Serielle Bussysteme im Automobil

Alles über CAN, LIN, FlexRay und MOST

vector

Sehr geehrter Leser,

das waren noch Zeiten, als es im Auto noch keine Elektronik gab und die Autofahrer bei der Führung ihrer Fahrzeuge komplett auf sich allein gestellt waren. Heute regelt beispielsweise das Motormanagement abhängig vom Betriebspunkt des Motors den für Kraftstoffverbrauch und Abgasverhalten des Motors optimalen Zündwinkel. Zudem sorgen eine Vielzahl von elektronischen Systemen für ein Mehr an Komfort und Sicherheit beim Autofahren.

Das wohl bekannteste aktive Sicherheitssystem im Kfz ist das elektronische Stabilitätsprogramm ESP, welches mittels einer Vielzahl von Sensoren bis zu 150 mal pro Sekunde Informationen über die Rotationsgeschwindigkeit der Räder, Quer- und Längsbeschleunigung und die Bewegungen des Lenkrads erhält. Sobald das Auto unter- oder übersteuert, leitet das ESP Korrekturen ein, wozu es gezielt Räder abbremst und blitzschnell über den Datenbus die Motorleistung drosselt.

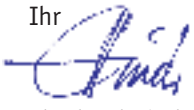
Das ESP ist nur ein Beispiel für die zunehmende informationelle Kopplung von elektronischen Systemen im Kfz. Aufgrund der steigenden Zahl von steuergeräteübergreifenden Funktionen und der damit einhergehenden immer intensiveren Datenkommunikation ergeben sich für die Fahrzeughersteller und -zulieferer neue Herausforderungen. Schon heute, und in Zukunft noch viel mehr, stellt die Beherrschung der wachsenden Systemkomplexität einen wesentlichen wettbewerbsentscheidenden Faktor dar.

Von Anfang an unterstützt Vector Hersteller und Zulieferer der Automobilindustrie bei der Entwicklung von elektronischen Steuergeräten und der Vernetzung dieser Steuergeräte mit CAN, LIN, FlexRay und MOST durch Werkzeuge für Entwurf, Simulation, Analyse, Test, Kalibrierung und Diagnose sowie durch Softwarekomponenten, Entwicklungsdienstleistungen und Schulungen. Die Zusammenarbeit von Vector mit Ausbildungsstätten und Hochschulen gibt Schülern, Auszubildenden und Studenten die Gelegenheit, am Know-how von Vector zu partizipieren.

An diese Tradition möchte Vector mit dieser Sonderausgabe zum Thema „Serielle Bussysteme im Automobil“ anknüpfen. Sie setzt sich aus fünf ausgewählten Beiträgen zusammen und richtet sich an alle, die sich mit der Entwicklung von elektronischen Steuergeräten und der Vernetzung elektronischer Systeme im Kfz beschäftigen. Nach einer Einführung in die serielle Datenkommunikation lernen Sie die im Moment gängigen seriellen Bussysteme im Kfz kennen, also CAN, LIN, FlexRay und MOST.

Ich hoffe, Sie können mit dieser Sonderausgabe aufschlussreiche Erkenntnisse gewinnen und freue mich schon jetzt auf Ihr Feedback.

Ihr



Eberhard Hinderer
Geschäftsführer Vector Informatik GmbH

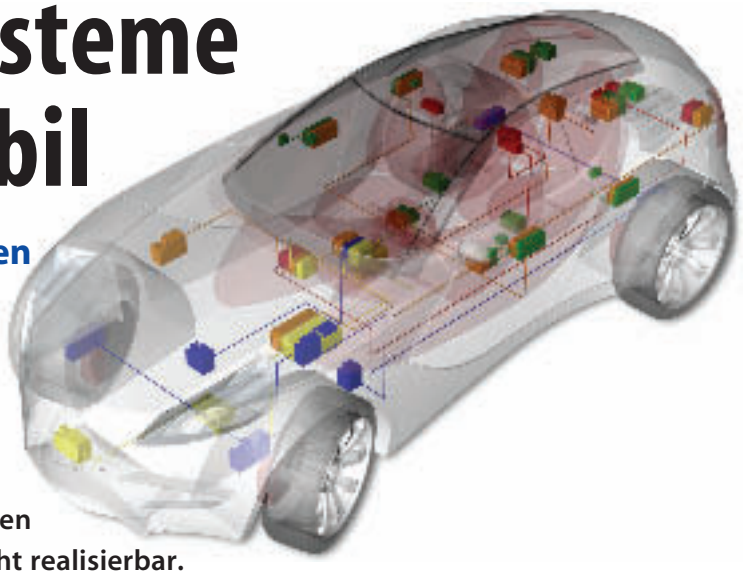
>> Inhalt

Elektronik automotive
Fachmagazin für Entwicklungen in der Kfz-Elektronik und Telematik

Sonderdruck

Serielle Bussysteme im Automobil

Teil 1: Architektur, Aufgaben und Vorteile



Viele Funktionen im Automobil wären ohne Datenaustausch zwischen elektronischen Komponenten gar nicht realisierbar. Bevor in den nächsten Folgen dieser Artikelreihe auf die speziellen Charakteristika der einzelnen Bussysteme eingegangen wird, erklärt dieser Beitrag die technischen Grundlagen der seriellen Bussysteme in modernen Kraftfahrzeugen und vergleicht die verschiedenen Konzepte, die dabei Anwendung finden.

Von Eugen Mayer

Die jüngere Geschichte des Automobils ist durch eine intensive Elektronifizierung gekennzeichnet. Die treibende Kraft dafür geht in der Hauptsache von den immer anspruchsvolleren Wünschen der Kunden an ein modernes Automobil aus. Des Weiteren werden vom Gesetzgeber immer strengere Vorgaben zur Abgasemission gemacht. Aber auch die Globalisierung sorgt durch gestiegenen Wettbewerbs- und Kostendruck für stetigen Innovationsdruck. Mit der Elektronik haben die Kfz-Hersteller einen Weg gefunden, dieser multiplen Herausforderung zu begegnen. Dies spiegelt sich vor allem in der Ende der 1970er Jahre beginnenden Migration elektronischer Steuergeräte in das Automobil wider.

Die ersten eingebetteten elektronischen Systeme verrichteten damals ihre Aufgaben noch völlig autonom. Schon sehr früh erkannte man, dass durch die Koordination von Anwendungen, die auf unterschiedlichen elektronischen Steuergeräten untergebracht waren, die Fahrzeugfunktionalität immens erhöht werden konnte. Dies war der Auslöser für die Integration von Kommunikationssystemen in das Automobil.

Allen voran beherrschte damals die elektronische Fahrdynamikregelung die Vorentwicklung. Der intensive Verkabelungsaufwand ließ jedoch nur einen eingeschränkten Datenaustausch auf Basis von Einzelleitungen zu. Als Ausweg aus diesem Dilemma kam der bitserielle Austausch von Daten über einen einzigen Kommunikationskanal in Frage. Dieser integriert alle individuellen Kommunikationskanäle und wird als Bus bezeichnet. Mittels dieses Busses und entsprechender serieller Schnittstellen können alle elektronischen Steuergeräte zu einem Systemverbund, dem seriellen Bussystem, zusammengeschlossen werden (**Bild 1**). Elektronische Steuergeräte werden in diesem Kontext als Busknoten (Nodes) bezeichnet.

Seit dem Einsatz serieller Bussysteme gehören die komplexen und oft unterschiedlich gearteten Kabelbäume

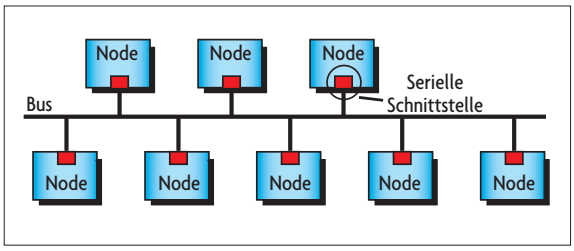


Bild 1. Alle elektronischen Steuergeräte (Busknoten oder „Nodes“) werden mittels eines Busses und entsprechender serieller Schnittstellen zu einem Systemverbund, dem seriellen Bussystem, zusammengeschlossen.

im Automobil der Vergangenheit an. Die Bussysteme vereinfachen nicht nur die Projektierung und Installation, sondern senken auch das Gewicht und den Platzbedarf für die Verkabelung. Darüber hinaus reduziert die geringere Zahl an Steckverbindungen die Störanfälligkeit deutlich. Die vielen Vorteile erkauft man jedoch durch zahlreiche Kommunikationsaufgaben, die das serielle Bussystem bewältigen muss. Im Folgenden werden die wichtigsten Kommunikationsaufgaben erläutert.

Kommunikationsaufgaben

Voraussetzung für einen reibungslosen seriellen Datenaustausch ist die eindeutige Zuordnung der zu versendenden Daten zu den Busknoten. Grundsätzlich unterscheidet man zwischen senderselektiver und empfängerselektiver Zuordnung (Adressierung). Bei der senderselektiven Adressierung bestimmt der Sender den gewünschten Empfänger über eine eindeutige Busknotenadresse. Im Gegensatz dazu werden bei der empfängerselektiven Adressierung nicht die Busknoten, sondern die zu versenden Daten adressiert. Dadurch stehen alle Daten prinzipiell jedem Busknoten zum Empfang zur Verfügung (Broadcast). Sämtliche Busknoten haben daher die Aufgabe, die für sie relevanten Daten herauszufiltern. Dies geschieht mit Hilfe der Adresse, die man hier als Identifier bezeichnet.

Damit der Empfänger die Daten und die Adresse als Einheit auffasst, packt der Sender beides zu einem Frame zusammen. Ein typischer Frame umrahmt die Adresse und die Daten mit einer Anfangs- und Endkennung, die vor allem zur Herstellung der Synchronität zwischen Sender und Empfänger dienen. Statt „Frame“ spricht man auch von „Rahmen“, „Nachricht“ oder „Botschaft“.

Zu den vordringlichsten Aufgaben eines seriellen Bussystems gehören die echtzeitfähige Datenübertragung und die Datensicherung. Ein verteiltes System wird nur dann seiner Bestimmung gerecht, wenn sämtliche Daten rechtzeitig und fehlerfrei die jeweilige Anwendung auf den Zielknoten erreichen. Die Leistungsfähigkeit und das Einsatzgebiet eines seriellen Bussystems im Automobil hängen entscheidend davon ab, in welchem Ausmaß es Störungen vermeidet, abwehrt, erkennt und korrigiert sowie die rechtzeitige Datenübertragung garantieren kann.

Datensicherheit durch Fehlererkennung und -korrektur

Quantitativ lässt sich die Datensicherheit durch die Restfehlerwahrscheinlichkeit beschreiben. Diese ist ein statistisches Maß für die Verletzung der Datensicherheit. Unter Restfehlerwahrscheinlichkeit versteht man das Produkt aus der Wahrscheinlichkeit A, mit der die zu übertragenden Daten verfälscht sind, und aus der Wahrscheinlichkeit B, mit der verfälschte Daten unerkannt bleiben. Die Datensicherheit eines seriellen Bussystems hängt also einerseits davon ab, wie weitreichend es Datenverfälschungen vermeidet, und andererseits, in welchem Ausmaß es verfälschte Daten detektiert.

Als Ursachen für Datenverfälschungen kommen im Automobil die verschiedensten Wechselwirkungen durch galvanische, kapazitive oder induktive Kopplungen bzw. elektromagnetische Felder in Frage. Verantwortlich sind dafür im Einzelnen z.B. Verstell- und Lüftermotoren, hochfrequente Signale durch den Kommutierungsprozess in Gleichstrommotoren und durch schnelle Datenübertragungen oder Reflexionen an den Bus-Enden. Je besser es gelingt, diese Ursachen zu eliminieren, desto störfester und sicherer ist die Datenübertragung.

Um die Störfestigkeit eines seriellen Bussystems zu erhöhen, sind einige wichtige Maßnahmen notwendig. Neben der Abschirmung des Übertragungsmediums sowie sämtlicher elektrischer und elektronischer Komponenten ist für einen ausreichend großen Abstand zwischen Daten- und Energieübertragungsleitungen sowie zwischen elektrischen und elektronischen Komponenten zu sorgen. Weiterhin gilt es, die Datenübertragungsfrequenz sowie die Anzahl und die Steilheit der Datensignalflanken zu begrenzen, das Prinzip der Differenzsignalübertragung zu nutzen und schließlich die beiden Bus-Enden mit dem Wellenwiderstand des Übertragungsmediums zu terminieren.

Selbst bei optimaler physikalischer Systemauslegung lassen sich Übertragungsfehler nicht vollständig ausschließen. Deshalb sind Fehlererkennungsmechanismen unerlässlich. Zu

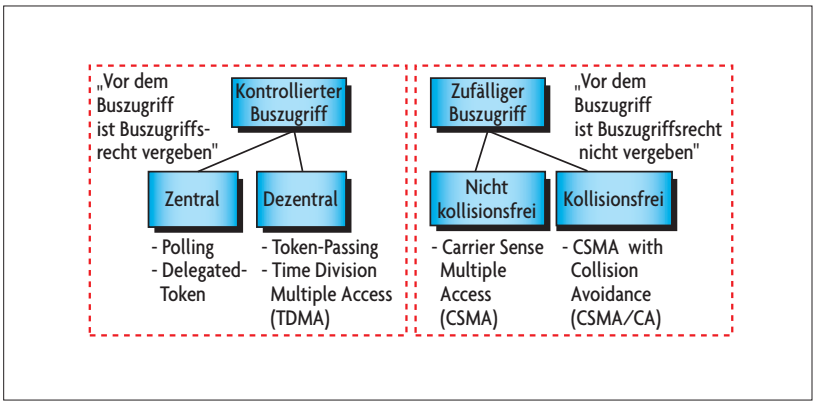


Bild 2. Je nach Anwendungsbereich wird auf einen zufälligen oder kontrollierten Buszugriff gesetzt.

den am häufigsten eingesetzten Methoden zählt das Prüfsummenverfahren. Dabei berechnet der Sender über einen definierten Algorithmus aus dem zu versendenden Datenblock eine Prüfsumme, die er im Anschluss an den Datenblock überträgt. Mit Hilfe dieser Prüfsumme ist der Empfänger in der Lage, den empfangenen Datenblock zu verifizieren.

Je ausgeklügelter der Algorithmus, je kürzer der zu sichernde Datenblock und je länger die Prüfsumme, desto besser die Fehlererkennungsfähigkeit. Allerdings ist wegen der beschränkten Bandbreite und der zeitlichen Anforderungen ein Kompromiss zwischen der Fehlererkennungsfähigkeit und dem Verhältnis zwischen Datenblock und Prüfsumme (Übertragungseffizienz) zu schließen. Zudem muss berücksichtigt werden, dass auch eine Prüfsumme während der Übertragung nicht immun gegen Störungen ist.

Wann immer das System einen Übertragungsfehler detektiert, ist eine Fehlerkorrektur notwendig, beispielsweise mit einer fehlerkorrigierenden Prüfsumme. Dies erfordert allerdings gegenüber der bloßen Fehlererkennung eine deutlich längere Prüfsumme. Aus Effizienzgründen setzt man im Automobil keine fehlerkorrigierenden Prüfsummen ein. Die Fehlerkorrektur erfolgt vielmehr durch die Botschaftswiederholung: entweder hervorgerufen durch eine vom fehlererkennenden Busknoten übertragene Fehlersignalisierung oder automatisch im Falle einer zyklischen Botschaftsübertragung.

Echtzeit-Fähigkeit für sicherheitskritische Anwendungen

Ein echtzeitfähiges System muss die Übertragung sämtlicher auszutauschenden Daten zwischen den verschiedenen Busknoten innerhalb eines definierten Zeitfensters garantieren können. Die wesentlichen Einflussfaktoren sind Anzahl und Größe der Botschaften, die zur Verfügung stehende Bandbreite und insbesondere die Art des Buszugriffs. Bei Letzterem unterscheidet man grundsätzlich zwischen einem kontrollierten und einem zufälligen Buszugriff (**Bild 2**).

Bei seriellen Bussystemen mit kontrolliertem Buszugriff ist das Buszugriffsrecht bereits vor dem Buszugriff eindeutig festgelegt. Solche Systeme bieten deterministischen Botschaftsverkehr als wichtige Voraussetzung für die Verwirklichung echtzeitfähiger serieller Bussysteme. Da der gesamte Kommunikationsablauf planmäßig abläuft und nicht beeinflussbar ist, zeichnen sich serielle Bussysteme mit kontrolliertem Buszugriff allerdings durch ein schlechtes dynamisches Verhalten aus.

Diesen Nachteil kennen serielle Bussysteme mit unkontrolliertem Buszugriff nicht. Jeder Busknoten hat zu jeder Zeit das Recht, den Bus zu belegen, z.B. aufgrund eines aktuellen Ereignisses. Daraus resultiert einerseits ein sehr schneller Buszugriff; andererseits birgt dies aber in Abhängigkeit von der Ereignisdichte, den Botschaftsgrößen und der zur Verfügung stehenden Datenrate eine mehr oder weniger akute Kollisionsgefahr, was keine guten Voraussetzungen für eine echtzeitfähige Datenübertragung darstellt.

