

# Wechsel von CAN zu FlexRay

AUTOMOBIL-ELEKTRONIK beschreibt, wie ein vorhandenes CAN-basiertes System vermessen, modelliert und simuliert wird. Im Modell wird die geplante Änderung auf FlexRay vorgenommen und so frühzeitig verfügbar gemacht. Das **ZUSAMMENSPIEL DER ECHTZEITSYSTEME** (Steuergerät asynchron zum FlexRay) wird **IM ECHTZEITSIMULATOR** dargestellt. Mittels Sensitivitätsanalysen können die kritischen Systemzustände ermittelt werden.

Steuergeräte mit CAN-Bus-Kommunikation sind Standard geworden. Mit ständig steigender Funktionalität und Variantenanzahl bietet die Steuergeräte-Entwicklung durchaus noch Herausforderungen. Insbesondere die Performance des Gesamtsystems sowie die Bus- und CPU-Last lassen sich nur schwer bis ins letzte Detail durchplanen.

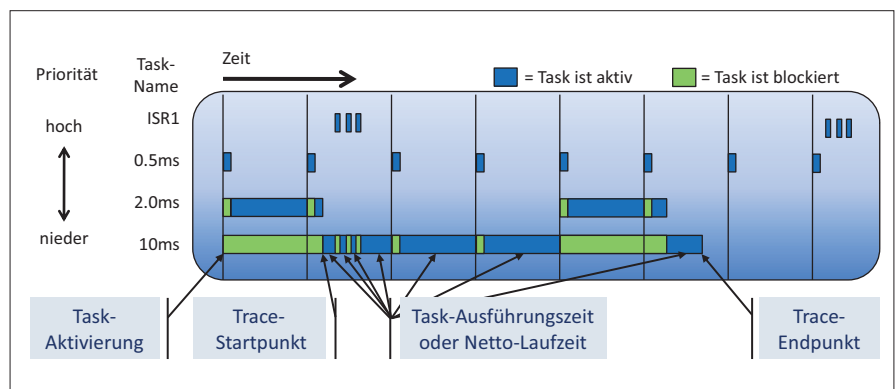
Werkzeuge, die sowohl in einer frühen Architekturphase als auch spät, wenn es langsam knapp wird, die schnelle Analyse von Design-Varianten erlauben, sind rar. Mit Hilfe der Echtzeitmodellierung können die dynamischen Abläufe in den Steuergeräten beschrieben und in der Echtzeitsimulation effektiv analysiert werden. Auch eine Umstellung von CAN auf FlexRay lässt sich mit der Echtzeitsimulation deutlich besser beherrschen.

Für die Ausgangssituation soll als Beispiel ein Serienprojekt dienen, bei dem die B-Muster-Phase bereits abgeschlossen ist. Da mehrere neu entwickelte Soft-

## Was zu einer Verdrängung oder einer Unterbrechung geführt hat, lässt sich erst durch Kenntnis der Task-Zustände erkennen

ware-Anteile von Projektpartnern zugeführt wurden, war das Echtzeitverhalten schwer gemeinsam zu definieren und später einzuhalten.

Das System ist im Verlauf der Entwicklung über die gesetzten Grenzen hinaus gewachsen. Es zeigt neben einer zu hohen CPU-Last auch unerklärliche Effekte in den Messspuren sowie Probleme mit dem Jitter einiger Funktionen. Nun gilt es, in kurzer Zeit Verständnis für das dynamische Systemverhalten zu gewinnen, um Optimierungspotenziale zu finden sowie die Bedingungen zu identifizieren, die zu Worst-Case-Situationen führen.



Die schematische Darstellung eines Task-Gefüges verdeutlicht, wie Tasks niedriger Priorität blockiert und verdrängt werden, so dass die Brutto-Ausführungszeit wesentlich länger als die reine Rechenzeit ist.

### Zuerst messen

Den Anfang macht eine Aufzeichnung wesentlicher Ereignisse im Ablauf der Software als Trace (Messspur) im Speicher des Steuergeräts. Durch Einfügen entsprechender Befehle im Source-Code werden die Start- und Endzeitpunkte der Ausführung von Tasks und ISR (Interrupt Service Routinen) ermittelt. Leider ist diese Methode nicht beeinflussungsfrei, da zusätzliche Befehle mehr Rechenzeit beanspruchen. Zudem ist der Speicher des Steuergeräts beschränkt, so dass nur ein kurzer Zeitausschnitt einer Betriebsituation erfasst wird.

Mit der Kenntnis der Perioden und Prioritäten der Tasks und ISR können aus den im Trace enthaltenen Brutto-Laufzeiten die Netto-Laufzeiten ermittelt werden (Tasks niedrigerer Priorität werden von ISR und höherer Tasks höherer Priorität unterbrochen und benötigen damit scheinbar länger). Je nachdem, welche Betriebsituationen erfasst werden und welche Ereignisse der aufgezeichnete Zeitausschnitt enthält, stellen die Netto-

Laufzeiten eine recht gute Auflistung der dynamischen Ausführungszeiten des Source-Codes dar.

Doch was hilft das beste Zahlenwerk, wenn daraus kein Gesamtbild der Abläufe deutlich wird? Im Trace sieht man zunächst nur, wann eine Task vom Scheduler auf der CPU zur Ausführung ausgewählt wurde. Was zu einer Verdrängung oder einer Unterbrechung geführt hat, lässt sich erst durch Kenntnis der Task-Zustände erkennen. Durch den begrenzten Speicher des Steuergeräts, werden im demzufolge nur kurzen Trace keine langfristigen Effekte sichtbar, wie sie durch Drift der verschiedenen Uhren im System entstehen.

