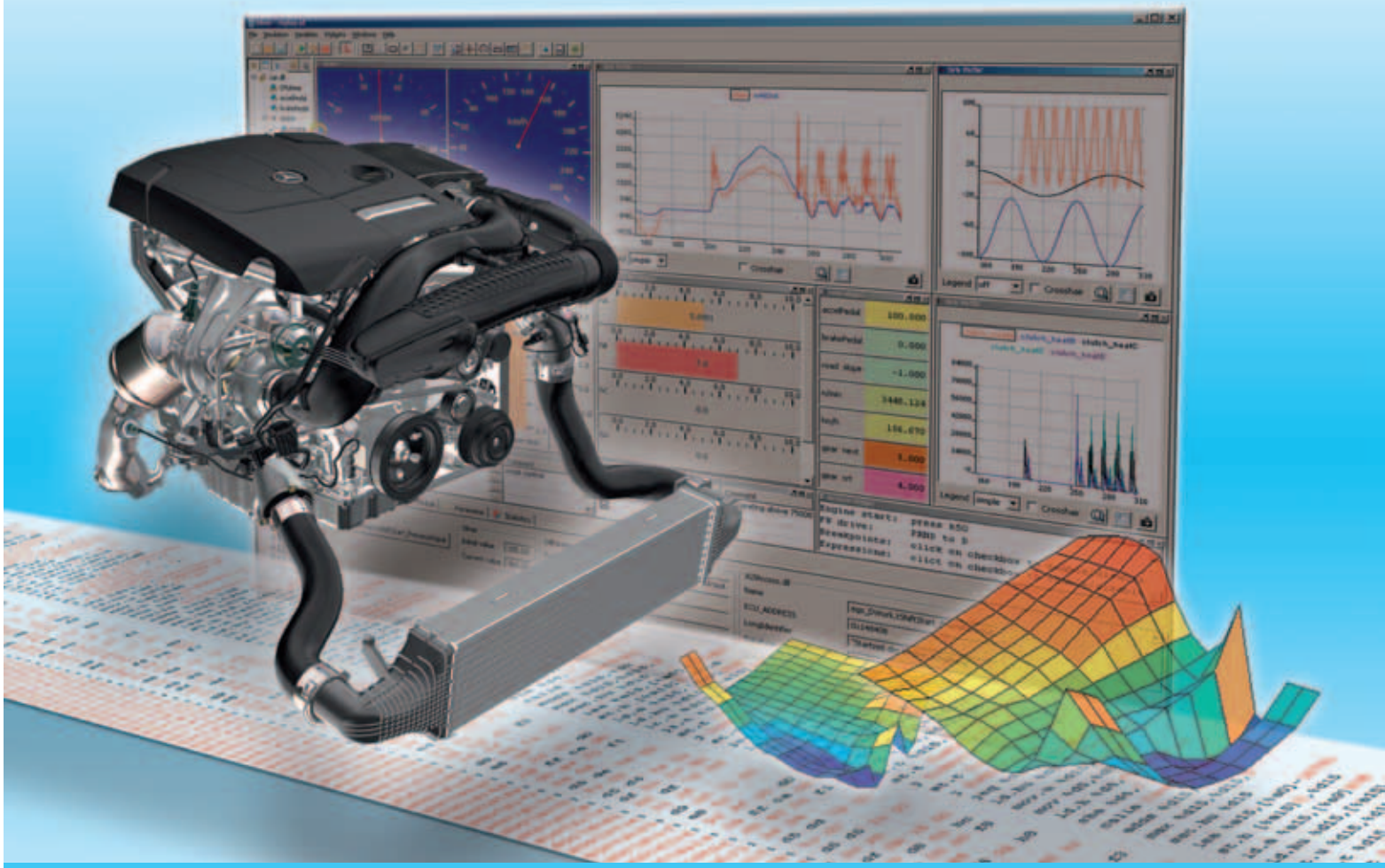


STEUERGERÄTE-SIMULATION AUF PC MITTELS TRICORE-EMULATION

Motorsteuergeräte enthalten heute Zehntausende von einstellbaren Softwareparametern. Das Kalibrieren dieser Parameter ist arbeitsintensiv und komplex. Die Simulation auf einem PC ermöglicht es, das Kalibrieren teilweise zu automatisieren und so zu beschleunigen, besonders wenn die Simulation viel (bis zu 20 mal) schneller als die Echtzeit läuft. QTronic und Daimler beschreiben den Simulationsprozess und wie dieser in der Motorenentwicklung beim OEM eingesetzt wird.

