Grundkonzepte und einfache Kfz-Bussysteme

2

In diesem Kapitel werden die elektrotechnischen Grundlagen dargestellt und einfache Kfz-Kommunikationsschnittstellen wie K-Line, SAE J1850 sowie einige Sensor-Aktor-Busse betrachtet.

2.1 Grundlagen

Zunächst werden die für die Anwendung im Kfz wichtigen Grundeigenschaften von Bussystemen vorgestellt. Dabei wird davon ausgegangen, dass der Leser mit den allgemeinen Grundlagen und Grundbegriffen von Datennetzen vertraut ist [1–3].

2.1.1 Elektrotechnische Grundlagen

Die Datenübertragung im Kfz erfolgt in der Regel bitweise seriell. Im einfachsten Fall sind zwei Geräte direkt miteinander verbunden (Abb. 2.1).

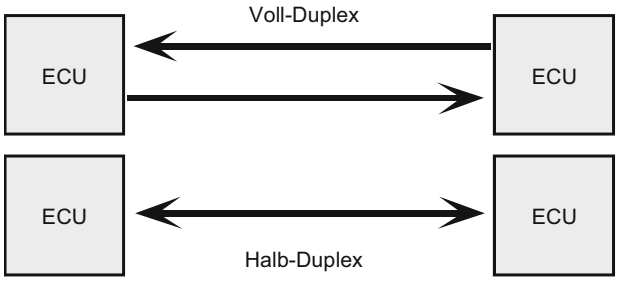

Abb. 2.1 Punkt-zu-Punkt-Verbindung

W. Zimmermann und R. Schmidgall, *Bussysteme in der Fahrzeugtechnik*, ATZ/MTZ-Fachbuch,
DOI 10.1007/978-3-658-02419-2_2,

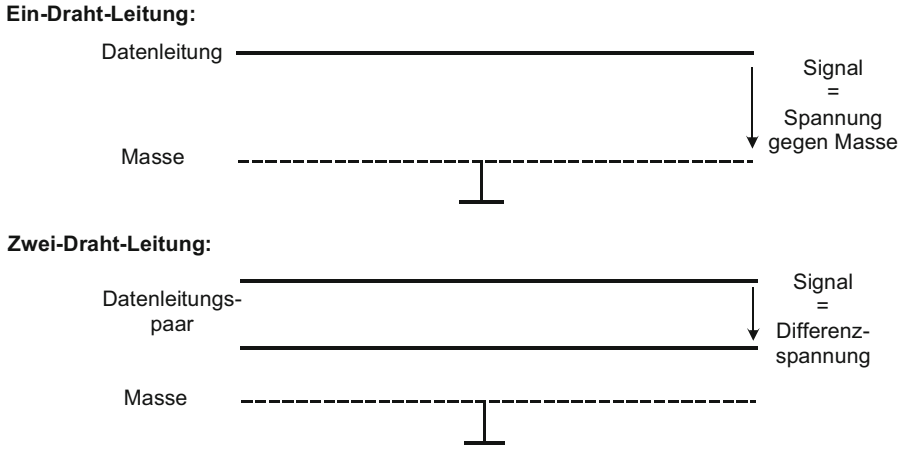

Abb. 2.2 Ein-Draht- und Zwei-Draht-Leitung

Abhängig davon, ob eine gemeinsame bidirektionale Leitungsverbindung oder ein Paar von jeweils unidirektionalen Leitungsverbindungen vorgesehen wird, ist eine Halb- oder Voll-Duplex-Verbindung möglich:

Voll-Duplex: Gleichzeitiges Senden und Empfangen ist möglich.
Halb-Duplex: Jedes Steuergerät kann nur abwechselnd senden oder empfangen. Standardverfahren bei Kfz-Bussystemen.

Die elektrischen Verbindungen können als Ein-Draht- oder als Zwei-Draht-Leitung ausgeführt werden (Abb. 2.2). Bei der kostengünstigen Ein-Draht-Leitung erfolgt die Signalrückführung über die Fahrzeugkarosserie (Masse). Derartige Verbindungen sind anfällig für die Ein- und Abstrahlung elektromagnetischer Störungen. Daher muss mit relativ großen Signalpegeln, im Kfz häufig mit Batteriespannungspegel, und niedrigen Bitraten gearbeitet werden. Üblich ist dies z. B. bei LIN oder ISO 9141 mit einer bidirektionalen Ein-Draht-Leitung (K-Line bei ISO 9141) als Halb-Duplex-Verbindung, seltener mit zwei unidirektionalen Ein-Draht-Leitungen als Voll-Duplex-Verbindung.

Zwei-Draht-Leitungen, deren Adern oft noch zusätzlich verdrillt werden (*Twisted Pair*) sind weniger anfällig für EMV-Probleme (Elektromagnetische Verträglichkeit). Dabei wird die Spannung zwischen den beiden Leitungen zur Signalübertragung verwendet (Differenzsignal). Daher kann mit kleineren Signalpegeln und höheren Bitraten gearbeitet werden. Im Kfz werden für Class B und C Netze, z. B. CAN oder FlexRay, üblicherweise Halb-Duplex-Zwei-Draht-Verbindungen eingesetzt. Abgeschirmte Zwei-Draht-Leitungen oder Paare von Zwei-Draht-Leitungen, also insgesamt vieradrige Verbindungen, wie sie heute bei Voll-Duplex-Ethernet-LANs üblich sind, werden im Kfz in der Regel aus Kostengründen nicht verwendet. Dasselbe gilt auch für optische Netze mit Lichtwellenleitern (LWL), die mit Ausnahme des Infotainment-Bussystems MOST im Kfz bisher nicht eingesetzt wer-

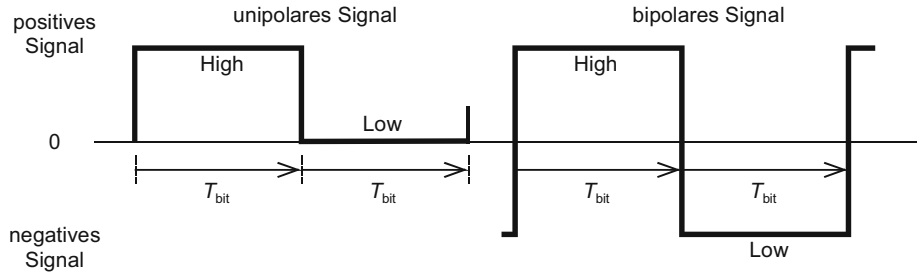

Abb. 2.3 Unipolare und bipolare Signale

den. Neben den teuren, in jedem Steuergerät notwendigen elektrooptischen Wandlern sind auch die Steckverbindungen für LWL fehleranfällig und die LWL selbst knickempfindlich, so dass die Verlegung zusammen mit den üblichen Kabelbäumen problematisch sein kann.

Bei den Ein-Draht-Verbindungen sind die Signale in der Regel *unipolar*, das Differenzsignal bei den Zwei-Draht-Verbindungen, z. B. CAN, meist *bipolar* (Abb. 2.3). Die digitale Übertragung erfolgt üblicherweise mit positiver Logik, d. h. die höhere Spannung (*High*) entspricht der *logischen 1*, die kleinere Spannung (*Low*) der *logischen 0* (*binäre* Signale). Verwendet man wie bei FlexRay einen dritten, zwischen *High* und *Low* liegenden Spannungspegel um z. B. den Ruhezustand des Bussystems abzubilden, spricht man von *ternären* Signalen.

Da die EMV-Störstrahlung umso größer ist, je häufiger Signalflanken auftreten, arbeitet man in der Regel mit Non-Return-to-Zero-(NRZ)-Codierung, d. h. das Signal bleibt für die gesamte Dauer eines Bits T_{bit} konstant auf Low oder High. Lediglich das veraltete SAE J1850 arbeitet mit pulsweitenmodulierten Signalen.

Der Wert

$$f_{bit} = \frac{1}{T_{bit}}$$

wird als *Bitrate* bezeichnet. Heute übliche Bitraten liegen im Bereich von 10 kbit/s (Class A-Netz, z. B. K-Line), d. h. *Bitdauer* $T_{bit} = 100$ µs, bis zu 500 kbit/s (Class C-Netz, z. B. CAN), d. h. $T_{bit} = 2$ µs. Bei Class D-Netzen (z. B. FlexRay) werden heute bis zu 10 Mbit/s, d. h. $T_{bit} = 100$ ns, verwendet und im Infotainmentbereich (z. B. MOST) sind schon heute 25 Mbit/s, d. h. $T_{bit} = 40$ ns, im Einsatz, zukünftig 150 Mbit/s.

Ebenfalls aus EMV-Gründen versucht man, die Steilheit (*Slew Rate*) der Signalflanken durch entsprechende Treiberschaltungen so langsam zu machen wie für eine gegebene Bitrate gerade noch zulässig.

Die Ausbreitung der Signale auf den Leitungen erfolgt mit Lichtgeschwindigkeit. Aufgrund der dielektrischen Eigenschaften des Kabelmaterials reduziert sich diese gegenüber der Lichtgeschwindigkeit im Vakuum etwa um den Faktor 2 … 3 auf ca.

$$c = 10 \ldots 15 \frac{\text{cm}}{\text{ns}} .$$

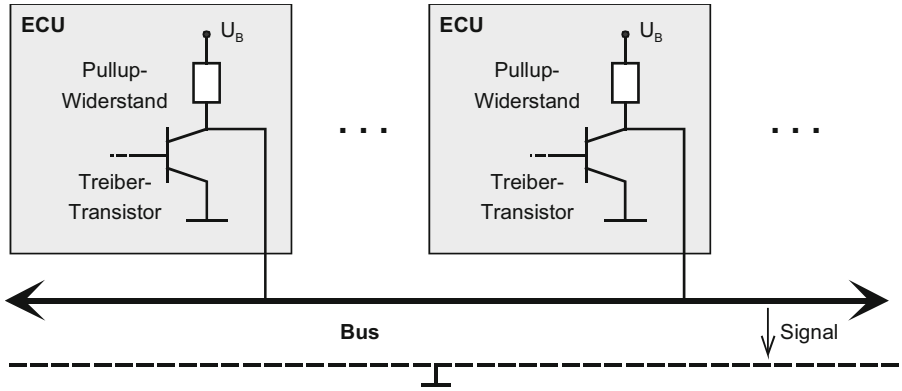

Abb. 2.4 Linien-Bus

Bei einer $l = 20$ m langen Verbindungsleitung, wie sie in einer LKW-Zugmaschine mit Auflieger oder einem Omnibus durchaus denkbar ist, beträgt die Signallaufzeit auf der Leitung in einfacher Richtung

$$T_d = \frac{I}{c} = 125 \ldots 200\text{ns} .$$

Zusätzliche Verzögerungen entstehen in den Leitungstreibern und -empfängern in den Steuergeräten. Für $T_d > 0{,}1 \cdot T_{bit}$ treten in der Praxis bereits deutliche Welleneffekte auf (Reflexionen an den Leitungsenden und an Stoßstellen wie Abzweigungen und Steckverbindern), welche die Übertragungssicherheit beeinträchtigen. Für Bussysteme mit höheren Bitraten müssen die Kabelführung, die Steckverbinder und die Kabelausführung sorgfältig gewählt werden. Zusätzlich müssen die Kabel elektrisch *abgeschlossen* werden, d. h. an den beiden Kabelenden sollten an das Kabel angepasste Abschlusswiderstände vorgesehen werden, um die genannten Welleneffekte zu vermeiden. Meist werden diese Abschlusswiderstände in diejenigen Steuergeräte eingebaut, die an den beiden Kabelenden angeschlossen werden, oder sie werden in das Kabel bzw. die Steckverbinder integriert. Steuergeräte, die bei Bussystemen nicht an den Leitungsenden sitzen sondern über Stichleitungen dazwischen angeschlossen sind (Abb. 2.4), dürfen keine Abschlusswiderstände enthalten.

