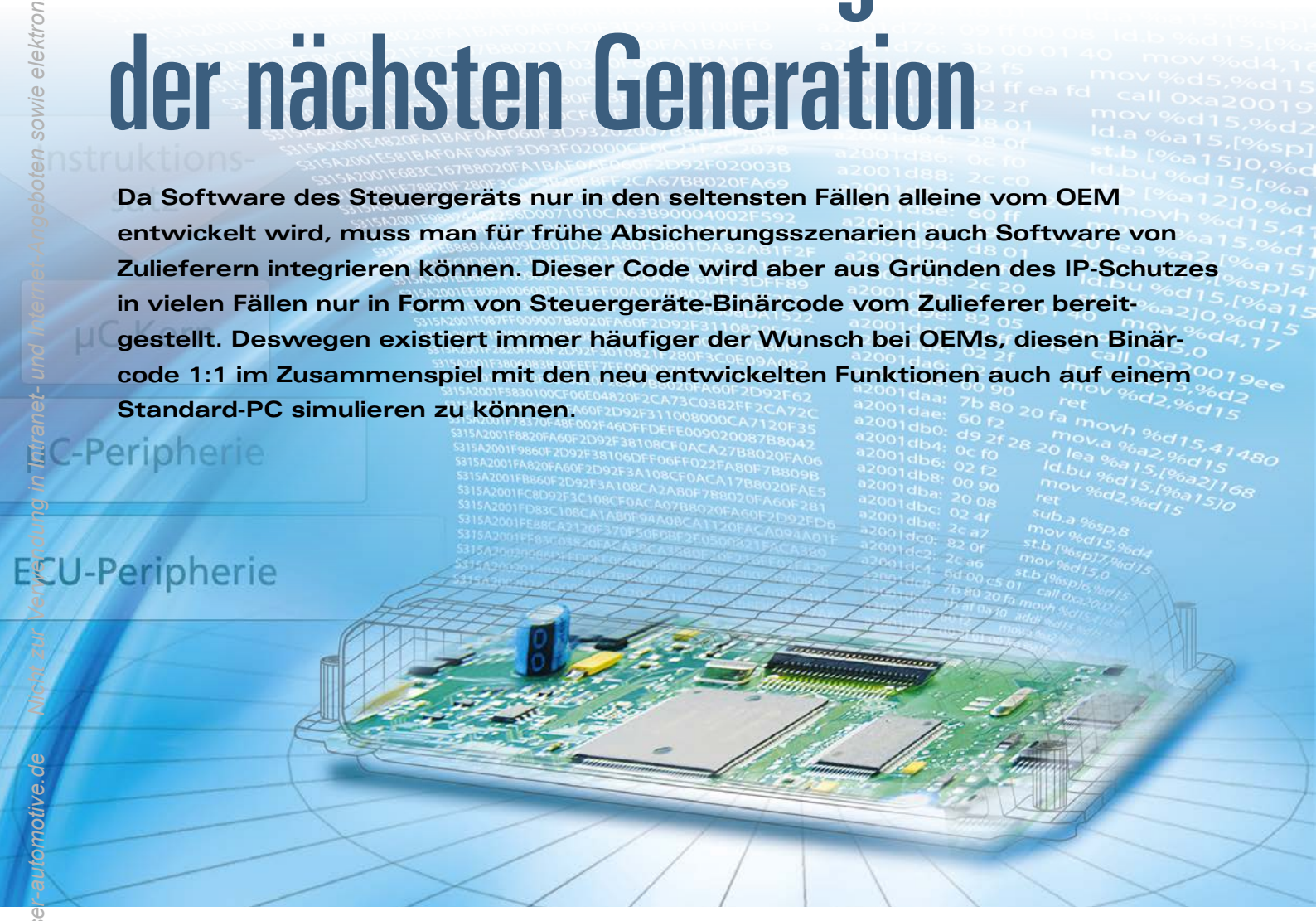
# Virtuelle Absicherung der nächsten Generation

**Da Software des Steuergeräts nur in den seltensten Fällen alleine vom OEM entwickelt wird, muss man für frühe Absicherungsszenarien auch Software von Zulieferern integrieren können. Dieser Code wird aber aus Gründen des IP-Schutzes in vielen Fällen nur in Form von Steuergeräte-Binärcode vom Zulieferer bereitgestellt. Deswegen existiert immer häufiger der Wunsch bei OEMs, diesen Binärcode 1:1 im Zusammenspiel mit den neu entwickelten Funktionen auch auf einem Standard-PC simulieren zu können.**




