

# **Model-based Development of Hybrid-specific ECU Software for a Hybrid Vehicle with Compressed-Natural-Gas Engine**

Dipl.-Ing. Tobias Mauk

IVK, Universität Stuttgart  
Pfaffenwaldring 12, 70569 Stuttgart  
Telephone: +49 711 685 68129, Fax: +49 711 6773216  
eMail: Tobias.Mauk@ivk.uni-stuttgart.de

Dr. phil. nat. Dieter Kraft, Robert Bosch GmbH

Dr.-Ing. Joachim Quarg, Adam Opel GmbH

Dipl.-Ing. Michael Böhm, Universität Stuttgart

Prof. Dr.-Ing. Michael Bargende, Universität Stuttgart

Prof. Dr.-Ing. Hans-Christian Reuss, Universität Stuttgart

## **Abstract**

This paper describes the development of hybrid control software for a parallel hybrid vehicle with compressed-natural-gas engine. The underlying project is shortly introduced. Then the software development process is presented with emphasis on different stages of testing and the requirements for the test environment software. Hereafter the tasks of the hybrid control software are explained. Finally a short description of the control strategy is given.

## **Introduction**

In 2006 Adam Opel GmbH, Robert Bosch GmbH and Universität Stuttgart joined forces to build up a prototypical hybrid vehicle with minimal CO<sub>2</sub> emissions [1]. The project is supported by the Federal Ministry of Economics and Technology (Bundesministerium für Wirtschaft und Technologie). The concept incorporates a down-sized, highly turbocharged natural-gas engine [2] combined with an electric motor in a parallel hybrid configuration [3][4]. The vehicle is based on the Astra Caravan. It is equipped with an automated manual transmission and two friction clutches.

In addition to the control units of the several aggregates like combustion engine, electric motor and transmission, a superordinate control unit is necessary. This so-called Hybrid Control Unit (HCU) coordinates the interaction between the aggregates. More detailed, its

most important tasks include the intelligent determination of the appropriate operational mode, the split-up of the desired driving torque between combustion engine and electric motor, the selection of the appropriate gear and the consideration of the battery state-of-charge in all its decisions.

Furthermore, the hybrid control unit receives predictive information about the upcoming road section from a forward-looking unit. This information includes geographical map data like curves, slopes and other information. It is used by the hybrid control unit to optimize its strategy.

Lastly, the hybrid control unit operates as a network gateway, as the control units of the additional hybrid components cannot be connected to the powertrain CAN bus directly due to their incompatibility. Therefore a second CAN bus is used.

Figure 1 shows the structure of the prototype vehicle powertrain including the most important control units and the two CAN busses.

