

# Das optimale Betriebssystem für FlexRay

Als skalierbares Highspeed-Kommunikationssystem mit deterministischem Verhalten ist FlexRay die passende Lösung als Data-Backbone für verteilte Regelungen oder sicherheitsrelevante Anwendungen im Automobil-Bereich. Im Gegensatz zur ereignisgesteuerten Kommunikation über CAN werden alle Botschaften einem festen Kommunikationsraster zugeordnet. Daraus ergibt sich für jedes beteiligte Steuergerät eine zeitlich eindeutige Verfügbarkeit der Eingangsdaten. Der Entwurf eines FlexRay-Netzwerkes erfordert bereits in einer sehr frühen Entwicklungsphase grundlegende Festlegungen wie Baudrate, Zykluslänge, Anzahl und Länge der Slots im statischen und dynamischen Segment bzw. Macro-Tick-Dauer für alle beteiligten Netzwerkknoten. Diese Planung der Kommunikationsabläufe ist in den kommunikationsspezifischen Software-Komponenten abgebildet, sie kann aber auch Einfluss auf die zeitliche Gestaltung der Anwendungs-Software haben.

Das Betriebssystem (OS) sorgt für das Zusammenspiel aller beteiligten Software-Komponenten. Zur Verfügung stehen ereignisgesteuerte (event triggered) oder zeitgesteuerte (time triggered) Betriebssysteme mit jeweils unterschiedlichen Diensten. Als weitere Option sind Betriebssysteme mit Speicherschutz verfügbar. Bei der Auswahl des optimalen Betriebssystems für FlexRay-basierte Anwendungen stellt sich insbesondere die Frage, ob ein zeitgesteuertes Betriebssystem für die synchronisierte Kommunikation über FlexRay zwingend erforderlich ist.

## ■ Ereignisgesteuerte und zeitgesteuerte Betriebssysteme

Bisher wurden im Automobilbereich meist ereignisgesteuerte Betriebssysteme eingesetzt. Dabei genießt das OSEK/VDX-OS [1], das künftig auch in einer ISO-Norm vorliegen soll, die

## Kriterien zur Betriebssystem-Auswahl für zeitkritische Anwendungen

FlexRay wird je nach Anwendung wegen des deterministischen Verhaltens oder wegen der schnellen Datenübermittlung eingeführt. Die Nutzung für sicherheitsrelevante Anwendungen spielt derzeit nur eine geringe Rolle. Für die Entscheidung, welches Betriebssystem in Verbindung mit FlexRay eingesetzt werden soll, sind neben Deterministik und Performance weitere Kriterien wie Speicherschutz oder Zeitüberwachung zu berücksichtigen. Der Beitrag erläutert, worauf es bei der Wahl des Betriebssystems ankommt, und zeigt konkrete Lösungen, die Vector Informatik auch im Rahmen von AUTOSAR anbietet.

Von Pascale Morizur, Dirk Grossmann und Winfried Janz

breiteste Akzeptanz. Ziel eines Betriebssystems ist es, unter optimaler Nutzung der verwendeten Hardware eine zusätzliche Laufzeitumgebung zur Verwaltung von Funktionseinheiten bereitzustellen. Definierte Betriebssystem-Dienste bieten diese Funktionalität an. Beim Design einer Anwendung werden zunächst zeitlich unabhängige Teilaufgaben definiert. Die daraus resultierenden Tasks oder Interrupts konkurrieren gemäß dem Scheduling-Algorithmus des Betriebssystems um die Laufzeitzuweisung:

- ▶ Tasks werden durch Alarmer oder Ereignisse (Events) ausgelöst. Dabei werden Extended Tasks und Basic Tasks unterschieden. Extended Tasks zeichnen sich durch die Fähigkeit aus, auf Ereignisse warten zu können.

- ▶ Interrupt Services Routines (ISR) sind Hardware-spezifisch und werden durch die Hardware-Peripherie getriggert. Sie haben höhere Priorität als Tasks und sollen deshalb nur für Teilaufgaben reserviert werden, die eine möglichst schnelle Reaktionszeit erfordern.