Bei OEMs geht der Trend dahin, in der Entwicklung immer mehr Prototyp-Fahrzeuge einzusparen. Gleichzeitig muss im Bereich des autonomen Fahrens mehr und mehr virtuell getestet werden. Daher kommt der realitätsnahen Absicherung von Steuergeräte-Software mithilfe virtueller Steuergeräte auf Standard-PC eine immer größere Bedeutung zu. Mittlerweile hat sich in der Automobilindustrie als ein neuer Validierungsschritt etabliert, die Applikationssoftware und Teile der Basis-Software mithilfe von Offline-Simulatoren wie z. B. VEOS von dSpace abzusichern. Damit können sowohl funktionale als auch Integrationstests der Original-Steuergeräte-Software, für die bereits C-Code vorliegt, in frühe Entwicklungsphasen vorverlagert und jedem Funktionsentwickler zur Verfügung gestellt werden, lange bevor erste Steuergeräte-Hardware-Prototypen existieren. Da allerdings die Software des Steuergeräts nur in den seltensten Fällen alleine vom OEM entwickelt wird, muss man für diese frühen Absicherungsszenarien auch Software von Zulieferern integrieren können. Dieser Code wird aber aus Gründen des IP-Schutzes in vielen Fällen nur in Form von Steuergeräte-Binärcode vom Zulieferer bereitgestellt. Deswegen existiert immer häufiger der Wunsch bei OEMs, diesen Binärcode 1:1 im Zusammenspiel mit den neu entwickelten Funktionen auch auf einem Standard-PC simulieren zu können.

Virtuelle Hardware-Plattformen versprechen hier eine Lösung, doch meistens stellt sich heraus, dass entweder kein komplettes Modell der Steuergeräte-Hardware vorliegt oder die Simulation des kompletten Steuergeräte-Codes mit Hardware-Modellen zu langsam und kostenintensiv ist. Vielfach zeigt sich nach näherer Betrachtung des gewünschten Anwendungsfalls beim OEM, dass auch einfachere und günstigere Modelle ausreichend sind, um das Testziel zu erreichen. Dabei muss aller-

Bilder: dSpace

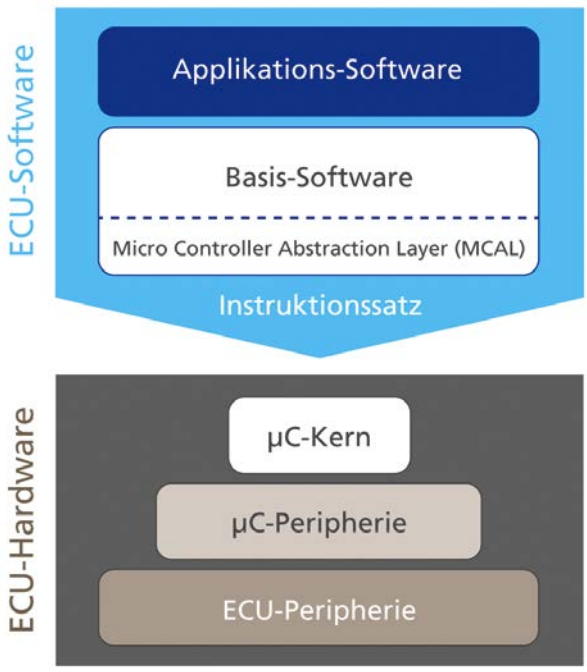



**Bild 1: dSpace bietet zusammen mit Partnern eine Simulator-Plattform an, auf der es möglich ist, Hardware-Modelle von µC-Cores und ECU-Komponenten zu integrieren und so vollständigen Steuergeräte-Binärcode zu simulieren.**

dings beachtet werden, dass diese virtuellen Hardware-Plattformen gut mit den bestehenden Absicherungsstrategien des OEMs zusammenspielen, sprich die dort eingesetzten Strecken- und Umgebungsmodelle sowie Experimentier- und Testwerkzeuge weiterhin nahtlos eingebunden werden können.

## Simulation von Steuergeräte-Binärcode

Im Allgemeinen besteht jede Software (Binärcode), die auf einem Steuergerät ausgeführt wird, aus einzelnen Befehlen, die vom verwendeten Mikrocontroller interpretiert und ausgeführt werden. Die Summe aller Befehle, die ein spezieller µC versteht, nennt man Instruktionssatz. Innerhalb eine µC-Familie gibt es meist einen standardisieren Instruktionssatz mit Abweichungen für einzelne µC-Derivate. Bei diesen Standards spricht man auch von Instruktionssatz-Architektur. Beispiele hierfür sind „Book E" für PowerPC oder „AArch32" für ARM Controller.