AUTOREN

DR. JAKOB MAUSS
ist Mitglied der Geschäftsführung der QTronic GmbH in Berlin.

MATTHIAS SIMONS
ist Entwicklungsingenieur bei der Daimler AG in Stuttgart.

FEHLERQUELLEN AUSSCHALTEN

Motorsteuerungen werden typischerweise vom Automobilhersteller appliziert, während der Programmcode der Steuerung von einem Zulieferer entwickelt wird. Der OEM ist dann außer Stande, eine Steuergerätesimulation auf Basis des Programm-Codes aufzubauen. Stattdessen muss der OEM per „Reverse Engineering“ Ersatzmodelle der zu applizierenden Funktionen der Steuerung entwickeln – ein aufwendiges und fehleranfälliges Vorgehen. Um diese Situation zu verbessern, hat QTronic einen Chip-Simulator in das ECU-Simulationswerkzeug Silver integriert. Dadurch kann ein „hex File“, das für einen TriCore-Prozessor kompiliert wurde, direkt auf einem PC ausgeführt werden. Dazu wird folgendes benötigt:

- ein hex File, das den Programmcode und die Parameter der zu simulierenden Funktionen enthält
- die Startadressen der zu simulierenden Funktionen
- ein A „SAP2/a2l-File“, das die Skalierungsregeln und Adressen der beteiligten Eingänge und Ausgänge sowie Parameter enthält.

Die Startadressen der Funktionen können beispielweise durch ein „map File“ gegeben werden, das zusammen mit dem hex File von einem TriCore-Kompiler erzeugt wurde. Silver verwendet das a2l File, um während der Simulation Integerwerte der Inputs, Outputs und Parameter in physikalische Einheiten umzurechnen. Eine TriCore-Simulation kann auch als S-Funktion (mexw32 File) exportiert werden und ist dann in Matlab/Simulink ausführbar. Auf einem durchschnittlichen PC läuft eine solche Simulation mit 40 MIPS. Wenn man nur einige Funktionen einer Motorsteuerung simuliert, läuft eine Simulation damit in vielfacher Echtzeit. In diesem Artikel wird auch beschrieben, wie solche Chip-Simulationen heute für die Entwicklung von Ottomotoren bei Daimler eingesetzt werden.