Bei einem zeitgesteuerten Betriebssystem sind sämtliche Abläufe und Aktionen unter der Kontrolle des Be-

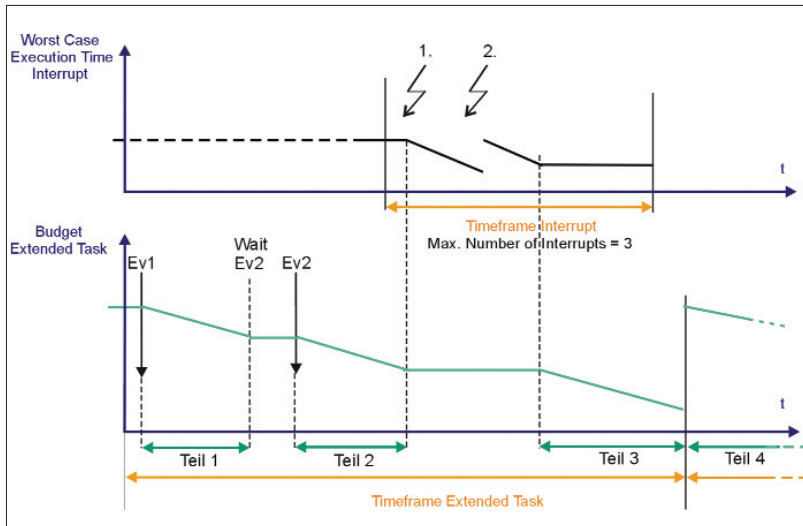
triebssystems ausschließlich abhängig von der Zeit. Für die Applikation ergibt sich damit ein streng deterministisches Zeitverhalten.

## ■ AUTOSAR-OS

AUTOSAR hat das OSEK-OS als Basis genommen und erweitert, so dass auch zeitgesteuerte Funktionalitäten unterstützt werden können. In vier Ausbaustufen, den so genannten Scalability Classes (SC), werden die spezifischen Eigenschaften des AUTOSAR-OS [2] zur Verfügung gestellt. Tabelle 1 stellt einen Überblick ihrer Zuordnung zu den Scalability Classes dar.

Scalability Class	Schedule Tables	Zeitüberwachung	Globalzeit/Synchronisierung	Speicherschutz	Kommentar
SC1	X	-	-	-	Ereignisorientiert
SC2	X	X	X	-	Weiterentwicklung von OSEKtime
SC3	X	-	-	X	Weiterentwicklung von HIS-Protected OS
SC4	X	X	X	X	-

Das AUTOSAR-OS bietet vier Ausbaustufen mit den für FlexRay relevanten Eigenschaften.



**Bild 1.** Wird eine Task nicht innerhalb ihres Überwachungszeitraumes abgeschlossen oder durch mehrere Interrupts unterbrochen, muss nicht zwangsläufig ein Fehler ausgelöst werden.

Es sind nur die Eigenschaften eingetragen, die im Zusammenhang mit einer FlexRay-basierten Anwendung von Bedeutung sind. Die derzeit verfügbaren Spezifikationen von BSW (Basic Software) und RTE (Runtime Environment) decken Speicherschutz noch nicht ab. Dies wird in zukünftigen Versionen der BSW- und RTE-Spezifikationen nachgezogen.

### Schedule Tables

Das Betriebssystem verwaltet mehrere Schedule Tables. Jede Schedule Table besteht aus einer Liste von zeitlich definierten Aktionen, die entweder Tasks aktivieren oder Events an Tasks verschicken.

### Zeitüberwachung

Bei einem Echtzeit-System kommt es darauf an, alle Aufgaben rechtzeitig abzuarbeiten. In der Entwurfsphase werden Teilaufgaben festen Zeitfenstern zugeordnet. Damit zur Laufzeit keine Verzögerung entsteht, darf eine Aufgabe die CPU nicht länger in Anspruch nehmen, als im Design vorgegeben. Deshalb überwacht das Betriebssystem die Laufzeit jeder einzelnen Task bzw. jedes Interrupts. OSEKtime [3] sieht dafür ein klassisches Deadline Monitoring vor: Tasks müssen fertig sein, bevor ihre zugeordnete Deadline erreicht ist. Liegt eine Verletzung vor, löst das gesamte System ein Reset aus. Bei Extended Tasks, die auf Events warten

können, ist diese Art von Überwachung nicht ausreichend. Deshalb hat die AUTOSAR-OS-Arbeitsgruppe eine Laufzeitüberwachung für jede einzelne Task und jeden Interrupt definiert. Sie wird mit unterschiedlichen Parametern im Konfigurator festgelegt.

