

Messen und Simulieren am FlexRay-Bus

Analyse und Simulation von FlexRay-Bussen und deren Anwendungen

Mit der Einführung des FlexRay-Busses steht die Automobilelektronik vor einem großen Umbruch. Durch den Schritt weg von der Ereignis- hin zu einer Zeitsteuerung werden höhere Ansprüche an das Zeitverhalten der Applikation gestellt. Dies schafft bei Entwicklung und Test neue Herausforderungen.

Von Dr. Carsten Böke

Die Aufgabenstellungen für die Entwickler von FlexRay-Systemen sind bekannt: Von der Netzwerkanalyse und der Restbus- und Umgebungssimulation der Architektur über Stimulus- oder Conformance-Tests von Steuergeräten bis hin zur Einbindung der Tools verschiedener Anbieter müssen Arbeiten und Untersuchungen erledigt werden, die auch bei den bereits etablierten Bussystemen notwendig sind. In der Praxis kommen jedoch aufgrund der Besonderheiten des FlexRay-Protokolls neue, ganz unterschiedliche Heraus-

forderungen auf die Entwicklerteams zu – im Sinne von Analyse- und Simulationsaufgaben. Und dafür benötigt man entsprechende Tools.

FlexRay: Seine Charakteristika sind auch Herausforderung

Die grundlegende Neuerung, die mit dem FlexRay-Bus Einzug ins Fahrzeug hält, ist das deterministische Verhalten. Innerhalb eines fest vorgegebenen Zeitrasters sind jedem Steuergerät dedizierte Sendefenster zugewiesen, in denen es senden darf. Das Kommuni-

kationsverhalten aller Teilnehmer (Knoten) am Bus muss dafür zuvor genau geplant werden (Bild 1). Festgelegt ist dies in der so genannten FIBEX-Datei (Field Bus Exchange Format), die alle relevanten Parameter wie Topologie, Konfigurationsparameter, Frames, Schedule usw. enthält. Alle Busteilnehmer, sofern sie aktiv an der Kommunikation teilnehmen wollen, müssen zueinander synchron arbeiten, damit sie auch nur die ihnen zugeordneten Sendeslots verwenden.

Diese Synchronität ist ein wesentliches Merkmal der FlexRay-Kommunikation. Sie garantiert, dass unter den Busknoten eine zeitgenaue Kommunikation erfolgt und verteilte Regelsysteme exakt arbeiten können. Das Protokoll sieht daher das regelmäßige Versenden so genannter Sync-Frames zur Wahrung der Synchronität aller Teilnehmer über den Bus vor.

In der Praxis ergeben sich zwei Fälle, in denen diese Synchronität nicht (oder noch nicht) gewährleistet ist. Zum einen kann es aufgrund von technischen Defekten zum Ausfall mehrerer Sync-Frames kommen. Der zweite Fall tritt immer auf: In der Startup-

Phase, also beim Hochfahren des Busses, kann noch keine Synchronität bestehen, da sich die Teilnehmer erst nach Erhalt der ersten Sync- bzw. Startup-Frames synchronisieren können. Für einen Startup sind definitionsgemäß mindestens zwei Startup-Knoten im Bussystem notwendig.

Für die Netzwerkanalyse möchte man selbstverständlich alle Phasen der Kommunikation überwachen können. Vom Wecken der Busteilnehmer über die