Im Gegensatz zu einer Punkt-zu-Punkt-Verbindung sind bei einem echten Bus mehrere Steuergeräteausgänge an dieselbe Leitung angeschlossen. Die Signalverknüpfung erfolgt dabei nach dem Wired-OR-Grundprinzip, wie es im Abb. 2.5 für eine Ein-Draht-Busleitung dargestellt ist.

Wenn beide Steuergeräte ein *High*-Signal oder gar kein Signal senden wollen, sperren sie den jeweiligen Treibertransistor. Durch die so genannten *Pull-Up*-Widerstände wird die Busleitung dann auf den Pegel der Versorgungsspannung U_B gezogen. Sobald ein Steuergerät ein *Low*-Signal senden will, schaltet es den jeweiligen Treibertransistor durch. Da der leitende Transistor viel niederohmiger ist als die *Pull-Up*-Widerstände, wird die Busleitung damit praktisch kurzgeschlossen und das Signal *Low* gesendet. Wenn dagegen zwei Steuer-

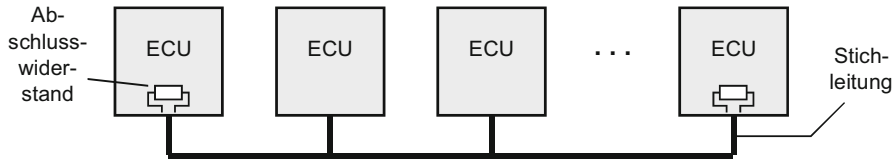

Abb. 2.5 Wired-OR-Prinzip

geräte gleichzeitig zu senden versuchen und ein Gerät *Low*, das andere Gerät *High* sendet, „gewinnt" dasjenige Steuergerät, das das *Low*-Signal auf den Bus legt. Man bezeichnet das *Low*-Signal in diesem Fall als dominant, das bei einer solchen Kollision „unterlegene" *High*-Signal als rezessiv. Um solche Kollisionen zu erkennen, muss jeder Sender die tatsächlich auf dem Bus liegenden Signalpegel überwachen und mit seinem Sendewunsch vergleichen.

2.1.2 Topologie und Kopplung von Bussystemen

Im Gegensatz zu einer Punkt-zu-Punkt-Verbindung können bei einem Datennetz mehrere Steuergeräte (*ECU Electronic Control Unit*) untereinander Daten austauschen. Für solche Datennetze existiert eine Reihe von unterschiedlichen Aufbauformen (Topologien), von denen die in der Praxis im Kfz am häufigsten zu findende Topologie der so genannte *Bus* ist. Aus diesem Grund wird der Begriff Bus häufig, aber nicht ganz präzise, als Synonym für alle Datennetze im Kfz verwendet.

Ein Bus, noch präziser ein *Linien-Bus*, entsteht, wenn mehrere Steuergeräte über kurze Stichleitungen an dieselben Verbindungsleitungen angeschlossen werden (Abb. 2.4). Durch ein Buszugriffsverfahren muss geregelt werden, welches Gerät zu welchem Zeitpunkt senden darf, um Kollisionen zu verhindern bzw. wie Kollisionen erkannt und aufgelöst werden.

Die gesendeten Daten können von allen anderen Steuergeräten empfangen werden. Wenn die gesendeten Daten nicht für alle angeschlossenen Geräte (*Broadcast*) bestimmt sind, muss bei Beginn der Datenübertragung oder mit jedem Datenpaket eine Adressierungsinformation übertragen werden, die einen einzelnen Empfänger (*Unicast*) oder mehrere Empfänger (*Multicast*) auswählt.

FlexRay unterstützt zwar auch Linien-Busse, bevorzugt aber Stern-Strukturen (Abb. 2.6). Ringe sind beim Infotainment-Bus MOST üblich.

Baum-ähnliche Netze findet man heute als Zusammenschaltung mehrerer Linien-Bussysteme. So kann man aus Sicht des Diagnosetesters das Fahrzeugnetz in Abb. 1.1 als Baum sehen, von dessen zentralem Knoten, dem Gateway, die Bussysteme für den Triebstrang, die Karosserie und die Infotainment-Kommunikation als Zweige abgehen, wobei der Karosserie-Bus weiter in die Türbusse verzweigt. An denjenigen Stellen, an denen Steuergerät und Bus bzw. mehrere Bussysteme zusammengeschaltet werden, muss die Koppelstelle bestimmte Umsetzungsaufgaben erledigen (Tab. 2.1).

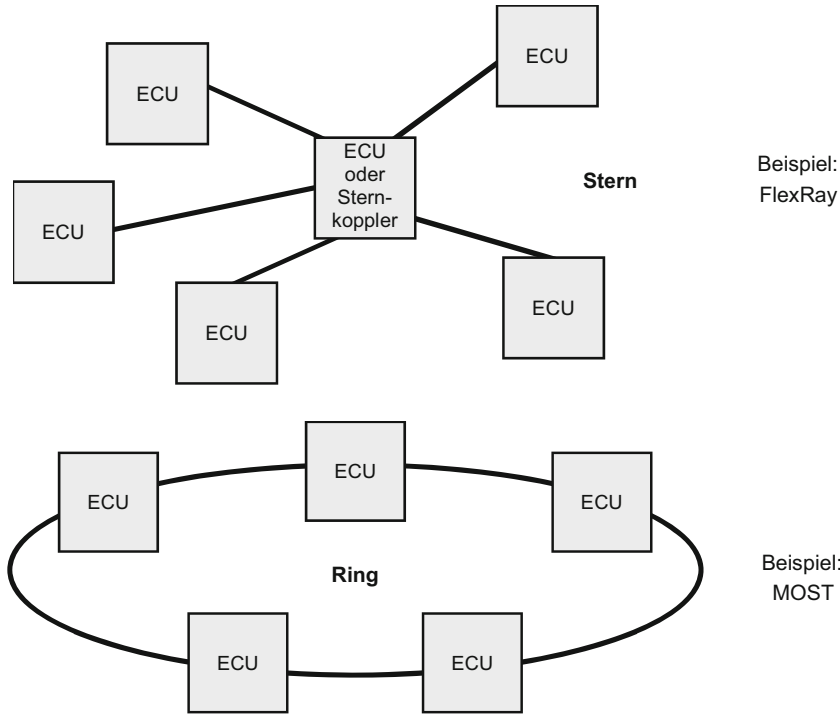

Abb. 2.6 Weitere Netz-Topologien

2.1.3 Botschaften, Protokollstapel, Dienste (Services)

Bei der Übertragung werden die Nutzdaten mit zusätzlichen Informationen versehen (Abb. 2.7). Vor den eigentlichen Nutzdaten (*Payload*) wird üblicherweise ein Vorspann (*Header*) übertragen, der Adressinformationen enthält (Sender und Empfänger der Daten), Angaben über die Anzahl der zu übertragenden Daten und gegebenenfalls zu deren Art. An die zu übertragenden Nutzdaten wird häufig ein Nachspann (*Trailer*) angehängt, der Informationen zur Fehlerprüfung und -korrektur enthält.

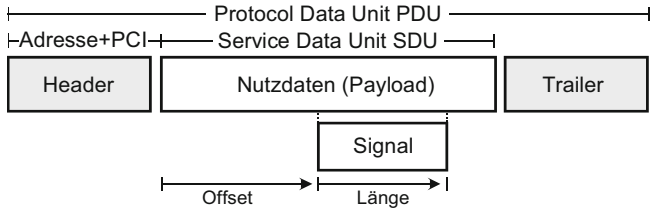

Abb. 2.7 Aufbau einer Botschaft

Tab. 2.1 Kopplung von Bussystemen

Bus-Koppler	Aufgabe
Transceiver (Abb. 2.11)	Umsetzung der Bussignale auf die Steuergeräte-internen Spannungspegel für die Sendesignale (Transmitter) und die Empfangssignale (Receiver), andere Bezeichnung *Bustreiber* oder *Leitungstreiber*, oft auch *physikalisches Businterface* (PHY). Findet sich in jedem Steuergerät.
Repeater	Signalverstärker zwischen Leitungsabschnitten ohne *Eigenintelligenz*. Erlaubt aus elektrotechnischer Sicht längere Leitungen, führt aber zu zusätzlichen Signallaufzeiten. Logisch bilden die beiden über einen Repeater gekoppelten Leitungsstücke einen einzigen Bus. Üblich im aktiven Sternpunkt von FlexRay; möglich bei CAN zur Kopplung von Zugmaschine und Anhänger, wird aber in der Regel als *Gateway* realisiert.
Gateway (Abb. 1.1)	Kopplung mehrerer Netze mit unterschiedlichen Protokollen, z. B. Umsetzung von CAN-Botschaften auf einen LIN-Bus, oder Bussen mit unterschiedlichen Adressbereichen und Bitraten, z. B. zwischen einem Triebstrang-CAN-Bus mit 500 kbit/s und 11 bit-Adressen (Identifier) und einem Karosserie-CAN-Bus mit 125 kbit/s und 29 bit-Adressen.
Switch, Hub, Router	Bei Kfz-Netzen nicht üblich. Router-Funktion oft im Gateway.

Bei einigen Protokollen werden auch zwischen die Nutzdaten nach bestimmten Vorschriften noch weitere Daten zur Übertragungssteuerung oder Trennung von Datenblöcken eingefügt, z. B. *Stuffing-Bits* oder ähnliches.

Im Bereich der Nutzdaten werden häufig mehrere, inhaltlich nicht notwendigerweise direkt zusammengehörige Datenwerte übertragen, z. B. die Motordrehzahl und die Kühlwassertemperatur. Wenn es sich bei diesen Daten um Messwerte handelt, spricht man von *Signalen*. Ein derartiges *Signal* wird durch seine Länge sowie seine Lage bezogen auf den Anfang des Nutzdatenbereichs (Offset in Bit oder Byte) und gegebenenfalls eine Umrechnungsvorschrift definiert, die den Zusammenhang zwischen dem eigentlichen physikalischen Wert, z. B. Drehzahl in 1/min, und dessen hexadezimaler Codierung beschreibt.