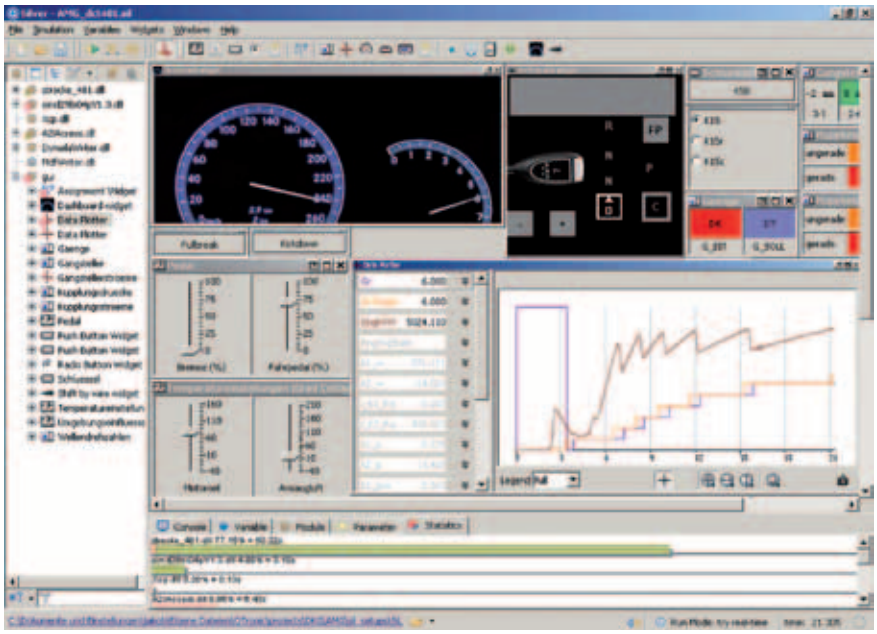
VIRTUELLE STEUERGERÄTE IM ENTWICKLUNGSPROZESS

Simulation bietet glänzende Möglichkeiten, den Entwicklungsprozess für Steuergeräte zu optimieren. Simulation hilft, Entwicklungsschritte auf PC zu verlagern, wo sie oft schneller, billiger oder in anderer Beziehung besser ausgeführt werden können. Einige Beispiele:

- Auf einem PC kann eine Simulation jederzeit angehalten werden, um den sogenannten Call Stack und alle Variablen eines virtuellen Steuergeräts zu inspizieren. Die Simulationen sind auf dem PC exakt und so oft wie nötig wiederholbar. Dagegen muss eine Simulation auf dem Prüfstand oder HiL in Echtzeit ablaufen. Anhalten oder "Steppen" ist dann unmöglich oder erfordert beträchtlichen Zusatzaufwand, zum Beispiel auf Basis der JTAG-Debug-Schnittstelle zum Prozessor. Die exakte Wiederholung von Experimenten ist hier wegen nicht-deterministischer Effekte nicht möglich.
- Ein auf PC laufendes virtuelles Steuergerät kann mit Mess- und Kalibrierwerkzeugen wie Inca (Etas) und CANape (Vector) via XCP appliziert werden. Dadurch können viele Applikationsaufgaben auf einer relativ billigen und hoch verfügbaren Plattform durchgeführt werden, ohne dabei Prüfstände und Versuchsfahrzeuge zu blockieren.
- Eine PC-Simulation kann mehrfach schneller als Echtzeit laufen. In Kombination mit Testautomatisierung erzielt man so einen mehrfach höheren Testdurchsatz als auf einem deutlich teureren HiL Testsystem.
- Ein Funktionsentwickler kann dank inkrementellem "Build" schon fünf Minuten nach Editieren einer Funktion den neuen Stand seines Steuergerätes auf PC probefahren. Dies hilft, Probleme früh zu entdecken und reduziert die Anzahl der Probleme, die erst relativ spät, während der Integration aller Module entdeckt werden. Die Erfahrung zeigt, dass solche frühen Checks die Entwicklungszeit spürbar verkürzen.

Um diese und andere Vorteile zu nutzen, muss das Steuergerät zunächst auf PC portiert werden, üblicherweise auf Basis des Programm-Codes der Steuerung, wie mit Ascet (Etas), TargetLink (dSpace) oder Embedded Coder (MathWorks) erzeugt. Silver [1] von QTronic ermöglicht das schnelle Portieren solcher Steuergeräte auf PC. Dazu werden:

- die zu simulierenden Tasks für Windows PC kompiliert
- das RTOS und andere Dienste (CAN, XCP) auf PC emuliert



❶ Getriebesteuerung in Silver [5]

: das resultierende virtuelle Steuergerät in der Schleife mit einer Fahrzeugsimulation oder „Open-loop“ auf PC ausgeführt.

Fahrzeugmodelle werden dabei aus Matlab/Simulink, Dymola, SimulationX oder MapleSim nach Silver importiert, beispielsweise als „FMU for model exchange“ [4]. Typische Anwendungen dieses Vorgehens sind in [2, 5] beschrieben, ❶.

Leider ist der Programmcode der Steuerung aber nicht immer verfügbar. Gründe hierfür sind:

: der Schutz von IP – die Steuerung wurde ganz oder teilweise von einem Zulieferer entwickelt; dem OEM (oder seinem Entwicklungsdienstleister) steht der Programmcode dann oft nicht zur Verfügung
: fehlende Portabilität des Programm-Codes, zum Beispiel wegen Verwendung von „inline Assembler Code“ oder anderen Kompiler-spezifischen Konstrukten, die die Kompilation für PC verhindern.