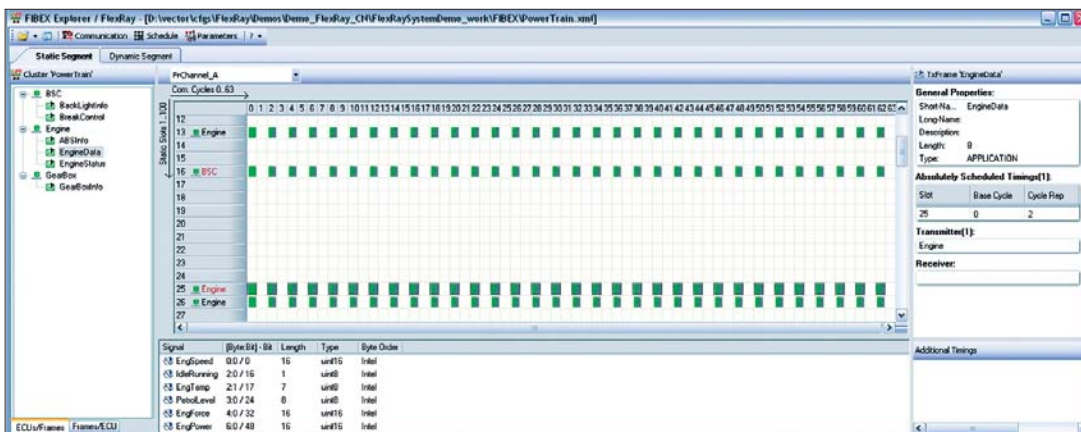


Bild 1. Darstellung des Schedule in einer FIBEX-Datei mit dem FIBEX-Explorer in CANoe.FlexRay 7.0, verfügbar ab November 2007.

(Screenshots: Vector Informatik)

Startup-Phase, dem „normalen“ synchronen Betrieb bis hin zum „Sleep-Mode“.

Das Tool CANoe von Vector Informatik (www.vector-informatik.de) bietet diese Möglichkeiten. Es arbeitet in beiden Modi und überwacht daher die Startup-Phase lückenlos. Beim Verlust der Synchronisation wechselt das Tool optional selbstständig vom synchronen Modus in die asynchrone Überwachung. Das Senden von Frames ist im asynchronen

Modus dann allerdings protokollbedingt nicht möglich. Darüber hinaus kann es Wakeup-Befehle senden oder den eigenen Transceiver in den Sleep-Modus versetzen, um einen abgeschalteten Knoten zu simulieren. So ist eine vollständige Analyse und Simulation des Busses gewährleistet – auch im und für den Fehlerfall.

Ein Steuergeräte- bzw. Bus-Test ist unter Umständen auch dann notwendig, wenn keine aktuelle FIBEX-Datei vorliegt. In einer frühen Phase der Steuergeräte-Entwicklung kann dies durchaus vorkommen, ebenso nach Versionsänderungen oder der ersten allgemeinen Kommunikationsprüfung eines neuen Steuergerätes (ECU). Um so einen ersten Test durchführen zu können, muss CANoe lediglich die Übertragungsrate des Busses bekannt sein, damit eine Initialisierung des Bus-Interfaces erfolgen kann.

Wenn Fahrzeughersteller Steuergeräte von ihren Zulieferern bekommen, werden Conformance-Tests durchgeführt – zunächst nur die Kommunikationseigenschaften betreffend. Damit wird festgestellt, ob sich das Steuergerät wie gewünscht und entsprechend der vorgegebenen Schedule verhält: Sendet und empfängt die ECU in den vorgesehenen Slots die richtigen Nachrichten?

Der Cluster Monitor, das entsprechende Analysewerkzeug im Tool CANoe, zeigt, welche Nachrichten auf dem Bus gesendet werden, welche Frames einem Knoten in der Datenbank zugeordnet sind und gibt mittels einer einfachen Grün/Rot-Anzeige sofort den Hinweis, ob das Kommunikationsverhalten korrekt ist und wo Un-

stimmigkeiten auftreten (Bild 2). Auch Botschaften des dynamischen Segments werden angezeigt.

In AUTOSAR werden optional neben Frames auch PDUs (Protocol Data Units) beschrieben. Mit einer proprietären Abwandlung der FIBEX-Semantik können auch PDUs in OEM-spezifischen Datenbanken modelliert sein. CANoe unterstützt Systembeschreibungen in den FIBEX-Formaten der Version 1.1.5, 1.2 und 2.0 sowie die proprietäre PDU-FIBEX-Variante. Somit können Entwickler Analysen und Simulationen solcher Netzwerksysteme problemlos durchführen.