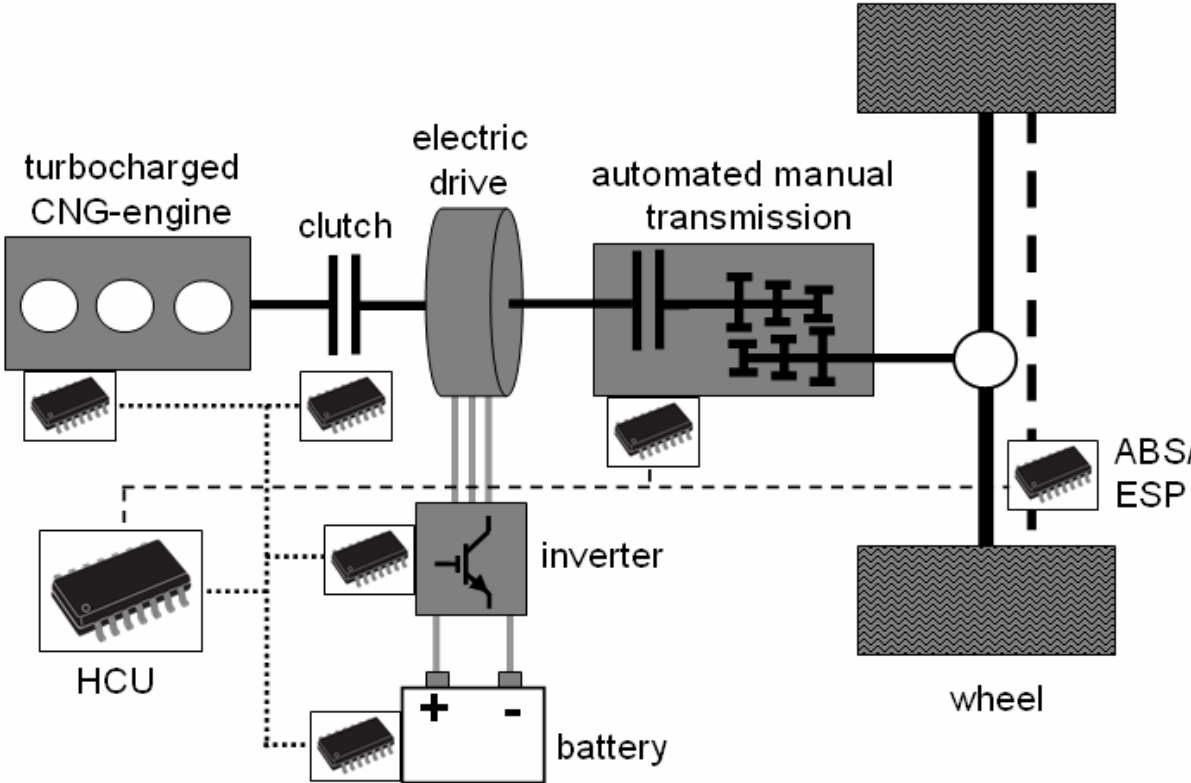


Figure 1: Hybrid powertrain including the most important control units

## Software Development Process

The following section describes aspects of the software development process for the hybrid control unit. Being able to test the software already at early development stages is very important in order to identify and correct software design imperfections as early as possible. As the software development for the hybrid control unit and the construction and assembly of the prototype vehicle take place simultaneously in order to save time, the software cannot be tested in the vehicle at early stages. Even if the vehicle was built up earlier, it would be too risky to perform early tests of the software on the real vehicle hardware. It is therefore necessary to model and simulate the environment of the hybrid control unit in order to perform tests. The environment model consists of a vehicle model, a driver model and a driving cycle. It ought to provide a virtual environment as it will be perceived by the hybrid control unit later in the real vehicle. This also enables tests of critical software functions without danger of damaging the vehicle or injuring people.

Due to these requirements an appropriate software development process is defined. For the development of the software functions a PC-based environment model is used. This approach saves a lot of time, as the test simulations can be calculated very fast and do not have to be slowed down to real-time speed. In the PC-based environment an additional CAN bus model simulates effects like quantisation, time discretisation and bus latency.

In the vehicle the hybrid control software will be executed on the hybrid control unit, which is in fact a rapid-prototyping ECU. To verify the real-time capability of the hybrid control software it is exported to that rapid-prototyping ECU using automatic code generation. A further real-time computer is used to provide the real-time environment model, which is also generated from the PC-based environment model using automatic code generation. After connecting the hybrid control unit and the environment simulator, hardware-in-the-loop (HiL) tests can be performed. The I/O of the hybrid control unit is identical to the one needed in the real vehicle. An advantage of using real CAN busses at this stage of testing is that also aspects as bus load and variable bus latency are taken into account. This approach supports an easier transfer to the real vehicle. Figure 2 shows the HiL simulation setup.

An even higher level of realistic environment for the hybrid control unit can be achieved using a test bench. Several parts of the environment model are replaced by real hardware components, e.g. the combustion engine, the electric motor, the clutch between them and the battery. All affected signal connections in the environment model must also be replaced by

connections to signals from the real components. Those components not present in the hardware setup have to be further on simulated. Also the driver and the driving cycle remain part of the environment simulation.