Um in solchen Situationen trotzdem auf PC simulieren zu können, haben Daimler und QTronic kürzlich einen Chip-Simulator in Silver integriert. Damit kann jetzt ein virtuelles Steuergerät auf Basis des für den Zielprozessor kompilierten hex File aufgebaut werden. Dazu wird kein Programm-Code (C-Code) mehr benötigt. Stattdessen führt der Chip-Simulator die im hex File kodierten Maschineninstruktionen direkt auf Windows PC aus.

CHIP-SIMULATION FÜR TRICORE-PROZESSOREN

Viele Fahrzeugsteuergeräte arbeiten mit Prozessoren der TriCore-Familie von Infineon, besonders im Antriebsstrang. Beispiele sind Motorsteuerungen der MED- und EDC-Familie von Bosch und Getriebesteuerungen von Continental.

Silver 2.5 verwendet ein Spezifikations-File, ❷, das die zu simulierenden Tasks (Funktionen) eines hex File auflistet. Silver erzeugt aus einer solchen Spezifikation automatisch ein ausführbares Silver-Modul (dll) oder eine SFunktion,

„Spez.File“ listet zunächst die verwendeten Files, ② (Zeilen 2 bis 5). Das Map File" ist optional. Falls vorhanden, darf man im Spez.File symbolische Funktionsnamen verwenden (wie ABCDE_20ms), statt Startadressen (beispielsweise 0x80081cde). „File frames.s“, ② (Zeile 5), enthält den „Startup-Code“ der wie üblich Stacks, Register, Timer und andere Ressourcen initialisiert. Die Zeilen, ② (10 bis 14), benennen die zu simulierenden Funktionen, und legen Zeitpunkt und Reihenfolge der Ausführung fest.

Silver verwendet diese Information für die RTOS-Emulation. Für ereignisgetriggerte Funktionen (zum Beispiel kurbelwellensynchron, ② (Zeile 12), bietet Silver zwei verschieden akkurate Ausführungsmodelle. ② (Zeilen 17 bis 19) schließlich definieren die Eingänge, Ausänge und Parameter des generierten Silver-Moduls beziehungsweise der SFunktion. In ② wird das Interface eines a2l Function Element namens ABCDE wiederverwendet. Alternativ können auch einzelne Variablen (a2l Measurements) verwendet werden, sofern ihre Eigenschaften (Adresse, Skalierungsregel, Datentyp etc.) im a2l File hinterlegt sind. Außerdem kann im Spez.File Folgendes festgelegt werden:

: Konfiguration der XCP-Emulation, für Online-Kalibrierung auf PC mit INCA oder CANape
: Data Sections des hex File, die in das generierte Silver Modul (beziehungsweise die SFunktion) eingebettet werden sollen. Dadurch kann das Laden des hex File zur Laufzeit umgangen werden, was die Ausführungszeit verkürzt
: Speicherbereiche, die beim Startup in anderen (schnelleren) Speicher kopiert werden sollen
: Ersatzfunktionen für bestimmte Funktionen. Damit werden etwa sogenanntgen getter- und setter-Funktionen für Sensoren und Aktuatoren durch Funktionen ersetzt, die direkt auf das Streckenmodell oder Messungen zugreifen
: Logging-Optionen, beispielsweise für Lese- oder Schreibzugriff auf bestimmte Speicherstellen während der Simulation.