► Für jede Task wird der Timeframe (Überwachungszeitraum) und das Execution Time Budget (maximale Ausführungszeit pro Überwachungszeitraum) definiert. Soll im Konfigurator des Betriebssystems die Mehrfachaktivierung einer Basic Task zugelassen sein, beinhaltet das Execution Budget die gesamte Laufzeit aller Aktivierungen der betroffenen Task. Bei Extended Tasks sind grundsätzlich keine Mehrfachaktivierungen möglich.

► Für jeden Interrupt sind der Überwachungszeitraum, die „Worst Case Exe-

cutation Time per Execution“ und die maximale Anzahl von Interrupts pro Überwachungszeitraum zu definieren. Je Timeframe können mehrere Interrupts zugelassen werden, die Worst Case Execution Time bezieht sich aber nur auf einen einzigen Durchlauf.

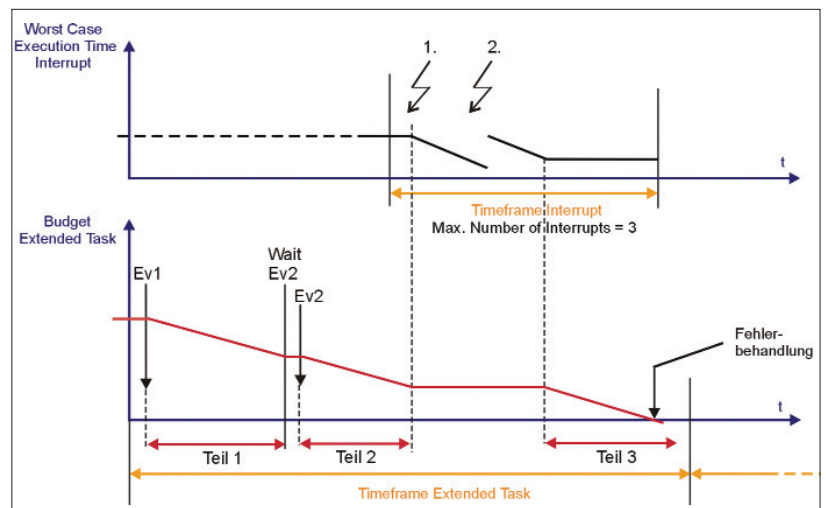
Bei Verletzung eines Überwachungsparameters leitet das Betriebssystem je nach Konfiguration entweder ein „Reset\_OS“, „Kill\_Application“, „Kill\_Task“ oder „Kill\_Interrupt“ als Fehlerreaktion ein.

**Bild 1** zeigt die Überwachung einer Anwendung, bestehend aus einer Extended Task, die von einem Interrupt mehrmals unterbrochen wird. Wegen einer längeren Laufzeit der Extended Task, z.B. für die Behandlung des Events 1 (Ev1) kann es zur Fehlerbehandlung kommen (**Bild 2**). Diese betrifft ausschließlich den Verursacher. Die Zeitüberwachung garantiert je nach Budget Tasks und Interrupts mit niedriger Priorität den Zugang zum Prozessor für die Abwicklung ihrer Aufgabe.

Sollte wie in **Bild 3** ein Interrupt zu lange dauern, wird dieser abgebrochen damit die Extended Task weiterlaufen kann. Die Bilder 1 bis 3 stellen ein AUTOSAR-OS der Scalability Class 2 mit Zeitüberwachungs-Tasks und Interrupts dar.

### Synchronisierung über Globalzeit

Verteilte Regelungen, die eine Synchronisierung von Tasks über Steuer-



**Bild 2.** Die längere Laufzeit einzelner Tasks kann zur Fehlerbehandlung führen.

geräte-Grenzen hinweg erfordern, benötigen die Unterstützung einer Globalzeit, die regelmäßig dem Betriebssystem zur Verfügung gestellt wird. Die Synchronisierung der Applikation auf diese Globalzeit kann entweder schrittweise erfolgen („smooth“) oder durch ein einmaliges, vollständiges Anpassen („hard“).

