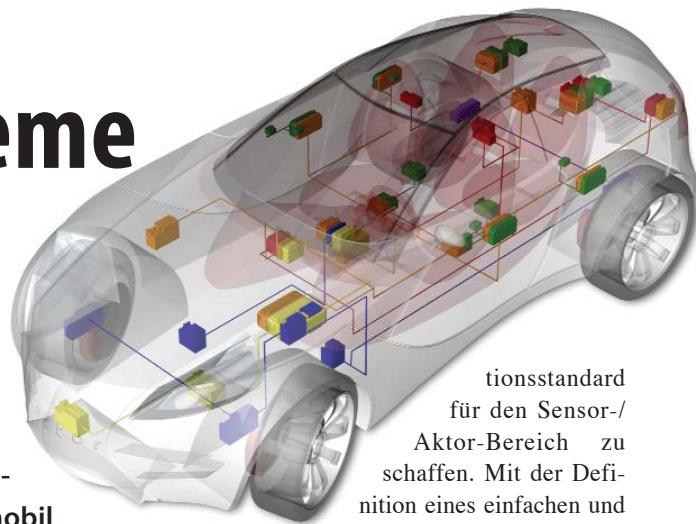


Serielle Bussysteme im Automobil

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



tionsstandard 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 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 be-

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.

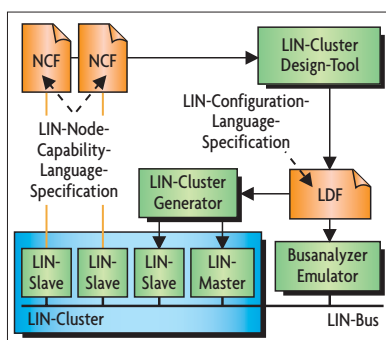
Nach und nach entstanden zahlreiche kostengünstige und einfache, aber proprietäre Sensor-/Aktor-Busse. Im Jahr 2000 kam mit LIN ein weiteres serielles Bussystem für den Sensor-/Aktor-Bereich auf den „Vernetzungsmarkt“. Diese Technologie setzte sich auf breiter Front durch, so dass man LIN inzwischen 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 zusammenschlossen hatten, um einen herstellerübergreifenden Kommunika-

Bild 1. LIN Work Flow: der standardisierte und schnelle Weg zum LIN-Cluster.

(Quelle: alle Bilder Vector Informatik)



schreibt 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.

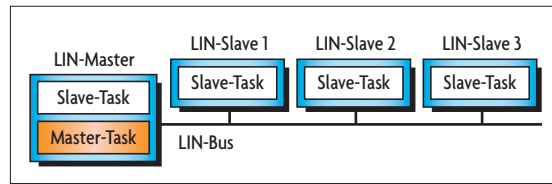
Mit Hilfe der beiden Datenaustauschformate und dem in der LIN-Spezifikation definierten Konfigurationsprozess ist es möglich, einen LIN-Slave-Typ (z.B. Schrittmotor) mehrmals in einem LIN-Cluster einzusetzen beziehungsweise einen LIN-Slave in verschiedenen LIN-Clustern einzusetzen, was die Wiederverwendbarkeit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



■ 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.

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 meh-

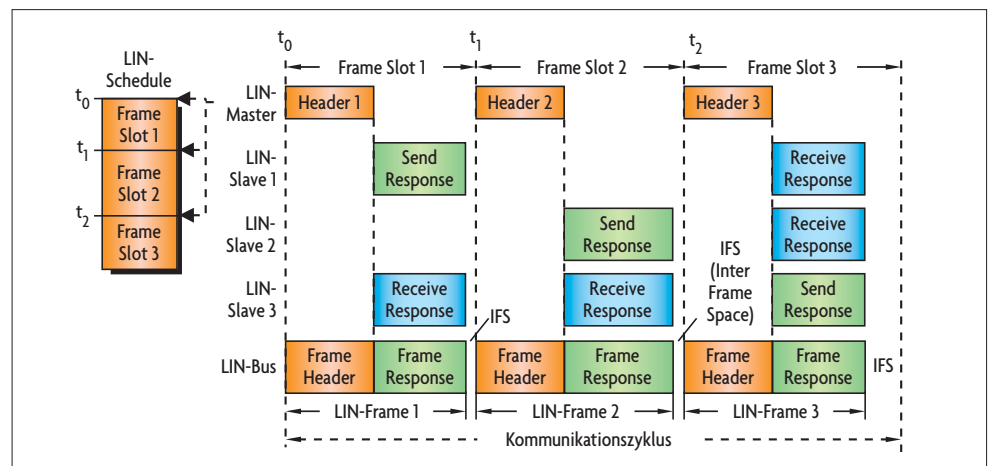
rerer Slave-Knoten (LIN-Slaves). Man verzichtet aus Kostengründen auf explizite Kommunikationscontroller und realisiert stattdessen die LIN-Kommunikation über Software-Tasks, die so genannten Slave-Tasks, in jedem Knoten. Der LIN-Master besitzt zusätzlich eine Master-Task, mit deren Hilfe er die