Ein so generiertes Silver-Modul (beziehungsweise SFunktion) führt auf einem PC exakt dieselben Berechnungen aus wie auf dem Zielprozessor, weil der Effekt jeder einzelnen Maschineninstruktion auf Prozessorregister und Steuergerätespeicher exakt auf PC nachgebildet ist. Dabei bestehen allerdings folgende Einschränkungen:

: die Simulation ist instruktionsakkurat, aber nicht Cycle-akkurat: Die Simulation auf PC kann also nicht exakt vorhersagen, wie viele Prozessorzyklen (Takte) eine bestimmte Funktion auf dem Zielprozessor benötigt. Zum Beispiel sind Pipeline-Effekte und unterschiedliche Speicherzugriffszeiten (zum Beispiel Cache) nicht nachgebildet.
: Tasks werden in der Simulation „unendlich schnell“ ausgeführt. Das emulierte RTOS unterbricht daher nie einen Task. Entsprechende Effekte kön-

```
# specification of sfunction or Silver module
hex_file(m12345.hex, TriCore_1.3.1)
a2l_file(m12345.a2l)
map_file(m12345.map)      # a TASKING or GNU map file
frame_file(frame.*)       # assembler code to emulate RTOS
frame_set(STEP_SIZE, 10) # Silver step size in ms
frame_set(TEXT_START, 0xa0000000) # location of frame code

# functions to be simulated, in order of execution
task_initial(ABCDE_ini)
task_initial(ABCDE_inisyn)
task_triggered(ABCDE_syn, trigger_ABCDE_syn)
task_periodic(ABCDE_10ms, 10, 0)
task_periodic(ABCDE_200ms, 200, 0)

# interface of the generated sfunction or Silver module
a2l_function_inputs(ABCDE)
a2l_function_outputs(ABCDE)
a2l_function_parameters_defined(ABCDE)
```

❷ Spezifikations-File für eine Chip-Simulation

nen daher hier nicht beobachtet und analysiert werden.

: Silicon Bugs sind nicht simuliert. Falls ein Kompiler für den Zielprozessor einen solchen Prozessorbug nicht korrekt kompensiert, ist dies in der Regel nicht per PC-Simulation entdeckbar.

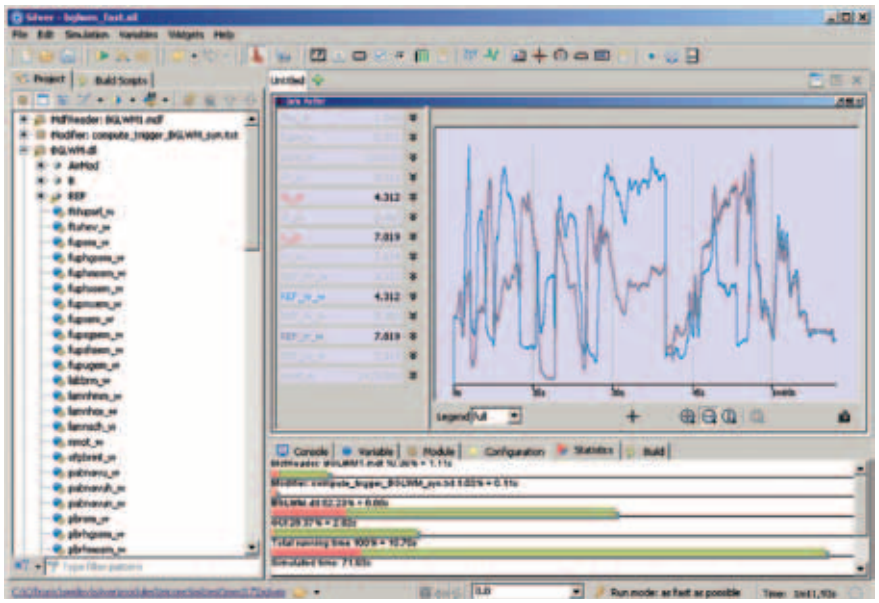
: On-Chip Devices wir der PCP-Koprozessor oder CAN Controller sind nicht simuliert.

Das verwendete Spezifikations-File kann Fehler enthalten. Silver enthält daher für die Fehlersuche auch einen Debugger, und zwar auf Basis des Instruction-Set-Simulators tsim von Infineon.