**Speicherschutz**

Mechanismen zum Speicherschutz sind erforderlich, um die Störung durch andere Module frühzeitig zu erkennen und zu unterbinden. Für den Fall, dass in einem Steuergerät Software-Module verschiedener Hersteller integriert werden, lassen sich damit zur Laufzeit die unterschiedlichen Software-Pakete bezüglich ihrer Speicherbereiche trennen. Dazu ist ein Prozessor mit Hardware-Unterstützung notwendig. Er sollte zudem über einen großen Speicher und hohe Verarbeitungsgeschwindigkeit verfügen, da die Speicherschutz-Mechanismen die Betriebssystem-

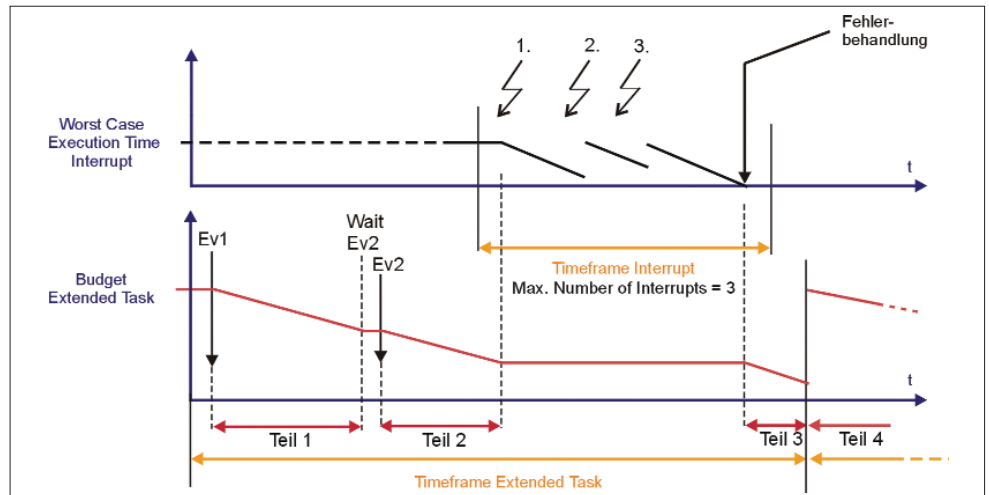


Bild 3. Interrupts, die zu viel Zeit in Anspruch nehmen, werden zu Gunsten der Extended Task abgebrochen.

tem-Dienste durchschnittlich um den Faktor 1,5 bis 2 verlangsamen.

**Datenaustausch zwischen FlexRay und Anwendungs-Software**

Hinsichtlich der Serienimplementierung sind Kosten und Nutzen von Sche-

dule Tables, Zeitüberwachung, Synchronisierung und Speicherschutz abzuwägen. Dabei muss berücksichtigt werden, dass das gewählte OS die Synchronisation des Datenaustauschs zwischen FlexRay und Applikation optimal unterstützt.