Die Überwachung des Busses durch sendewillige Busknoten reduziert die Kollisionsgefahr erheblich. Ganz verhindert werden kann sie durch das zusätzliche Einführen von Botschaftsprioritäten. Allerdings können auch diese auf Bus-Monitoring und Botschaftsprioritäten basierenden zufälligen Buszugriffsverfahren keine Rechtzeitigkeit garantieren. Denn durch die Priorisierung besteht die Gefahr, dass niederpriore Botschaften unverhältnismäßig lange verzögert werden.

Architektur serieller Bussysteme

In Anlehnung an das von der ISO (International Standardization Organization) spezifizierte Referenzmodell der Datenkommunikation wird die serielle Schnittstelle eines Busknotens im Automobil typischerweise in zwei (Kommunikations-)Schichten unterteilt: untere Schicht (Physical Layer) und darüber liegende Schicht (Data Link Layer).

Der Data Link Layer übernimmt die Aufgaben von Adressierung, Framing, Buszugriff, Synchronisation sowie Fehlererkennung und -korrektur.

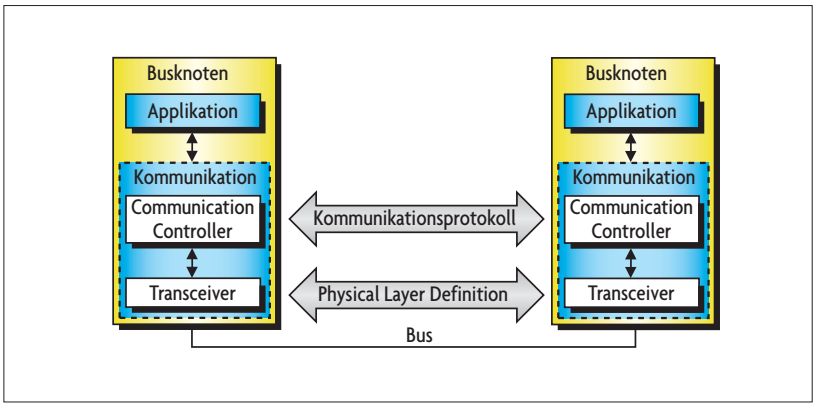


Bild 3. Vereinfachte Architektur serieller Bussysteme.

Definiert werden diese Aufgaben durch ein Kommunikationsprotokoll. Die Physical-Layer-Spezifikation umfasst dagegen alle Aspekte des Physical Layers, angefangen von der physikalischen Busankopplung bis hin zur physikalischen Signalübertragung mittels Bus.

In der Regel realisiert man die physikalische Busankopplung mit Hilfe eines Transceivers und die Ankopplung des Data Link Layer über einen Kommunikationscontroller. Wenn sich alle Busknoten innerhalb des Systems an dasselbe Kommunikationsprotokoll und dieselbe Physical-Layer-Spezifikation halten, sind die grundlegenden Voraussetzungen für einen reibungslosen Datenaustausch zwischen den Busknoten gegeben.

Bei der seriellen Kommunikation übergibt die Applikation des Senders dem Kommunikationscontroller den zu versendenden Datenblock. Dieser ergänzt den Datenblock um die Adresse und um Prüf- und Synchronisa-

tionsinformationen, so dass ein Frame entsteht. Der Transceiver überträgt nun den Frame über den Bus. Als physikalische Verbindungsstruktur kommt im Automobil hauptsächlich die wegen der passiven Busankopplung sehr einfach handhabbare Linientopologie zum Einsatz. Auf der Empfängerseite nimmt wieder der Transceiver den Frame entgegen und übergibt ihn dem

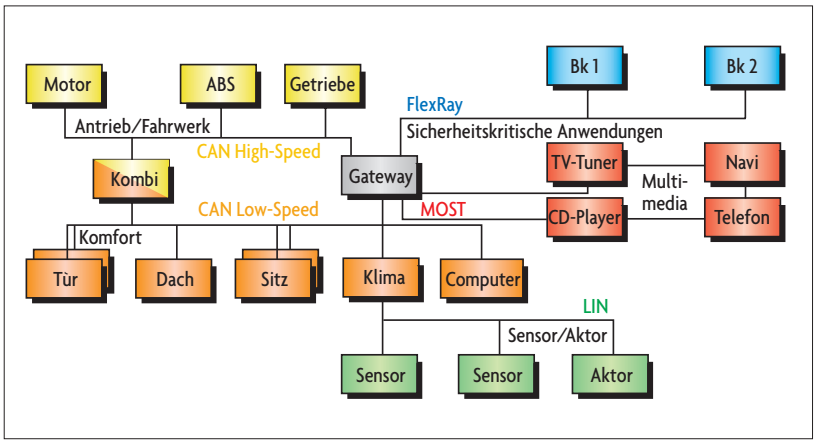


Bild 4. Beispiel zur Vernetzung von elektronischen Steuergeräten mit seriellen Bussystemen im Automobil.

Kommunikationscontroller, welcher die an ihn übertragenen Informationen auswertet und im Falle des korrekten Datenempfangs den Datenblock an die Applikation weitergibt.

Das Ergebnis ist ein hierarchischer und somit transparenter Kommunikationsfluss. Dieser wird durch das Erledigen der den Schichten zugeordneten Kommunikationsaufgaben sowie durch das Kommunikationsprotokoll und die Definition des Physical Layers garantiert (**Bild 3**).

Für manche Aufgaben, wie beispielsweise das Busmanagement (u.a. Sleep- und Wake-Up-Funktion) oder die Diagnose und Konfiguration von Busknoten reicht die vom Data Link Layer zur Verfügung gestellte Kommunikationsfunktionalität nicht aus. Durch die Definition höherer Schichten bzw. höherer Kommunikationsprotokolle kann die Kommunikationsfunktionalität erweitert werden.

Transparenz durch herstellerübergreifende Bustechnologien

Der verschärfte Wettbewerb sorgt für immer mehr Sicherheits- und Komfortfunktionen im Automobil. Die Folge ist nicht nur ein permanenter Anstieg der Anzahl an elektronischen Komponenten in den Fahrzeugen, sondern auch ein deutlich höherer Vernetzungsgrad mit rasant steigendem Datenaufkommen, da die meisten neuen Automobil-Funktionen ohne Datenaustausch nicht mehr auskommen. Damit für die Kfz-Hersteller die zunehmende Komplexität der Fahrzeugelektronik beherrschbar bleibt, schaffen sie auf den System-, Funktions- und Kommunikationsebenen verschiedene Standards. Auf der System- bzw. Funktionsebene soll in Zukunft „AUTOSAR“ (AUTomotive Open System ARchitecture) für die nötige Transparenz sorgen. Herstellerübergreifende Kommunikationsstandards wie die seriellen Bussysteme CAN [1], LIN [2], MOST [3] und FlexRay [4] sorgen für mehr Transparenz auf der Kommunikationsebene.

CAN (Controller Area Network) wird hauptsächlich in den Bereichen Antrieb, Fahrwerk und Komfort eingesetzt. LIN (Local Interconnect Network) dient zur einfachen und kostengünstigen Datenübertragung im Sensor/Aktor-Bereich. MOST (Media Oriented System Transport) setzt man im Infotainment zur Übertragung von Video- und Audiosignalen ein. Und FlexRay ermöglicht schließlich anspruchsvollste Kommunikation in sicherheitskritischen verteilten Anwendungen. **Bild 4** zeigt ein Beispiel zur Vernetzung von elektronischen Steuergeräten mit seriellen Bussystemen im modernen Automobil. Im Gegensatz zu CAN, LIN und MOST muss sich FlexRay im Automobil aber erst noch etablieren. Im Herbst dieses Jahres geht die erste FlexRay-Serienanwendung auf die Straße. Der Münchner Autobauer BMW wird im neuen X5 erstmals das innovative Bussystem in einer aktiven Dämpferkontrolle einsetzen.

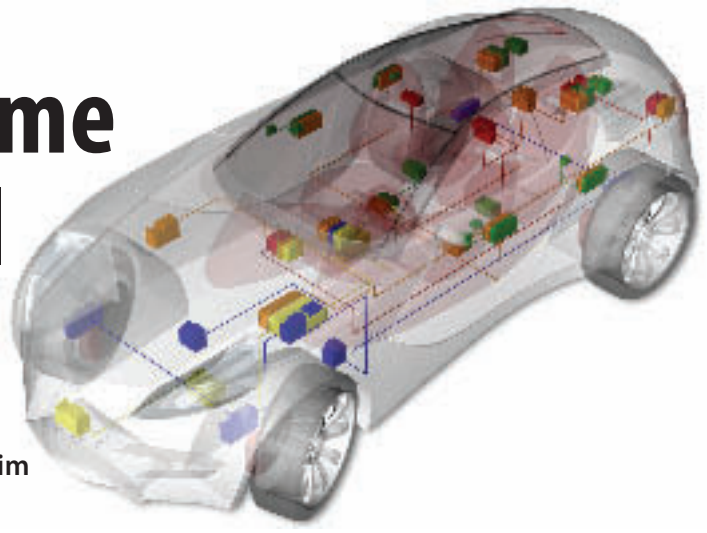
CAN wurde Anfang der 1980er Jahre von der Robert Bosch GmbH entwickelt und 1994 international standardisiert (ISO 11898). Drei Geschäftsführer der Vector Infomatik waren an der Entwicklung maßgeblich beteiligt. LIN, MOST und FlexRay gingen aus den herstellerübergreifenden Organisationen LIN-Konsortium, MOST-Kooperation und FlexRay-Group hervor. Sie sind zwar nicht offiziell standardisiert, können aber als De-facto-Standards aufgefasst werden.

Vector Informatik [5] unterstützt Fahrzeughersteller und -zulieferer bei der CAN-, LIN-, FlexRay- und MOST-Vernetzung mit einer durchgängigen Werkzeugkette aus Design- und Entwicklungstools sowie mit Softwarekomponenten und Basissoftware für AUTOSAR-Steuergeräte. Beratung, Consulting Services und Werkzeuge für das Prozessmanagement ergänzen die Anwendungsgebiete. Abgerundet werden die Leistungen durch ein umfangreiches Schulungsangebot, wie beispielsweise das eintägige Seminar „Serielle Bussysteme im Automobil“ oder Grundlagenseminare zu CAN, LIN, FlexRay und MOST. Ergänzende und vertiefende Informationen zu seriellen Bussystemen bietet das Unternehmen in Form von elektronischen Informationen im Bereich „Training“ an. *sj*

Links

[1] www.can.bosch.com
[2] www.lin-subbus.com
[3] www.mostcooperation.com
[4] www.flexray.com
[5] www.vector-informatik.de

Serielle Bussysteme im Automobil



Teil 2: Sicherer Datenaustausch mit CAN

Die immer komplexer werdenden elektronischen Systeme sorgen für ein höheres Maß an Sicherheit und Komfort beim Autofahren. Durch seine spezifischen Merkmale – so wird etwa der sichere Datenaustausch auch unter widrigen Umgebungsbedingungen sichergestellt – leistet dabei das serielle Bussystem CAN (Controller Area Network) einen entscheidenden Beitrag.

Von Eugen Mayer

Die von Bosch [1] entwickelte CAN-Technik ist seit 1993 genormt und liegt, gegliedert in mehrere Teile, als ISO-Norm 11898 vor (**Bild 1**). Der erste Teil umfasst das CAN-Protokoll und deckt vollständig den Data Link Layer (Framing, Adressierung, Buszugriff, Datensicherung) und teilweise den Physical Layer (Physical Signaling) des standardisierten Referenzmodells der Datenkommunikation (ISO 7498) ab. Das CAN-Protokoll wird in Hardware implementiert, wofür mittlerweile eine Vielzahl von kostengünstigen CAN-Controllern zur Verfügung steht.

Der zweite Teil beschreibt den CAN-High-Speed-Physical-Layer, der dritte Teil den CAN-Low-Speed-Physical-Layer. Beide decken den Physical Layer der ISO 7498 ab (unter anderem die physikalische Busankopplung, Datenraten und die Spannungspegel). Der CAN-High-Speed-Physical-Layer kommt vor allem in Antriebs- und Fahrwerksapplikationen zum Einsatz. Man realisiert ihn im Wesentlichen durch den CAN-High-Speed-Transceiver, der eine maximale Datenrate von 1 Mbit/s unterstützt. Für den hauptsächlich im Komfortbereich eingesetzten CAN-Low-Speed-Physical-Layer nutzt man in der Regel den CAN-Low-Speed-Transceiver mit einer maximalen Datenrate von 125 kbit/s.

Die CAN-Schnittstelle (**Bild 2**) besteht demnach aus einem CAN-Controller und einem CAN-Transceiver. Während der CAN-Controller das CAN-Protokoll abwickelt, übernimmt der CAN-Transceiver die Aufgabe, den CAN-Controller physikalisch an den im Differenzsignalmodus betriebenen CAN-Bus zu koppeln. Die Differenzsignalübertragung verbessert die Störempfindlichkeit und erfordert zwei Kommunikationsleitungen (CAN-High- und CAN-Low-Leitung), die zur Vermeidung von Reflexionen an den Enden mit dem Wellenwiderstand abgeschlossen werden.

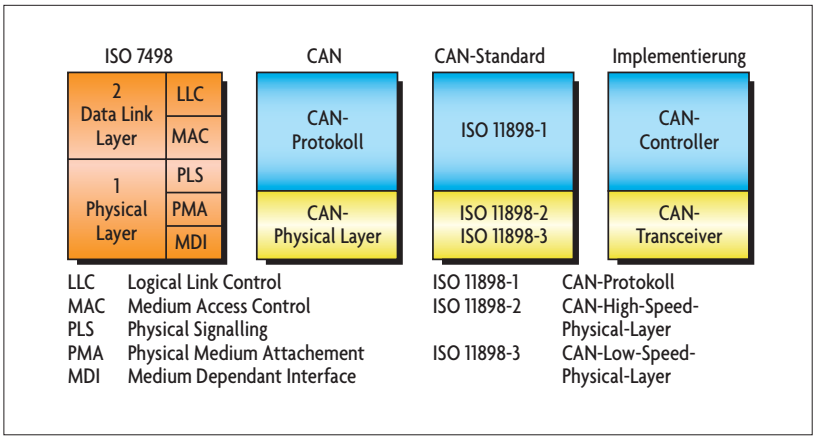


Bild 1. Die CAN-Technik ist seit 1993 genormt und liegt als ISO-Norm 11898 vor, die den Data Link Layer und teilweise den Physical Layer des standardisierten Referenzmodells der Datenkommunikation (ISO 7498) abdeckt.

Zuordnung von Knoten und Nachrichten über Nachrichtenadressen und -filter

Die Nachrichtenadressen, üblicherweise als Identifier (ID) bezeichnet, bestimmen nicht die CAN-Zielknoten, sondern die Identität der Nachrichten selbst. Prinzipiell stehen so alle CAN-Nachrichten allen CAN-Knoten zum Empfang zur Verfügung (Nachrichtenverteilung). Über einen Filter selektiert jeder CAN-Knoten die für ihn relevanten CAN-Nachrichten aus dem Nachrichtenstrom (empfängerselektives System). Aufgrund des 11-bit-breiten Identifiers lassen sich in einem CAN-Netzwerk bis zu 2048 CAN-Nachrichten spezifizieren.

Diese Form der Nachrichtenverteilung bietet folgende Vorteile:

- Kosteneinsparung durch Mehrfachausnutzung von Sensoren.

- Einfache Realisierung und Synchronisation von verteilten Prozessen.
- Hohe Flexibilität hinsichtlich der Konfiguration.

Der Verzicht auf Knotenadressen erlaubt die Integration von weiteren Busknoten, ohne dass die Hard- oder Software vorhandener Busknoten modifiziert werden muss. Dies gilt allerdings nur, wenn es sich beim hinzukommenden Busknoten ausschließlich um einen Empfänger handelt.

Datenübertragung erfolgt ereignisgesteuert

Die übertragenen Nachrichten und deren Reihenfolge sind in einem CAN-Netzwerk nicht vom Fortschreiten der Zeit abhängig, sondern vom Auftreten spezieller Ereignisse. Jeder CAN-Knoten ist prinzipiell berechtigt, sofort nach Auftreten eines Ereignisses auf den CAN-Bus zuzugreifen. In Verbindung mit der vergleichsweise kurzen Nachrichtenlänge von maximal 130 bit im Standard-Format und der hohen Datenübertragungsrate bis zu 1 Mbit/s ermöglicht das Verfahren schnelle Reaktionen auf asynchrone Vorgänge. Dies ist eine wichtige Voraussetzung für eine echtzeitfähige Datenübertragung im Millisekundenbereich (1 bis 10 ms), die vor allem die Applikationen des Antriebs und des Fahrwerks verlangen.

