Automated Software Verification at TEMIC

CUSTOMERS

- **Test automation with ControlDesk 2.0**
- **Central database for single tests or test sequences**
- **Automatic generation of standardized test documentation**

The complexity of functionalities offered by automotive systems is growing at an enormous speed. This makes high demands on software complexity, and coupled with shorter and shorter innovation cycles, it also means that new development approaches are needed. Temic faced this challenge very early on and has now, in cooperation with dSPACE, developed an automated test environment for software verification. This had its debut in the electronic control unit (ECU) designed to get active suspension on board the Mercedes CL Coupé.

Going Automatic

The use of rapid prototyping tools in functional development, along with procedural method models for software creation, have become today’s new standard. Automated software development processes – from model-based descriptions to the use of automatic code generators and right through to automated software verification – are firmly on the agenda for software development in the future.

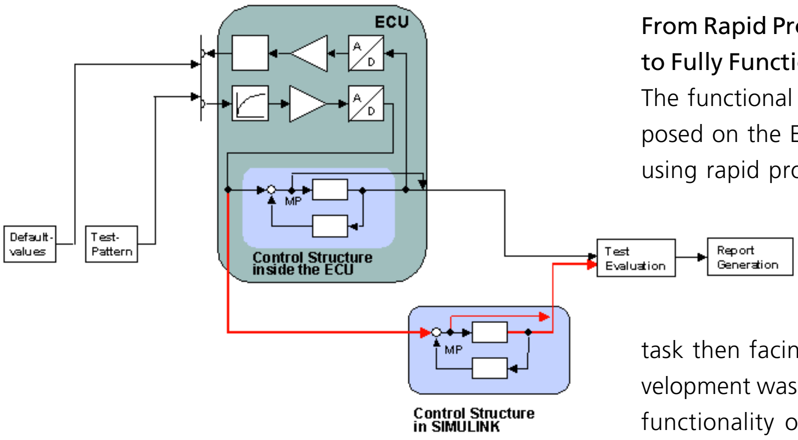
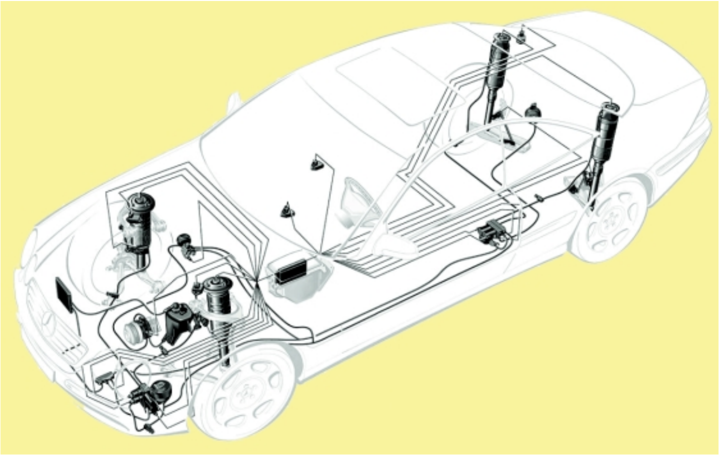


Diagram of a test run.



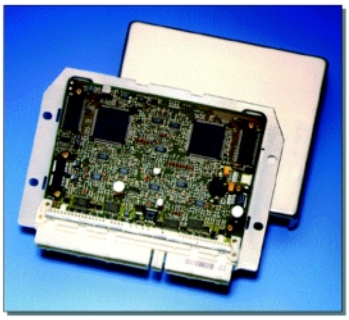
System arrangement of the ABC components.

Actual Project

TEMIC developed and produced the ECU for a new, actively controlled suspension, which was first used in production in the new Mercedes CL Coupé . This system, known as Active Body Control or ABC, enhances both driving safety and ride comfort. Driven by the software, electronically controlled hydraulic cylinders in the suspension system adjust the spring and damping properties of the vehicle according to road surface, driving style and the driver’s preferences. The ABC system has enormous requirements in terms of processing power and real-time control behavior. This in turn creates the need for an appropriate test environment as described in this article.

From Rapid Prototyping Model to Fully Functional ECU

The functional requirements imposed on the ECU were defined using rapid prototyping, by creating a MATLAB/Simulink model. The task then facing production development was to implement the functionality on the target system – the ECU itself. When model-based control algorithms are implemented, particular attention needs to be paid to verification: it has to be demonstrated that the software implementation completely matches the model in terms of functions.