■ Simulation und Stimulation

Während der Steuergeräte-Entwicklung möchte man möglichst frühzeitig wissen, ob sich das FlexRay-System auch unter voller Buslast spezifikationskonform verhält und alle Nachrichten zum richtigen Zeitpunkt gesendet und empfangen werden. Da aber weder alle ECUs gleichzeitig fertig gestellt werden noch ein einzelner Zulieferer auch alle anderen Steuergeräte eines Bussystems zur Verfügung hat, werden fehlende Komponenten mit Hilfe

einer Simulation nachgebildet. Durch den einfachen Import einer FIBEX-Datei in CANoe kann eine Kommunikationsmatrix und der dazugehörige Sendezeitplan (Schedule) für die Simulation implementiert werden. Auf die Definitionen der Datenbank greift das Tool ganz einfach symbolisch zu.

Eine ECU wird sinnvollerweise „gegen den Rest“ der Bussystem-Teil-

Cluster:	Cluster Parameter - PowerTrain	
Statistikgruppe	Wert	Einheit
Parameter		
Zykluslänge	5	ms
Statische Slots	100	
Statische Payload	8	bytes
Dynamische Payload	24	bytes
Knoten gesamt	5	
Knoten konfiguriert	5	
Statistiken		
Frames	600	fr/s
Static Frames	600	fr/s
Dynamic Frames	0	fr/s
Zähler		
Zyklen	1024	
Frames	2736	
Static Frames	2736	
Nullframes	0	
Dynamic Frames	0	

Netzknoten	Total (A)	Total (B)	Statisch & Null (A)	Statisch & Null (B)	Dynamisch (A)	Dynamisch (B)
BLU (PowerTrain)	0 [0]	0 [0]	0 [0]	0 [0]	0	0
B5C (PowerTrain)	64 [64]	0 [0]	64 [64]	0 [0]	0	0
Dashboard (PowerTrain)	0 [0]	0 [0]	0 [0]	0 [0]	0	0
Engine (PowerTrain)	96 [96]	0 [0]	96 [96]	0 [0]	0	0
GearBox (PowerTrain)	32 [32]	0 [0]	32 [32]	0 [0]	0	0
Nicht zugeordnet	0	0	0	0	0	0

Name	ID	Kanal	Offset	Repetition	Duration [ms]	Invalid frames	Missing frames	Null frames	Frame Typ
ABSInfo	13	A	0	2	10	0	0	0	APPLICATION
EngineData	25	A	0	2	10	0	0	0	APPLICATION
EngineStatus	26	A	0	2	10	0	0	0	APPLICATION

! Bild 2. Der Cluster-Monitor zur Anzeige der Sendekonformität der FlexRay-Knoten und ihrer Botschaften.

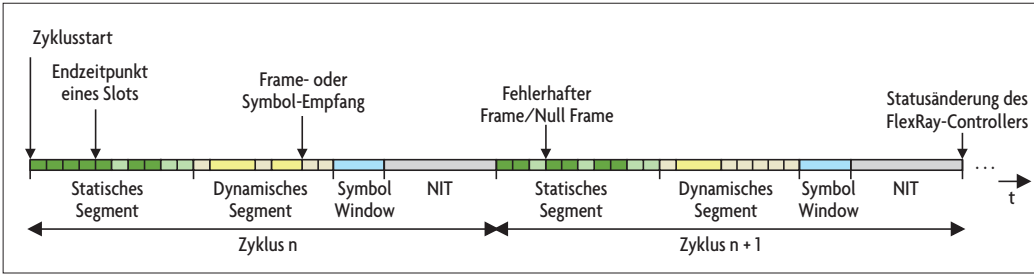


Bild 3. Benachrichtigungskonzept in CANoe für FlexRay-Ereignisse.

nehmer getestet, daher spricht man von einer Restbussimulation. Die Aufgabenstellung ist bei FlexRay deutlich schwieriger als beispielsweise bei CAN, denn immerhin arbeitet FlexRay bei etwa der zehnfachen Übertragungsgeschwindigkeit, dazu noch zeitgesteuert. Entsprechend schnell und rechtzeitig müssen auch die Daten für ein Frame-Update zur Verfügung stehen.