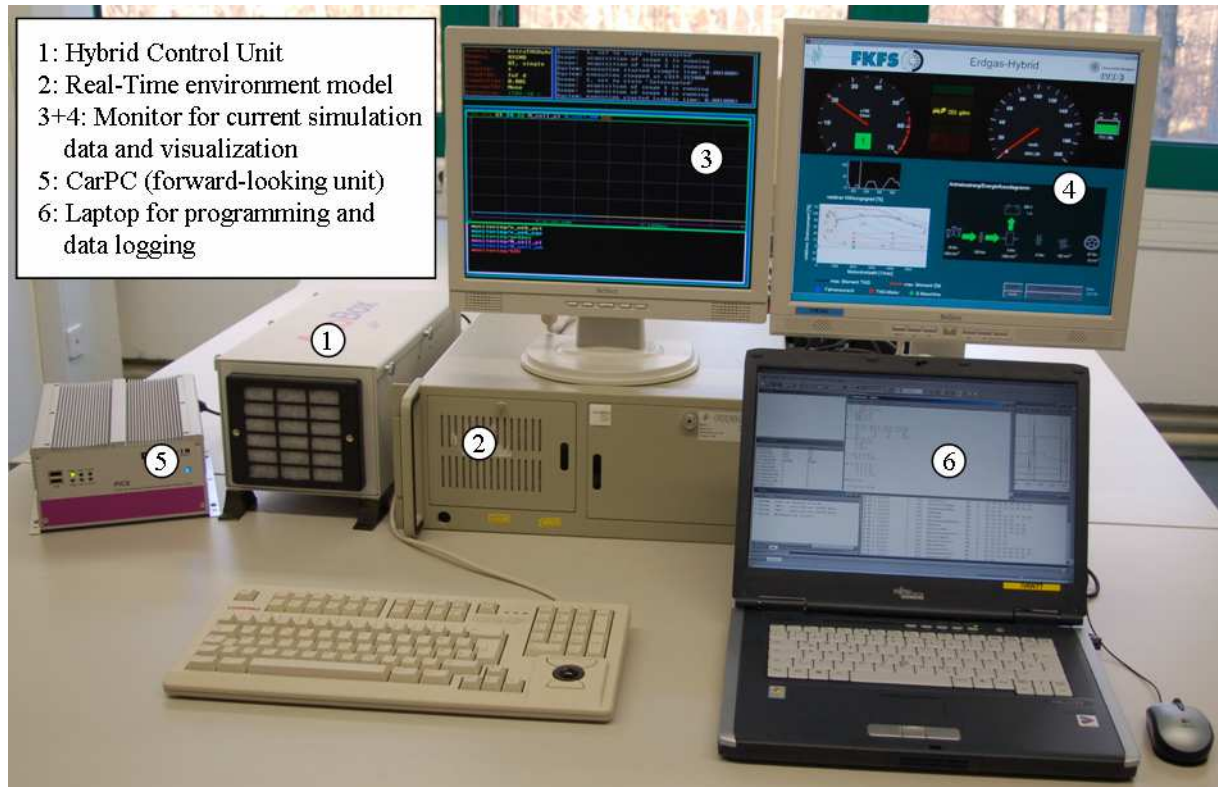


Figure 2: Hardware-in-the-loop simulation setup

The last level of performing tests will take place in the prototype vehicle, as soon as the hybrid control unit has passed all previous tests. No further environment simulation is necessary at this stage. Figure 3 shows the concept of the presented development process. The steps need to be iterated.

Essential prerequisite for this development process is a strictly modular structure of the environment model. This includes the consequent separation of function and interface in every module. Otherwise it would not be possible to easily use the same function in all the stages of testing, as the modules have to be interchangeable with real hardware components on the test bench and the interfaces differ from stage to stage (e.g. real bus vs. simulated bus).

The functionality of the environment model software may not be altered between the different stages of testing. The same holds for transmission rate, resolution and feasible value range of the signals. This is ensured by using automatic code generation and the use of a library

concept. The interface modules for the different stages are stored in separate model libraries and linked to the respective function modules, which are also stored in libraries. Thus, code can be automatically generated for all needed configurations (i.e. PC-based, real-time, test bench configuration).