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 (Identifizier, ID), die alle LIN-Slaves über ihre Slave-Task auswerten. Ein LIN-Slave überträgt im Anschluss an den Frame-Header mit der ID assoziierte Frame Response. Der LIN-Frame, gebildet aus Frame-Header und Frame-Response, steht wegen der Nachrichtenadressierung mittels 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



■ 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.

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 Delegation 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.

■ 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.

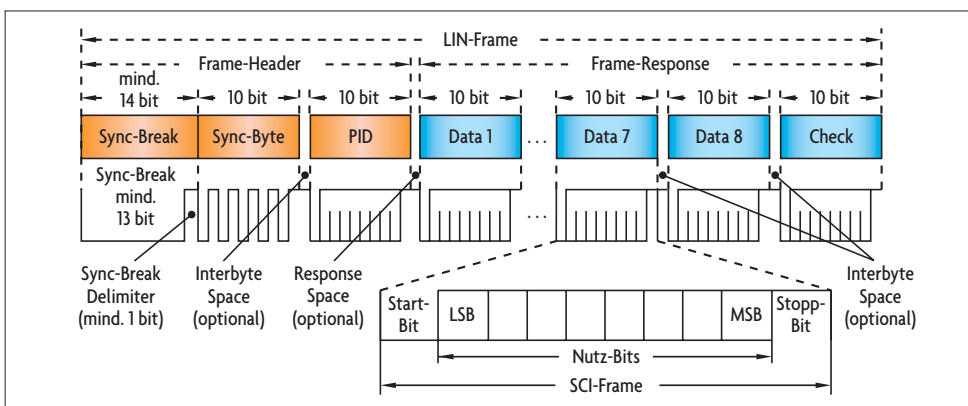
Der Event Triggered Frame wurde eingeführt, um sporadische Veränderungen oder Ereignisse auf Seiten der LIN-Slaves zu kommunizieren. Er entspricht prinzipiell einem Unconditional Frame, nur mit dem Unterschied, dass dem Frame-Header mehrere Frame-Responses von unterschiedlichen LIN-Slaves zugeordnet 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].

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 beispiels-

I 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.



weise 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

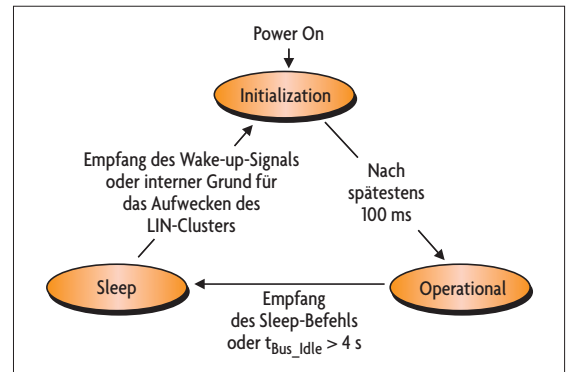
sowie Time Outs sind in der Spezifikation festgelegt.

■ 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 vertie-



fende 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.

Bild 5. Kommunikationszustandsmodell eines LIN-Slaves.



**Dipl.-Ing., Dipl.-Techpaed.
Eugen Mayer**

hat nach der Berufsausbildung zum Kommunikationselektroniker 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