Beim FlexRay-Protokoll sind bis zu 64 Grundzyklen in einer sich stän-

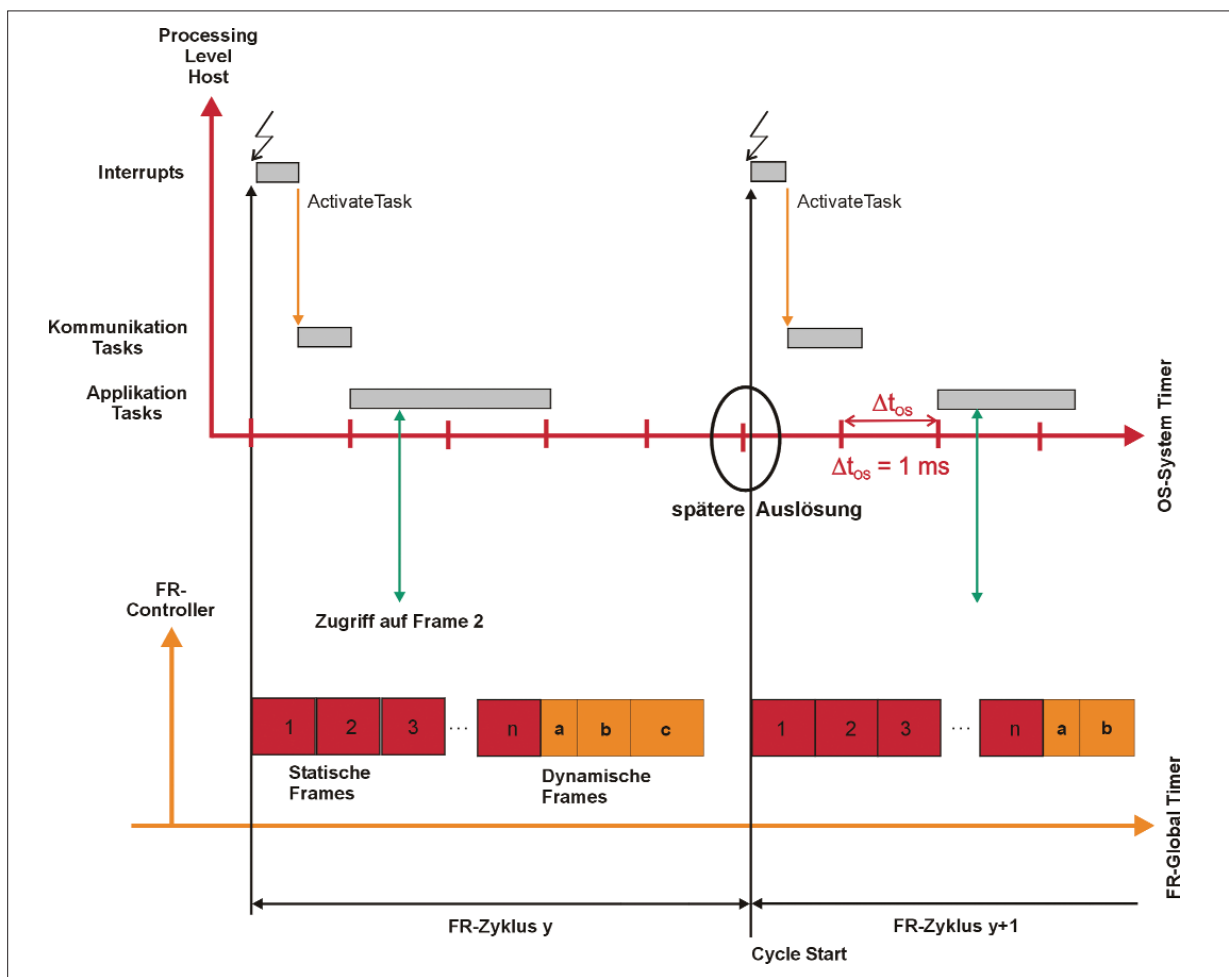


Bild 4. Beim Datenaustausch zwischen der FlexRay-Hardware und der Software-Applikation mit einem OSEK-OS führt eine mangelhafte Synchronisierung zu einer verspäteten Abarbeitung.