Die für die Ausführung von Binärcode zuständige Einheit im µC ist der µC-Kern. Er ist dafür verantwortlich, die Befehle aus dem Speicher zu holen, zu interpretieren und auszuführen. Darüber hinaus realisiert er den Registersatz, Adressierungsarten, Datentypen, die Unterbrechungslogik, das Memory Management und vieles mehr.

Will man nun Binärcode auf einem Simulator ausführen, der einen anderen Instruktionssatz hat (beispielsweise PowerPC-Binärcode auf einem x86-Prozessor), braucht man ein Stück Software, das den µC-Core auf dem Simulator nachbildet. Oft wird in diesem Zusammenhang von Instruktionssatz-Simulator gesprochen. An dieser Stelle kann man schon erahnen, dass dieser Begriff etwas irreführend ist, denn das, was eigentlich simuliert werden muss, ist der µC-Core. Allerdings reicht eine Simulation des Cores alleine noch nicht aus, um den Binärcode vollständig auszuführen: Wie Bild 1 zeigt, besteht die Steuergeräte-Hardware aus weiteren Komponenten wie der µC-Peripherie (AD Converter, Bit I/O) und der ECU-Peripherie, zum Beispiel ASICs für spezielle I/O. Eine Simulation des kompletten Steuergeräte-Binärcodes gelingt nur, wenn für alle diese Komponenten Modelle in einer geeigneten Modellierungssprache wie SystemC im Simulator hinterlegt sind.

Existieren diese Modelle, dann steht der Simulation nichts mehr im Weg. dSpace bietet hierfür zusammen mit Partnern eine Simulator-Plattform an, auf der es möglich ist, Hardware-Modelle von µC-Cores und ECU-Komponenten zu integrieren und so vollständigen Steuergeräte-Binärcode zu simulieren.

In konkreten Projekten stellt sich dabei allerdings oft heraus, dass der Nutzen den Aufwand und die Kosten, Hardware-Modelle zu erstellen, nicht aufwiegt. Darüber hinaus haben vollständige Hardware-Modelle, zum Beispiel auf Register-Transfer- bzw. Transaktionsebene, aufgrund ihrer Komplexität zumeist eine zu lange Ausführungszeit. Nur wenn man genaue Aussagen über Timing, Stromverbrauch oder eine Performance-Abschätzung in frühen Projektphasen unbedingt braucht und keine Chance hat, an ein Prototyp-Steuergerät zu kommen, ist diese Art der Virtualisierung der richtige Weg.

## Effiziente Lösungen

In den allermeisten Fällen ist es nicht notwendig, die Hardware im Detail zu simulieren. Wie Bild 1 zeigt, besteht die ECU-Software – vor allem, wenn sie nach Standards wie AUTOSAR entwickelt wurde – aus mehreren Schichten, wobei höhere Software-Schichten (Application) deutlich weniger Abhängigkeiten zur Hardware besitzen als niedrige (Basic Software). Es reicht in »

| Im Original-Instruktionssatz zu simulierende Software | Benötigte Informationen | Aufwand |
|---|---|---|
| **Applikationssoftware** | Map File und Funktions-einsprungspunkte | gering |
| **Applikationssoftware und Teile der Basis-Software** | s. o., zusätzlich Informationen zum Stubben der BSW | mittelmäßig |
| **Vollständiger Steuergeräte-Binärcode** | komplettes ECU-Hardware-Modell | sehr hoch (initial) |

**Tabelle 1: Benötigte Informationen und Aufwandsabschätzung für die Binärcode-Simulation.**

den meisten Fällen aus, die oberen, hardwareunabhängigen Software-Schichten zu simulieren, um das Testziel zu erreichen: Will beispielsweise der OEM Teile der Anwendungssoftware als Binärcode von Zulieferern zusammen mit seinen neu entwickelten Funktionen absichern, spielt das Verhalten von Hardware und Basis-Software keine Rolle. Neben einem Modell des µC-Core muss der Simulator hierfür lediglich die Software-Einsprung-Punkte kennen und die Bereitstellung der Ein- und Ausgabesignale realisieren. Hierfür stellt dSpace das Werkzeug „ECU Interface Manager" zur Verfügung. Dieser kann nach erfolgreichem Parsing die Binärcode-Datei automatisiert instrumentieren, das heißt Code-Stücke einfügen, die die Kommunikation zum Beispiel mit einem Streckenmodell herstellen. Dabei wird der zu simulierende Code nicht verändert, sondern lediglich an die Simulation angebunden. Voraussetzung dafür ist das Vorhandensein des Linker Map Files für die betreffenden Stellen im Steuergeräte-Binärcode.