Da der CAN-Kommunikation kein Zeitplan zugrunde liegt, ergibt sich der Nachrichtenverkehr stets erst zur Laufzeit und birgt deshalb implizit die Gefahr von Kollisionen. Diese steigt mit zunehmender Buslast und stellt die Echtzeit-Fähigkeit in Frage. Um trotz zufälligen Buszugriffs Echtzeit-Datenübertragung zu gewährleisten, kommt im CAN-Netzwerk das CSMA/CA-Buszugriffsverfahren (Carrier Sense Multiple Access/Collision Avoidance) zum Einsatz.

CSMA/CA-Zugriffsverfahren sorgt für zerstörungsfreien, prioritätsgesteuerten Buszugriff

Der Buszugriff beginnt damit, dass ein sendewilliger CAN-Knoten zunächst den CAN-Bus abhört (Carrier Sense). Ist der CAN-Bus frei, darf der CAN-Knoten sofort mit der Nachrichtenübertragung beginnen. Stellt er hingegen Busaktivitäten fest, muss er seinen Sendewunsch so lange zurückstellen, bis der CAN-Bus frei und die gerade laufende Nachrichtenübertragung abgeschlossen ist. Zudem hat er eine Pause von drei Bitzeiten (ITM – Intermission) abzuwarten. Eine laufende Nachrichtenübertragung wird nicht unterbrochen, der Buszugriff ist damit zerstörungsfrei. Im Falle mehrerer sendewilliger CAN-Knoten verhindert die bitweise Arbitrierung (**Bild 3**) trotz simultanen Buszugriffs (Multiple Access – MA) das Auftreten von Kollisionen. Im Rahmen der bitweisen Arbitrierung legen alle sendewilligen CAN-Knoten den ID der zu übertragenden CAN-Nachrichten bitweise vom höchst- zum niederwertigsten Bit an. Die dem CAN-Netzwerk zugrunde liegende Wired-AND-Buslogik (0 = dominant) sorgt dafür, dass sich immer ein eindeutiger Buspegel einstellt. Nach dem Aufschalten eines ID-Bits vergleicht jeder CAN-Knoten den Buspegel mit dem aufgeschalteten Pegel. Die Arbitrierungslogik entscheidet, ob ein CAN-Knoten weiter senden darf oder das Senden einstellen muss.

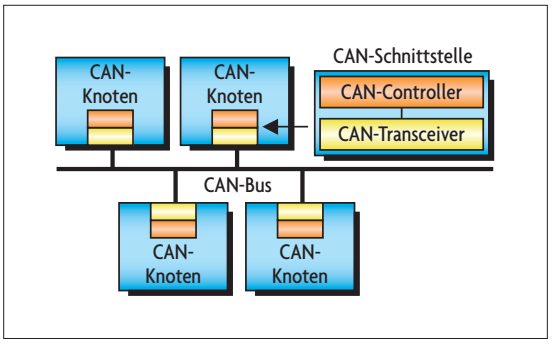


Bild 2. Die CAN-Schnittstelle besteht aus dem eigentlichen Controller zur Abwicklung des CAN-Protokolls und dem Transceiver zur physikalischen Ankopplung an das differenzielle CAN-Netzwerk.

Am Ende der Arbitrierungsphase bekommt jener CAN-Knoten die Sendeberechtigung, der die CAN-Nachricht mit dem niederwertigsten Identifier überträgt. Unterlegene CAN-Knoten wechseln zunächst in den Empfangszustand und greifen für einen erneuten Sendeversuch auf den CAN-Bus zu, sobald dieser wieder frei ist. So verhindert die Bus- und Arbitrierungslogik nicht nur Kollisionen (Collision Avoidance), sondern sorgt auch für einen prioritätengesteuerten Buszugriff: Je niederwertiger ein Identifier ist, desto höher ist die Priorität der CAN-Nachricht und damit ein schnellerer Buszugriff möglich. Die CAN-Nachricht mit dem kleinsten Identifier (ID = 0) wird deshalb ohne Verzögerung übertragen.

Ist die Buslast nicht zu hoch, sorgt diese Art des zufälligen, zerstörungsfreien und prioritätengesteuerten Buszugriffs für einen gerechten und sehr schnellen Buszugriff. Allerdings muss einerseits berücksichtigt werden, dass mit zunehmender Buslast die Verzögerungen vor allem von niederprioren CAN-Nachrichten anwachsen. Das kann so weit führen, dass CAN-Nachrichten im schlimmsten Fall zu spät bei Empfängern ankommen oder völlig unterdrückt werden. Andererseits verursacht das CSMA/ CA-Buszugriffsverfahren einen reziproken Zusammenhang zwischen Netzwerkausdehnung und maximaler Datenrate. Während der bitweisen Arbitrierung muss ein rezessiv sendender CAN-Knoten einen dominanten Pegel sicher erkennen. Die Größe des Bit-Zeitintervalls ist deswegen so zu wählen, dass die Signallaufzeiten auf dem CAN-Bus vollständig kompensiert werden. Steigende Netzausdehnung bedingt somit ein verlängertes Bit-Zeitintervall, woraus schließlich eine maximal nutzbare Datenrate resultiert.

Datenübertragung erfolgt mittels Data-Frames

Neben den für die Datenübertragung hauptsächlich eingesetzten Data-Frames (**Bild 4**) gibt es auch Remote-Frames zur Anforderung von Daten. Sie kommen aber kaum zur Anwendung, da die Datenübertragung im Automobil nicht auf Nachfrage, sondern im Wesentlichen auf Informationserzeugerinitiative basiert. Beide Frame-

Typen sind identisch aufgebaut. Allerdings entfällt beim Remote-Frame das Data-Field.

Grundvoraussetzung für die Übertragung von Data- und Remote-Frames ist ein Gleichlauf zwischen Sender und Empfänger. Da aus Kosten- und Aufwandsgründen auf eine Taktleitung verzichtet wird, realisiert man den Gleichlauf mittels Signalflanken und einem definierten Re-Synchronisationsmechnismus. Jede Nachrichtenübertragung beginnt mit dem Übertragen des dominanten Synchronisationsbits (SOF, Start of Frame) und erzeugt so die erste Signalflanke (Bus-Idle entspricht einem rezessiven Buspegel). Der Empfänger stellt während der gesamten Übertragung durch Auswertung jeder ankommenden Signalflanke die Synchronisation sicher und passt notfalls sein eigenes Bit-Timing entsprechend an. Das Bit-Stuffing-Verfahren sorgt dafür, dass spätestens nach fünf homogenen Bits ein komplementäres Bit und somit eine Signalflanke erscheint.

Im Anschluss an das SOF folgt der ID, der entweder 11 bit (Standard-ID) oder 29 bit (Extended-ID) lang sein kann. Im Automobil dominiert das Standardformat. Das Extended-Format spielt typischerweise im Zusammenhang mit höheren Protokollen wie SAE J1939 eine Rolle. Mittels IDE-Bit (Identifier Extension) wird das ID-Format angezeigt. Ein anderer Bitschalter (RTR-Bit, Remote Transmission Request) zeigt an, ob es sich um ein Data- oder ein Remote-Frame handelt.

Zur Übertragung von Nutzinformationen steht das 64 bit breite Data-Field zur Verfügung, bei dem die genaue Anzahl der Nutzbytes mittels DLC (Data Length Code) angegeben wird. Dem Data-Field folgt die CRC-Sequence (Cyclic Redundancy Check). Anhand aller zu übertragenden Bits, eines Generatorpolynoms und eines definierten Algorithmus generiert der Sender die CRC-Sequence. Unabhängig von der Nachrichtenfilterung geschieht dasselbe empfängerseitig mit den ankommenden Bits. Es folgen der Vergleich der beiden Sequenzen und die Quittierung nach dem rezessiven CRC-Delimiter im Acknowledge-Slot (ACK-Slot). Am Ende eines Data-Fames steht nach dem rezessiven ACK-Delimiter das 7 bit lange und rezessive EOF (End of Frame).

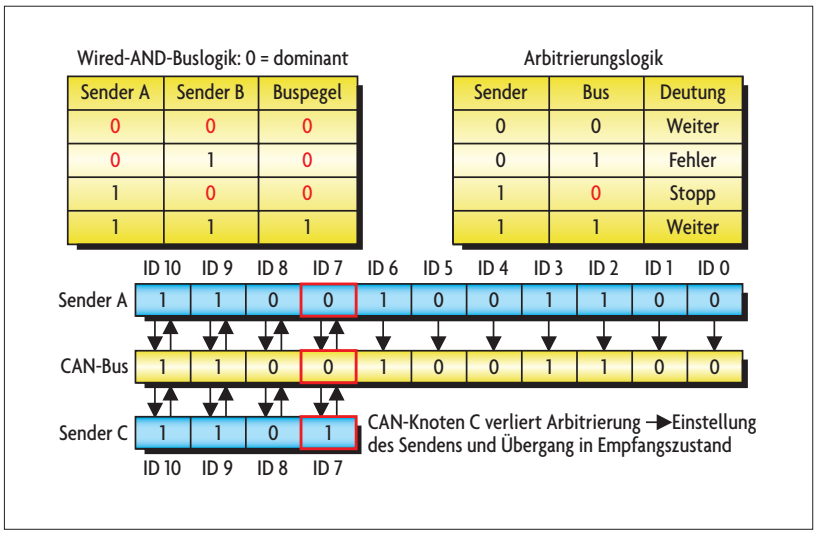


Bild 3. Die Wired-AND-Buslogik sorgt, hier am Beispiel von zwei CAN-Knoten, für einen eindeutigen Buspegel. Die Arbitrierungslogik entscheidet, ob Busteilnehmer senden dürfen.

Fehlererkennungsmechanismen sorgen für hohe Datensicherheit

Die Wahrscheinlichkeit, dass verfälschte CAN-Nachrichten unerkannt bleiben, ist außerordentlich gering. Sie liegt bei $4{,}7 \times 10^{-11}$ [2]. Dafür verantwortlich sind die im CAN-Protokoll definierten Fehlererkennungsmechanismen. Dazu gehören auf der Empfängerseite neben dem von der Nachrichtenfilterung unabhängigen CRC, mit dem sich bis zu fünf Fehler innerhalb einer CAN-Nachricht detektieren lassen, die Überprüfung des Formats (Form Check) und die Bit-Stuffing-Regel (Stuff Check). Der Sender führt ein Bit-Monitoring durch und wertet den ACK-Slot aus.

Legt man einem CAN-Netzwerk eine Fehlerrate von 10^{-3} zugrunde, dann treten bei einer jährlichen Betriebsdauer von 1000 Stunden, einer Datenrate von 500 kbit/s, einer mittleren Buslast von 25 % und einer mittleren Nachrichtenlänge von 80 bit statistisch knapp alle 4000 Jahre vom CAN-Protokoll unerkannte verfälschte CAN-Nachrichten auf. Unter der Fehlerrate versteht man das Verhältnis gestörter CAN-Nachrichten zur Anzahl aller übertragenen CAN-Nachrichten.

Sobald ein Fehlererkennungsmechanismus einen Übertragungsfehler signalisiert, bricht der CAN-Knoten, der den Fehler erkannt hat, die Nachrichtenübertragung ab, indem er ein Error-Flag (sechs dominante Bits) auf den CAN-Bus auflegt. Das Error-Flag verletzt bewusst die Bit-Stuffing-Regel, so dass netzwerkweit jeder CAN-Knoten den bis dahin lokalen Fehler wahrnimmt und infolgedessen die Nachrichtenübertragung mit dem Aufschalten eines Error-Flags abbricht.

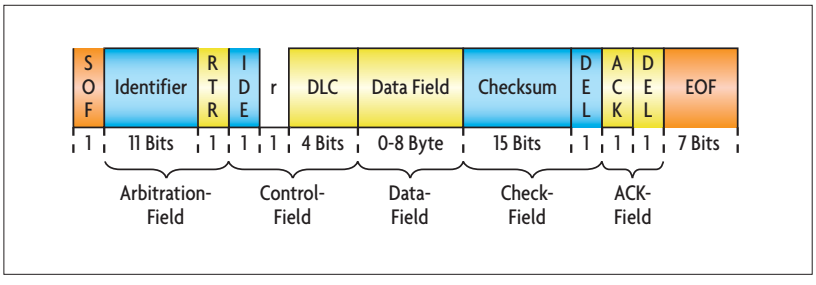


Bild 4. Frames enthalten einen 11-bit- (Standard) oder 29-bit-Identifier (Extended). Bei Remote-Frames ist, im Gegensatz zu Data-Frames, die Länge des Data-Field gleich Null.

Diese Methode stellt die für die Datenintegrität verteilter Anwendungen so wichtige netzweite Datenkonsistenz sicher.

Die Fehlerkorrektur besteht darin, die abgebrochene CAN-Nachricht durch denselben Sender zu wiederholen, sobald der CAN-Bus wieder frei ist (nach dem Error-Delimiter und ITM). Bei der Auslegung des Systems muss man berücksichtigen, dass es aufgrund des CSMA/CA-Buszugriffverfahrens keine Garantie für das sofortige Wiederholen gibt. Die Fehlererholzeit hängt von der Nachrichtenpriorität und der Buslast ab.

Die Fehlersignalisierung via Error-Flag versetzt jeden CAN-Knoten in die Lage, laufende Nachrichtenübertragungen abzubrechen. Da dies auch für defekte CAN-Knoten gilt, sind solche in der Lage, die gesamte CAN-Kommunikation zum Erliegen zu bringen. Um dies zu verhindern, verfügt jeder CAN-Knoten über eine Netzknotenüberwachung (**Bild 5**), die mittels Fehlerzähler und Regeln zur Steuerung der Fehlerzähler den defekten Knoten abschalten kann (Bus Off).

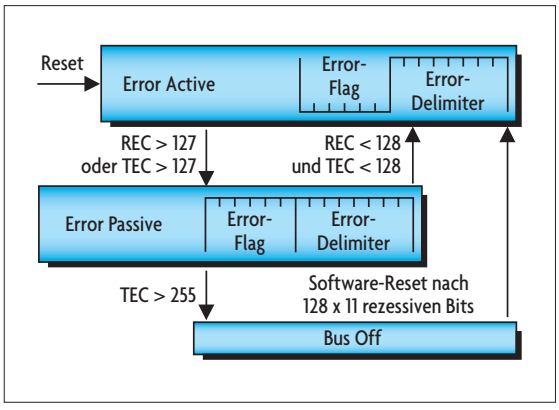


Bild 5. Jeder CAN-Knoten verfügt über eine Netzknotenüberwachung, die mittels Fehlerzähler defekte Knoten erkennen und abschalten kann.

Quittierung empfangener CAN-Nachrichten

In einem CAN-Netzwerk wird unabhängig von der Nachrichtenfilterung jede Nachrichtenübertragung von allen Empfängern gleichzeitig im ACK-Slot quittiert (In-Frame-Acknowledgement). Ein dominanter Pegel entspricht einer positiven, ein rezessiver Pegel einer negativen Quittierung. Da der Sender den ACK-Slot rezessiv belegt, reicht eine positive Quittierung aus, um die Korrektheit der Nachrichtenübertragung zu bestätigen. Aufgrund dieses knotenneutralen positiven Acknowledgement werden negativ quittierende CAN-Knoten überschrieben und bleiben ungehört. Deshalb senden diese nach dem ACK-Delimiter ein Error-Flag.

Liegt keine einzige positive Quittierung vor, wird also der ACK-Slot von keinem Empfänger überschrieben, detektiert der Sender einen ACK-Fehler und bricht die laufende Nachrichtenübertragung mit dem Aufschalten eines Error-Flags ab.

Forderung nach Echtzeit-Fähigkeit und hoher Datenübertragungsrate überlastet CAN

Noch bis vor einigen Jahren war CAN die am meisten gefragte Bustechnik im Automobil. Mit voranschreitender Elektronifizierung stößt CAN zunehmend an seine Grenzen. Speziell bei echtzeit-kritischen und höchst sicherheits-kritischen Kfz-Anwendungen, die zudem eine hohe Datenübertragungsrate erfordern wie beispielsweise das Fahrerassistenzsystem „Lane Keeping Assistance“, kommen zunehmend neue Techniken zum Einsatz, aber auch in kostensensitiven Komfortanwendungen stellen die Fahrzeugentwickler den CAN-Bus in Frage.

Deswegen haben sich im Laufe der Zeit neben CAN zwei andere Busse für den Einsatz im Automobil etabliert oder sind auf dem besten Wege dorthin: LIN und FlexRay. LIN (Local Interconnected Network) wird bereits zur kostengünstigen Vernetzung von Sensoren und Aktoren im Komfortbereich eingesetzt. FlexRay ist aufgrund der zeitgesteuerten Kommunikationsmethode, einer Datenrate von bis zu 20 Mbit/s und der Möglichkeit, auf zwei Kommunikationskanälen zu senden, auf dem Vormarsch. Zum weltweit ersten Serieneinsatz kommt FlexRay im neuen BMW X5 in einem aktiven Dämpferkontrollsystem.