ECU for ABC on board the Mercedes CL Coupé.

Classic test procedures are dominated by manual execution and have considerable drawbacks when it comes to dynamic functionalities:

- Large amount of time needed due to the high number of individual test cases
- Unsatisfactory reproducibility of dynamic test patterns
- The documentation has to be produced by hand

In drawing up the requirements for the next generation of tools for software tests, we were essentially guided by the need to eliminate these disadvantages. The individual objectives were as follows:

- Verification of correct implementation of control and function specifications in the software for the target system (ECU)
- Short-term repeatability of test cases with full integration into the concurrent change management
- Complete, standardized test documentation
- Reduction in time and costs by means of completely automated test runs

To achieve these objectives, we opted for close cooperation with dSPACE, using ControlDesk 2.0. The completely new kinds of options offered by this software, particularly its transparent connection to MATLAB/Simulink, and the professional engineering support that dSPACE staff gave us in adjusting ControlDesk to our needs, were major factors in the success of the pilot project.

Papers

By dSPACE

K. Lamberg, P. Wältermann:
"Using HIL Simulation to Test Mechatronic Components in Automotive Engineering"

English 3 **03**
German 3 **04**

T. Pöhlmann:
"The Internal Combustion Vehicle with the Lowest Emissions"

English 3 **05**
German 3 **06**

Test Environment

The heart of the automated test environment is a fully automated comparison of the functionality of the MATLAB/Simulink model with its implementation in the ECU. The basic idea is to apply the same stimulus signals to the Simulink model as to the implemented ECU software (see diagram on p. 12). The evaluation compares the behavior patterns of the implementation in the ECU to those of the reference model according to mathematical, computational criteria, and rates the results as pass or fail.

To generate suitable test patterns, we use ControlDesk's Stimulus Editor. This allows stimulus patterns for the planned tests to be defined freely. Parameterizable signal patterns, graphically editable signal patterns and signal behaviors gathered from test drives in the vehicle are all options that can be used for test case generation.

The ECU's electronic connection to the test environment is controlled centrally by ControlDesk. Control and communication are via parameterizable inputs and outputs, CAN bus connection and data bus based on the dSPACE ISO9141 Python module.

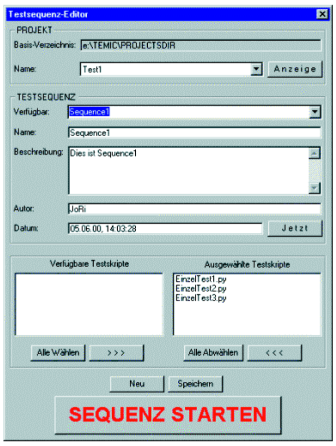
At the start of each test, the ECU is put into a defined initial state by means of stimuli, before the actual test stimuli are fed to it.

Both the internal computing results from the initialization phase in the ECU and the test case that immediately follows are transferred and recorded to the test environment via the data bus. The next step is to start the Simulink model directly from the test environment and feed in the signals transferred by the ECU, all under the control of ControlDesk.

The documentation including the graphical results is produced in a standardized Microsoft Word file after each test step. A database administered under ControlDesk is used as the central data basis for all system files, initialization and test stimuli, and test results .

Test Database

For the intended reductions in time and costs, entire test sequences and even complete release tests have to be run completely automatically. dSPACE specially designed a sequence editor for this task. This allows any desired individual tests to be grouped together to produce complex test sequences.



Basis-Verzeichnis: e:\TEMIC\PROJECTSDIR
Name: Test1
Verfügbar: Sequence1
Name: Sequence1
Beschreibung: Dies ist Sequence1
Autor: JoRi
Datum: 05.06.00, 14:03:28
EinzelTest1.py
EinzelTest2.py
EinzelTest3.py

The sequence editor's graphical interface.

All of the individual tests and the test sequences are placed in the central database. They can subsequently be performed and reproduced as often as required.

Success Story

The resulting enhancement of test quality and reduction in time and cost sets new standards in software verification. This is largely made possible by ControlDesk and its new test automation features. So the project definitely marks the start of the next step in software development.

Udo Teubert
Reinhard Helldörfer
TEMIC
Germany