Gemäß dem OSI-Schichtenmodell durchlaufen die Daten von der Anwendung bis zur eigentlichen Übertragung über die Busleitungen mehrere Schichten (vgl. Tab. 1.3), die jeweils eigene Header und Trailer verwenden. Die Botschaft der nächsthöheren Ebene wird beim Senden auf der nächstniedrigeren Ebene als Payload verstanden und um die Header und Trailer der aktuellen Schicht ergänzt (Abb. 2.8). Beim Empfangen dagegen werden Header und Trailer der aktuellen Schicht entfernt, bevor die Payload dieser Schicht an die nächsthöhere Ebene weitergegeben wird. Innerhalb einer Schicht werden im Idealfall nur die zugehörigen Header und Trailer verarbeitet, der Inhalt der jeweiligen Payload dagegen ist für die Schicht selbst bedeutungslos. Auf diese Weise entsteht ein *Protokollstapel*. Der Header einer Schicht wird häufig als *Protocol Control Information PCI* bezeichnet, weil er das für diese Schicht spezifische Protokoll beschreibt, die Nutzdaten als *Service Data Unit SDU* und der gesamte Datenblock als *Protocol Data Unit PDU*.

In der Regel ist die Länge einer Botschaft auf dem Bussystem begrenzt. So erlaubt z. B. CAN nur maximal 8 Byte, d. h. 64 bit Nutzdaten in einer Botschaft. In diesem Fall wird eine

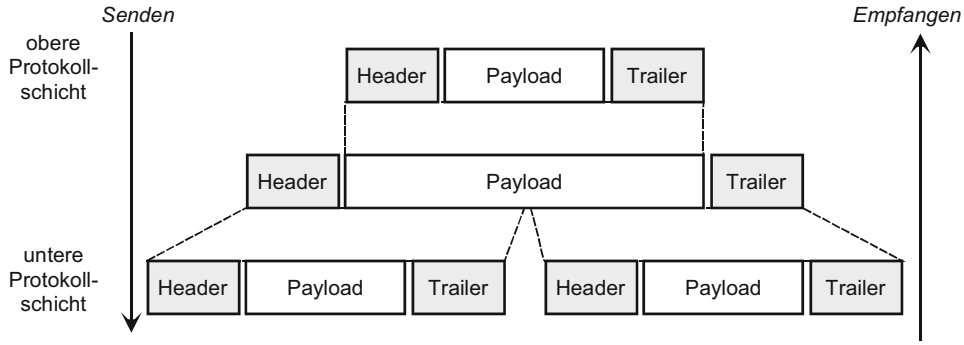

Abb. 2.8 Protokollstapel (*Protocol Stack*)

längere Botschaft der oberen Schicht beim Senden in mehrere Botschaften aufgeteilt (*Segmentierung*) und beim Empfangen wieder aus mehreren Botschaften der unteren Schicht zusammengesetzt (*Desegmentierung*). Der umgekehrte Fall, dass eine untere Ebene längere Botschaften erlaubt als eine höhere Ebene und daher mehrere Botschaften beim Senden zu einer Botschaft zusammengefasst werden, ist bei Kfz-Bussen derzeit noch selten, kommt aber z. B. bei FlexRay in Betracht.

Bei sauber spezifizierten Protokollen besitzt jede Schicht des Protokollstapels eine exakt definierte Reihe von Funktionen, so genannte *Dienste* (*Services*). Die übergeordnete Protokollschicht verwendet diese Dienste, um Botschaften an die darunter liegende Schicht zu senden, von dieser zu empfangen oder Konfigurations- und Statusinformationen mit dieser Schicht auszutauschen.

Der Begriff *Botschaft* (engl.: *Message*) ist dabei nicht eindeutig. Je nach Ebene in einem Protokoll und in unterschiedlichen Standards mit unterschiedlicher Bedeutung gebraucht, finden sich auch die Begriffe *Datenblock, Paket* und *Rahmen* (*Frame*).

2.1.4 Kommunikationsmodelle, Adressierung

Der Datenaustausch zwischen zwei Steuergeräten über einen Protokollstapel vermittelt den Anwendungen im Idealfall die Illusion, ohne zwischengeschaltetes Bussystem direkt miteinander zu kommunizieren (Abb. 2.9). Der Protokollstapel kapselt die Details der Kommunikation, wobei auch innerhalb des Protokollstapels jede Ebene den Eindruck haben sollte, direkt mit der entsprechenden Ebene des anderen Kommunikationspartners zu interagieren.

In der Regel wird die Kommunikation zwischen den Steuergeräten im Fahrzeug während der Entwicklungsphase festgelegt, da die verbauten Geräte und die Struktur der auszutauschenden Informationen vorab bekannt sind und damit keine Laufzeitkonfiguration mehr nötig ist. Bei manchen Protokollen ist trotzdem am Beginn der Kommunikation eine gegenseitige Verständigung vorgesehen, ähnlich der Wahl der Telefonnummer und

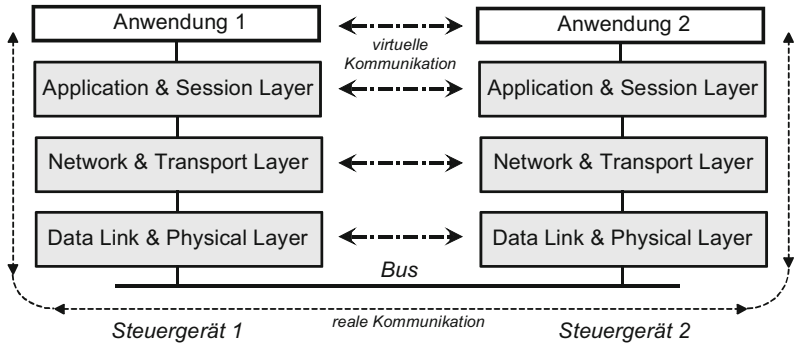

Abb. 2.9 Kommunikation zwischen Anwendungen über einen Protokollstapel

dem folgenden Verbindungsaufbau bei Telefongesprächen. Derartige Protokolle werden als *verbindungsorientiert* bezeichnet und bestehen üblicherweise aus den drei Phasen *Verbindungsaufbau*, *Datenaustausch* (Kommunikationsphase) und *Verbindungsabbau*. Können Geräte dagegen spontan ohne vorherige Absprache kommunizieren, bezeichnet man dies als *verbindungslose* Kommunikation.

Bei Kfz-Bussystemen werden Steuer- und Regeldaten praktisch immer im *Broadcast*-Verfahren versendet (Abb. 2.10). Das Steuergerät sendet seine Botschaften unaufgefordert. Jeder Empfänger entscheidet, ob er die Botschaften ignoriert oder den Dateninhalt auswertet (*Empfangs- oder Akzeptanzfilterung*). In der Regel besitzen die Protokolle zusätzlich *Adressierungsmechanismen*, mit denen ein einzelner (*Unicast*) oder mehrere Empfänger (*Multicast*) gezielt angesprochen werden können.

Für die Adressierung von Botschaften gibt es zwei gängige Verfahren:

- Gerätebasierte Adressierung (Stations- oder Teilnehmeradressierung)
- Inhaltsbasierte Adressierung (Botschafts-Kennung, Identifier).

Bei der *gerätebasierten Adressierung* (Stations- oder Teilnehmeradressierung) wird jedes Steuergerät durch eine eindeutige Adresse (Nummer) identifiziert, wobei jede Botschaft neben der *Zieladresse*, d. h. der Adresse des Empfängers, in der Regel auch die *Quelladresse*, d. h. die Adresse des Senders enthält. Für die Aussendung von Botschaften an mehrere

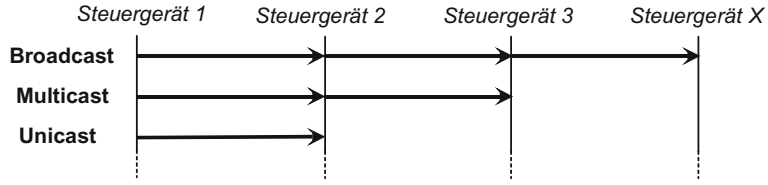

Abb. 2.10 Botschaftsversand an alle, eine Gruppe oder einen einzelnen Empfänger

(*Multicast*) oder alle Empfänger (*Broadcast*) existieren spezielle Adressen, die gelegentlich als funktionale Adressen bezeichnet werden. Da die Anzahl der Steuergeräte in einem Fahrzeug relativ klein ist, genügen für Adressen einige Bit, typ. 5…8. Werden mehrere Busse über ein Gateway verknüpft, wird in den höheren Protokollschichten meist eine für das Gesamtsystem eindeutige Geräteadresse verwendet, die vom Gateway in einer unteren Protokollschicht in eine nur für das adressierte Bussystem eindeutige Geräteadresse umgesetzt wird.

Anstelle der gerätebasierten Adressierung kann auch eine *inhaltsbasierte Adressierung* verwendet werden, wie z. B. bei CAN. Dabei wird eine Botschaft mit einer bestimmten, adressähnlichen Kennung (Identifier) versehen, die den Inhalt der Botschaft kennzeichnet, also anzeigt, dass eine Botschaft beispielsweise Informationen über die Motordrehzahl enthält. Da in heutigen Fahrzeugen sehr viele solcher Informationen über die Bussysteme gesendet werden, benötigt man hierfür längere *Adressen*, bei CAN z. B. sind alternativ 11 bit oder 29 bit Kennungen üblich. Die inhaltsbasierte Adressierung führt zu einer geringeren Busbelastung und zu einer besseren Datenkonsistenz im gesamten Netzwerk, wenn Daten häufig an mehrere Teilnehmer versendet werden müssen.

Sowohl bei gerätebasierter als auch bei inhaltsbasierter Adressierung sehen bei einem Bussystem alle angeschlossenen Steuergeräte jede Botschaft und müssen auf Basis der Adresse entscheiden, ob sie die Botschaft tatsächlich empfangen und weiterverarbeiten oder einfach ignorieren wollen (*Akzeptanzfilterung*). Bei Bussystemen mit niedriger Bitrate findet die Akzeptanzprüfung in der Regel in der Protokollsoftware statt. Bei Protokollen mit höheren Bitraten findet die Protokollverarbeitung für die Schichten 1 und 2 und damit nach Möglichkeit auch die Akzeptanzfilterung dagegen im Kommunikationscontroller statt, einem speziellen IC oder einem auf dem Mikrocontroller des Steuergerätes integrierten Schaltungsteil, häufig auch als *Media Access Controller* (MAC) bezeichnet (Abb. 2.11). Bei der gerätebasierten Adressierung ist die Akzeptanzfilterung verhältnismäßig einfach, da nur die eigene Zieladresse sowie ggf. die Multicast- und Broadcast-Adressen erkannt werden müssen, während bei der inhaltsbasierten Adressierung oft mehrere Dutzend oder gar Hunderte *Adressen* (Botschaftskennungen) erkannt werden müssen. Gängige Kommunikationscontroller können typischerweise so programmiert werden, dass sie die Kennung von 8 bis 16 Botschaften hardwaremäßig erkennen, wobei mit Hilfe von Bitmasken auch ganze Bereiche von Adressen festgelegt werden können. Für die übrigen Botschaftskennungen, die ebenfalls empfangen werden sollen, muss eine Multicast-Kennung eingerichtet werden, für die die Akzeptanzfilterung dann softwaremäßig erfolgt.