Zuverlässige Steuergerätevernetzung

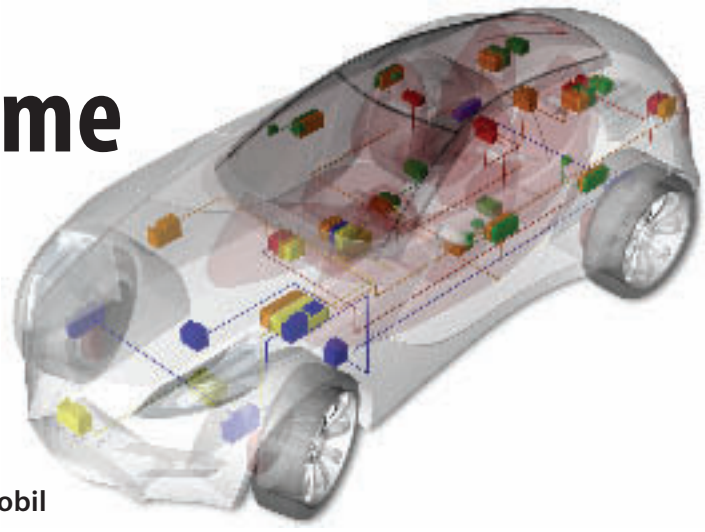
Die Spezialisten der Vector Informatik [3] unterstützen Fahrzeughersteller und -zulieferer nicht nur bei der CAN-Vernetzung, sondern auch bei den Bussystemen LIN, FlexRay und MOST. Für Kundenprojekte sind durchgängige Werkzeugketten aus Design- und Entwicklungstools wahlweise mit Softwarekomponenten und Basissoftware für AUTOSAR-Steuergeräte verfügbar. Für den Einstieg in die Steuergerätevernetzung bzw. den Datenaustausch im Automobil bietet das Stuttgarter Unternehmen das eintägige Seminar „Serielle Bussysteme im Automobil“ an. Grundlagenseminare zu CAN, LIN, FlexRay und MOST vermitteln das notwendige Basiswissen, um sich schnell mit den vielfältigen Entwicklungstätigkeiten rund um die Automobilelektronik vertraut zu machen [4].

Der erste Teil dieser Beitragsreihe [5] befasste sich allgemein mit seriellen Bussystemen im Automobil. In den Folgen drei bis fünf werden die seriellen Bussysteme LIN, FlexRay und MOST behandelt. Der interessierte Leser findet zu den bereits veröffentlichten Themen auf der Internetseite der VectorAcademy [4] ergänzende und vertiefende Informationen. *sj*

Literatur und Links

[1] www.can.bosch.com
[2] *Unruh, J.; Mathony, H. J.; Kaiser, K.H:* Error Detection Analysis of Automotive Communication Protocols. SAE International Congress 1990.
[3] www.vector-informatik.de
[4] www.vector-academy.de
[5] *Mayer, E.* Datenkommunikation im Automobil. Teil 1: Architektur, Aufgaben und Vorteile serieller Bussysteme. *Elektronik Automotive* 2006, H. 7, S. 70 bis 73.

Serielle Bussysteme im Automobil



Teil 3: Einfacher und kostengünstiger Datenaustausch mit LIN

Die LIN-Bustechnik hat sich in kürzester Zeit für den einfachen und kostengünstigen Datenaustausch im Automobil etabliert. Viele Kfz-Hersteller setzen heute zur Übertragung von unkritischen Signalen im Komfortbereich auf LIN. Der folgende Beitrag zeigt die Gründe für den Siegeszug von LIN (Local Interconnect Network) im Automobil auf und erläutert die dahinter stehende Technik.

Von Eugen Mayer

Die steigenden Ansprüche der Autofahrer an den Fahrkomfort führten zu einer breiten Elektronifizierung in diesem Bereich der Fahrzeugtechnik. Dies spiegelt sich in der Migration vieler elektronischer Komponenten in den Komfortbereich wider. Lange Zeit war es üblich, die immer größere Zahl von Sensoren, Aktoren und Motoren direkt mit einem zentralen Steuergerät zu verbinden.

Diese Entwicklung stieß allmählich an ihre Grenzen: Hatte sie doch einen rasanten Anstieg des Verkabelungsaufwands zur Folge, begleitet von größerem Platzbedarf, zunehmendem Gewicht und einer deutlich höheren Störanfälligkeit. Zudem erforderte die zunehmende Individualisierung viele unterschiedliche Kabelbaum- und Steckervarianten, was wiederum die Produktion, Installation und Wartung erheblich erschwerte.

Die Entwickler erkannten schnell, dass auch in dieser Kfz-Applikationsdomäne eine Vernetzung der Komponenten über ein Bussystem die ideale Lösung darstellen würde. Da jedoch der CAN-Bus für den kostensensiblen Sensor-/Aktor-Bereich nicht in Frage kam, begannen bereits Mitte der 1990er Jahre viele Kfz-Hersteller und -zulieferer mit der Entwicklung eigener Sensor-/Aktor-Bussysteme.

Nach und nach entstanden zahlreiche kostengünstige und einfache, aber proprietäre Sensor-/Aktor-Busse. Im Jahr 2000 kam mit LIN ein weiteres serielles Bussystem für den Sensor-/Aktor-Bereich auf den „Vernetzungsmarkt“. Diese Technologie setzte sich auf breiter Front durch, so dass man LIN inzwischen in fast jedem Fahrzeug findet, typischerweise in den Komfort-Anwendungen wie Klima-, Sitz-, Tür- und Spiegelsteuerung.

Konsortium verhilft LIN zum Durchbruch

Ein wesentlicher Grund für die schnelle Etablierung von LIN war die Gründung des LIN-Konsortiums [1], in Rahmen dessen sich namhafte Kfz-Hersteller und -Zulieferer sowie Halbleiter- und Tool-Hersteller zusammengeschlossen hatten, um einen herstellerübergreifenden Kommunikationsstandard für den Sensor-/ Aktor-Bereich zu schaffen. Mit der Definition eines einfachen und kostengünstigen Physical Layer, der sich am ISO-Standard 9141 orientiert, sowie eines einfachen und schlanken Kommunikationsprotokolls legte das LIN-Konsortium das Fundament für den Erfolg. Denn dadurch wurden die Voraussetzungen für die Realisierung einfacher und kostengünstiger Busknoten geschaffen.

Da sich das LIN-Konsortium nicht nur auf die eigentliche LIN-Kommunikation fokussierte, sondern auch eine Entwicklungsmethodik (LIN Work Flow) verfügbar machte, konnte es die Akzeptanz des Bussystems erheblich erhöhen. Mit Hilfe des LIN Work Flow lässt sich die Entwicklung eines LIN-Netzwerks (LIN-Cluster) automatisieren – somit lassen sich Zeit und Kosten sparen. Im Mittelpunkt der Entwicklungsmethodik stehen zwei Datenaustauschformate, mit deren Hilfe der gesamte LIN-Cluster sowie die einzelnen LIN-Knoten einheitlich beschrieben werden (**Bild 1**).

Zur Beschreibung eines gesamten LIN-Clusters dienen die einheitliche Syntax (LIN Configuration Language) und das standardisierte LIN Description File (LDF). Im LDF sind die kompletten Eigenschaften eines LIN-Clusters definiert, insbesondere die Kommunikationsbeziehungen. Mit Hilfe des

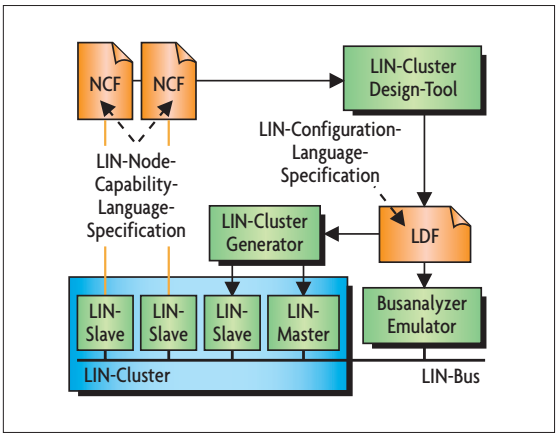


Bild 1. LIN Work Flow: der standardisierte und schnelle Weg zum LIN-Cluster. (Quelle: alle Bilder Vector Informatik)

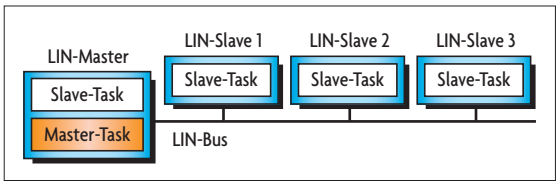


Bild 2. Master-Slave-Kommunikationsarchitektur: alle LIN-Knoten besitzen eine Slave-Task zur Teilnahme an der Kommunikation in einem LIN-Cluster. Ein LIN-Knoten besitzt zusätzlich eine Master-Task zur Steuerung der LIN-Kommunikation.

LDF können die Generierungswerkzeuge die Software-Komponenten für die LIN-Kommunikation erzeugen. Zusätzlich versorgt das LDF die Analyse-, Mess- und Testwerkzeuge oder Restbus-Emulatoren mit den nötigen Informationen. Analog dazu gestaltet sich die Beschreibung der einzelnen LIN-Knoten (LIN-Slaves) über die einheitliche Syntax der Node Capability Language und die standardisierten Node Capability Files (NCF). Das NCF beschreibt die Leistungsmerkmale eines LIN-Slaves, etwa die Frame- und Signaldefinitionen, Bitraten oder Diagnosefunktionen und stellt im Rahmen des Systemdesigns die Grundlage zur automatisierten Erstellung des LDF dar.

Mit Hilfe der beiden Datenaustauschformate und dem in der LIN-Spezifikation definierten Konfigurationsprozess ist es möglich, einen LIN-Slave-Typ (z.B. Schrittmotor) mehrmals in einem LIN-Cluster einzusetzen beziehungsweise einen LIN-Slave in verschiedenen LIN-Clustern einzusetzen, was die Wiederverwendbarkeit von LIN-Slaves erhöht.

Einen ebenso wichtigen Beitrag zum Erfolg von LIN leistet die detaillierte Dokumentation der Spezifikation. Die seit November 2006 vorliegende LIN-Spezifikation 2.1 [2] definiert den Physical Layer, das Kommunikationsprotokoll, den LIN Work Flow, die LIN API sowie die Diagnose und Konfiguration der LIN-Knoten.

LIN erlaubt Eindraht-Kommunikation mit bis zu 20 Kbit/s

Das Ziel, ein kostengünstiges Kommunikationsprotokoll für den seriellen Datenaustausch im sicherheitsunkritischen Sensor-/Aktor-Bereich zu schaffen, wirkte sich vor allem auf das Design des Physical Layer aus. So nutzt man für die physikalische Signalübertragung in einem LIN-Cluster nicht die von CAN bekannte Differenzsignalübertragung, sondern verwendet eine gewöhnliche Eindrahtleitung (Single Wire). Um trotzdem eine ausreichende Störfestigkeit zu gewährleisten, verwendet man als Bezugspotential für die Buspegel die Versorgungsspannung der Steuergeräte-Elektronik und die Fahrzeug-Masse. Ein LIN-Transceiver sorgt für die physikalische Busankopplung. Ein Pegel unterhalb 40 % der Versorgungsspannung wird vom Empfänger als eine logische Null interpretiert. Als logische Eins interpretieren Empfänger Pegel oberhalb von 60 % der Versorgungsspannung.

Die maximale Datenrate ist auf 20 Kbit/s begrenzt, damit sich die Abstrahlungen in Grenzen halten. Bei Leitungslängen bis 40 Meter ergibt sich unter Berücksichtigung der Knoten- und Leitungskapazitäten sowie der maximal zulässigen Zeitkonstante eines LIN-Clusters, welche die LIN-Spezifikation vorgibt, eine maximal empfohlene Knotenanzahl von 16.

Schaltungstechnisch entspricht ein LIN-Cluster einer Open-Collector-Schaltung. Ein Pull-up-Widerstand sorgt dafür, dass der Buspegel nahezu der Versorgungsspannung (High-Pegel) entspricht, wenn die Sendetransistoren aller LIN-Knoten sperren. Der Buspegel wird auf nahezu Masse (Low-Pegel) gezogen, sobald ein Sendetransistor öffnet. Entsprechend werden der Low-Zustand als dominanter Pegel und der High-Zustand als rezessiver Pegel bezeichnet.

LIN arbeitet mit Master-Slave-Kommunikation

Der Kommunikation in einem LIN-Cluster liegt eine Master-Slave-Architektur zugrunde. Ein Cluster besteht aus einem Master-Knoten (LIN-Master) und mindestens einem oder mehreren Slave-Knoten (LIN-Slaves). Man verzichtet aus Kostengründen auf explizite Kommunikationscontroller und realisiert statt dessen die LIN-Kommunikation über Software-Tasks, die so genannten Slave-Tasks, in jedem Knoten. Der LIN-Master besitzt zusätzlich eine Master-Task, mit deren Hilfe er die Cluster-Kommunikation koordiniert (**Bild 2**).

Die Koordination erfolgt mittels der zyklischen Abarbeitung der in Frame Slots organisierten LIN-Schedule (**Bild 3**). Jeweils zu Beginn eines Frame Slots überträgt die Master-Task einen Frame-Header mit einer Frame-Kennung (Identifier, ID), die alle LIN-Slaves über ihre Slave-Task auswerten. Ein LIN-Slave überträgt im Anschluss an den Frame-Header die mit der ID assoziierte Frame Response. Der LIN-Frame, gebildet aus Frame-Header und Frame-Response, steht wegen der Nachrichtenadressierung mit-

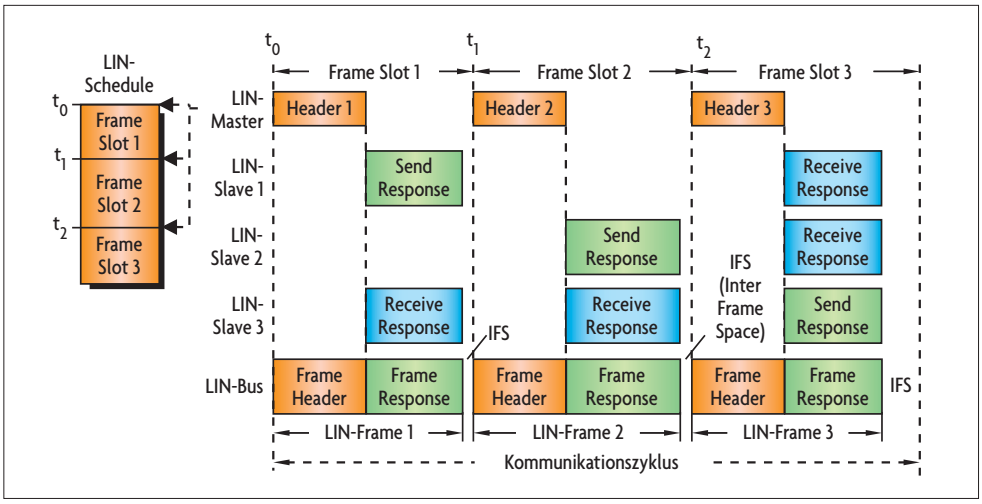


Bild 3. Zentrales Nachrichtenverteilsystem: Der LIN-Master steuert mittels Master-Task und LIN-Schedule das Senden und Empfangen aller LIN-Frames. Ein Frame Slot muss so lang sein, dass der entsprechende LIN-Frame übertragen werden kann. Die Länge des IFS hängt u.a. vom Abarbeitungstakt (Time Base) der Master-Task ab.

tels ID jedem LIN-Knoten zum Empfang zur Verfügung (Broadcast).

Datenübertragung mittels LIN-Frames

Die serielle Datenübertragung in einem LIN-Cluster wird wegen des fehlenden Kommunikationscontrollers über die serielle Schnittstelle (Serial Communication Interface, SCI) des Mikrocontrollers abgewickelt und erfolgt byteorientiert. Jedes Byte wird vom SCI mit dem LSB (Least Significant Bit) zuerst übertragen und von einem Start- und einem Stopp-Bit eingerahmt (SCI Frame). Ein LIN-Frame setzt sich folglich aus einer Anzahl von SCI Frames zusammen, aufgeteilt auf Frame-Header und Frame-Response (**Bild 4**).

Mit der Übertragung des Frame-Headers erledigt der LIN-Master zwei zentrale Kommunikationsaufgaben: Er synchronisiert die LIN-Slaves und delegiert die Kommunikation, indem er einen Sender beziehungsweise einen oder mehrere Empfänger für die Frame-Response bestimmt. Die LIN-Slaves dürfen aufgrund der Kostensensibilität On-Chip-Resonatoren mit einer Frequenz-Toleranz von bis zu 14 % benutzen. Deshalb überträgt der LIN-Master zunächst einen Sync-Break, um alle LIN-Slaves davon in Kenntnis zu setzen, dass die Übertragung eines LIN-Frames beginnt. Der Sync-Break setzt sich aus mindestens 13 dominanten Bits zusammen und ruft bei allen LIN-Slaves einen SCI-Fehler hervor. Er wird vom Sync-Break-Delimiter terminiert (mindestens ein rezessives Bit). Mit dem folgenden Sync-Byte (SCI Frame mit dem Wert 0x55) überträgt der LIN-Master den Kommunikationstakt.