Um die Ausführungsgeschwindigkeit der Chip-Simulation zu bestimmen, hat QTronic eine komplexe Funktion einer Motorsteuerung (aus MED17 mit TC1797), ❸, auf PC portiert. Die Funktion besteht auf Top-Level aus fünf Tasks, die alle 10 und 200 ms laufen, beziehungsweise nur initial oder kurbelwellensynchron. Das Spez.File dazu ähnelt dem aus ②. Die Funktion hat 114 skalare Inputs, 102 skalare Outputs und 108 Parameter (a2l Charateristics), viele davon Achsen und Kennfelder. Alle Inputs und Outputs wurden auf einem Motorprüfstand für ein Szenario von 3,5 min Dauer gemessen. Mit dem resultierenden Messfile (mdf/dat) wurde dann in Silver-open-loop auf PC eine Simulation getrieben. Jede Simulation umfasste 380.205256 Millionen Instruktionen (von tsim gezählt) und wurde fünf mal auf PC (Intel-i5-Prozessor mit 2,4 GHz und 2,92 GB RAM) wiederholt. ❹ zeigt die Mittelwerte der so erhaltenen Ausführungszeiten. Silver kann für ein gegebenes Spezifikations-File auch eine SFunktion (mexw32 File) generieren. Die SFunktion kann dann in Matlab/Simulink für die automatische Optimierung von Applikationsparametern verwendet werden. Die generierte SFunktion akzeptiert alle im Spez.File gelisteten Parameter als Funktionsargumente. Die SFunktion kann also zum Beispiel mit Matlab-Workspace-Variablen gerufen werden, die dann von einer Optimierungsfunktion in Matlab/Simulink automatisch variiert werden. Eine SFunktion läuft in Simulink mit ungefähr 40 MIPS.

ANWENDUNGEN DER CHIP-SIMULATION

Funktionsentwickler möchten gelegentlich vorgefundene Steuergerätefunktionen durch selbst entwickelte Funktionen ersetzen. Dafür werden Werkzeuge wie E-Hooks (Etas) oder No-Hooks (ATI) eingesetzt. Diese Werkzeuge ersetzen im hex File original Funktionen durch neue Funktionen. Dies ist enorm nützlich, birgt aber auch gewisse Risiken. Manchmal werden die Ersatzfunktionen nicht wie gewünscht gerufen oder Variablen nicht wie erwartet gelesen oder geschrieben. Dies wird üblicherweise erst erkannt, nachdem das manipulierte hex File auf das Steuergerät



❸ Simulation der BGLWM-Funktion in Silver, getrieben von einem Messfile

geflasht und auf einem Prüfstand oder im Fahrzeug in Betrieb genommen wurde. Die Simulation auf PC bietet hier die Möglichkeit, solche Fehler früh und ohne jedes Risiko für Prüfstand oder Fahrzeug zu erkennen.

Ein zweite Anwendungen der Chip-Simulation ist die numerische Optimierung von Steuergeräteparametern [3]. Daimler hat die Chip-Simulation wie oben beschrieben mit einem numerischen Verfahren zur Berechnung optimaler Parameter kombiniert. Das Optimierungsverfahren benötigt ein genaues und schnell rechnendes Modell der zu optimierenden Motorfunktion. In der Vergangenheit haben die Software-Ingenieure hierzu handkodierte Simulink-Modelle verwendet. Die Entwicklung der benötigten Ersatzfunktionen war arbeitsintensiv und fehleranfällig. QTronic konnte diese handkodierten Modelle jetzt teilweise durch SFunktionen ersetzen, die mit Silver aus hex Files automatisch erzeugt wurden. Diese generierten Funktionen laufen in Simulink genauso schnell, wie ihre handkodierten Gegenstücke.

ZUSAMMENFASSUNG

Ein hex File, das für einen Zielprozessor kompiliert wurde, ist wie oben gezeigt per Chip-Simulation auf Windows PC ausführbar, entweder Open-loop getrieben von einem Messfile, oder in der Schleife mit einem Fahrzeugmodell. Dabei können je nach Anwendungsfall einzelne Funktionen oder fast komplette Steuergeräte auf PC simuliert werden.