Während der Sender bei *Broadcast*-Kommunikation in der Regel keine Bestätigung der Empfänger erhält (*Unacknowledged Communication*), erwartet der Sender bei *Unicast*-Sendungen oft eine positive Bestätigung (*Acknowledged Communication*) des Empfängers, dass die Botschaft korrekt empfangen wurde (Abb. 2.12). Verzichtet man, um das Bussystem zu entlasten, auf die positive Bestätigung, so wird zumindest eine Fehlermeldung (*Not Acknowledged*) erwartet, wenn einer der Empfänger einen Übertragungsfehler erkannt hat. In der Regel versendet der Sender die Botschaft darauf erneut. Ein Kommunikationsprotokoll, bei dem sich die Anwendung darauf verlassen kann, dass der *Network and*

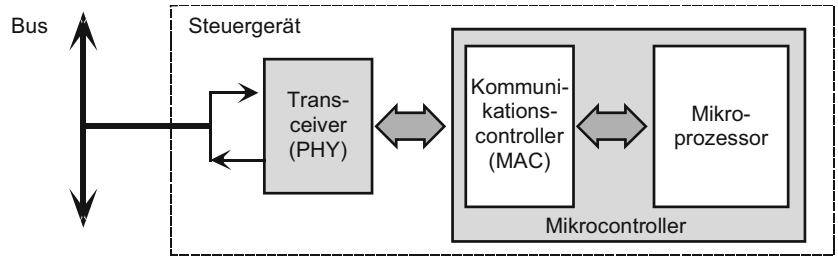

Abb. 2.11 Typisches Bus-Interface eines Steuergerätes (PHY Physical Interface, MAC Media Access Control)

Transport bzw. *Data Link Layer* selbstständig gewährleisten, dass die Information beim Empfänger korrekt ankommt, bezeichnet man als *gesicherte* Übertragung, das Gegenteil als *ungesichert*.

Das *Broadcast*-Verfahren, bei dem der Sender „ungefragt" Informationen versendet, ähnelt der Ausstrahlung von Rundfunk- und Fernsehprogrammen. Im Gegensatz dazu fordert ein Gerät beim *Request-Response*-Verfahren (Abb. 2.13) ein anderes Gerät durch eine Anfrage-Botschaft (*Request*) gezielt auf, Informationen (*Response*) zu liefern. Dieses Kommunikationsmodell wird unter dem Namen *Client-Server*-Verfahren auch im Internet verwendet, bei dem ein Web-Browser (*Client*) eine Anfrage an einen Web-*Server* stellt und von dort eine Antwort, z. B. eine HTML-Seite, erhält. Wird die Anfrage mit *Multicast*-Adressierung gesendet, muss der Sender mit Antworten von mehreren Empfängern rechnen. Benötigt man dieselbe Information wiederholt, entlastet das *Publisher-Subscriber*-Verfahren das Bussystem. Dabei meldet der an der Information interessierte Partner sein Interesse an der Information einmalig an (*Subscribe*), worauf der andere Partner die Information dann ohne weitere Anfragen regelmäßig (*Periodic*) oder bei Änderungen der Daten (*On Event*) liefert.

Ist die Informationsmenge größer als die Nutzdatenlänge einer einzelnen Botschaft, muss die Information vor der Übertragung auf der Senderseite zerstückelt (*Segmentierung*) und auf der Empfängerseite wieder zusammengesetzt (*Desegmentierung*) werden. Dies erfolgt in der Regel im *Network and Transport Layer* des Protokollstapels, dem sogenannten *Transportprotokoll* (siehe Kap. 4). Um den Empfänger vor zu großen Datenmengen oder zu schnell aufeinanderfolgenden Botschaften zu schützen, verfügen diese Protokolle in der

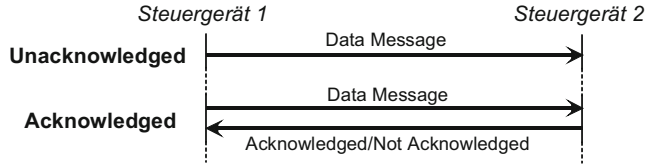

Abb. 2.12 Unbestätigte und bestätigte Kommunikation

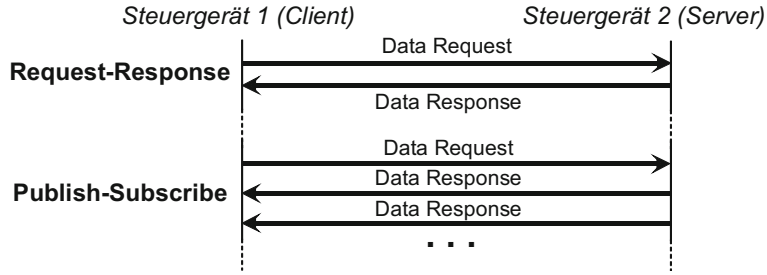

Abb. 2.13 Abfrage von Informationen

Regel auch über eine *Flusssteuerung*, bei der der Empfänger dem Sender mitteilt, wie viele Botschaften in welchem Abstand er empfangen bzw. den Sender zu einer Sendepause veranlassen kann.

Die Schnittstelle zwischen Anwendung und Protokollstapel bzw. den verschiedenen Schichten eines Protokollstapels verwendet üblicherweise die in Abb. 2.14 dargestellte Ablaufreihenfolge. Die Anwendung beauftragt den Protokollstapel mit dem Versenden einer Botschaft (*Transmit Request*) und lässt sich optional die durchgeführte Übertragung bestätigen (*Transmit Confirmation*). Danach wartet die Anwendung im einfachsten Fall, bis die Antwort eintrifft (*synchroner* oder *blockierender Betrieb*). Falls die Anwendung dagegen weiterarbeitet (*asynchroner* oder *nicht-blockierender* Betrieb), muss der Protokollstapel die Anwendung informieren, wenn eine Antwort eingetroffen ist. Diese Rückmeldung kann durch Setzen eines Flags erfolgen, das die Anwendung von Zeit zu Zeit abfragt (*Polling*), oder indem der Protokollstack eine Interrupt- oder Rückruffunktion der Anwendung aufruft (*Receive Indication*, *Notification* oder *Callback*).

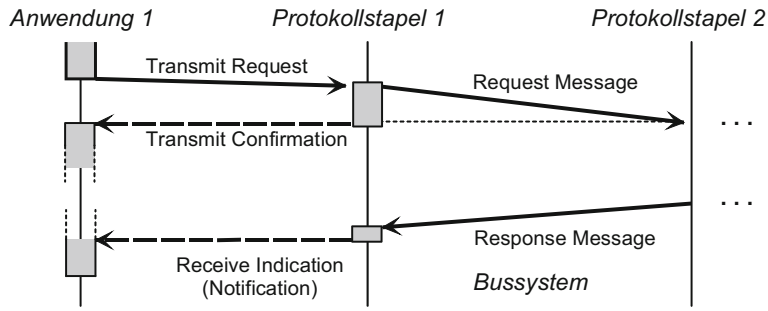

Abb. 2.14 Ablaufreihenfolge beim Senden und Empfangen von Botschaften

2.1.5 Zeichen- und Bitstrom-basierte Übertragung, Nutzdatenrate

Da bei den Bussystemen die Bitrate f_{bit}, d. h. die Geschwindigkeit, mit der die einzelnen Bits über die Busleitungen übertragen werden, durch die elektrotechnischen Randbedingungen vorgegeben ist, und der Header und der Trailer ebenfalls Übertragungszeit benötigen (Protokoll-Overhead), reduziert sich die Nutzdatenrate f_{Daten} mindestens im Verhältnis

$$f_{\text{Daten}} = f_{\text{bit}} \cdot \frac{\text{Anzahl der Nutzdatenbits}}{\text{Anzahl der Nutzdatenbits} + \text{Anzahl der Steuerbits}}$$

mit *Anzahl der Steuerbits = Anzahl der Bits im Header + Trailer + Sonstige.*

Für die eigentliche Datenübertragung über die Busleitungen werden zwei verschiedene Verfahren eingesetzt:

- Zeichen-basierte Übertragung, üblich bei niedrigen Bitraten, z. B. K-Line, LIN,
- Bitstrom-basierte Übertragung, üblich bei hohen Bitraten, z. B. CAN, FlexRay.

Bei der *Zeichen-basierten Übertragung* (andere Bezeichnungen: *asynchrone Übertragung*, Start-Stopp-Verfahren, UART-basierte Übertragung) werden die zu übertragenden Daten einer Botschaft einschließlich Header und Trailer in Gruppen von je 8 bit (ein Zeichen) unterteilt und jedes Zeichen einzeln übertragen (Abb. 2.15). Zwischen den Zeichen befindet sich die Übertragungsleitung im Ruhezustand (*logisch 1*). Als Kennzeichnung für den Beginn eines Zeichens wird ein Startbit (*logisch 0*) übertragen, anschließend die 8 bit des Zeichens beginnend mit dem niederwertigsten Bit (LSB). Danach wird eventuell noch ein Paritätsbit gesendet, mit dem einfache Übertragungsfehler erkannt werden können. Anschließend wird ein Stoppbit (*logisch 1*) gesendet. Bis zum Beginn des nächsten Zeichens bleibt die Leitung wieder im Ruhezustand. Vorteil dieses Verfahrens ist, dass der Kommunikationscontroller sehr einfach ist und als so genannter *UART* (Universal Asynchronous Receiver and Transmitter) auf praktisch jedem Mikrocontroller bereits integriert ist. Der UART wickelt das Senden und Empfangen eines Zeichens inklusive der Erzeugung von Start- und Stoppbit sowie Paritätsbit und dessen Überprüfung selbstständig ab. Die Bereitstellung und Weiterverarbeitung jedes Zeichens dagegen muss durch die Protokollsoftware erfolgen, so dass das Verfahren zu einer relativ hohen Rechnerbelastung führt bzw. in der Praxis auf niedrige Datenraten beschränkt ist. Die Bitrate ist konstant und wird durch den UART erzeugt. Der zeitliche Abstand zwischen den einzelnen Zeichen T_{ICB} (*Inter Character Break*) sowie der Abstand zwischen dem letzten Zeichen einer Botschaft und dem ersten Zeichen der nächsten Botschaft T_{IFB} (*Inter Frame Break*) wird in den Protokollen typischerweise durch Minimal- und Maximalwerte vorgegeben, um der Protokollsoftware ausreichend Zeit für das Bereitstellen und Verarbeiten der Daten zu geben. Der Overhead der Übertragung ist hoch. Selbst wenn alle Zeichen ohne Pause übermittelt werden könnten, lässt sich wegen der für jeweils 8 Datenbits zusätzlich nötigen Start- und Stoppbits im günstigsten Fall (ohne Paritätsbit und Pausen) eine Nutzdatenrate

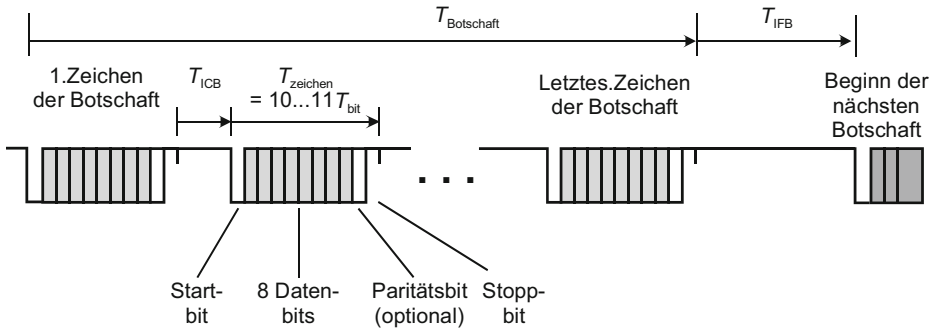

Abb. 2.15 Zeichen-basierte Übertragung (ICB … Inter Character Break, IFB … Inter Frame Break)

von maximal

$$f_{\text{Daten}} = f_{\text{bit}} \cdot \frac{8}{10}$$

erreichen. Durch die Pausenzeiten zwischen den Zeichen und Botschaften wird die Nutzdatenrate nochmals verringert.