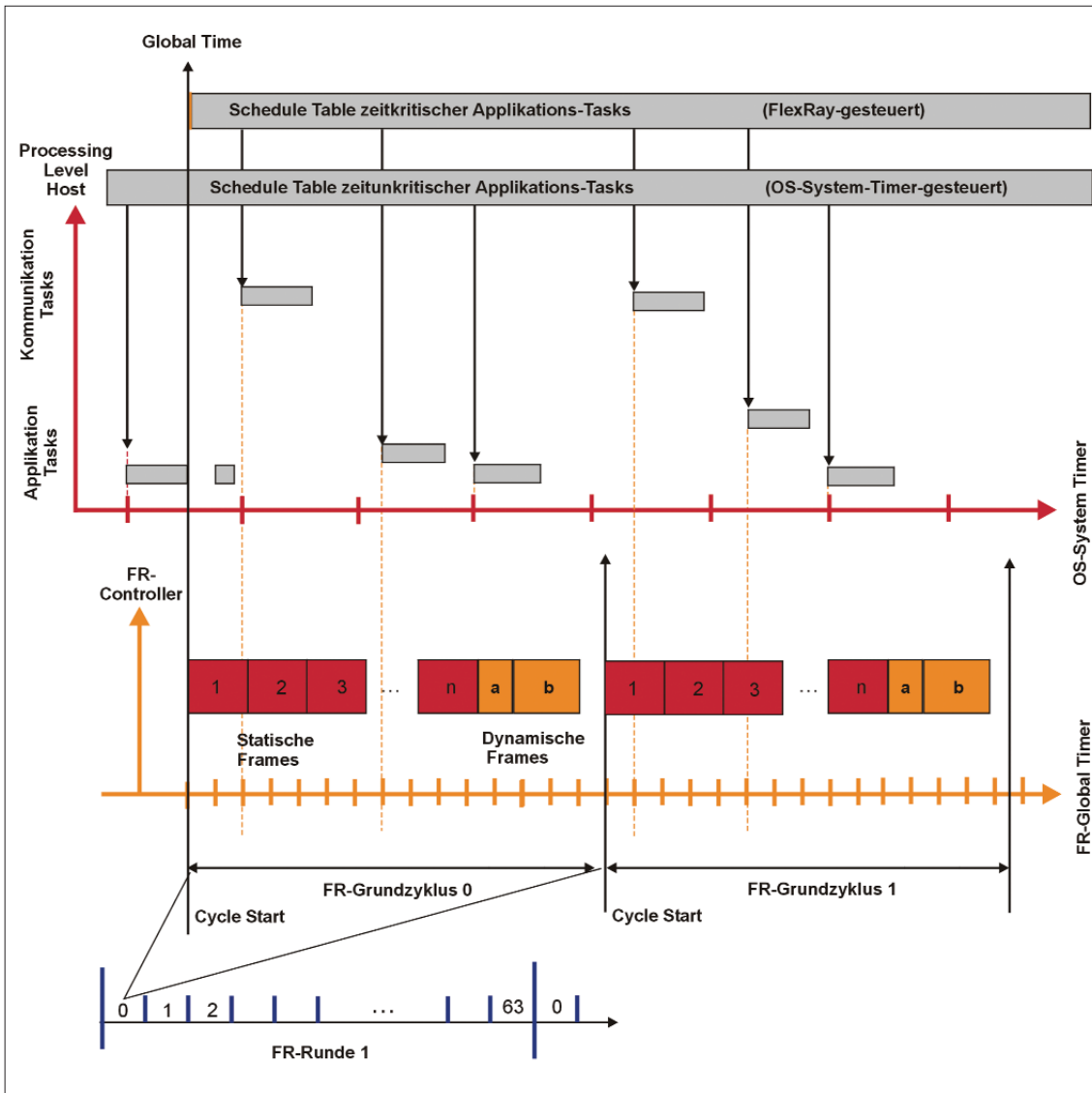


Bild 5. Die Einführung einer globalen Zeitbasis und der Schedule Tables führt zu einer stabilen Verbindung zwischen FlexRay-Hardware und AUTOSAR-OS (SC1).

das TDMA-Verfahren (Time Division Multiple Access) ist allen beteiligten Steuergeräten der genaue Sendebzw. Empfangszeitpunkt jeder Botschaft im statischen Bereich des FlexRay-Zyklus bekannt. Diese beiden Eigenschaften minimieren die Unschärfe bezüglich der Signallaufzeit einer FlexRay-basierten Anwendung.

Für die Synchronisierung des Datenaustauschs bei einer FlexRay-basierten Anwendung lassen sich je nach Bedarf und verfügbaren Betriebssystem-Eigenschaften mehrere Lösungsansätze realisieren. Die Kommunikations-Tasks sollten grundsätzlich mit dem FlexRay-Timer des CC getriggert werden. Die Aktivierung der Applikations-Tasks könnte hingegen durch einen beliebigen Timer erfolgen. Hieraus ergeben sich vier mögliche Lösungsansätze:

dig wiederholenden Runde zusammengefasst.