Diese Art der Simulation eröffnet neue Möglichkeiten, bestimmte Entwicklungsschritte vom Fahrversuch, Prüfstand und HiL auf PC zu verlagern, wo sie schneller, billiger oder sonst wie besser ausgeführt werden können, ohne Zugriff auf den den Programmcode (Modell oder C Code) der Steuerung. Daimler setzt diesen neuen Ansatz inzwischen für die Entwicklung von Benzinmotoren ein. Weitere Anwendungen sind denkbar, zum Beispiel die Online-Applikation von Steuergerätefunktionen auf PC via XCP.

LITERATURHINWEISE

[1] Junghanns, A.; Serway, R.; Liebezeit, T.; Bonin, M: Building Virtual ECUs Quickly and Economically, ATZelektronik 03/2012, Juni 2012
[2] Brückmann, H.; Strenkert, J; Keller, U.; Wiesner, B.; Junghanns A.: Model-based Development of a Dual-Clutch Transmission using Rapid Prototyping and SiL. International VDI Congress Transmissions in Vehicles 2009, Friedrichshafen, Germany, 30.06.-01.07.2009. http://QTronic.de/doc/DCT_2009.pdf
[3] Röpke , R.(ed.): Design of Experiments (DoE) in Engine Development – Innovative Development Methods for Vehicle Engines. Expert Verlag, 2011
[4] Blochwitz, T.; Otter M. et. al.: Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. 9th International Modelica Conference, Munich, 2012
[5] Tatar, M.; Schaich, R.; Breitinger, T.: Automated test of the AMG Speedshift DCT control software. 9th International CTI Symposium Innovative Automotive Transmissions, Berlin, 2010. http://QTronic.de/doc/TestWeaver_CTI_2010_paper.pdf

AUSFÜHRUNGSPLATTFORM	LAUFZEIT [s]	MIPS
SILVER IM DEBUG-MODUS (TSIM)	919,15	0,41
GENERIERTES SILVER-MODUL ODER MATLAB/SIMULINK-SFUNCTION	9,30	40,80
MED17 MIT TC1797, 180 MHz	210,00	270

❹ Laufzeiten der Chip-Simulation für die BGLWM-Funktion

LIGHTING-LÖSUNGEN FÜR AUTOMOTIVE-CLUSTER

■ 12-kanaliger LED-Treiber ■ High-Brightness RGB-LEDs ■ Komplettlösung

Der neue LED-Treiber BD18377 bietet zusammen mit den branchenweit flachsten RGB-LEDs ein komplettes Design für beleuchtete Anzeigen in Automotive-Cluster-Lösungen.

12-kanaliger LED-Treiber

BD18377

- Programmierbarer Konstantstrom Ausgang mit 15 bis 50 mA je Kanal; Programmierung mit nur einem externen Widerstand
- Optimierte Genauigkeit bei 30 mA:
  Kanalstreuung: ± 2.5%
  Bauteilstreuung: ± 1.7%
- 4-adriges SPI-Interface (Clock, Data-In, Data-Out, Latch)
- Diagnoseausgang für LED-Status
- PWM-Dimmung 0,1..100 % (global)
- Individuelle Kalibrierung der LED-Helligkeit (6 Bit)
- Daisy-Chain-kompatibel (zu BD8377 und BD18377)
- Kompaktes HTSSOP-20-Gehäuse
- AEC-Q100-qualifiziert

Ultraflache, dreifarbige High-Brightness-RGB-LEDs

SRGB-Serie

- Branchenweit flachstes PLCC6-Gehäuse (h = 0,6 mm)
- Hohe Helligkeit: 1,8 cd (weiß)
- Herausragende Farbübereinstimmung
- Gleichförmige Lichtverteilung
- Optimierte Farbmischung

SRGB2 (PLCC6) **SRGB-S**

Kompakte, dreifarbige High-Brightness-RGB-LEDs

Die neue GC-RGB-Serie

- Branchenweit kleinste RGB-LED in Reflektorbauart (1816-Format)
- Schwarze Reflektorbauart
- Hohe Helligkeit
- Überragende Farbmischung
- Mit 4 oder 6 Pins verfügbar