Bei der *Bitstrom-basierten Übertragung* (Abb. 2.16, andere Bezeichnung: *synchrone Übertragung*) werden alle zu einer Botschaft gehörenden Daten inklusive Header und Trailer ohne Pause als ein zusammengehörender Block (Frame) im Bittakt übertragen (so genannter Bitstrom). Die Pause zwischen den Frames T_{IFB} (*Inter Frame Break*) wird im Protokoll wieder mit Minimal- und Maximalwerten vorgegeben. Gegenüber der zeichenbasierten Übertragung ergibt sich durch die innerhalb der Botschaft pausenfreie Übertragung eine deutlich höhere Nutzdatenrate. Allerdings ist auch ein wesentlich aufwendigerer Kommunikationscontroller notwendig, der die gesamte Übertragung in Hardware abwickelt, so dass die Protokollsoftware lediglich die Botschaft zur Verfügung stellen bzw. weiterverarbeiten muss.

Sender und Empfänger arbeiten in der Regel mit jeweils einem eigenen lokalen Taktgenerator für den Bittakt. Obwohl Sender und Empfänger stets mit derselben nominalen Bitrate arbeiten, sind kleinere Toleranzen in der Praxis nicht vermeidbar.

Bei der Zeichen-basierten Datenübertragung wird der Bittaktgenerator des Empfängers durch die Signalflanke zu Beginn des empfangenen Startbits synchronisiert. Die bis zum

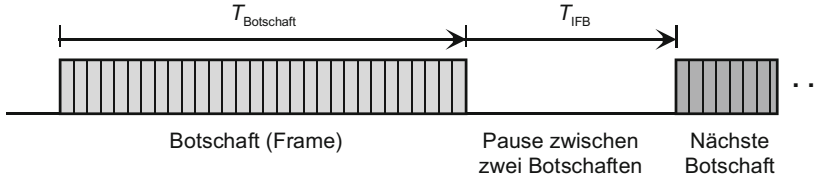

Abb. 2.16 Bitstrom-basierte Übertragung (IFB … Inter Frame Break)

Ende eines Zeichens aufsummierten Toleranzen der Bitzeiten müssen deutlich kleiner sein als eine Bitdauer, sonst werden die Bits am Zeichenende nicht mehr korrekt empfangen. Da ein Zeichen aus relativ wenig Bits besteht, typ. 10 bit (1 Start-, 8 Daten-, 1 Stopp-Bit), sind verhältnismäßig große Toleranzen zulässig, z. B. bei einem aufsummierten, zulässigen Fehler von 0.5 T_{bit} bis zu ± 5 %.

Bei der Bitstrom-basierten Zeichenübertragung mit Botschaften von zum Teil weit über 100 bit würden sich die erforderlichen Toleranzen von weniger als ein Zehntel der Toleranzen der Zeichen-basierten Übertragung praktisch nicht mehr realisieren lassen. Aus diesem Grund muss der Bittaktgenerator während des Empfangs einer Botschaft nachsynchronisiert werden. Während bei Büro-LANs dazu spezielle Bit-Codierverfahren verwendet werden (z. B. Manchester-Codierung), die eine höhere Übertragungsbandbreite erfordern und größere EMV-Probleme verursachen, müssen bei der in Kfz-Bussystemen üblichen NRZ-Codierung zusätzliche Synchronisationsbits in den Datenstrom eingefügt werden. Ein Beispiel dafür ist das so genannte *Bit-Stuffing*, bei dem nach einer bestimmten Anzahl von Bits einer Botschaft (bei CAN z. B. 5 bit) immer dann ein zusätzliches Stuff-Bit eingefügt wird, wenn zu viele aufeinanderfolgende Datenbits alle *logisch 0* oder *logisch 1* sind und so im übertragenen Signal keine Signalflanke zur Nachsynchronisation des Bittaktgenerators vorhanden wäre. Bit-Stuffing vergrößert die Länge einer Botschaft abhängig von den Datenbits, bei CAN z. B. im ungünstigsten Fall um bis zu 25 %.

2.1.6 Buszugriffsverfahren, Fehlererkennung und Fehlerkorrektur

Wie in Abschn. 2.1.1 bereits dargestellt, kommt es zu einer Kollision, wenn mehrere Steuergeräte gleichzeitig versuchen, über den Bus Daten zu senden. Alle verbreiteten Bussysteme müssen daher eine Strategie für den Buszugriff einsetzen, mit der Kollisionen erkannt bzw. vermieden werden. Gemeinsamer Grundsatz bei allen Kfz-Bussystemen ist dabei die Nicht-Unterbrechbarkeit laufender Übertragungen. Die gerade auf dem Bus übertragene Botschaft wird in jedem Fall vollständig übertragen. Jeder Sender muss überprüfen, ob der Bus frei ist, bevor er mit einem Sendeversuch beginnt. Welches Steuergerät senden darf, wird mit einem der in Tab. 2.2 dargestellten Zugriffsverfahren bestimmt.

Alle Protokolle verwenden Verfahren zur Fehlererkennung und teilweise auch Fehlerkorrektur auf verschiedenen Ebenen des Protokollstapels. Bei der Übertragung werden die gesendeten Datenbits praktisch immer mit den tatsächlichen Datenbits auf den Busleitungen verglichen, um Kollisionen und Übertragungsfehler zu erkennen. Bei der Zeichenübertragung können die einzelnen Zeichen durch Paritätsprüfung, ganze Botschaften durch Prüfsummen, in der Regel *Cyclic Redundancy Check CRC*, gesichert werden. Neben der Korrektheit der Daten werden auch die Zeitbedingungen des Übertragungsprotokolls überwacht (*Timeout*). In der Regel erhält der Sender eine Quittung für die fehlerfreie Übertragung vom Empfänger. Dies kann, wie bei CAN, durch ein unmittelbar am Ende der Botschaft gesendetes Quittungsbit, eine spezielle Bestätigungsbotschaft (*Acknowledge ACK*) oder im Rahmen der regulären Antwortbotschaft (*Response*) des Empfängers gesche-

Tab. 2.2 Buszugriffsverfahren

Master-Slave	Es gibt genau ein Steuergerät (Master), das eine Datenübertragung beginnen darf. Dieses Steuergerät fragt die anderen Steuergeräte (Slaves) regelmäßig oder bei Bedarf ab. Slaves dürfen Daten nur als Antwort auf eine an sie gerichtete Abfrage senden. Sie müssen selbst dann auf die Anfrage durch den Master warten, wenn sie sehr dringende Informationen zu senden haben. Untereinander können die Slaves nicht direkt kommunizieren. Wenn der Master ausfällt, ist entweder gar keine Kommunikation mehr möglich oder es muss ein Verfahren vorhanden sein, um aus den Slaves einen neuen Master zu bestimmen. Das Verfahren wird z. B. beim LIN-Bus verwendet.
Asynchron (Event Based)	Jedes Steuergerät kann zu beliebigen Zeiten auf den freien Bus zugreifen. Wenn mehrere Steuergeräte zeitgleich den Bus belegen, werden die elektrischen Signale auf dem Bus verfälscht. Da jedes Steuergerät zur Fehlererkennung ohnehin die Signale auf dem Bus überwacht, wird die Kollision erkannt und die Steuergeräte stellen die Übertragung sofort ein und versuchen es zu einem späteren Zeitpunkt nochmals.
CSMA/CD Carrier Sense Multiple Access Collision Detect	Mit diesem relativ einfach zu implementierenden Verfahren lässt sich bei zeitlich statistisch verteilten Botschaften im Mittel der beste Datendurchsatz erzielen. Aus diesem Grund wurde das **CSMA/CD**-Verfahren ursprünglich bei dem im Bürobereich dominierenden Ethernet-LAN verwendet. Das Verfahren ist allerdings nicht deterministisch. Da alle Botschaften gleichberechtigt sind und mit jeder anderen Botschaft kollidieren können, kann für keine Botschaft garantiert werden, wie lange es höchstens dauert, bis sie übertragen wird. Bei sehr hoher Busbelastung kann es infolge ständiger Kollisionen sogar dazu kommen, dass keine Daten mehr übertragen werden können.
CSMA/CR Carrier Sense Multiple Access Collision Resolution	Für die Anwendung im Kfz wird dieses Verfahren daher z. B. bei CAN so modifiziert, dass bei Kollisionen derjenige Sender „gewinnt“ (*Arbitrierung*), dessen Botschaft die höhere Priorität hat. Ein Sender mit einer Botschaft, die eine niedrigere Priorität hat, stellt das Senden sofort ein. Die Botschaft mit der höheren Priorität wird ohne Kollision weiter übertragen (**CSMA/CR**). Die Priorität einer Botschaft wird im Header der Botschaft codiert. Bei diesem Verfahren ist die Übertragung zumindest für Botschaften mit hoher Priorität deterministisch.
Token-Passing	Die Sendeberechtigung (Token) läuft reihum. Nach einem bestimmten Verfahren gibt ein Steuergerät, wenn seine Sendezeit abgelaufen ist oder wenn es keine weiteren Daten zu senden hat, die Sendeberechtigung an ein anderes Gerät ab. Da das Weitergeben des Tokens überwacht werden sollte, ist das Verfahren relativ aufwendig und wird daher im Kfz nur bei den asynchronen Kanälen von MOST verwendet.