Jeder Grundzyklus enthält Frames mit verschiedenen Prozessdaten beziehungsweise Signale für unterschiedliche Applikations-Tasks. Das FlexRay-Protokoll liefert mit einer Globalzeit die Grundlage für den synchronisierten Datenaustausch. Sobald sich der Datenverbund synchronisiert hat, steht diese jedem Communication Controller (CC) des Verbunds zur Verfügung.

Bei einer verteilten Regelung wird eine Funktion über mehrere Steuergeräte im Netzwerkverbund verteilt. Die Tot-

zeit, verursacht durch die Signallaufzeit von Senderapplikation zu Empfängerapplikation, kann dabei einen entscheidenden Einfluss auf die Regelgüte haben. Grundsätzlich wirkt sich eine geringe Totzeit günstig aus.

Bei ereignisgesteuerten Kommunikationssystemen wird der Datenaustausch zwischen CC und Host grundsätzlich per Interrupt gesteuert; beim Zugriff auf den Bus entstehen gegebenenfalls Wartezeiten. Die Signallaufzeit lässt sich nicht genau vorhersagen, es wird daher typischerweise eine Worst-Case-Abschätzung durchgeführt. Erst durch die Bereitstellung einer globalen Zeit des FlexRay-Protokolls wird es dem Betriebssystem ermöglicht, Dienste zur Synchronisation auf diese Zeit anzubieten. Durch

Alarmer, die am OS-System-Timer hängen, triggern die Applikations-Tasks. Am Anfang jedes Grundzyklus wird der „FlexRay-Cycle Start Interrupt“ durch den FlexRay-Controller ausgelöst. Die Kommunikations-Task wird sofort aktiviert. Die Kette von Applikations-Tasks wird unabhängig davon vom Betriebssystem-Timer angestoßen. Bild 4 stellt vereinfacht mit nur einer Applikations- und einer Kommunikations-Task diesen Lösungsansatz dar. Zu beachten ist, dass eine spätere Auslösung eines Cycle Start Interrupts, zum Beispiel durch eine Drift zwischen CC- und Host-Quarz verursacht, eine verspätete Abarbeitung der übertragenen FlexRay-Daten (z.B. aus Frame 2) von  $\Delta t_{OS} = 1 \text{ ms}$  als Konsequenz hat. Erwartet

die Applikation keine längere Reaktionszeit, ist dafür ein OSEK-OS ausreichend.

► Erfordert die Regelung eine kürzere Antwortzeit (weniger als 1 ms), dann ist für die Auslösung der Applikations-Tasks ein High Resolution Timer (Auflösung ca. 10  $\mu$ s) beim OSEK-OS notwendig. Dafür wird allerdings ein geeigneter Timer benötigt.

► Falls das Steuergerät nicht über einen High Resolution Timer verfügt, kann der FlexRay-Timer zusätzlich für die Aktivierung der zeitkritischen Applikations-Tasks benutzt werden (Bild 5). FlexRay-unabhängige Tasks können weiterhin an den OS-System-Timer gehängt werden. Beim Aufstartverhalten ist zu beachten, dass der FlexRay-Timer und damit die durch ihn aktivierten Tasks erst zur Verfügung stehen, nachdem der FlexRay-Controller sich erstmalig auf das Netzwerk synchronisiert hat. Sollte anschließend die Synchronisierung verloren gehen, kann der FlexRay-Timer auf Basis seiner lokalen Zeit weiterhin arbeiten, falls der FlexRay-Controller entsprechend konfiguriert wurde. Ein AUTOSAR-OS der Scalability Class 1 mit Schedule Tables ist dazu ausreichend. Diese Lösung erlaubt eine über mehrere Steuergeräte verteilte Regelung, da die FlexRay-Globalzeit die Synchronisierung über alle Steuergeräte sicherstellt.

► Falls in obigen Fällen zusätzlich die Einhaltung der Task- und Interrupt-Laufzeiten überwacht werden soll, ist ein AUTOSAR-OS der Scalability Class 2 oder 4 erforderlich. Damit werden Laufzeitüberschreitungen verhindert; ein zeitlich deterministisches Verhalten wird erreicht.