Würde man alle statischen Frames eines FlexRay-Systems mit Botschaften belegen, wären 2047 Slots ×

Tool automatisch – gemäß den Vorgaben aus der FIBEX-Datei.

Um sowohl den Synchronbetrieb und den asynchronen Modus sowie auch – wie eingangs beschrieben – den Startup des Bussystems ohne weiteren Startup-Knoten zu ermöglichen, haben die VN-Interfaces zwei Communication Controller. Zur gleichzeitigen Analyse mehrerer FlexRay-Cluster oder zur Simulation von FlexRay-zu-FlexRay-Gateways können sogar mehrere der genannten Interfaces eingebunden werden.

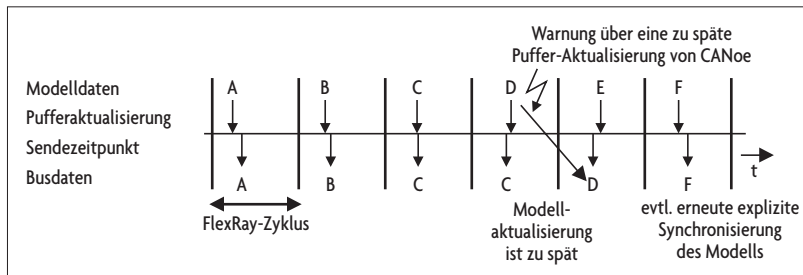


Bild 4. Optionale Überprüfung der Sendeaufträge auf Rechzeitigkeit (TX Check for Slot Miss).

64 Zyklen × 2 Kanäle auszufüllen und zu senden, insgesamt also 262 016 Frames. Die Botschaften müssen in Sendepuffern bereitgestellt werden, die vorab zu definieren sind – Ressourcen, die ein Controller „von der Stange“ nicht bietet. Vector Informatik hat für diese Aufgabe die Interfaces der Reihe VN3x00 entwickelt, die zusammen mit CANoe beliebig viele Frames senden können, ohne bei der Pufferbelegung an Grenzen zu stoßen. Lediglich der Pufferspeicher kann die Anzahl an Frames beschränken. Allerdings ist dieser mit 2 Mbyte sehr großzügig bemessen. Somit kann in den meisten Fällen eine Buslast von 100 Prozent simuliert werden. Über die notwendige Konfiguration und Reservierung der Sendepuffer muss sich der Entwickler keine Gedanken machen. Auch diese Aufgabe erledigt das

Natürlich müssen auch die richtigen Daten in den Botschaften stehen und sie müssen zum richtigen Zeitpunkt bereitgestellt werden. Die Modelle mit den dazugehörigen Informationen, wann ein Frame mit welchen Daten zu senden ist, müssen synchron zum Kommunikationsmedium

abgearbeitet werden. Dazu hat Vector das „Notification Concept“ entwickelt (Bild 3). Das Antriggern eines Modells kann dazu führen, dass erst eine größere Rechenaufgabe zu bewältigen ist, die das Bereitstellen einer Antwort-Botschaft so lange verzögert, bis der vorgesehene Sendeslot verpasst ist. Hier muss der Entwickler eine

„Worst Case Execution Time“ berücksichtigen. Der Notification Handler ist in der Lage, zu überprüfen, ob der nächste Slot erreicht wurde, und bei einem Fehler diesen zu melden (Bild 4).

Hardware als Nadelöhr

Als Grund für eine zu lange Berechnungszeit kommt manchmal die Laufzeitumgebung in Frage. Wenn die CPU der Entwicklungsplattform mit weiteren Tasks belegt ist, z.B. GUI oder Virens Scanner, und deshalb die Berechnung eines Regelalgorithmus verzögert wird, ist eine wirklichkeitstreuere Ausführungszeit nicht immer gewährleistet. Das Rechenergebnis steht dann unter Umständen nicht rechtzeitig für ein Update zur Verfügung.