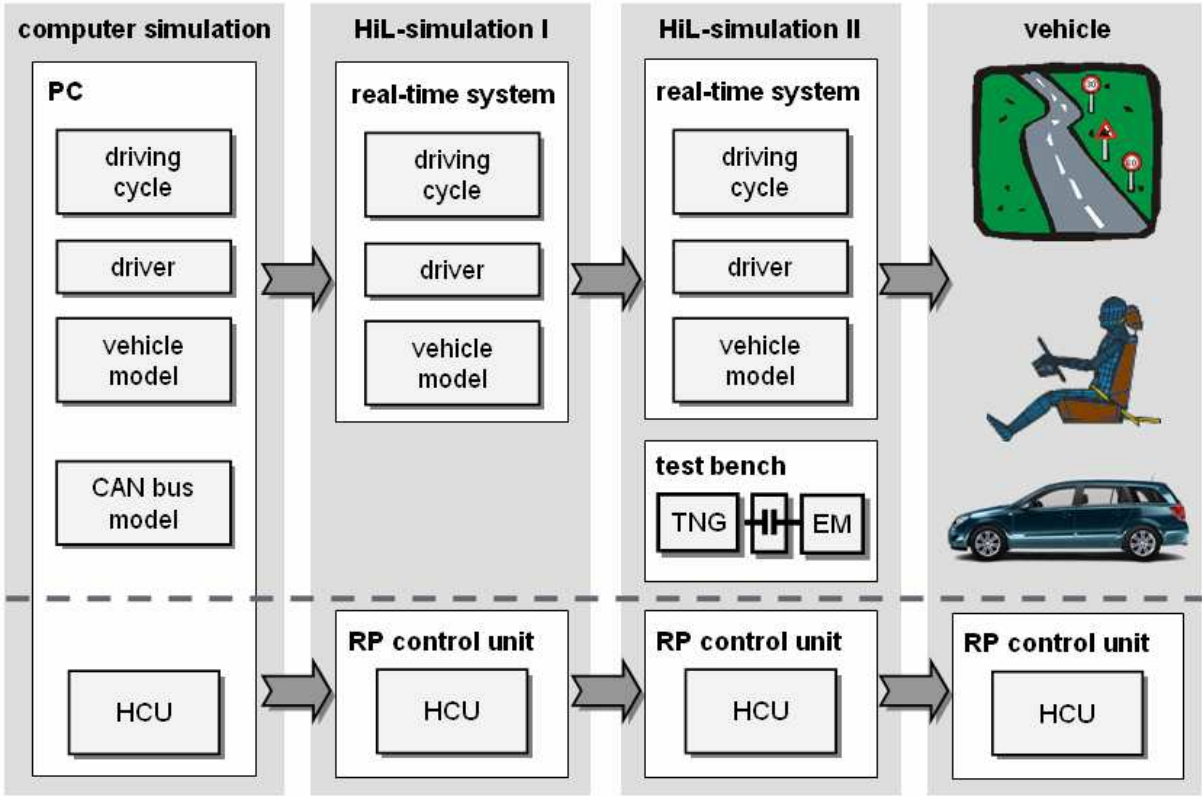


Figure 3: Development process: several stages of testing

Defining all the module interfaces in an exact manner and at an early stage is essential for the success of this development process, as later changes would be very time-consuming, expensive and error-prone. Additionally the compatibility to old software versions would be lost. It must be distinguished between physical inter-module connections (i.e. physical couplings of the corresponding hardware components) and communication signals (mostly CAN signals and messages). Physical connections are mechanical, electrical or thermal quantities, for example the actual engine speed and torque or the actual current of the electric motor.

In this project communication signals are mainly CAN signals. As there are several hundred different signals, a special “helper tool” was developed to automatically generate a module with I/O blocks for all the CAN signals out of a CAN communication matrix description file.