Zur Delegierung der Kommunikation dient ein sechs Bit langer ID, der durch zwei Paritätsbits mit Even Parity und Odd Parity gesichert ist. Der ID und die beiden Paritätsbits werden zusammen als PID (Protected Identifier) bezeichnet. Die ersten 60 IDs stehen für die Nutzdatenkommunikation zur Verfügung. Die letzten vier Identifier, ID 60 bis ID 63, sind reserviert, ID 60 und ID 61 davon für Diagnosezwecke.

Die Frame-Response setzt sich aus bis zu acht Datenbytes und einer Checksumme für die Fehlererkennung zusammen. Man unterscheidet die klassische von der erweiterten Checksumme. Die klassische Checksumme entspricht der invertierten Modulo-256-Summe aller Datenbytes. Ein Übertragungsfehler liegt vor, wenn sich die Modulo-256-Summe mit den ankommenden Datenbytes nicht zu 0xFF addiert. Die erweiterte Checksumme berücksichtigt zusätzlich den PID bei der Bildung der invertierten Modulo-256-Summe.

Weil die LIN-Slaves meistens mit sehr einfachen und kostengünstigen Mikrocontrollern ausgestattet sind, ist ihnen nicht nur erlaubt, die Übertragung der Frame-Response ein wenig zu verzögern (Response Space), sondern auch Sendepausen zwischen der Übertragung der SCI Frames (Interbyte Spaces) einzulegen. Insgesamt darf sich die Frame-Response so um 40 % verlängern. Dasselbe gilt für den Frame-Header, vor allem, weil es unterschiedliche Methoden gibt, den Sync-Break zu erzeugen. Die Zeitreserve von 40 % muss man beim Design der LIN-Schedule unbedingt berücksichtigen.

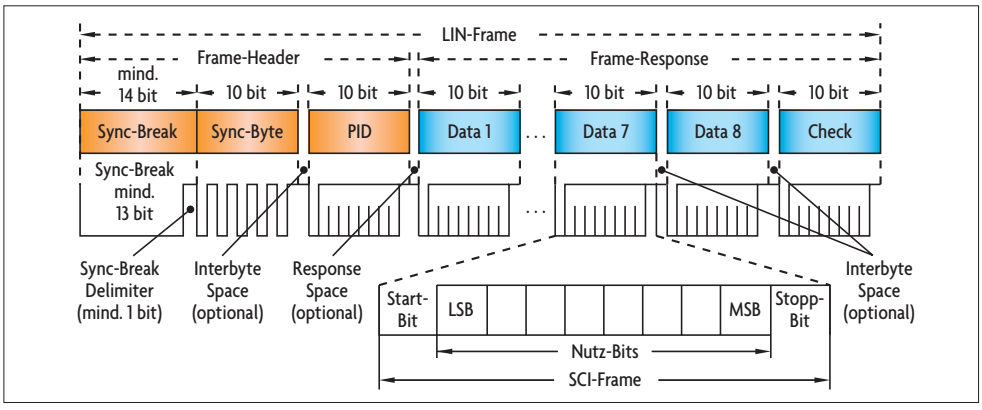


Bild 4. Jeder LIN-Frame setzt sich aus einem Frame-Header und einer Frame-Response zusammen. Der Frame-Header wird immer vom LIN-Master übertragen. Die Frame-Response kann von einem LIN-Slave oder vom LIN-Master übertragen werden.

Sporadic, Event Triggered und Diagnostic Frames

Die LIN-Spezifikation ermöglicht es, den durch die LIN-Schedule vorgegebenen starren Kommunikationszyklus zu flexibilisieren beziehungsweise zu vereinfachen. Dazu stehen die beiden Frame-Typen „Sporadic Frame“ und „Event Triggered Frame“ zur Verfügung. Im Zuge der Einführung der zusätzlichen Frame-Typen bezeichnet man den herkömmlichen LIN-Frame fortan als „Unconditional Frame“.

Unter einem Sporadic Frame versteht man einen Unconditional Frame, der sich mit anderen Unconditional Frames denselben Frame Slot teilt. Sporadic Frames werden vom LIN-Master bedarfsabhängig komplett übertragen, so dass Kollisionen ausgeschlossen sind. Wenn kein Bedarf seitens des LIN-Masters vorliegt, bleibt der entsprechende Frame Slot einfach leer.

Der Event Triggered Frame wurde eingeführt, um sporadische Veränderungen oder Ereignisse auf Seiten der LIN-Slaves zu kommunizieren. Er entspricht prinzipiell einem Unconditional Frame, nur mit dem Unterschied, dass dem Frame-Header mehrere Frame-Responses von unterschiedlichen LIN-Slaves zugeordnet sind. Mit welcher Frame-Response der Event Triggered Frame Header komplettiert wird, hängt vom Bedarf der entsprechenden LIN-Slaves ab. Ein Bedarf liegt dann vor, wenn neue Daten zu übertragen sind. Die Frame-Response des Event Triggered Frame wird im ersten Byte durch den PID des assoziierten Unconditional Frame identifiziert.

Im Gegensatz zum Sporadic Frame lassen sich beim Event Triggered Frame jedoch Kollisionen nicht ausschließen. Im Fall einer Kollision ist der LIN-Master für die Übertragung aller dem Event Triggered Frame zugeordneten Unconditional Frames verantwortlich. Dies erreicht er durch die Aktivierung und Abarbeitung einer Collision Resolving Schedule.

Zur Diagnose von LIN-Slaves eignen sich sowohl gewöhnliche Unconditional Frames als auch spezielle Diagnostic Frames. Unconditional Frames werden für die einfache signalbasierte Diagnose eingesetzt, während man Diagnostic Frames entweder zur benutzerdefinierten Diagnose oder zur Diagnose auf Basis eines standardisierten Transportprotokolls [3] und einheitlicher Diagnosedienste einsetzt [4, 5].

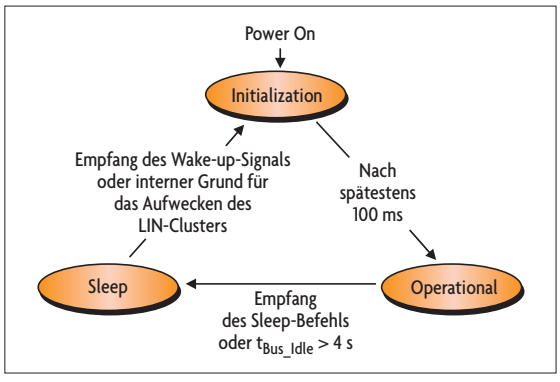


Bild 5. Kommunikationszustandsmodell eines LIN-Slaves.

Die LIN-Spezifikation definiert zwei Diagnostic Frames: Master Request Frame und Slave Response Frame. Der Master Request Frame (ID = 0x60) entspricht dem Diagnose-Request. In diesem Fall überträgt der LIN-Master sowohl den Frame-Header als auch den Frame-Response. Ein Master Request Frame wird beispielsweise dann übertragen, wenn ein Diagnose-Request über CAN anliegt. Der Slave Response Frame (ID = 0x61) entspricht der Diagnose-Response. In diesem Fall überträgt der LIN-Master den Header, der entsprechende LIN-Slave die Response.

LIN legt Status- und Netzwerkmanagement fest

Die LIN-Spezifikation definiert ein Statusmanagement und ein Netzwerkmanagement. Das Statusmanagement schreibt den LIN-Slaves vor, dass sie dem LIN-Master erkannte Übertragungsfehler wie Paritäts- oder Checksummenfehler anzuzeigen haben. Dazu dient ein Response Error Signal in einem Unconditional Frame (Status Frame), das allerdings keine weiteren Informationen über die Art des Fehlers enthält. Die LIN-Spezifikation enthält keine Definitionen zur Fehlerbehandlung, sondern überlässt diese Aufgabe dem Anwender.

Das LIN-Netzwerkmanagement regelt in erster Linie den Übergang aller Slaves eines LIN-Clusters vom normalen Kommunikationszustand (Operational) in den Sleep-Zustand und umgekehrt (**Bild 5**). Erkennen die LIN-Slaves für vier Sekunden keine Busaktivität, wechseln sie vom Operational-Zustand in den Sleep-Zustand. Dasselbe bewirkt ein Sleep-Kommando vom LIN-Master, bei dem es sich um einen speziellen Master Request Frame handelt.

Umgekehrt wechselt ein LIN-Slave vom Sleep- über den Initialisierungs- in den Operational-Zustand, wenn er ein Wake-up-Signal, gefolgt von einem gültigen Header erkennt. Das Wake-up-Signal besteht aus einem dominanten Puls von mindestens 250 Mikrosekunden und höchstens 5 Millisekunden Dauer und kann von jedem LIN-Knoten gesendet werden. Die LIN-Spezifikation schreibt eine maximale Initialisierungsphase von 100 Millisekunden vor, das heißt, der LIN-Master muss spätestens nach dieser Zeitspanne mit der Abarbeitung der LIN-Schedule beginnen. Bleibt er passiv, sendet der entsprechende LIN-Slave ein weiteres Wake-up-Signal. Die Anzahl von und der zeitliche Abstand zwischen den Wiederholungen sowie Time Outs sind in der Spezifikation festgelegt.

Bei Vernetzungsfragen auf externes Know-how zurückgreifen

Bei der Vernetzung mit LIN, CAN, FlexRay und MOST unterstützt Vector Informatik [6] Fahrzeughersteller und -zulieferer mit einer durchgängigen Werkzeugkette, Embedded-Software-Komponenten, Basis-Software für AUTOSAR-Steuergeräte und Hardware-Schnittstellen. Die Anwender des Werkzeugs CANoe.LIN profitieren beispielsweise während des gesamten Entwicklungsprozesses von praxisgerechten Funktionen für Modellerstellung, Simulation, Funktionstest, Konformitätstest, Diagnose und Analyse. Den busübergreifenden Entwurf und die Pflege der Kommunikationsdaten eines vernetzten Systems unterstützen beispielsweise die DaVinci Network Designer für LIN, CAN und FlexRay. Werkzeuge für die Entwicklung, Kalibrierung und Diagnose von Fahrzeugsteuergeräten ergänzen das umfangreiche Vector-Angebot. Für den Entwicklungsprozess von elektronischen Systemen bietet das Stuttgarter Unternehmen neben Beratung auch eine Werkzeugumgebung. Abgerundet werden die Leistungen durch ein umfangreiches Schulungsangebot zu den Vector-Tools, Software-Komponenten und seriellen Bussystemen.

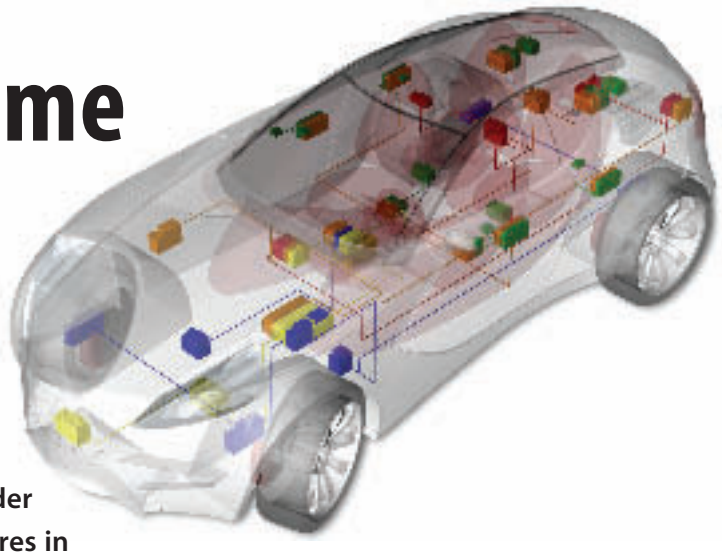
Für den Einstieg in den seriellen Datenaustausch im Automobil bietet die Vector Academy [7] die eintägige Schulung „Serielle Bussysteme im Automobil“ an. Grundlagenschulungen zu CAN, LIN, FlexRay und MOST vermitteln das notwendige Basiswissen, um sich schnell mit den vielfältigen Entwicklungstätigkeiten rund um die Automobilelektronik vertraut zu machen.

Die ersten beiden Teile dieser Beitragsreihe befassten sich mit dem seriellen Datenaustausch im Automobil [8] und mit CAN [9]. In zwei folgenden Beiträgen werden die seriellen Bussysteme FlexRay und MOST behandelt. Der interessierte Leser findet zu den bereits veröffentlichten Themen auf der Internetseite der Vector Academy [7] ergänzende und vertiefende Informationen in Form von E-Books. *sj*

Literatur und Links

[1] www.lin-subbus.org
[2] LIN Specification Package Revision 2.1
[3] Road vehicles – Diagnostics on Controller Area Network (CAN) – Part 2: Network layer services. International Standard ISO 15765-2.4, Issue 4, 2002-06-21.
[4] Road vehicles – Diagnostics on controller area network (CAN) – Part 3: Implementation of diagnostic services. International Standard ISO 15765-3.5, Issue 5, 2002-12-12.
[5] Road vehicles – Diagnostic systems – Part 1: Diagnostic services. International Standard ISO 14229-1.6, Issue 6, 2001-02-22.
[6] www.vector-informatik.de
[7] www.vector-academy.de
[8] *Mayer, E.:* Serielle Bussysteme im Automobil – Teil 1: Architektur, Aufgaben und Vorteile. *Electronic automotive* 2006, H. 7, S. 70 bis 73.
[9] *Mayer, E.:* Datenkommunikation im Automobil – Teil 2: Sicherer Datenaustausch mit CAN. *Electronic automotive* 2006, H. 8, S. 34 bis 37.

Serielle Bussysteme im Automobil



Teil 4: FlexRay für den Datenaustausch in sicherheitskritischen Anwendungen

FlexRay geht zum ersten Mal mit dem BMW X5 in Serie. Er wurde im August 2006 auf dem Pariser Autosalon der Öffentlichkeit vorgestellt und ist ab März dieses Jahres in Deutschland erhältlich. FlexRay sorgt dabei innerhalb des aktiven Fahrwerksystems für die sichere und zuverlässige Datenübertragung zwischen dem zentralen Steuergerät und den vier jeweils einem Stoßdämpfer zugeordneten Satellitensteuergeräte. Der Beitrag zeichnet den Weg von FlexRay ins Automobil nach und erläutert die wichtigsten Prinzipien des FlexRay-Bussystems.

Von Eugen Mayer

Laut Statistischem Bundesamt [1] war das Fahren auf Deutschlands Straßen noch nie so sicher wie im Jahr 2005. Bei beträchtlichem Zuwachs des Kraftfahrzeugbestandes ereigneten sich im Vergleich zum Vorjahr fast ein Prozent weniger Unfälle mit Personenschäden (336 619). Stark zurückgegangen ist dabei die Zahl der im Straßenverkehr Getöteten (5361, –8,2 %), Schwerverletzten (76 952, –4,6 %) und Leichtverletzten (356 491, –1 %). 2006 setzte sich dieser Trend fort: Zwischen Januar und August wurden 3260 Verkehrsteilnehmer getötet, was einem Rückgang um 7,8 % im Vergleich zum Vorjahreszeitraum entspricht. Die Zahl der Verletzten sank im gleichen Zeitraum um 5,8 %.

Entscheidend zur Reduzierung der Unfälle und Minderung der Unfallfolgen tragen aktive Sicherheitssysteme beziehungsweise Assistenzsysteme bei, die den Fahrer beim Führen seines Fahrzeugs unterstützen. Eine Studie namhafter Fahrzeughersteller zeigt zum Beispiel, dass ESP die Anzahl der Schleuderunfälle um bis zu 80 % verringert [2]. Einen ebenso wichtigen Beitrag zur Minderung der Unfallfolgen leisten die immer sichereren Fahrgastzellen und optimierten Rückhaltesysteme.

Mit dem Ziel, bis 2010 die Zahl der Verkehrstoten zu halbieren, forciert die Kfz-Branche vor allem die Weiter- und Neuentwicklung von innovativen aktiven Sicherheits- und Fahrerassistenzsystemen. Da es in diesem Zusammenhang nicht nur um das Informieren und Anweisen geht, sondern vielmehr um korrigierendes Eingreifen und Übernehmen von Fahraufgaben, kommt man ohne elektronische Schnittstellen zum Fahrwerk und zum Antrieb nicht mehr aus. Großes Potential wird der Kombination aus Brake-by-Wire- und Steer-by-Wire-Systemen zugeschrieben.