### Echtzeitmodellierung

Man muss längst keine Lanze mehr für Simulation brechen. Das bessere Verständnis komplexer Zusammenhänge durch Simulationsmodelle ist in vielen Bereichen der Naturwissenschaften und Ingenieursdisziplinen längst bewiesen. Bildet man das System durch ein Simulationsmodell nach, so sind innerhalb kurzer Zeit detaillierte Analysen über lange Zeiträume möglich. Im Simulator kann das System beliebig lange Simulations-

läufe durchführen und verschiedene Stimulationsszenarien durchlaufen.

Grundvoraussetzung dafür ist jedoch, dass das Modell möglichst viel Information über das System korrekt darstellt und die dynamischen Abläufe sowie die kausalen Zusammenhänge (Ablaufreihenfolgen, Reaktion auf bestimmte Stimuli etc.) mit simuliert werden. Durch Import der OIL-Files (OSEK-Beschreibung der Prioritäten der Tasks) und die Modellierung in C ist die Erstellung eines aussagekräftigen Simulationsmodells in wenigen Manntagen möglich.

Das Echtzeitmodell beschreibt die einzelnen Tasks und ISR mit ihren Ausführungszeiten auf einer abstrakten Ebene, die es erlaubt das Scheduling leicht nachzuvollziehen. Der Simulator stößt die zeit- und ereignisgesteuerten Vorgänge gemäß dem Scheduling des Betriebssystems an und stellt sie in aussagekräftigen Diagrammen und Statistiken dar. Ein Vergleich der gemessenen und simulierten Traces gibt dem Entwickler ein Maß, wie gut das Modell mit der Realität übereinstimmt.

### Effektive Analyse und Optimierung

Mit diesem Hilfsmittel können die Ingenieure effektiv an die Lösung der Probleme des weit fortgeschrittenen Projekts gehen. Zunächst wird das System in diversen Betriebszuständen simuliert sowie die Auswirkung auf das Timing einzelner Tasks und der CPU-Last beobachtet. Durch gezieltes Variieren der Stimuli und Ausführungszeiten wird eine Sensitivitätsanalyse möglich. Sofern in den Traces auch Messpunkte innerhalb der Tasks erfasst wurden, lässt sich sogar untersuchen, wie sich eine Verschiebung von Funktionsblöcken in andere Tasks auf das Zeitverhalten auswirken würde.

In unserem Beispiel diente das beschriebene Vorgehen dazu, diejenigen

Programmteile zu identifizieren, in denen eine Optimierung des Source-Codes signifikante Vorteile für die System-Performance erwarten ließen. Aus der Sensitivitätsanalyse wurde deutlich, welche Interrupt-Rate und -Phasenlage (Interrupts vom CAN-Bus und von Sensoren rotierender Mechanik) zu Interferenzen mit den Zyklen der zeitgesteuerten Tasks führten. Wenn auch nicht alle Erkenntnisse automatisch zu erfolgreichen Ab-

## Beim Wechsel von CAN auf FlexRay wird eine systematische Überprüfung der Software-Architektur notwendig

hilfemaßnahmen geführt haben, so hat die Echtzeitsimulation doch das Systemverständnis erhöht und das Vertrauen des OEM in das System trotz der hohen CPU-Last verbessert.

### Von CAN zu FlexRay

In der nächsten Variante soll das gleiche System hauptsächlich mittels FlexRay im Fahrzeug angebunden sein. Die erste naive Erwartung könnte nun sein, dass mit diesem Echtzeitbus und der deutlich höheren Datendurchsatzrate auch die System-Performance leichter erreichen lässt. Allerdings setzt sich im Markt auch hier zunehmend die Erkenntnis durch, dass der CAN-Bus nicht „einfach so“ durch FlexRay ersetzt werden kann.

Der Wechsel von CAN- auf FlexRay-Kommunikation ist ein Wechsel von einer ereignis- zu einer zeitgesteuerten Architektur, wodurch die Entwickler die Anregung des Echtzeitsystems grundlegend umstellen müssen. Dadurch wird eine systematische Überprüfung der Software-Architektur notwendig.

Das bekannte Zeitverhalten der CAN-Variante liefert die Daten, um eine Flex-

Ray-taugliche Software-Architektur schnell zu ermitteln. Da der Großteil der Applikation unverändert bleiben soll, können deren Ausführungszeiten im Echtzeitmodell der FlexRay-Variante ebenso wiederverwendet werden. Das neu geplante Zeitraster muss aber die festen Zyklen des FlexRay berücksichtigen, um Interferenzen eines asynchronen Timings zu vermeiden. Die veränderte Bus-Kommunikation bildet der Simulator mit den vorhandenen Modellen der FlexRay-Controller ab. Der Simulator berücksichtigt deren Aufstart- und Synchronisationsmechanismen wie zuvor schon die Konfiguration und das Scheduling des CAN-Busses.

Im Simulationsmodell der geplanten FlexRay-Version des Steuergeräts kann der Systemarchitekt in kurzen Iterationszyklen das optimale Timing und Task-Gezüge ermitteln. Verschiedene Synchronisationsmöglichkeiten lassen sich im Task-Modell schnell hinzufügen und in der Echtzeitsimulation analysieren. Dabei behält der Architekt die Performance des Gesamtsystems im Auge und kann frühzeitig das Echtzeitverhalten überprüfen. Schon in der ersten Implementierung des FlexRay-Systems wird ein so gutes Zeitverhalten erreicht, dass eine Iterationschleife in der Entwicklung eingespart werden kann. ←



Dipl.-Ing. Tapio Kramer ist verantwortlich für das technische Marketing bei Inchron

infoDIRECT [www.all-electronics.de](http://www.all-electronics.de)

Link zu Inchron:

331AELeS08