So every signal on the real bus corresponds to a signal in the model. According to the distinction between physical and communicational signals, the component modules consist of two parts: one represents the physical part, and the other one the control unit part of the respective component. Of course it is not possible to exactly replicate the complete control unit software of all components into these control unit modules – due to the multitude of functions in today’s control units –, but fortunately it is not necessary to complete this very complex and expensive task to that extent. Only those signals which are of special interest for the hybrid control unit have to be modelled in detail. Three levels of detail are distinguished:

- Some signals are not relevant for the hybrid control unit, so it is sufficient to choose a constant value for those signals. Irrelevant CAN messages (i.e. messages consisting of not relevant signals only) are not omitted but still transmitted in order to preserve the correct bus load. An example for such a message is the tire pressure message.
- Another set of signals is relevant for the hybrid control unit, but only for supervising purposes, for example the airbag activation status signal. The hybrid control unit has to switch off the high-voltage electrical system in case of an airbag activation. Reasonable constant values can be used to model these signals. To test the hybrid control unit these values may be changed manually.
- The remaining signals are especially relevant for the hybrid control unit and must be carefully modelled, for example the engine speed and torque signals or the battery voltage. Of course only a simplified model of the real control unit software is practicable.

### **Environment Model**

In the following section the vehicle model (as part of the environment model) will be described more detailed, since it is a very important element of the development process. Its structure is strictly modular, which makes it possible to easily exchange interface modules between different stages of testing, or to replace component modules by real hardware components at the test bench. The model is limited to the longitudinal dynamics of the vehicle. The most important modules and their interfaces are shown in Figure 4. The transmission and the adjacent clutch C2 are handled together in one module, since they are controlled by a common control unit. The same holds for the electric motor and the inverter.