Etwas aufwendiger wird die Simulation, wenn Teile der Basis-Software wie Kommunikations-Stacks für Automotive-Busse mitsimuliert werden sollen. Dadurch können zum Beispiel Fehler in der Netzwerk-Konfiguration aufgedeckt werden, bevor erste Steuergeräte-Prototypen existieren. Hier muss zunächst eine Software-Ebene gefunden werden, oberhalb derer der originale Code im Simulator ausgeführt werden soll. In nach AUTOSAR entwickelter Software bietet sich dafür der „Microcontroller Abstraction Layer" (MCAL) an. Um die Simulation dafür korrekt aufzusetzen, benötigt man allerdings etwas mehr Informationen über den Steuergeräte-Code als bei der reinen Applikationssimulation. Beispielsweise liegen Ein- und Ausgabesignale hier nicht mehr als „applikationsgerechte" Signale vor, sondern lediglich auf der unteren Treiberebene. Auf Applikationsebene muss der Simulator für ein Bit-Eingabesignal nur den Wert vorgeben (0 oder 1). Auf Treiberebene muss er zusätzlich wissen, auf welchem Digital Input Port und auf welcher Bit-Position das Signal erwartet wird. Darüber hinaus hat die Basis-Software auch mehr Schnittstellen zum Betriebssystem als der Applikationscode, zum Beispiel für Events und Synchronisationsmittel. Da diese vom Simulator bedient werden müssen, muss er auch einen größeren Teil der Ablauflogik übernehmen als im ersten Fall. Auch hier helfen dSpace-Werkzeuge, den Binärcode auf effiziente Weise zu instrumentieren und so die Anbindung an den Simulator herzustellen (siehe Tabelle 1).

## Ausblick

Binärcode-Simulation ist ein wichtiger Schritt in Richtung vollständig virtualisierter Absicherung von Steuergeräte-Code. Zur Vorbereitung und Durchführung einer Simulation beim OEM sind aufgrund der komplexen Materie effektive Werkzeuge nötig, um einfach, schnell und sicher zum Ziel zu gelangen. Es ist wichtig, sich bereits im Vorfeld sehr genau zu überlegen, welche Effekte man in der hardwarenahen Simulation betrachten möchte. Dies hat eine große Auswirkung auf die Auswahl eines passenden Simulators und der zu erstellenden Modelle und damit im Endeffekt auf die Kosten dieser Simulationsszenarien. dSpace kann mit seiner Erfahrung in diesem Bereich beratend zu Seite stehen und helfen, solche hardwarenahen Simulationen kostengünstig in bestehende Absicherungsprozesse einzubinden. ■ (oe)

» **www.dspace.de**

### i RTI AUTOSAR Blockset

Das neue RTI AUTOSAR Blockset von dSpace wertet das kompakte Prototyping-System MicroAutoBox II zur universellen Echtzeitentwicklungs- und Validierungsplattform im AUTOSAR-Kontext auf.
Auf dieser Plattform können in MATLAB/Simulink entworfene Regler jetzt nicht mehr nur zusammen mit einzelnen AUTOSAR-Software-Komponenten eingesetzt werden. Erstmals besteht auch die Möglichkeit, komplette virtuelle Steuergeräte (V-ECUs) aus dem Systemarchitekturwerkzeug SystemDesk auszuführen und zu validieren. Diese V-ECUs können auf der MicroAutoBox II für Aufgaben wie Software-Entwicklung, Rapid Control Prototyping, Benchmarking und Test verwendet werden, lange bevor erste Prototypen des Seriensteuergeräts zur Verfügung stehen.


Die V-ECUs werden inklusive Konfigurationen für Betriebssystem und Basis-Software auf die MicroAutoBox II übertragen. Die Unterstützung der AUTOSAR-Versionen 3.x und 4.x sowie die Erfüllung der AUTOSAR OS Conformance Class 1 lassen dem Anwender dabei stets die größtmögliche Freiheit beim Einsatz von AUTOSAR-Software. Die I/O lässt sich im Gegensatz zu serienbasierten Basis-Software-Werkzeugen auf der MicroAutoBox II gewohnt einfach und komfortabel konfigurieren.

**Dipl. Inf. Dominik Holler** ist Softwareentwickler Experment Software bei der dSpace GmbH in Paderborn.

**Dr. rer. nat. Karsten Krügel** ist Senior Produktmanager Virtual Validation bei der dSpace GmbH in Paderborn.

**Dipl. Ing. Robert Leinfellner** ist Gruppenleiter Real-Time Processor Core bei der dSpace GmbH in Paderborn.