Für diesen Fall bietet sich das Tool „CANoe RT“ (Real-Time) als Software-Lösung an. Um hohen Echtzeitanforderungen gerecht zu werden, wird dies durch explizit darauf abgestimmte Hardware ergänzt, auf der exklusiv ein Teil des Tools und der Modelle ausgeführt wird.

Verfügbar sind für dieses Tool auch die „RT-Box“ und das „RT-Rack“, welche fertig konfektioniert und konfiguriert sind (Bild 5). Diese dienen als erweitertes Multi-Bus-Interface (op-

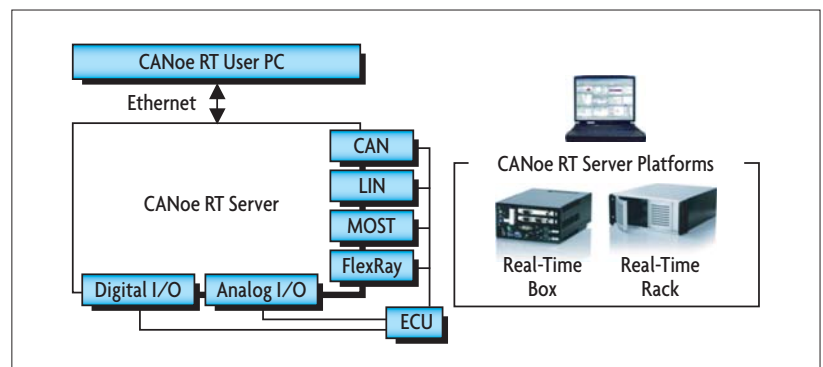


Bild 5. CANoe RT als echtzeitfähige und deterministische Ausführungsplattform.

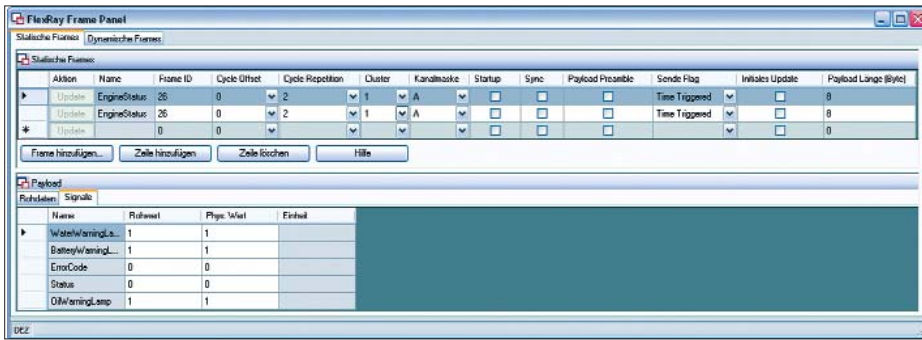


Bild 6. Das „FlexRay Frame Panel“ zum einfachen interaktiven Versenden von FlexRay-Frames.

tional auch mit digitalem und analogem I/O und weiteren Bus-Interfaces) für CANoe RT. Auf dieser RT-Hardware werden nicht nur die gesamte Bus-Kommunikation und die I/Os ver-

(Bild 6). Dies kann jederzeit zur Laufzeit geschehen. Ein Frame kann auch mehrfach mit unterschiedlicher Payload vordefiniert werden. Dies vereinfacht die Handhabung des Tools. Zusätzlich können mit Hilfe der Programmiersprache CAPL programmierbare Stimuli erzeugt werden, die auf zuvor empfangenen Botschaften beruhen. Da die Ausführung des Programms nativ auf Maschinenebene erfolgt, werden sehr schnelle Antwortzeiten erreicht.