Erhöhte Anforderungen durch steigende Vernetzung

Die Realisierung von immer anspruchsvolleren Sicherheits- und Fahrerassistenzfunktionen geht einher mit einer immer intensiveren Kopplung der elektronischen Steuergeräte und erfordert deshalb sehr hohe Datenraten zur Übertragung der zunehmenden Anzahl von Steuer- und Statussignalen. Es handelt sich dabei in erster Linie um Signale, die aufgrund sehr zeitkritischer Regelungen nicht nur äußerst schnell, sondern auch absolut deterministisch zu übertragen sind. Folglich wächst die Bedeutung von Kommunikationssystemen im Automobil, die schnelle und deterministische Datenübertragung garantieren. Wegen des potentiellen Einsatzes von By-Wire-Systemen sind sie zudem mit fehlertoleranten Strukturen und Mechanismen auszulegen. Denn so vielfältig das Potential und die Vorteile von By-Wire-Systemen durch neue konstruktive Freiheiten, vereinfachte Montage oder einer Personalisierung des Fahrzeugs auch sein mögen, sie erhöhen die Anforderungen an die Datenübertragung in erheblichem Maße, weil sie zur Klasse der Fail-Operational-Systeme gehören. Diese müssen auch im Falle eines auftretenden Fehlers noch einwandfrei funktionieren.

Diesen Anforderungen wird CAN mit seinem ereignis- und prioritätengesteuerten Buszugriff der durch physikalische Randbedingungen im Automobil hervorgerufenen beschränkten Übertragungsrate von 500 Kbit/s sowie dem Mangel an fehlertoleranten Strukturen und Mechanismen nicht gerecht [3].

FlexRay beantwortet die Frage nach Determinismus und Fehlertoleranz

Die Gewissheit, dass CAN den wachsenden Anforderungen an die Daten-

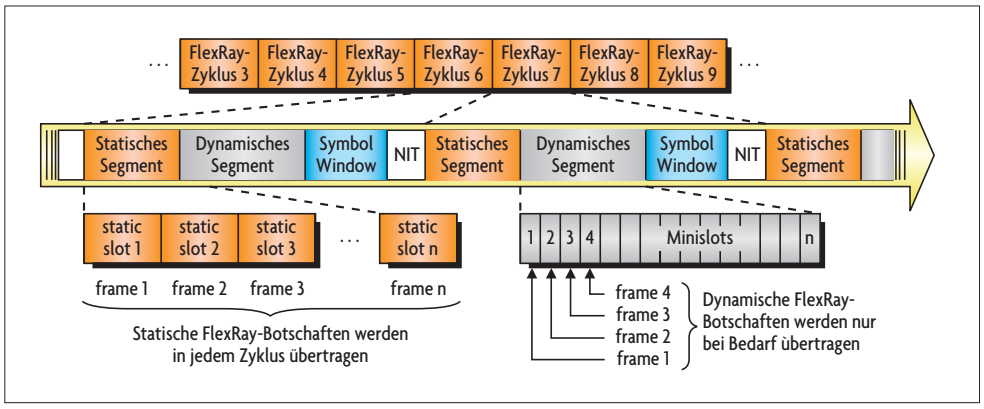


Bild 1. Die Übertragung von Nachrichten kann bei FlexRay im statischen oder dynamischen Segment erfolgen. Während des „Symbol Window" wird die Funktion des Buswächters überprüft; die „Network Idle Time" (NIT) wird zur Uhrensynchronisation genutzt. (Quelle aller Bilder: Vector Informatik)

übertragung im Automobil mittelfristig kaum mehr gerecht werden kann, führte zur Entwicklung mehrerer deterministischer und fehlertoleranter serieller Bussysteme mit höheren Datenraten. Dazu gehören zum Beispiel TTP (Time Triggered Protocol) [4], Byteflight [5] oder auch TTCAN (Time Triggered CAN) [6].

Ein wesentlicher Grund für den Erfolg von FlexRay war die Gründung des FlexRay-Konsortiums [8], in dessen Rahmen sich im Jahre 2000 die beiden Kfz-Hersteller DaimlerChrysler und BMW sowie die beiden Chiphersteller Motorola und Philips zusammenschlossen. Basierend auf dem ursprünglich von BMW entwickelten Byteflight-Bussystem schuf das FlexRay-Konsortium den herstellerübergreifenden, deterministischen und fehlertoleranten Kommunikationsstandard FlexRay mit einer Datenrate von 10 Mbit/s für äußerst sicherheits- und zeitkritische Anwendungen im Automobil.

Heute setzt sich das FlexRay-Konsortium aus sieben Core-Partnern zusammen: BMW, Bosch, DaimlerChrysler, Freescale, General Motors, Philips und Volkswagen. Nach und nach schloss sich dem FlexRay-Konsortium eine Reihe von Premium Associate Members an, wie beispielsweise Vector Informatik [7] und Associate Members.

Einen wichtigen Beitrag zum Erfolg von FlexRay leistet die detaillierte Dokumentation der FlexRay-Spezifikation. Die beiden wichtigsten Spezifikationen, das Kommunikationsprotokoll und der Physical Layer, liegen im Moment in der Version 2.1 vor. Diese und weitere Spezifikationen der FlexRay-Bustechnologie stehen auf der Homepage des FlexRay-Konsortiums zum Abruf bereit [8].

Definierter Kommunikationszyklus verbietet unkontrollierten Buszugriff

Ebenso wie bei der Datenkommunikation in einem CAN-Cluster liegt auch der Datenkommunikation in einem Flex-Ray-Cluster eine Multi-Master-Kommunikationsstruktur zugrunde. Allerdings dürfen die FlexRay-Knoten nicht wie bei CAN im Zuge von anwendungsbezogenen Ereignissen unkontrolliert auf den Bus zugreifen. Sie müssen sich vielmehr an einen exakt definierten Kommunikationszyklus halten, der jeder FlexRay-Botschaft ein bestimmter Zeitschlitz zuordnet (Time Division Multiple Access, TDMA) und dadurch die Sendezeitpunkte sämtlicher FlexRay-Botschaften vorgibt (**Bild 1**).

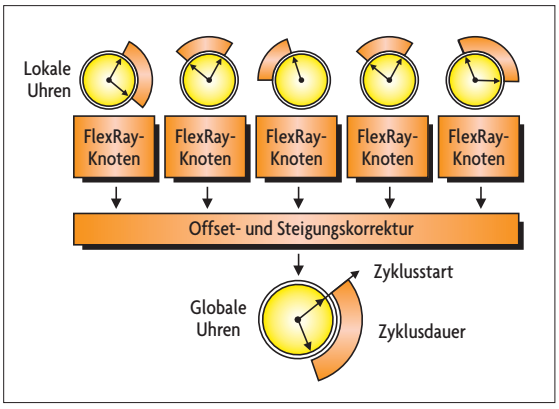


Bild 2. Die FlexRay-Knoten führen während des Kommunikationszyklus eine Steigungskorrektur durch, eine Offsetkorrektur erfolgt bei Bedarf am Ende der „Network Idle Time".

Die zeitgesteuerte Kommunikation sorgt nicht nur für deterministische Datenkommunikation, sondern auch dafür, dass alle Knoten eines FlexRay-Clusters unabhängig voneinander entwickelt und getestet werden können. Zudem wirkt sich das Entfernen oder die Integration von FlexRay-Knoten in ein bestehendes Cluster nicht auf den Kommunikationsablauf aus, was der in der Automobilentwicklung häufig propagierten Wiederverwendung entgegenkommt.

Dem Paradigma zeitgesteuerter Kommunikationsarchitekturen folgend, besteht die grundlegende Logik der FlexRay-Kommunikation in der Auslösung aller Systemaktivitäten durch das Erreichen bestimmter Punkte im Zeitablauf. Den dazu erforderlichen netzwerkweiten Gleichlauf der FlexRay-Knoten wird mittels verteiltem, fehlertoleranten Uhrensynchronisationsmechanismus sichergestellt: Alle FlexRay-Knoten korrigieren mittels regelmäßig übertragener Synchronisationsbotschaften nicht nur laufend den Beginn (Offsetkorrektur), sondern auch die Dauer (Steigungskorrektur) der Kommunikationszyklen (**Bild 2**). Dadurch erhöht sich sowohl die Bandbreiteneffizienz als auch die Robustheit der Synchronisation.

Der FlexRay-Kommunikation kann entweder ein elektrischer oder ein optischer Physical Layer zugrunde gelegt werden. Für die elektrische Signalübertragung spricht die Einfachheit, woraus sich Kostenvorteile ergeben. Die vergleichsweise kostenintensive optische Signalübertragung zeichnet sich durch eine im Vergleich zur elektrischen Signalübertragung deutlich bessere elektromagnetische Verträglichkeit (EMV) aus.

Die FlexRay-Kommunikation ist nicht an eine bestimmte Topologie gebunden. Eine einfache passive Bus-

struktur ist genauso möglich wie eine aktive Sterntopologie oder eine Kombination aus beidem (**Bild 3**). Die Vorteile der aktiven Sterntopologie bestehen primär in der Möglichkeit zum Abschalten von fehlerhaften Kommunikationszweigen beziehungsweise von FlexRay-Knoten sowie im Aufbau größerer Cluster mit idealen Busabschlüssen, wenn die physikalische Signalübertragung elektrisch erfolgt.

Zur Minimierung des Ausfallrisikos sieht FlexRay die redundante Auslegung des Kommunikationskanals vor (**Bild 4**). Dieser redundante Kommunikationskanal kann allerdings auch zur Erhöhung der Datenrate auf bis zu 20 Mbit/s herangezogen werden. Die Wahl zwischen Fehlertoleranz und erhöhter Übertragungsrate lässt sich für jede einzelne FlexRay-Botschaft treffen.

Schließlich sorgt ein unabhängiger Kontrollmechanismus (Buswächter – Bus Guardian) dafür, dass ein FlexRay-Knoten nur dann Zugang zum Bus erhält, wenn er laut Kommunikationszyklus an der Reihe ist. Dadurch wird die Busmonopolisierung durch einen defekten FlexRay-Knoten (Babbling Idiot) verhindert.

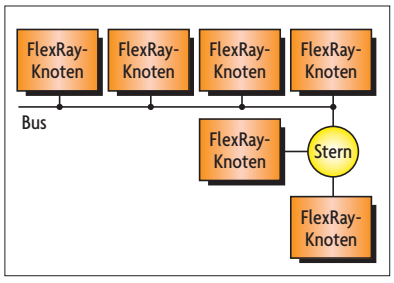


Bild 3. FlexRay erlaubt Bus- und Sternstrukturen, oder wie in diesem Beispiel eine kombinierte Topologie aus passivem Bus und aktivem Stern.

Statisches Segment überträgt echtzeit-relevante Daten

Jeder Kommunikationszyklus ist gleich lang und prinzipiell gegliedert in ein statisches und ein dynamisches Zeitsegment (Bild 1). Von zentraler Bedeutung ist das statische Segment, mit dem jeder Kommunikationszyklus beginnt. Es ist in eine frei definierbare Anzahl von maximal 1023 gleich langen statischen Zeitschlitzen (static slots) unterteilt.

Jedem statischen Zeitschlitz ist eine FlexRay-Botschaft zugeordnet, die von einem FlexRay-Knoten übertragen wird. Die Zuordnung von statischem Zeitschlitz, FlexRay-Botschaft und FlexRay-Knoten erfolgt mittels Slotnummer, Botschaftskennung (Identifier, ID) und dem Wert des auf jedem FlexRay-Knoten implementierten Slot-Counters. Damit pro Zyklus alle FlexRay-Botschaften in der richtigen Reihenfolge zeitlich korrekt übertragen werden, werden die Slot-Counter zu Beginn eines jeden statischen Zeitschlitzes auf allen FlexRay-Knoten synchron inkrementiert. Wegen der garantierten äquidistanten und somit deterministischen Datenübertragung ist das statische Segment prädestiniert für die Übertragung von echtzeitrelevanten Botschaften.

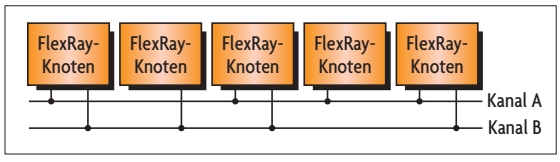


Bild 4. Um das Ausfallrisiko zu minimieren, arbeitet FlexRay wahlweise mit zwei getrennten Kommunikationskanälen.

Im Anschluss an das statische Segment folgt optional das je Kommunikationszyklus gleich lange dynamische Segment. Auch dieses Segment ist in Zeitschlitze gegliedert, aber nicht in statische Zeitschlitze, sondern in Minislots (Bild 1). Die Kommunikation im dynamischen Segment (Minislotting) basiert ebenfalls auf Zuordnungen und dem synchronen Inkrementieren der Slot-Counter.

Allerdings werden die den Minislots zugewiesenen FlexRay-Botschaften nicht zwangsläufig in jedem Kommunikationszyklus übertragen, sondern lediglich bei Bedarf. Wenn kein Bedarf vorliegt, werden die Slot-Counter nach der definierten Zeit eines Minislots inkrementiert. Bei der Übertragung einer dynamischen FlexRay-Botschaft verzögert sich die Inkrementierung der Slot-Counter um die Zeit der Botschaftsübertragung (**Bild 5**).

Die Zuordnung einer dynamischen FlexRay-Botschaft zu einem Minislot legt implizit die Priorität der FlexRay-Botschaft fest: Je niedriger die Nummer des Minislots, desto höher ist die Priorität der dynamischen FlexRay-Botschaft, dementsprechend früher erfolgt die Übertragung und folglich ist vor dem Hintergrund eines begrenzten dynamischen Zeitsegments auch die Übertragungswahrscheinlichkeit höher. Diejenige dynamische FlexRay-Botschaft, die dem ersten Minislot zugeordnet ist, wird bei Bedarf auf jeden Fall übertragen, wenn man von einem ausreichend langen dynamischen Zeitsegment ausgeht.

Beim Kommunikationsdesign muss man sicherstellen, dass auch die dynamische FlexRay-Botschaft mit der niedrigsten Priorität übertragen werden kann – zumindest, wenn sonst kein anderer Bedarf mit höherer Priorität vorliegt. Der Designer eines FlexRay-Clusters hat auch darauf zu achten, dass die Übertragung der längsten dynamischen FlexRay-Botschaft möglich ist.

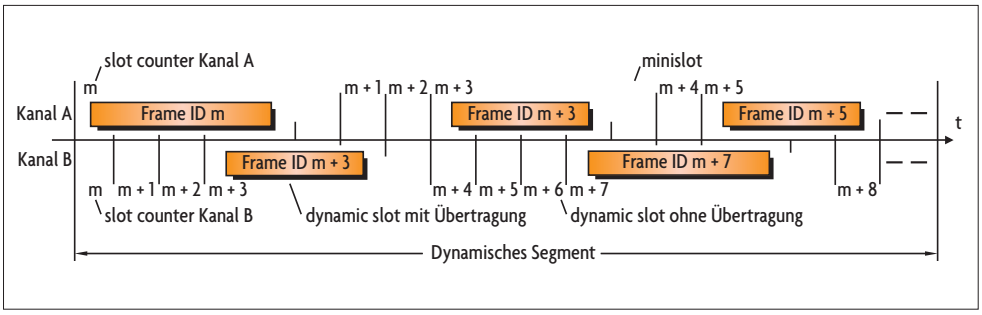


Bild 5. Im dynamischen Segment arbeitet FlexRay mit Minislots. Dabei ist jedem Minislot genau eine Botschaft zugewiesen, die jedoch nicht zwangsläufig in jedem Kommunikationszyklus übertragen werden muss.

Der Kommunikationszyklus wird durch zwei weitere Zeitsegmente komplettiert (Bild 1). Das Segment „Symbol Window“ dient zur Überprüfung der Funktion des Buswächters und das Zeitsegment „Network Idle Time“ (NIT) schließt den Kommunikationszyklus ab. Während der NIT berechnen die FlexRay-Knoten die zur Synchronisation der lokalen Uhren erforderlichen Korrekturfaktoren. Am Ende der NIT wird im Bedarfsfall die Offsetkorrektur durchgeführt (die Steigungskorrektur findet stets über den ganzen Kommunikationszyklus verteilt statt). Eine Datenübertragung findet während der NIT nicht statt.

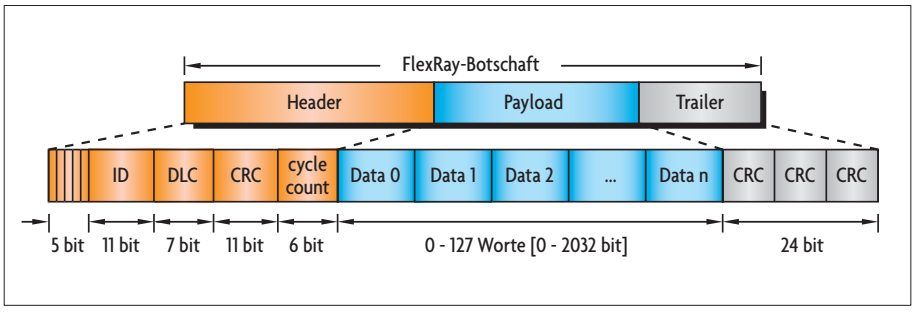


Bild 6. Eine FlexRay-Botschaft besteht aus Header, Payload und Trailer. Pro Nachricht können bis zu 254 Bytes an Nutzdaten übertragen werden.