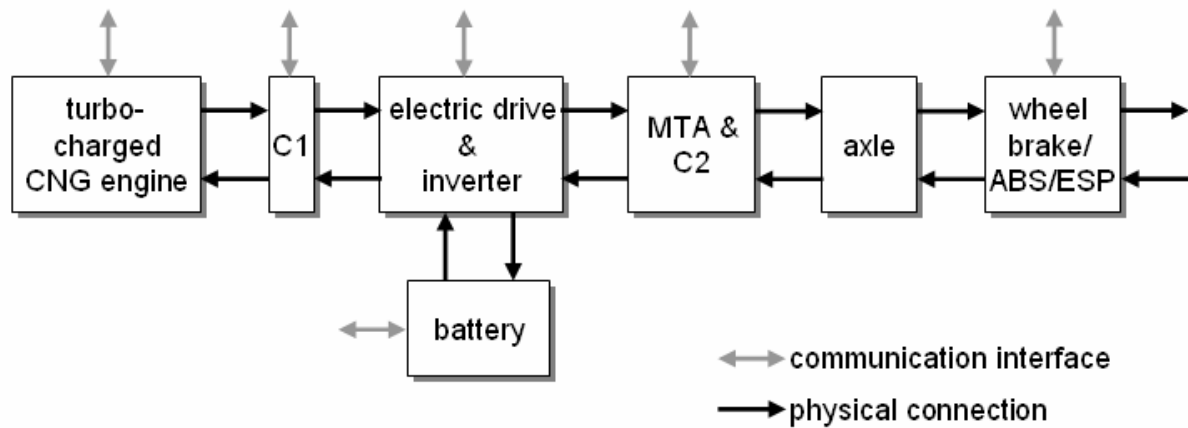


Figure 4: Module structure of the vehicle model

In addition to the modules shown in Figure 4, there is a CAN bus model, which is only needed to perform the PC-based tests. It simulates time discretisation, quantisation, bus latency and sampling effects. It is removed in both real-time test stages, since physical CAN busses are used here. Without the CAN bus model in the PC-based tests, PC simulation results deviate from the HiL simulation results. This is critical especially when the CAN bus is part of a closed loop control. When designing closed loop controllers the dead time caused by the bus must be taken into account. A good example is the necessary communication between the engine control unit and transmission control unit during a gear shift operation. Another example is the driver model. It is implemented as a controller, which uses the accelerator and the brake pedal to achieve a desired vehicle speed trajectory. Figure 5 shows the accelerator position in the HiL simulation (using real busses) and in two different PC-based simulations (with and without the CAN bus model). The curve shapes of the CAN bus simulation and the HiL simulation (using real CAN busses) are quite similar, remaining differences are reducible to the inherent non-determinism of CAN busses. However, the simulation without the CAN bus model reveals noticeable differences. Especially when accelerating the vehicle, the resulting accelerator pedal positions are higher when the CAN latencies are not neglected. This may lead to further discrepancies in the behaviour of both simulations, for example a gear shift might be triggered in one simulation (but not in the other) because of different pedal positions. So comparability between PC-based simulation and HiL simulation (including real busses) can only be preserved by using the CAN bus model in the PC-based simulation. Again, a special helper tool was developed to automatically generate the CAN bus model due to the large number of messages.

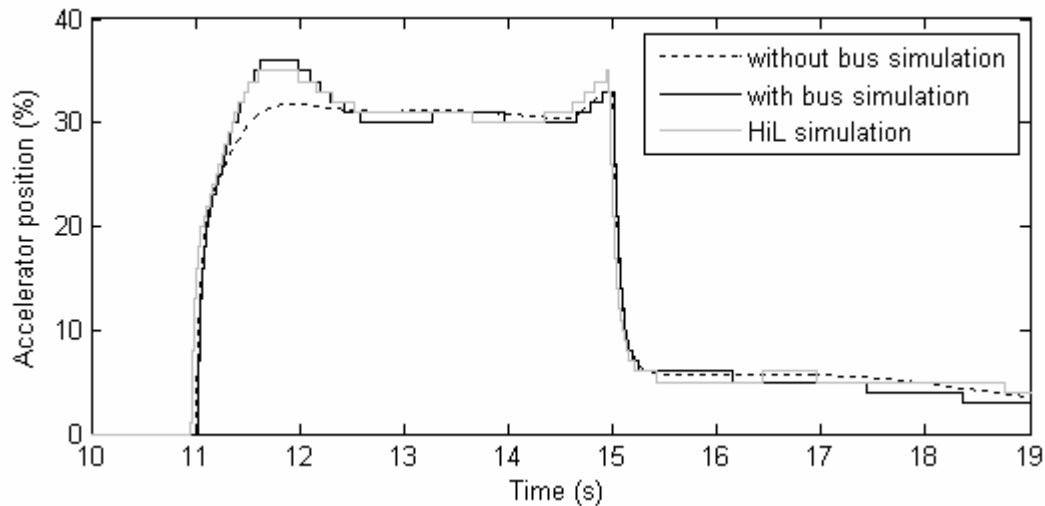


Figure 5: Influence of the CAN bus on simulation results

### The Hybrid Control Unit Prototype

Thanks to the very good congruence between PC-based and HiL simulation it is possible to effectively develop the hybrid control software mainly on the PC. Nonetheless it is necessary to perform regular HiL tests to validate the real-time capability of the software and the correct operation of the communication between hybrid control unit and its environment.

Due to incompatibilities between different control units it is not sufficient to use only one CAN bus for the whole powertrain. The control units from the original Astra vehicle are directly connected via one CAN bus (“Astra bus”). The other control units, which were added, e.g. engine, battery and electric motor control units, share another CAN bus (“hybrid bus”). Both busses are separately connected to the hybrid control unit (see Figure 1), which works as a gateway. For example, the hybrid control unit acts as a virtual engine control unit towards the other control units on the Astra bus, since the engine control unit is the only control unit which was removed from that bus. The hybrid control unit collects all CAN messages that are supposed to reach the engine control unit and passes them to the real engine control unit, which is connected to the hybrid bus – after having done all necessary translation and conversion work. Likewise it receives the messages which the engine control unit sends to the other powertrain control units (e.g. the transmission control unit or the ABS/ESP control unit) and translates them to the Astra bus.

