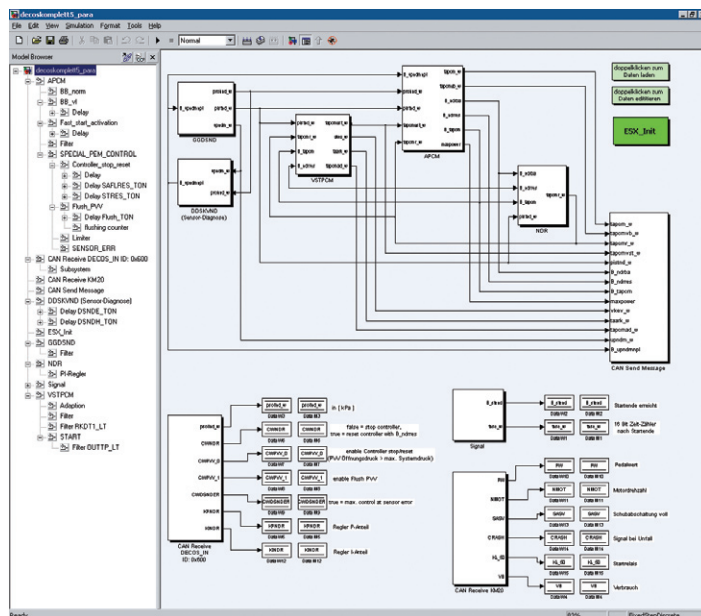
After minor adaptations to the structure of the Simulink library, the task was accomplished. The result was a highly complex model from which we could implement the full scope of the required functionality (see Figure 3). We used Simulink Fixed Point to specify the fixed-point data types used on the Infineon 80C167CS processor in the ESX Controller.

We thoroughly examined and tested the code automatically generated from the model, but found no errors. As a result, we can now run older versions of the vehicle with the new fuel system after minimal conversion work. Further test projects involving the newly developed ESX target blockset and automatic code generation yielded equally good results.

All the conditions for the official release of the Simulink ESX block library were met, and we now have a stable software platform for future development projects.

The ESX is now developed almost exclusively in Simulink, which greatly reduces the software developers' workload when we implement new projects. We are working on a number of extensions to the blockset, including adding I/Os. The near future will see the emergence of a new generation of the ESX control unit. We plan to use automatic production code generation here, too. ◀

Figure 3. Simulink model of the ESX fuel pump controller.



RESOURCES

- ▶ **Model-Based Design for Control Systems**
www.mathworks.com/res/controldesign
- ▶ **MathWorks Training: Automotive Tracks**
www.mathworks.com/res/autotraining
- ▶ **Book: Simulation Engineering: Build Better Embedded Systems Faster**
www.mathworks.com/res/book1775
- ▶ **Mercedes-AMG** www.mercedesAMG.com