Sichere Datenübertragung durch CRC

Die Übertragung von Signalen in einem FlexRay-Cluster erfolgt mittels der exakt definierten FlexRay-Botschaft, wobei es zwischen der im statischen Segment und der im dynamischen Segment übertragenen FlexRay-Botschaft im Aufbau prinzipiell keinen Unterschied gibt. Sie setzt sich stets aus einem Header, der Payload und dem Trailer zusammen (**Bild 6**).

Der Header umfasst das 5 bit breite Statusfeld, die ID, die Payload Length und den Cycle Counter. Die Header-CRC (11 bit) sichert Teile des Statusfeldes, die ID und die Payload Length mit einer Hamming-Distanz von sechs. Die ID kennzeichnet die FlexRay-Botschaft und korrespondiert mit einem Zeitschlitz im statischen oder dynamischen Segment. Im dynamischen Segment entspricht die ID der Priorität der FlexRay-Botschaft. Die einzelnen Bits des Statusfeldes spezifizieren die FlexRay-Botschaft genauer. Beispielsweise zeigt das „Sync Frame Indicator Bit“ an, ob die FlexRay-Botschaft zur Uhrensynchronisation verwendet werden darf.

Im Anschluss an den Header folgt die Payload. Insgesamt lassen sich mit einer FlexRay-Botschaft bis zu 254 byte an Nutzdaten transportieren. Der Trailer umfasst die den Header und die Payload sichernde CRC (24 bit). Bei einer Payload bis zu 248 byte garantiert die CRC eine Hamming-Distanz von sechs, für eine größere Payload liegt die Hamming-Distanz bei vier [8].

Durchgängige Lösung für Entwicklung, Simulation und Verifikation

Bereits im Jahr 2001 bot die Vector Informatik die erste Lösung für die Entwicklung von FlexRay-Systemen an. In der Zwischenzeit erhält der Entwickler ein umfassendes Portfolio [9]. Dazu gehören Werkzeuge für Design, Entwicklung, Simulation, Analyse, Test und Kalibrierung von Steuergeräten und verteilten Netzwerken. Mit dem DaVinci Network Designer FlexRay existiert eine Umgebung zum effizienten Design der Vernetzungsarchitektur und der Kommunikationsbeziehungen. Simulation, Analyse und Test von FlexRay-Systemen erfolgen mit CANoe.FlexRay, dessen Multibus-Konzept den gleichzeitigen Betrieb der Bussysteme FlexRay, CAN, LIN und MOST ermöglicht. Um das Systemverhalten des FlexRay-Netzwerkes bei Fehlern und Störungen genau zu untersuchen, generiert FRstress diese auf einem Kanal im FlexRay-Cluster.

Für den direkten Zugriff auf Steuergeräte-interne Größen benötigt der Entwickler ein spezielles Mess- und Verstellprotokoll: XCP on FlexRay. Im Rahmen der Entwicklung des aktiven Fahrwerksystems des neuen BMW X5 setzten die BMW-Ingenieure das Mess-, Kalibrier- und Diagnose-Tool CANape ein. Als XCP on FlexRay Master misst und verstellt CANape einzelne Steuergeräteparameter direkt über FlexRay. Neben Software entwickelt Vector Informatik auch Stacks für Steuergeräte. Mit den FlexRay-Softwarekomponenten lassen sich Applikationen unkompliziert an verschiedene Bus- oder Betriebssysteme anbinden. Für den hardwareseitigen Zugang zu FlexRay-Bussen sorgen passende Businterfaces für die USB-, PCI- und PCMCIA-Schnittstelle eines PC oder Notebooks.

Das notwendige Basiswissen, um sich mit den vielfältigen Entwicklungstätigkeiten rund um die Steuergerätekommunikation im Automobil vertraut zu machen, werden von der Vector Academy [10] im Rahmen von Seminaren zu CAN, LIN, FlexRay und MOST vermittelt.

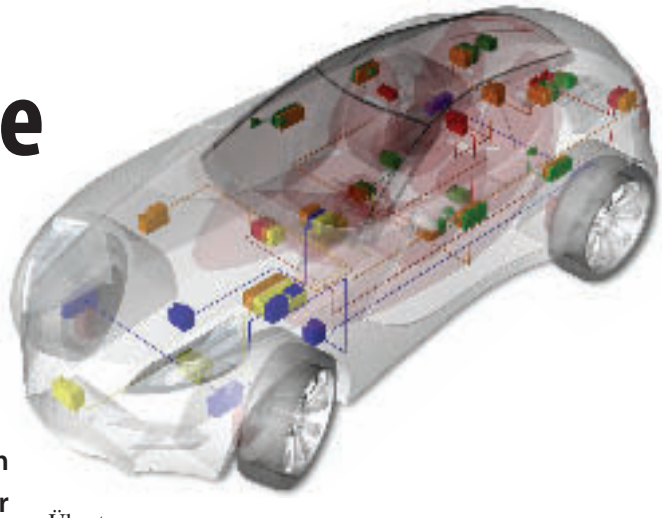
Die bisher erschienenen drei Teile dieser Beitragsreihe befassten sich allgemein mit dem seriellen Datenaustausch im Automobil, mit CAN und LIN. Im letzten Beitrag wird auf die Übertragung von Multimediadaten mit MOST eingegangen. Der interessierte Leser findet zu den bereits veröffentlichten Themen auf der Internetseite der Vector Academy ergänzende und vertiefende Informationen, unter anderem in Form von E-Books. *sj*

Literatur und Links

[1] www.destatis.de
[2] www.bosch.de
[3] *Mayer, E.:* Datenkommunikation im Automobil. Teil 2: Sicherer Datenaustausch mit CAN. *Elektronik automotive* 2006, H. 8, S. 34ff.
[4] www.tttech.com
[5] www.byteflight.com
[6] www.can-cia.org/can/ttcan
[7] www.vector-informatik.de
[8] www.flexray.com
[9] www.flexray-solutions.de
[10] www.vector-academy.de

Serielle Bussysteme im Automobil

Teil 5: MOST für die Übertragung von Multimediadaten



Ein Auto der Premiumklasse ähnelt zunehmend einem mobilen Büro. Auf Wunsch des Kunden drängen immer mehr moderne Unterhaltungs- und Informationsmedien in das Kraftfahrzeug. Zu den wichtigsten Herausforderungen gehört dabei, den Verkabelungsaufwand gering zu halten und gleichzeitig den gestiegenen Anforderungen an den Funktionsumfang eines Infotainmentsystems im Fahrzeug gerecht zu werden. In circa 50 Modellreihen kommt daher bereits das Bussystem MOST (Media Oriented System Transport) zur Übertragung von Audio- und Videosignalen zum Einsatz.

Von Eugen Mayer

War früher das Autoradio einziges Infotainment-Gerät, kamen im Laufe der Zeit CD- und MP3-Player, Navigationsgeräte und schließlich auch Bildschirme für die Wiedergabe von Video- und DVD-Filmen hinzu. Darüber hinaus lassen Bluetooth-Freisprecheinrichtungen mit integriertem Mikrofon und iPod-Steuerung das Cockpit allmählich zum Multimedia-Center werden, über das sich alle Playlisten und Verzeichnisse eines digitalen MP3-Players direkt auf dem Fahrzeugdisplay anzeigen und auch starten lassen.

Der ohnehin bereits umfangreiche Verkabelungsaufwand nimmt durch die weiter ansteigende Vernetzung der immer leistungsfähigeren Infotainmentgeräte kaum mehr handhabbare Ausmaße an. Glücklicherweise erkannten einige Kfz-Hersteller schon früh, welche Vorteile die Busvernetzung auch für diesen Bereich zu bieten hat. Mitte der 1990er Jahre begannen BMW und Daimler auf der Basis des von Matsushita und Philips entwickelten D2B-Bus (Digital Data Bus) eine einheitliche Kommunikationstechnologie zur seriellen Übertragung von Audio- und Videosignalen im Fahrzeug zu entwickeln.

Die MOST Cooperation

1998 gründeten BMW, Daimler, Harman/Becker und SMSC (vormals OASIS Silicon Systems) die MOST Cooperation [1]. Inzwischen hat sich MOST als De-facto-Standard für die Übertragung von Multimediadaten im Fahrzeug etabliert; die MOST Cooperation umfasst 15 internationale Fahrzeug- und mehr als 70 Gerätehersteller. Die MOST-Spezifikation liegt seit Oktober 2006 in der Version 2.5 vor. Sie gliedert sich in die Bereiche Application, Network und Hardware.

Der Bereich Application beschreibt ein logisches Gerätemodell zur transparenten Modellierung und Steuerung von verteilten Infotainment-Systemen und basiert in erster Linie auf Methoden der Objektorientierung. Weiterhin definiert er ein hierarchisches Kommunikationsmodell sowie Dienste zur Verwaltung eines Infotainment-Systems.

Die Network Section beschreibt den „MOST Network Interface Controller“ und seine Dienste, das Netzwerkmanagement sowie die Handhabung des Datentransports in einem MOST-System. Die Hardware Section setzt sich mit der Hardware-Struktur eines MOST-Gerätes auseinander.

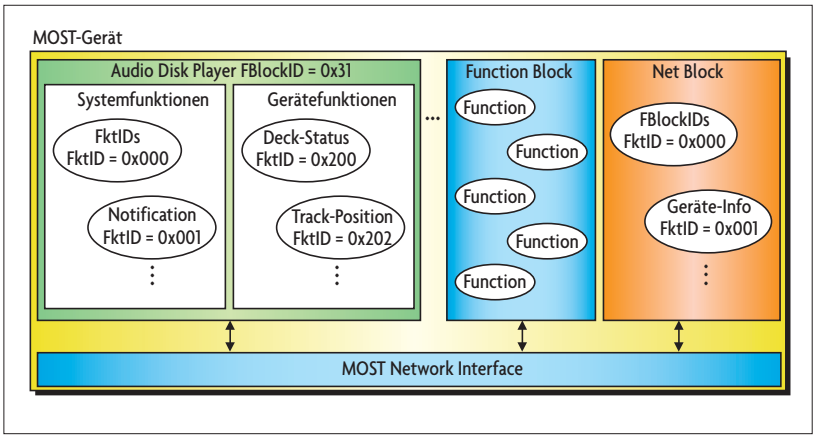


Bild 1. Struktur eines MOST-Gerätes, das unter anderem den Function Block „Audio Disk Player“ beherbergt. Zur Systemverwaltung sind für jedes MOST-Gerät der Net Block, für jeden Function Block Systemfunktionen obligatorisch.

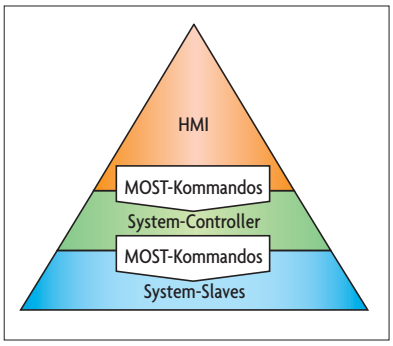


Bild 2. Hierarchie eines MOST-Systems.

Modellierung von Funktionen

Ein MOST-Gerät gliedert sich in eine Funktions- und in eine Netzwerkebene (MOST Network Interface). Auf der Funktionsebene sind die Infotainment-Funktionen als Funktionsblöcke (Function Blocks) untergebracht. Jeder Funktionsblock wie etwa der „Audio Disk Player“ stellt dem MOST-Netzwerk einen dedizierten Satz von Funktionen (z.B. Track Position) zur Verfügung, auf die mittels Operation Types (z.B. Set zum Setzen eines Tracks, oder SetGet zum Setzen und Lesen eines Tracks) zugegriffen werden können (**Bild 1**).

Sowohl den Funktionsblöcken als auch den von einem Funktionsblock bereitgestellten Funktionen sind Adressen (FBlockID, FktID) zugeordnet. Man kann sie, ebenso wie die Kennungen der Operation Types, dem Function Catalog entnehmen. So hat der FBlock „Audio Disk Player“ den FBlockID=0x31 – die Funktion „Track Position“ den FktID=0x202.

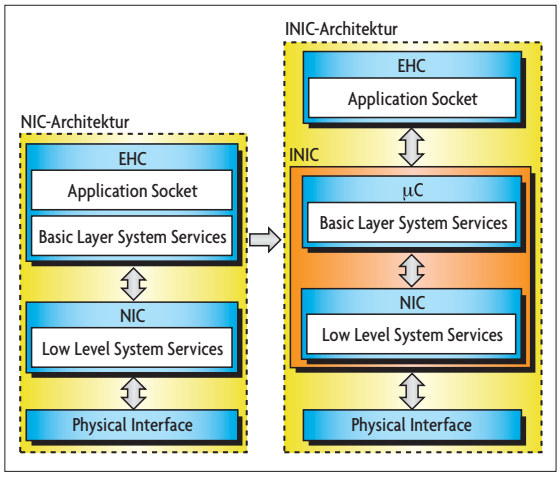


Bild 3. Unterschied zwischen der NIC- und der INIC-Architektur eines MOST-Gerätes.

Durch die Trennung von Funktion und Netzwerk und mit Hilfe der Funktionsmodellierung lässt sich ein Kommunikationsmodell realisieren, welches völlig unabhängig von physikalischen MOST-Geräten ist. Folglich spielt es keine Rolle, auf welchen MOST-Geräten man einen Funktionsblock unterbringt.

Hierarchisches Kommunikationsmodell

MOST-Systemen liegt eine dreistufige, hierarchische Steuerungsphilosophie nach dem Master-Slave-Prinzip zugrunde (**Bild 2**). Auf der obersten Hierarchieebene ist die HMI (Human Machine Interface) angesiedelt, ein exponierter Controller, der dem Anwender den gesamten Funktionsumfang zur Verfügung stellt. Auf der mittleren Hierarchieebene befinden sich die üblichen Controller. Sie decken einen Teil der Systemfunktionen ab und stellen ihr partielles Systemwissen der HMI als System-Master zur Verfügung.

Die unterste Hierarchieebene bilden die System-Slaves, deren Funktionen jeweils von einem oder mehreren Controllern nutzbar sind. Sie sind mit keinerlei Systemwissen ausgestattet, was die Flexibilität hinsichtlich der Konfiguration wesentlich erhöht. System-Slaves lassen sich einem MOST-System einfach hinzufügen oder entfernen. Zur Steuerkommunikation dienen MOST-Kommandos. Sie umfassen im Kern den FBlockID, den FktID, den Operation Type und bis zu 65 535 Nutzbyte.

Systemverwaltung

Die Application Section definiert auch übergeordnete Funktionsblöcke und Funktionen zur Systemverwaltung. Zu den Systemfunktionen zählt unter anderem die Funktion FktIDs (FktID=0x000), mit der man die von einem Funktionsblock unterstützten Funktionen abfragt. Die Systemfunktion Notification (FktID=0x001) erlaubt das Erstellen der Notification-Matrix für einen Funktionsblock. Aus der Notification-Matrix geht hervor, welches MOST-Gerät zu benachrichtigen ist, wenn sich eine bestimmte Eigenschaft eines Funktionsblocks verändert hat. Dieser Mechanismus verhindert ein unnötiges Ansteigen der Buslast im MOST-System.

Zur Abfrage seiner Funktionsblöcke und Adressen hat jedes MOST-Gerät den (System-)Funktionsblock Net Block mit der FBlockID=0x01. Mittels der Funktion FBlockIDs (FktID=0x000) können die auf einem MOST-Gerät implementierten Funktionsblöcke in Erfahrung gebracht werden. Über die FktIDs 0x002, 0x003 und 0x004 lassen sich die physikalische und die logische Adresse sowie die Gruppenadresse eines MOST-Gerätes ermitteln.

Eine wichtige Aufgabe bei der Verwaltung eines MOST-Systems hat der Network Master. Er ist für den Systemstart und die Verwaltung der Central Registry verantwortlich. Diese umfasst die logischen Adressen der in einem MOST-System implementierten MOST-Geräte und die Adressen der darauf untergebrachten Funktionsblöcke.

„MOST Network Interface“

Das „MOST Network Interface“ (**Bild 3**) sorgt dafür, dass die auf unterschiedlichen MOST-Geräten untergebrachten Funktionsblöcke in der Lage sind, miteinander zu kommunizieren. Dabei erbringen die „MOST System Services“ („Low Level System“ und „MOST Network Services“) die Kommunikationsfunktionen, die für den Transport aller multimediarelevanten Daten (zeitkontinuierliche Bitströme, Paketdaten und Steuerdaten) notwendig sind. Die „Low Level System Services“ (Schicht-2-Dienste) sind in Hardware (Network Interface Controller; NIC) implementiert und setzen auf dem Physical Layer auf.

Die „MOST Network Services“, die den Transport Layer in Form von „Basic Layer System Services“ und das höhere Management in Form eines Application Sockets umfassen, sind auf einem externen Host Controller (EHC) untergebracht und steuern den NIC. Dabei muss sichergestellt sein, dass der EHC auch die zeitkritischsten Teile des Network Interface bedienen kann. Durch die Weiterentwicklungen, von MOST25 über MOST50 zu MOST-

150, stößt diese Architektur inzwischen an ihre Grenzen.