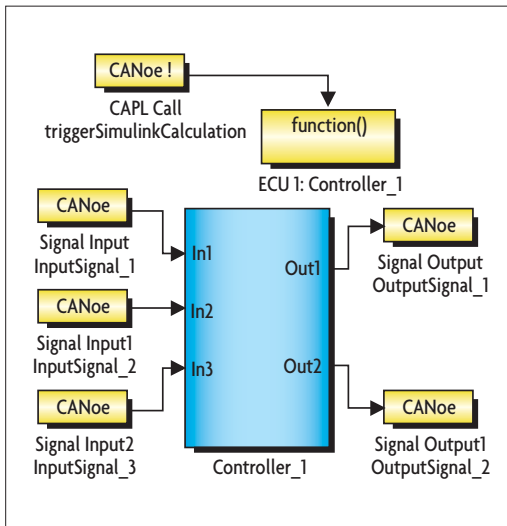


Bild 7. Modell mit integriertem CANoe-Blockset für den Real-Time-Workshop zum Empfangen und Lesen von Signalen sowie zur synchronen Modellausführung in MATLAB.

arbeitet, sondern es werden auch die kompletten Simulationsmodelle auf dieser RT-Hardware abgearbeitet. Losgelöst von der GUI-Verwaltung oder anderen Programmen und Tasks wird so die deterministische Ausführung der Simulationsmodelle gewährleistet. Ein normaler Windows-PC oder ein Notebook dient „nur“ noch als Front- und Back-End mit der Benutzerschnittstelle und den Analyse- und Logging-Funktionen.

Eine einfache Art, Frames zu senden, stellt das „Frame Panel“ dar – ein Tool, mit dem der Anwender innerhalb der Oberfläche Daten als Stimulus spezifiziert, die in die Sendepuffer gelegt werden, um zum korrekten Zeitpunkt ein Frame-Update durchzuführen

Letztlich ergibt sich mit dem Tool CANoe und den dazu erhältlichen Hardware-Komponenten eine praxisgerecht skalierbare Lösung, so dass es egal ist, ob der Anwender ein Notebook mit VN3600 Interface via USB nutzt, einen PC mit der PCI-Variante VN3300 oder die RT-Box:

Das gleiche Modell zur Analyse oder Simulation läuft ohne Änderung auf all diesen Plattformen. Werden allerdings mehr I/Os benötigt, ist das RT-Rack optimal, denn in diese Entwicklungsplattform können zusätzliche Karten mit analogen oder digitalen I/Os und weitere Bus-Schnittstellen eingesetzt werden.

Integration von MATLAB/Simulink-Modellen

Nützlich ist auch die direkte Koppelbarkeit an MATLAB/Simulink und dessen Modelle. Dabei stehen zwei Möglichkeiten zur Auswahl. Für eine nicht-echtzeitfähige Simulation kann MATLAB/Simulink als Master fun-

gieren, und CANoe wird als Client-Anwendung ausgeführt.

Der empfohlene Weg ist allerdings, CANoe als Master zu betreiben, damit der Kommunikationsbus als Zeitbasis dient. Mithilfe des Simulink-Real-Time-Workshop-Add-ons wird aus dem in MATLAB beschriebenen Modell des Knotens eine Node-Layer-

DLL erzeugt, die dann in CANoe ablaufen kann. Bei mehreren Knoten muss für jeden von ihnen eine eigene DLL generiert werden. Die Ausführung des Modells wird aus dem Notification Layer heraus gesteuert. Events wie Zyklusstart oder Frame-Empfang dienen dabei als Trigger. Über eine eigene API haben die MATLAB-Modelle automatisch eine Busan- kopplung und direkten Zugriff auf alle Bussignale (Bild 7). Alle Funktionen zum Benachrichtigen und zum Senden und Empfangen von Botschaften und Signalen stehen somit zur Verfügung.

Auch wenn im Automobilumfeld mittelfristig weiterhin nur ein FlexRay-Kanal benötigt und genutzt wird – die hier gezeigten Lösungen unterstützen bereits heute beide Kanäle. ha



Dr. rer. nat. Carsten Böke

studierte Informatik an der Universität Paderborn. Von 1995 bis 2004 war er wissenschaftlicher Mitarbeiter im Heinz-Nixdorf-Institut, Fachgebiet Entwurf Paralleler Systeme. Seit 2004 ist er Senior Software Development Engineer bei der Vector Informatik GmbH und entwickelt dort Werkzeuge für die Bus-Analyse und Bus-Simulation von FlexRay-Systemen. carsten.boeke@vector-informatik.de