Eine FlexRay-basierte Anwendung erfordert nicht zwingend ein zeitgesteuertes Betriebssystem. Die Festlegung des geeigneten Betriebssystems sollte unter Berücksichtigung der Anwendung und Architektur individuell für jedes Steuergerät erfolgen. Dazu ist eine Analyse in Bezug auf Steuergeräte-übergreifende Synchronität, Sicherheitsanforderungen, Antwortzeit und Zeitüberwachung notwendig. Für alle FlexRay-Anwendungen bietet Vector Informatik dem Entwickler ein entsprechendes Betriebssystem: das nach dem OSEK/VDX-Standard zertifizierte osCAN mit oder ohne High Resolution Timer oder osCAN AUTOSAR, das die Scalability Classes SC1-SC4 gemäß AUTOSAR-OS V2.0 abdeckt. Die Vector-FlexRay-Softwarekomponenten arbeiten mit jeder dieser OS-Varianten zusammen. Der osCAN TimingAnalyzer analysiert hierbei die Einplanbarkeit von Tasks mit einer Worst-Case-Betrachtung.

Für die durchgängige Entwicklung von FlexRay-Systemen bis zum Se-

rieneinsatz unterstützt Vector den Anwender mit Softwarekomponenten und individueller Dienstleistung. Ausgereifte und aufeinander abgestimmte Tools wie der DaVinci Network Designer für alle FlexRay-typischen Entwurfsaufgaben oder CANoe.FlexRay 5.2 für Simulation und Stimulation eines Netzwerkes, Integrationstests und Restbussimulation sowie Analyse des fertigen FlexRay-Netzwerks erleichtern die Entwicklung. Der Zugriff auf alle internen Parameter des FlexRay-Steuergerätes über das standardisierte Mess- und Kalibrierprotokoll „XCP on FlexRay“ erfolgt mit CANape 6.0. Für die erste, schnelle und flexible Implementierung eines FlexRay-Netzwerks sorgt das FlexRay Evaluation Bundle. Diese integrierte Umgebung aus Softwarekomponenten und Werkzeugen beinhaltet auch eine Beispielanwendung für ein FlexRay-System mit zwei Knoten. *sj*

#### Literatur

- [1] OSEK/VDX Operating System, Version 2.2.2. [www.osek-vdx.org](http://www.osek-vdx.org)
- [2] AUTOSAR – Specification of Module Operating System V2.0. [www.autosar.org](http://www.autosar.org)
- [3] OSEKtime – OSEK/VDX Time-Triggered Operating System, Version 1.0. [www.osek-vdx.org](http://www.osek-vdx.org)



**Dipl.-Ing. Pascale Morizur**

studierte Physik-Elektronik an der Grande Ecole ICPI in Lyon (F). Nach ihrem Abschluss 1986 arbeitete sie zehn Jahre bei MAN-Nutzfahrzeuge in der Vorentwicklung im Bereich CAN, J1939 und Diagnose. Sie ist bei Vector als Product Management Engineer im Bereich FlexRay-Embedded-Softwarekomponenten tätig.

[pascale.morizur@vector-informatik.de](mailto:pascale.morizur@vector-informatik.de)



**Dipl.-Ing. Winfried Janz**

studierte an der Universität Stuttgart Elektrotechnik. Nach vier Jahren in der Software-Entwicklung für Embedded Controller in der Regelungs- und Automatisierungstechnik kam er 1995 zu Vector. Hier ist er seit 1997 als Teamleiter und Produktmanager für die Entwicklung von OSEK-Echtzeit-Betriebssystemen zuständig. Er hat in den Arbeitskreisen OSEK und AUTOSAR die Spezifikationen Betriebssystem und Konfiguration mitgestaltet.

[winfried.janz@vector-informatik.de](mailto:winfried.janz@vector-informatik.de)



**Dipl.-Ing. Dirk Grossmann**

studierte Elektrotechnik an der Universität Stuttgart. Nach seinem Abschluss 1997 arbeitete er zunächst in der Betriebssystem-Entwicklung der ETAS GmbH, bevor er für zwei Jahre als Produktmanager Nordamerika den Bereich Betriebssystem verantwortete. Seit 2003 ist er bei Vector als Teamleiter für die Entwicklung der FlexRay-Embedded-Softwarekomponenten verantwortlich.

[dirk.grossmann@vector-informatik.de](mailto:dirk.grossmann@vector-informatik.de)