Apart from the various tasks already mentioned in the introduction, the hybrid control unit has another important task which goes beyond a normal gateway function. Not only does it have

to translate, re-arrange and requantise signals between both CAN busses, some signals even need to be re-interpreted, i.e. their meaning is altered in the hybrid context. This fact is explained by means of the following example. In the series-production vehicle the engine control unit periodically sends the engine speed in a CAN message. This signal is important for the transmission control unit (which also controls the adjacent clutch C2), since it needs to know the input (i.e. engine-sided) speed of the clutch. Engine output speed and clutch input speed are assumed to be always identical (not only equal) – and rightly so, since the combustion engine is directly attached to the clutch C2 in the series-production vehicle. However, these two speeds are not necessarily the same in the hybrid vehicle, as the electric motor and the other clutch C1 are inserted between the engine and the clutch C2 (and C1 may be open). In this case it is the speed of the electric motor that is identical to the input speed of clutch C2. So, although the transmission control unit still expects to receive the combustion engine speed (it does not know it is being hybridised), it must be told the speed of the electric motor instead. This implies a change in the meaning of signals. There are a number of signals that must be likewise “redirected” or even manipulated to make the series aggregates work as required. Another example is the input torque to clutch C2, which is labelled “engine torque” in the series car, but in the hybridised car it has to be calculated dependent on the engine torque, the electric motor torque and the status of clutch C1. This approach makes it possible to hybridise the car without modifying the series components. Needless to say, it is very important to test this complex software extensively in the PC-based and HiL simulations.

### **Hybrid Control Strategy Optimizations**

Hybrid vehicle concepts in general offer more degrees of freedom for powertrain control strategies than conventional vehicles. There are different targets which strategies may pursue [5], mainly saving fuel, improving driveability or variably weighted mixtures of both. Also other targets are possible, such as improving comfort or optimizing battery durability. The strategy in this project focuses on minimizing CO<sub>2</sub> emissions (which is quite close to saving fuel respectively natural gas) while complying current and future emission limits.

Therefore the strategy software is supported by a forward-looking unit which provides predictive information about the assumed upcoming road section, such as data about curves, slopes and other information. This data is used for optimization calculations. The strategy software calculates (approximates) a trajectory of gear shifts, state-of-charge and torque

distribution between engine and electric motor (with a variable time horizon of a few minutes), which is supposed to be optimal in the sense of minimized CO<sub>2</sub> emissions.

## Summary

A development process for hybrid-specific control software was presented in this paper. Emphasis was put on the multi-stage testing reaching from PC-based non-real-time simulation to real-time HiL simulation, and tests including several real aggregates on a test bench. Important prerequisites were described, especially the exact definition of different interfaces. Afterwards the tasks of the hybrid control software were explained, especially the extended gateway function. Finally the hybrid control strategy was shortly presented.

## References

- [1] Quarg, J. et al.: *Compressed-Natural-Gas Hybrid – Drive Concept based on Natural-Gas Hybrid Technology*, 4th Symposium on Hybrid Vehicles and Energy Management, 2007
- [2] Berner, H.-J.; Bargende, M.: *Ein CO<sub>2</sub>-minimales Antriebskonzept auf Basis des Kraftstoffes Erdgas*. Tagung „Gasfahrzeuge – Die passende Antwort auf die CO<sub>2</sub>-Herausforderung der Zukunft?“ Berlin, 2004.
- [3] Bargende, M.; Berner, H.-J.: *A Downsized, Turbocharged Natural Gas SI Engine - Including Hybridization - For Minimized CO<sub>2</sub> Emissions*. SAE 2005-24-026
- [4] Bargende, M.; Berner, H.-J.: *Auf der Suche nach der Grenze: 90 g/km CO<sub>2</sub> bei mehr als 1500 kg Fahrzeuggewicht?* Motortechnische Konferenz, Ingolstadt, 2005
- [5] Fried, O.: *Betriebsstrategie für einen Minimalhybrid-Antriebsstrang*, Universität Stuttgart, Dissertation, 2003