Tab. 2.2 (Fortsetzung)

Zeitsynchron (Time Based)	Jedes Steuergerät erhält ein bestimmtes periodisches Zeitfenster, in dem dieses und nur dieses Steuergerät senden darf. Wenn bestimmte Geräte mehr oder wichtigere Daten zu senden haben als andere, erhalten sie entweder längere Zeitfenster oder mehrere Fenster in kürzeren Zeitabständen. In Kfz-Anwendungen wird die Zuordnung der Zeitfenster in der Regel in der Entwicklungsphase statisch konfiguriert. Damit das Verfahren funktioniert, müssen alle Steuergeräte mit einer gemeinsamen Zeitbasis arbeiten. Im Gegensatz zu CSMA ist das Zeitverhalten der Übertragung streng deterministisch, d. h. es kann für jede Botschaft exakt vorhergesagt werden, wie lange es im ungünstigsten Fall dauert, bis sie übertragen wird. Wenn ein Steuergerät gerade keine Daten zu übertragen hat, bleibt sein Zeitfenster frei. Das Verfahren eignet sich besonders gut für periodische Übertragungen wie z. B. bei Messwerten und Regelkreisen, aber nur schlecht für seltene, spontane, aber eventuell sehr dringende Botschaften wie z. B. Fehlermeldungen, oder seltene, aber sehr lange Datenübertragungen, wie sie z. B. beim *Flashen* notwendig sind. Nach diesem Verfahren arbeiten z. B. FlexRay oder TTCAN, sehen aber aus den erwähnten Gründen ein spezielles Zeitfenster vor, in dem die Steuergeräte auch asynchron auf den Bus zugreifen können.
TDMA Time Division Multiple Access	

hen. Im Fehlerfall wiederholt der Sender meist die Übertragung. Der Empfänger verwirft die fehlerhaft empfangenen Daten.

Steuergeräte, die auch bei ausgeschalteter Zündung noch funktionsfähig sein müssen, z. B. die Steuerung für die Innenraumbeleuchtung, werden in der Regel in Ruheperioden in einen Stromsparmodus (*Stand By, Sleep*) versetzt, um die Batteriestandzeit zu verlängern. Durch ein externes Signal, z. B. den Türkontakt, werden sie wieder *aufgeweckt* und können dann andere Steuergeräte über spezielle oder beliebige Busbotschaften (*Wake Up*) ebenfalls wieder in den Normalbetriebszustand versetzen. Dies ist Aufgabe des sogenannten Netzmanagements, siehe Kap. 7.

2.1.7 Jitter und Latenz bei der Datenübertragung

Bei den meisten Anwendungen im Kfz ist eine zuverlässige Übertragung relativ kleiner Datenmengen unter Echtzeitaspekten (*Real Time*) notwendig. Im Gegensatz zu Büro-LANs steht nicht so sehr der absolute Datendurchsatz im Vordergrund, sondern die Frage, wie zuverlässig die Zeitbedingungen der Übertragung eingehalten werden:

- Bei periodisch übertragenen Daten, wie sie bei Regelungen üblich sind, muss die Periodendauer der Übertragung nicht nur für die gegebene Aufgabenstellung genügend klein sein, die tatsächliche Periodendauer der einzelnen Übertragungen darf auch nur bis zu einem vorgegebenen Maß schwanken (*Übertragungs-Jitter*).

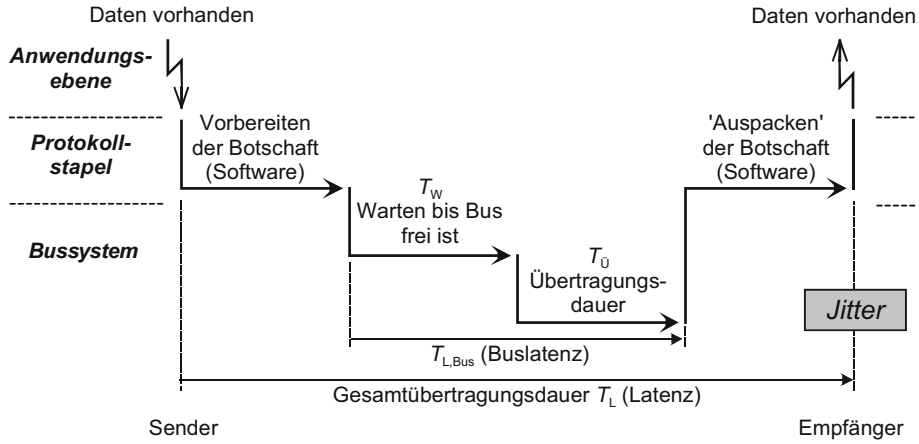

Abb. 2.17 Verzögerungen bei der Datenübertragung

- Bei nicht-periodisch zu übertragenden Daten, wie sie für Steuerungsaufgaben typisch sind, ist die maximale Verzögerung (*Übertragungs-Latenz*) wichtig.

Neben der eigentlichen Übertragungsdauer auf dem Bus, die durch die Bitrate, die Datenmenge und den Protokolloverhead vorgegeben ist, ist vor allem das Bus-Zugriffsverfahren für die zeitliche Unsicherheit verantwortlich. Daher greifen viele Optimierungen insbesondere bei neueren Kfz-Bussystemen an dieser Stelle an und führen oft zu heftigen Glaubenskriegen um das *am besten geeignete* Bussystem.

Es darf jedoch nicht vergessen werden, dass wegen des Protokollüberbaus sowie der vielen anderen Aufgaben, die die relativ leistungsschwachen Mikroprozessoren in Kfz-Steuergeräten noch zu erfüllen haben, ein erheblicher Teil der Verzögerungszeiten auf die softwareseitige Verarbeitung der Daten zurückgeht. Dies gilt insbesondere, je komplexer das Übertragungsprotokoll ist und je mehr Teile der Protokollverarbeitung nicht im Kommunikationscontroller in schneller Hardware abgearbeitet werden können. Aus diesem Grund ist immer eine Gesamtbetrachtung von der Bereitstellung der Daten auf der Anwendungsebene beim Sender bis zu deren Verfügbarkeit auf der Anwendungsebene beim Empfänger notwendig (Abb. 2.17). Insbesondere bei Class C-Bussen dominieren in der Praxis meist die Zeit für die softwareseitige Bearbeitung und der dort ebenfalls entstehende Jitter im Vergleich zur reinen Busübertragung.

2.1.8 Elektrik/Elektronik-(E/E)-Architekturen

Die ersten vernetzten Fahrzeuge in den 1990er Jahren hatten eine simple Elektronikarchitektur, bei der einfach alle Steuergeräte an einen gemeinsamen CAN-Bus angeschlossen wurden (Abb. 2.18). Für die Diagnose gab es zusätzlich eine K-Line-Schnittstelle, die teil-

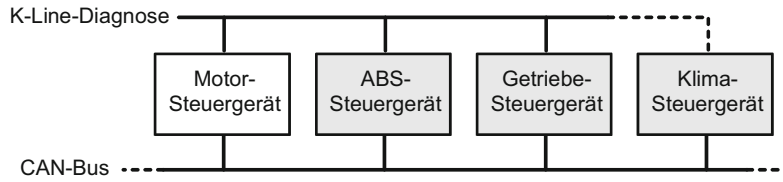

Abb. 2.18 *Klassische* Elektronik-Architektur

weise als Bus, teilweise aber auch für jedes einzelne Gerät nach außen geführt wurde. Zentrale Fahrzeugfunktionen wurden meist zusätzlich in das Motor-Steuergerät integriert, da dies das einzige Steuergerät war, das sicher in jedem Fahrzeug vorhanden war, während es sich bei den anderen Geräten oft um in unterschiedlichen Kombinationen anzutreffenden Sonderausstattungen handelte.

Nachdem immer mehr Steuergeräte eingebaut wurden, nahm die Busauslastung so weit zu, dass zuerst ein zweiter und später weitere Busse eingesetzt werden mussten. Die Busse wurden durch einzelne Steuergeräte verbunden, die an mehrere Bussysteme angeschlossen und zusätzlich zu ihren normalen Funktionen für den Datenaustausch zwischen den Bussystemen verantwortlich waren. Die Struktur der Systeme ergab sich primär aus dem Wunsch nach einer möglichst günstigen Verkabelung und aus den mehr oder weniger zufälligen Einbauorten der Geräte. In manchen Fällen war sie auch einfach ein Abbild der Organisationsstruktur des Herstellers. Weil die so gewachsenen Systeme schwer zu beherrschen waren und der Wildwuchs in der Steuergeräte-Kommunikation oft zu dubiosen Fehlerbildern führte, setzte sich in den 2000er Jahren allmählich eine klarer strukturierte Architektur durch, bei der die Steuergeräte nach Anwendungsbereichen gruppiert und die Bussysteme über ein zentrales *Gateway* gekoppelt werden (Abb. 2.19). Die klassischen Anwendungsbereiche (*Domänen*) sind Triebstrang (*Powertrain*), Fahrwerk (*Chassis*), Karosserie (*Body*) sowie Unterhaltungselektronik und Navigation (*Infotainment*).

Mit dem Aufkommen von Fahrerassistenzsystemen wird die Trennung in die unterschiedlichen Domänen allerdings immer weniger möglich, weil die Assistenzsysteme häufig domänenübergreifende Funktionen benötigen und damit der Kommunikationsbedarf zwischen den einzelnen Domänen massiv ansteigt. Das zentrale *Gateway*-Steuergerät (Abb. 2.19) erweist sich dabei als Flaschenhals. Möglicherweise werden zukünftige Fahrzeuge daher die *Gateway*-Funktion wieder dezentralisieren und die einzelnen Gateways über einen Hochgeschwindigkeitsbus (*Backbone*) miteinander verbinden (Abb. 2.20). Gleichzeitig ist denkbar, dass die dezentralen *Gateways* nicht mehr als normale Steuergeräte, sondern als Hochleistungsrechner (*Domain Controller*) ausgeführt werden. Diese *Domain Controller* sollten dann sämtliche rechenleistungsintensiven Aufgaben übernehmen, während die übrigen Steuergeräte zu intelligenten Sensor-Aktor-Ansteuereinheiten vereinfacht werden könnten.

Das Konzept für die Aufteilung der Fahrzeugfunktionen auf verschiedene Steuergeräte, deren Vernetzung, die Optimierung der Einbauorte und Verkabelung sowie die Vertei-

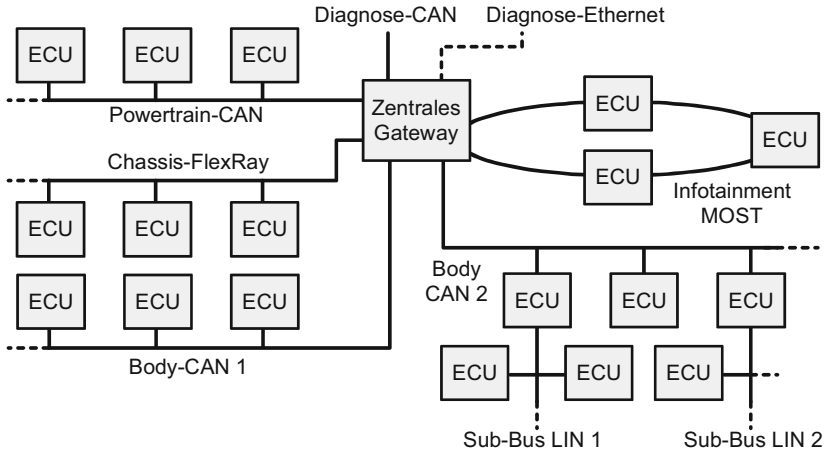

Abb. 2.19 Aktuelle heterogene Elektronik-Architektur

lung und Steuerung der notwendigen elektrischen Energie im Fahrzeug, die sogenannte Elektrik/Elektronik-(E/E)-Architektur, ist heute eine der komplexesten Aufgaben bei der Entwicklung einer Fahrzeugfamilie.