In Neuentwicklungen löst der INIC (Intelligent Network Interface Controller) den NIC ab. Während der INIC die Abwicklung der zeitkritischen Teile des Netzwerktreibers vom EHC übernimmt, läuft auf dem EHC nur noch ein relativ kleiner Teil des Netzwerktreibers, der im Wesentlichen einen Sockel für die Applikation darstellt. Die INIC-Architektur entlastet somit den EHC. Zur Steuerung stellt der INIC dem EHC bzw. der MOST API („MOST Network Services“) mit der INIC API eine Schnittstelle zur Verfügung. Die Funktionen des INIC sind durch einen Funktionsblock (FBlock INIC) gekapselt.

MOST Networking

MOST ermöglicht die Übertragung von kontinuierlichen Bitströmen (Bitstreaming) ohne Pufferung und unnötigen Overhead. Dazu speist ein ausgewiesenes MOST-Gerät (Timing Master) den MOST-Frame (**Bild 4**) mit einer festen Frequenz (44,1 kHz oder 48 kHz) in das üblicherweise optische Übertragungsmedium ein.

In einem MOST25-System stellt der MOST Frame 60 Streaming Channels zu je 8 bit (bzw. 15 Quadlets zu je 4 byte) zur Übertragung von kontinuierlichen Bitströmen zur Verfügung (Source Data Area). Die Übertragungsrate eines Streaming Channels ergibt sich entweder zu 352,8 kbit/s (44,1 kHz) oder zu 384 kbit/s (48 kHz).

Da die MOST-Geräte physikalisch zu einem Ring zusammengeschlossen sind, muss jeder MOST-Frame jedes MOST-Gerät mit der vom Timing Master vorgegeben Frequenz passieren. Sobald sich die entsprechenden Kommunikationspartner (Datenquelle und -senke) mit denselben Streaming Channels verbunden haben, beginnt das Bitstreaming (**Bild 5**).

Auf- und Abbau der Verbindung geschieht üblicherweise auf Anfrage durch den Funktionsblock Connection Master (CM; FblockID=0x03). Zu diesem Zweck stellt der CM die beiden Funktionen BuildSyncConnection und RemoveSyncConnection bereit.

Im Rahmen des Verbindungsaufbaus fordert der CM die entsprechende Datenquelle auf, zum Beispiel den TV-Tuner, sich die entsprechende Anzahl Streaming Channels beim Timing Master allokieren zu lassen. Denn der Timing Master ist für die Verwaltung der „Channel Resource Allocation Table“ zuständig. Die Adressen der allokierten Streaming Channels gibt der CM der Datensenke, zum Beispiel Display, weiter, damit sich diese mit den Streaming Channels verbinden kann. Zum Abschluss aktualisiert der CM die „Sync Connection Table“, über die er sämtliche synchronen Verbindungen verwaltet. Der Abbau funktioniert nach dem gleichen Schema.

Um auch Datenpakete übertragen zu können, hat der Anwender die Möglichkeit, die Anzahl der Streaming Channels mittels Boundary Descriptor bis auf 24 (sechs Quadlets) zu verringern. Denn alle Streaming Channels, die nicht für das Bitstreaming reserviert sind, werden zum Packet Channel zusammengefasst. Während bei 44,1 kHz eine maximale Übertragungsrate von bis zu 12,7 Mbit/s möglich ist, erreicht man bei 48 kHz eine maximale Übertragungsrate von bis zu 13,8 Mbit/s. Verwaltet wird der Boundary Descriptor vom Funktionsblock Network Master (FBlockID= 0x02). Über die Funktion Boundary (FktId=0xA03) lässt sich dieser setzen.

Zur Übertragung von Datenpaketen kommt ein Schicht-2-Protokoll zum Einsatz. Der Frame setzt sich aus dem Arbitrierungsfeld, der Quell- und Zieladresse, dem Data Length Code, dem Datenfeld (48 oder 1014 byte) und der Datensicherung zusammen. Ein im Ring zirkulierender Token regelt den Buszugriff. Jenes MOST-Gerät, welches den Token vom Ring nimmt, darf auf den Packet Channel zugreifen.

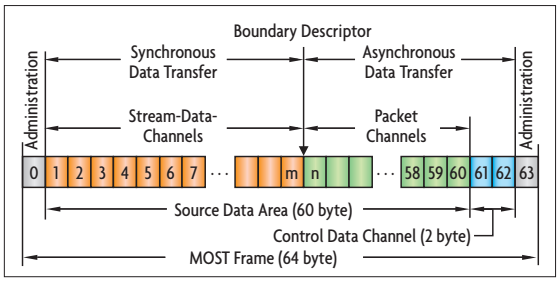


Bild 4. Aufbau des MOST-Frames: Im Administrations-Byte 0 werden Synchronisationsinformationen und der Boundary Descriptor, im Administrations-Byte 63 werden Status-Bits und ein Paritäts-Bit zur Sicherung des MOST-Frames übertragen.

Schließlich muss das MOST-System die zur Verwaltung und Steuerung notwendigen MOST-Kommandos übertragen. Dazu kommen Control Messages (**Bild 6**) zum Einsatz, die im Control Channel übertragen werden. Zur Übertragung einer Control Message sind 16 MOST-Frames (MOST-Block) erforderlich. Die Übertragungsrate beträgt bei 44,1 kHz 705,6 kbit/s und bei 48 kHz 768 kbit/s. Auch der Übertragung der Control Messages liegt ein Schicht-2-Protokoll zugrunde. Der Buszugriff erfolgt mittels CSMA-Verfahren (Carrier Sense Multiple Access).

Physical Layer

Zur Übertragung der Audio- und Videosignale im MOST-System sind

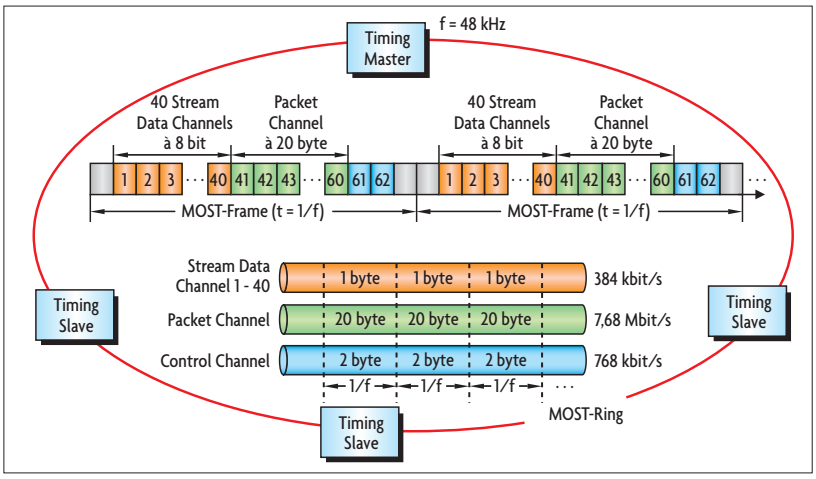


Bild 5. Prinzip des Bitstreamings: Der Timing Master überträgt mit der Frequenz von 48 kHz MOST-Frames. Es stehen 40 Streaming Channels (10 Quadlets) mit je 384 kbit/s zur Allokierung bereit (Boundary Descriptor = 0xA). Der Packet Channel (20 byte) stellt für die Übertragung von Datenpaketen eine Bandbreite von 7,68 Mbit/s zur Verfügung.

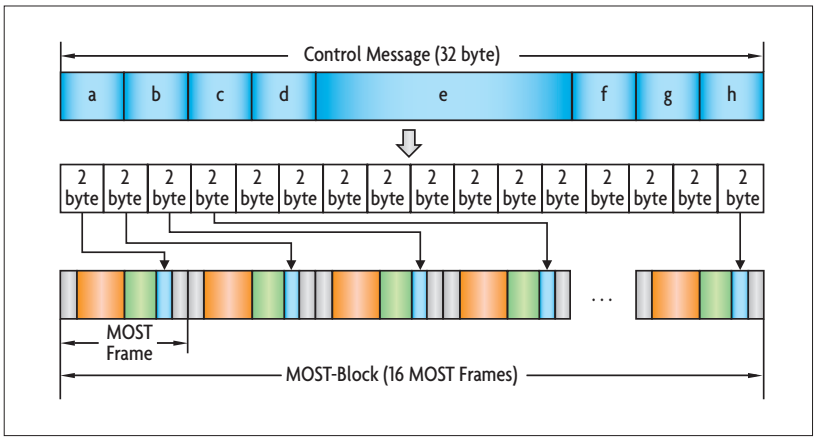


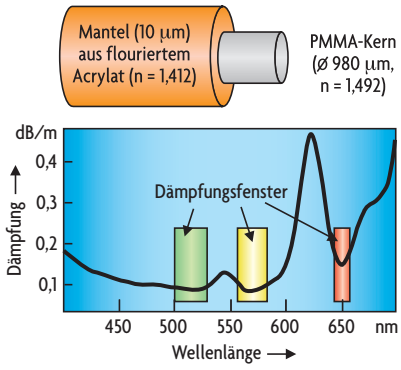
Bild 6. Zur Übertragung einer Control Message ist ein MOST-Block erforderlich. Die Control Message setzt sich zusammen aus: Arbitrierung (a), Zieladresse (b), Quelladresse (c), Message Typ (d), Datenfeld (e), Datensicherung (f), Bestätigung (g) und Reservierung (h).

heute Lichtwellenleiter aus Polymerfasern (POF; polymeroptische Faser) Stand der Technik (**Kasten**). In der Summe sind die technischen Eigenschaften der Polymerfasern denen elektrischer Übertragungsmedien weit überlegen. Zu erwähnen ist vor allem die hervorragende elektromagnetische Störfestigkeit und die vergleichsweise hohe Signalübertragungsgeschwindigkeit von bis zu 500 Mbit/s. Außerdem stellt die Kombination aus POF, roter Leuchtdiode als Lichtquelle und einer Silizium-PIN-Fotodiode als Empfänger eine sehr günstige und vergleichsweise einfach handhabbare Form der optischen Signalübertragung dar.

Mit MOST150 geht nach MOST50 aktuell ein MOST-System an den Start, dem diese Sender- und Empfänger-Technik zugrunde liegt und dem Anwender eine Übertragungsgeschwindigkeit von 150 Mbit/s zur Verfügung stellt. Die im Auto vergleichsweise kurzen Strecken von bis zu 20 m sind damit problemlos zu bewältigen.

Signalübertragung mittels POF

Tritt ein Lichtstrahl von einem transparenten Medium in ein anderes über, so wird er gebrochen. Die Brechung ist um so stärker, je größer der Einfallswinkel ist. Das Medium, in dem der Lichtstrahl mit dem Lot den kleineren Winkel bildet, ist das optisch dichtere Medium. Beim Übergang vom optisch dichteren zum optisch dünneren Medium wird der Strahl vom Lot weg gebrochen. Der Brechungswinkel α kann berechnet werden, wenn die so genannten Brechzahlen n der beiden Medien bekannt sind (Snellius-Gesetz). Überschreitet der Lichtstrahl beim Übergang vom optisch dichteren zum optisch dünneren Medium den Einfallswinkel α_0, so tritt Totalreflexion ein.
Aufgrund dieser Eigenschaft kann man Licht in einem optischen Leiter transportieren. Im MOST-System setzt man üblicherweise Polymerfasern zur optischen Signalübertragung ein, deren Kern aus PMMA (Polymethylmethacrylat) von einem dünnen Mantel aus fluoriertem Acrylat umgeben ist. PMMA hat eine größere Brechzahl als fluoriertes Polymer.



Wenn der eingehende Lichtstrahl über dem Grenzwinkel liegt, wird aufgrund der Totalreflexion das Licht im Kern geführt.
Die kleinsten Dämpfungen für das Übertragen von Licht in einer Step-Index-PMMA-Fasern erhält man bei 520 nm (grün), 560 nm (gelb) und 650 nm (rot). In erster Linie werden rote LEDs verwendet (Dämpfung: 0,14 dB/m), da sie sehr kostengünstig sind

Entwicklung, Test und Analyse von MOST-Systemen

Die Vector Informatik GmbH ist seit 1999 assoziiertes Mitglied der MOST Cooperation und bietet Unterstützung bei Entwicklung und Analyse von Infotainment-Lösungen im Automobil. Für Anwendungen wie High-End-Audiosysteme, Multimedia-Streaming, Telefonie und Navigation gibt es eine umfassende Palette an Analyse-, Entwicklungs- und Test-Werkzeugen, Hardware-Schnittstellen, Datenlogger sowie Schulungen und Dienstleistungen [2]. Das notwendige Basiswissen rund um die Steuergerätekommunikation im Automobil vermittelt die Vector Academy [3] im Rahmen von Seminaren zu CAN, LIN, FlexRay und MOST. *sj*

Literatur

[1] www.mostcooperation.com
[2] www.vector-group.net/most/de
[3] www.vector-academy.de
[4] *Mayer, E.:* Serielle Bussysteme im Automobil. Teil 1: Archtiektur, Aufgaben und Vorteile. *Elektronik automotive* 2007, H. 7, S. 70 – 73.
[5] *Mayer, E.:* Datenkommunikation im Automobil. Teil 2: Sicherer Datenaustausch mit CAN. *Elektronik automotive* 2007, H. 8, S. 34 – 37.
[6] *Mayer, E.:* Serielle Bussysteme im Automobil. Teil 3: Einfacher und kostengünstiger Datenaustausch mit LIN. *Elektronik automotive* 2008, H. 1, S. 40 – 43.
[7] *Mayer, E.:* Serielle Bussysteme im Automobil. Teil 4: FlexRay für den Datenaustausch in sicherheitskritischen Anwendungen. *Elektronik automotive* 2008, H. 2, S. 42 – 45.

Dipl.-Ing., Dipl.-Techpaed. Eugen Mayer
hat an der FH Ravensburg/Weingarten Elektronik und an der Universität Stuttgart Elektrotechnik und Berufspädagogik studiert. Er arbeitet seit 1999 bei der Vector Informatik und ist dort als Senior Trainer tätig.
eugen.mayer@vector-informatik.de

Es ist keine Kunst, Automobil-Vernetzung zu entwickeln,

wenn man es einmal gelernt hat.
Egal, ob Sie in das Thema Automobil-Elektronik neu einsteigen oder ob Sie schon ein versierter Profi sind: in der **VectorAcademy** finden Sie mit Sicherheit den richtigen Workshop für Ihren Wissens-Level. Ihr Vorteil: Sie buchen nur die Module, die Sie wirklich brauchen.

CAN, LIN, AUTOSAR, FlexRay, MOST, offene Protokolle... statt Antworten in Büchern zu suchen, löchern Sie doch unsere Trainer! Effektiver können Sie sich Wissen und Fähigkeiten nicht aneignen: wir vermitteln Ihnen strukturiert die Theorie, leiten Sie bei praxisorientierten Übungen an. Und Sie tauschen Erfahrungen mit Kollegen aus Ihrer Branche aus.

Mit weniger sollten Sie sich nicht zufrieden geben. Dafür kostet es weniger als Sie vielleicht denken. Den schnellen Überblick verschaffen Sie sich am besten auf unseren Web-Seiten. Falls Antworten offen bleiben, unsere Schulungsberater freuen sich auf Ihren Anruf:

Tel (07 11) 8 06 70-5 70 und -5 76
www.vector-academy.de

VectorAcademy. Mehr Praxis.

Vector Informatik GmbH
Ingersheimer Str. 24
70499 Stuttgart
Tel (07 11) 8 06 70-0 Fax (07 11) 8 06 70-1 11
info@vector-informatik.com
www.vector-informatik.com

vector

Ihre Ansprechpartner

Deutschland und alle Länder, soweit nicht unten genannt

Vector Informatik GmbH
Ingersheimer Str. 24
70499 Stuttgart
GERMANY
Tel.: +49 711 80670 0
Fax: +49 711 80670 111

USA, Kanada, Mexiko

Vector CANtech, Inc.
Suite 550
39500 Orchard Hill Place
Novi, Mi 48375
USA
Tel.: +1 248 449 9290
Fax: +1 248 449 9704

Frankreich, Belgien, Luxemburg

Vector France SAS
168, Boulevard Camélinat
92240 Malakoff
FRANCE
Tel.: +33 1 4231 4000
Fax: +33 1 4231 4009

Japan

Vector Japan Co., Ltd.
Seafort Square Center Bld. 18F
2-3-12 Higashi-shinagawa,
Shinagawa-ku
Tokyo 140-0002
JAPAN
Tel.: +81 3 5769 6970
Fax: +81 3 5769 6975

Schweden, Dänemark, Norwegen, Finnland, Island

VecScan AB
Lindholmspiren 5
41756 Göteborg
SWEDEN
Tel.: +46 31 764 76 00
Fax: +46 31 764 76 19

Korea

Vector Korea IT Inc.
Daerung Post Tower III, 508
Guro-gu, Guro-dong 182-4
Seoul 152-790
REPUBLIC OF KOREA
Tel.: +82 2 2028 0600
Fax: +82 2 2028 0604

www.vector-worldwide.com