2.2 K-Line nach ISO 9141 und ISO 14230

K-Line ist das älteste in europäischen Kfz eingesetzte Busprotokoll für Diagnoseaufgaben. Es handelt sich um ein zeichen-orientiertes Protokoll (vgl. Abschn. 2.1.5), das sich mit den

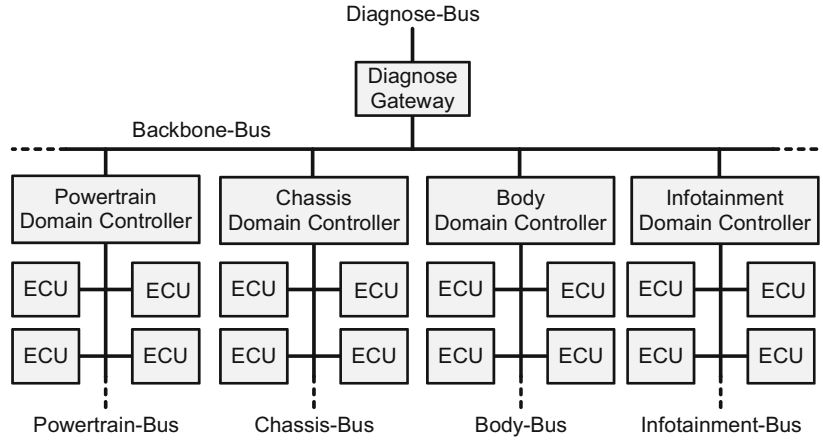

Abb. 2.20 Mögliche zukünftige Elektronik-Architektur

in nahezu jedem Mikrocontroller und jedem Rechner vorhandenen seriellen Schnittstellen auf UART-Basis einfach implementieren lässt.

2.2.1 Entwicklung von K-Line und KWP 2000

Die Schnittstelle ist in den 80er Jahren als Firmenstandard entstanden und wurde 1989 als ISO 9141 standardisiert. Der Standard definierte im Wesentlichen die elektrischen Eigenschaften, die Art der Bitübertragung und die Art der Kommunikationsaufnahme (Initialisierung, „Reizung“) zwischen Steuergerät und Diagnosetester. Während der Initialisierungsphase tauschen die Geräte ein so genanntes *Keyword* aus, durch das sich die beiden Geräte auf ein gemeinsames Datenprotokoll für den sich anschließenden Austausch von Diagnosedaten verständigen. Die zulässigen Werte für das Keyword und die darauf aufbauenden Diagnoseprotokolle selbst sind nicht Bestandteil der ISO 9141-Norm und waren lange Zeit genauso wie die verwendeten Leitungsverbindungen und Steckverbinder extrem herstellerspezifisch und untereinander inkompatibel.

Anfang der 90er Jahre wurden in USA auf Vorschlag zunächst der kalifornischen und dann der Bundesumweltbehörden CARB und EPA unter der Bezeichnung OBD (On Board Diagnose) gesetzliche Vorschriften für die Überwachung emissionsrelevanter Komponenten im Kfz erlassen. Neben Vorgaben, welche Arten von Fehlern im Fahrbetrieb erkannt und wie diese Fehler dem Fahrer angezeigt und im Gerät gespeichert werden müssen, wird eine definierte Diagnoseschnittstelle gefordert, mit der Behörden, Polizei und Werkstätten Testgeräte an das Fahrzeug anschließen können, um die emissionsrelevanten Komponenten zu prüfen. Neben der von den amerikanischen Herstellern favorisierten SAE J1850 Schnittstelle akzeptierten die Behörden auf Drängen der europäischen Hersteller auch eine auf ISO 9141 basierende Schnittstelle. Dabei wurden die im ursprünglichen Standard enthaltenen Freiheitsgrade und Wahlmöglichkeiten mit dem Standardzusatz ISO 9141-2 (ISO 9141-CARB) eingeschränkt und präzisiert, wobei aber immer noch über herstellerspezifische Keywords erhebliche Protokollvarianten insbesondere für den nicht emissionsrelevanten Teil der Diagnose zulässig waren. Parallel mit der Weiterentwicklung der OBD-Vorschriften in USA übernahm Mitte der 90er Jahre die EU die amerikanischen Vorschriften in modifizierter Form (European OBD, OBD-II, OBD-III). In diesem Rahmen wurde unter dem Sammelbegriff *Keyword Protocol 2000* (KWP 2000) auch die Spezifikation der Diagnoseschnittstelle als ISO 14230 weiter präzisiert und auf die höheren Protokollebenen ausgedehnt. Der Standard umfasst die in Tab. 2.3 aufgeführten Teile.

Der Physical Layer und der Data Link Layer einschließlich der für emissionsrelevante Komponenten wesentlichen Einschränkungen werden in den folgenden beiden Abschnitten beschrieben, der Application Layer in Kap. 5.

Tab. 2.3 Normen für die KWP 2000 K-Line-Diagnose

ISO 14230-1	**Physical Layer** der allgemeinen Diagnoseschnittstelle für KWP 2000 (Kompatibel zu ISO 9141-2)
ISO 14230-2	**Data Link Layer**
ISO 14230-3	Implementierungshinweise, die dem **Application Layer** zuzuordnen sind.
ISO 14230-4	Einschränkungen für Physical und Data Link Layer bei der Diagnose emissionsrelevanter Komponenten (**OBD**)

2.2.2 K-Line Bus-Topologie und Physical Layer

K-Line ist ein bidirektionaler Ein-Draht-Bus, über den der gesamte Datenverkehr abgewickelt wird (Abb. 2.21). Optional ist zusätzlich eine weitere, aber nur unidirektionale Ein-Draht-Leitung, die L-Line, möglich, die aber nur für die Initialisierungsphase verwendet wird. Eine Mischung von Geräten mit und ohne L-Line innerhalb desselben Fahrzeugs ist zulässig. Der Diagnosetester kann entweder direkt mit dem fahrzeuginternen Bus verbunden oder über ein Gateway-Steuergerät geführt werden. Die Konfiguration mit Gateway-Steuergerät findet sich in der Praxis meist dann, wenn der fahrzeuginterne Bus nicht auf K-Line sondern auf CAN basiert. Der KWP 2000 Application Layer für CAN wurde in ISO/DIS 15765-3 so definiert, dass er mit dem KWP 2000 Application Layer nach ISO 14230 weitgehend kompatibel war.

Die Logikpegel auf K- und L-Line sind relativ zur Bordnetzspannung (Batteriespannung U_B) mit > 0,8 U_B für *High* und < 0,2 U_B für *Low* definiert (auf der Senderseite im Fahrzeug, beim Tester werden kleinere Toleranzen gefordert).

Diagnosetester werden heute üblicherweise auf Basis von PCs aufgebaut. Da deren serielle RS232 C-Schnittstelle mit anderen Logikpegeln und getrennten Sende- und Empfangsleitungen arbeitet, ist für den Tester ein interner oder externer Schnittstellenkonverter notwendig.

Die Bitübertragung selbst erfolgt Standard-UART kompatibel zeichenweise mit 1 Start-Bit, 8 Daten-Bits und 1 Stopp-Bit je Zeichen.

Obwohl die Steuergeräte über die K-Line theoretisch auch untereinander kommunizieren können, findet die Kommunikation in der Praxis nur zwischen dem Tester und einem Steuergerät statt, gegebenenfalls mit dem Gateway als Zwischenstation.

Die Bitrate (Baudrate) wird vom jeweiligen Steuergerät im Bereich 1,2 kbit/s bis 10,4 kbit/s frei vorgegeben, der Tester muss sich darauf einstellen. Die einzelnen Steuergeräte können mit unterschiedlicher Bitrate arbeiten. Für emissionsrelevante Steuergeräte wird eine feste Bitrate von 10,4 kbit/s gefordert. Für Sonderanwendungen in der Applikations- und Fertigungsphase (*Messen, Verstellen, Flashen*) werden, außerhalb der elektrischen Spezifikationen des Standards auch höhere Bitraten verwendet, was zu Übertragungs- und Kompatibilitätsproblemen führen kann.

Passend zur Bitrate werden im Standard auch zulässige Leitungskapazitäten, Innenwiderstände sowie Pullup- und Pulldown-Widerstände der Leitungstreiber, geforderte Über-

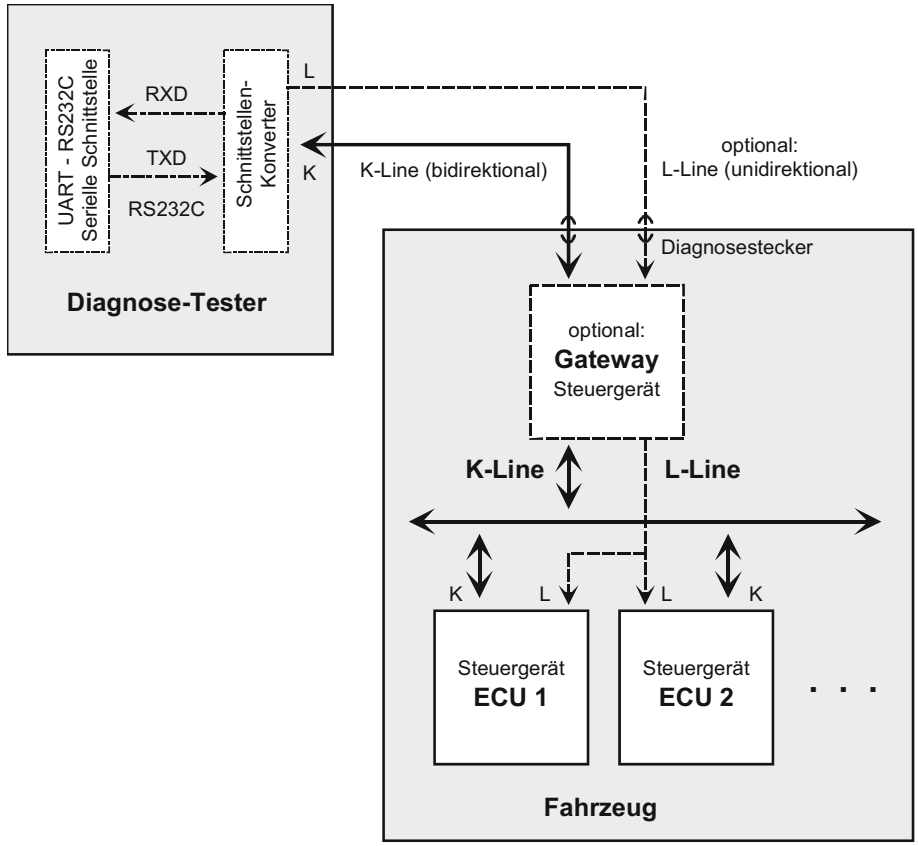

Abb. 2.21 K-Line Bus-Topologie (ohne Masse und Batterieverbindungen)

spannungsfestigkeit, Toleranzen der Bitrate usw. für eine maximale Bitrate von 10,4 kbit/s definiert. Die geforderten Innenwiderstände der Treiber für 12 V-Bordnetze (PKW, amerikanische LKW) und 24 V-Bordnetze (europäische LKW) sind unterschiedlich, doch können Treiberschaltungen für Steuergeräte so ausgelegt werden, dass sie für beide Bordnetzspannungen verwendbar sind. Dies gilt auf der Testerseite wegen der dort generell engeren Spezifikationen nur bedingt.

Die Form der Diagnosesteckdose ist durch ISO 15031-3 und SAE J1962 einheitlich vorgeschrieben (Abb. 2.22). Diese 16-polige Steckverbindung enthält Anschlüsse für Bus-Systeme nach ISO 9141, SAE J1850 und CAN sowie Batteriespannungsanschlüsse für den Tester. Die im Bild nicht bezeichneten Kontakte dürfen herstellerspezifisch belegt werden. Es ist zu erwarten, dass einige dieser Anschlüsse zukünftig durch weitere Bussysteme belegt werden, z. B. LIN, FlexRay oder Ethernet.

In Fahrzeugen amerikanischer Hersteller (z. B. Chrysler, Ford, GM) wurde lange SAE J1850 verwendet, in europäischen Fahrzeugen ISO 9141/ISO 14230 K-/L-Line, in neueren

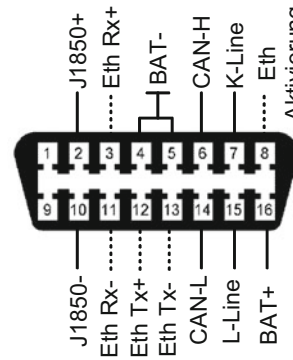

Abb. 2.22 OBD-Diagnosesteckdose (Buchse) nach ISO 15031-3/SAE J1962

Fahrzeugen mittlerweile ISO 15765 CAN. Die Buchse befindet sich typischerweise unter dem Armaturenbrett oder in der Mittelkonsole.

Die ursprüngliche ISO 9141 lässt im Gegensatz zur neueren ISO 14230 optional auch Varianten zu, bei denen die K- und die L-Leitung jeweils unidirektional als Datenleitungen verwendet werden. Außerdem kann ein einzelnes Steuergerät zusätzlich zu der weiter unten beschriebenen 5 Baud-Initialisierung bei einer reinen Punkt-zu-Punkt-Verbindung auch initialisiert werden, indem seine K- und/oder L-Leitung für min. 2 s auf Masse gelegt wird. Die Zeitfenster (Timeout) sind in ISO 9141 großzügiger spezifiziert als in der neueren ISO14230.

2.2.3 Data Link Layer

Bei der Kommunikation ist der Diagnosetester grundsätzlich der *Master* bzw. *Client*, von dem die Initiative ausgeht. Das Steuergerät ist der *Slave* bzw. *Server*, der auf Anfragen antwortet. Eine so genannte Diagnosesitzung (Diagnostic Session) zwischen Tester und Steuergerät läuft bei K-Line in 3 Phasen ab:

- Verbindungsaufbau (Initialisierung)
- Datenaustausch
- Verbindungsabbau

Verbindungsaufbau und Verbindungsabbau: Initialisierung („Reizung“) Für den Verbindungsaufbau definiert die ISO 14230 zwei zulässige Verfahren:

- *Schnelle Initialisierung* (Fast Initialisation), in ISO 9141 noch nicht definiert
- *5 Baud Initialisierung*, bereits in ISO 9141 definiert.

Die *schnelle Initialisierung* ist nur zulässig für Steuergeräte, die mit der festen Bitrate von 10,4 kbit/s arbeiten, und läuft folgendermaßen ab (Abb. 2.23):

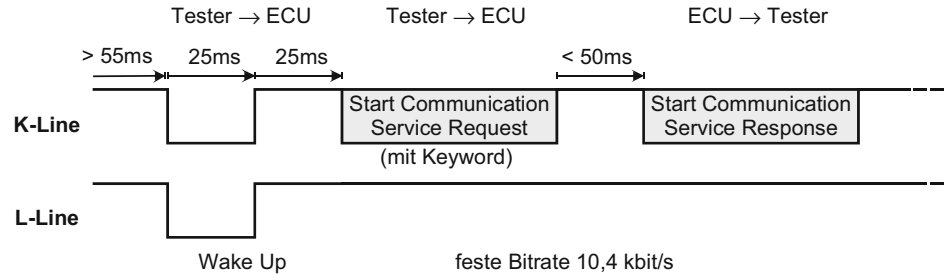

Abb. 2.23 Schnelle Initialisierung

- Die K- und L-Line waren nach dem Einschalten für min. 300 ms bzw. nach dem Abbau der vorigen Verbindung für min. 55 ms im Ruhezustand *High*.
- Der Tester legt K- und L-Line gleichzeitig für 25 ms auf *Low* und anschließend für 25 ms wieder auf *High* (Wake Up Pattern).
- Die weitere Kommunikation findet nur noch auf der K-Line statt, die L-Line bleibt dauernd auf *High*.
- Der Tester sendet eine *Start Communication Service Request*-Botschaft (im normalen Botschaftsformat nach Abb. 2.25) mit der Adresse des Steuergerätes, zu dem die Verbindung aufgebaut werden soll.
- Das adressierte Steuergerät antwortet innerhalb von 50 ms mit einer *Start Communication Service Response*-Botschaft, die das so genannte Keyword (siehe unten) enthält.

Diese Initialisierungssequenz dauert ca. 100 ms.

Soll mit einer anderen Baudrate als 10,4 kbit/s gearbeitet werden, muss die so genannte *5 Baud Initialisierung* eingesetzt werden (Abb. 2.24):

- Die K- und L-Line waren für min. 300 ms im Ruhezustand High.
- Der Tester sendet gleichzeitig auf der K- und der L-Line mit der Bitrate 5 bit/s ein 8 bit langes Adresswort mit der Adresse des Steuergerätes, zu dem die Verbindung aufgebaut werden soll.

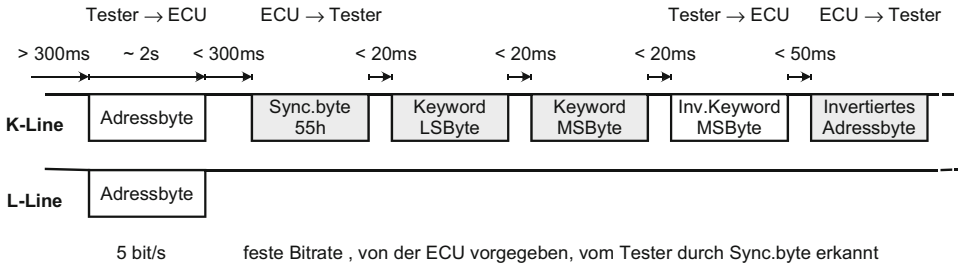

Abb. 2.24 5 Baud Initialisierung

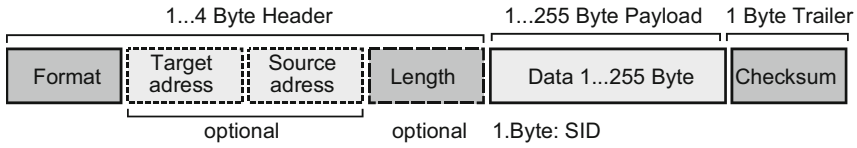

Abb. 2.25 KWP 2000 Botschaftsformat

- Die weitere Kommunikation findet nur noch auf der K-Line statt, die L-Line bleibt dauernd auf High.
- Das adressierte Steuergerät antwortet innerhalb von 300 ms, indem es ein Zeichen mit dem Synchronisationsmuster 55 h und anschließend im Abstand von jeweils max. 20 ms die beiden Bytes des Keywords (siehe unten) sendet. Das Steuergerät sendet dabei mit seiner eigenen, festen Bitrate im Bereich 1,2 … 10,4 kbit/s.
- Der Tester misst die Bitdauer im Synchronisationsmuster und erkennt damit, mit welcher Bitrate das Steuergerät arbeitet. Er schaltet selbst auf diese Bitrate um und sendet das zweite Keybyte innerhalb von 20 ms invertiert als Echo zurück.
- Das Steuergerät quittiert den Verbindungsaufbau, indem es innerhalb von 50 ms seine invertierte Adresse als Echo zurücksendet.

Diese Initialisierungssequenz dauert ca. 2,5 s.

Der Verbindungsabbau erfolgt mit Hilfe einer *Stop Communication Service Request*-Botschaft (siehe unten) ausgehend vom Tester.

Datenaustausch, Botschaftsformat Die gesamte Kommunikation erfolgt nach dem Request-Response-Verfahren. Der Diagnosetester sendet eine Botschaft als Anfrage (*Request*) an das Steuergerät, das Steuergerät sendet eine Botschaft mit der gewünschten Antwort (*Response*) oder einer Fehlermeldung zurück. Die übertragenen Botschaften haben das in Abb. 2.25 dargestellte Format.

Mit dem Formatbyte wird angezeigt, ob die beiden optionalen Adressbytes und das Längenbyte folgen oder nicht. Das Längenbyte gibt an, wie viele Datenbytes folgen. Für Botschaften mit ≤ 63 Datenbytes kann die Datenlänge innerhalb des Formatbytes codiert und das Längenbyte weggelassen werden. Die Prüfsumme im Trailer ist die mod 256 Summe aller Bytes (außer dem Prüfsummenbyte selbst).

Über das Keyword zeigt das Steuergerät dem Tester während der Initialisierung an, welche Header-Optionen es erwartet und selbst verwenden wird. Die Bezeichnung KWP 2000 für das Protokoll resultiert aus dem Wertebereich 2000 bis 2031, den das Keyword haben darf. Zusätzlich zur Festlegung, ob die optionalen Adressbytes und/oder das optionale Längenbyte verwendet werden, signalisiert das Steuergerät, ob für die Timeout-Überwachung bei der Übertragung der normale oder ein erweiterter Timeout-Parametersatz (Tab. 2.4) verwendet werden soll. Mit Hilfe der *Access Timing Parameter Request*-Botschaft kann der Tester dann später auch andere Timeout-Parameter einstellen, sofern das Steuergerät dies unterstützt.

Tab. 2.4 Default-Timeout-Werte, Bezeichnungen der Norm in ()

Abstand zwischen …	Min. … Max.
… zwei Bytes innerhalb einer Botschaft (ECU: P1, Tester: P4)	0 … 20 ms
… Tester-Anfrage und Steuergeräte-Antwort (P2)	25 … 50 ms
… Steuergeräte-Antwort und nächster Tester-Anfrage (P3)	55 ms … 5 s

Mit Ausnahme von Punkt-zu-Punkt-Verbindungen müssen Tester und Steuergeräte adressiert werden. Dabei wird über das Formatbyte zwischen so genannter funktionaler und physikalischer Adressierung unterschieden. Physikalische Adressen sind beliebige, innerhalb eines Fahrzeugs eindeutige 8 bit-Werte, die vom Kfz-Hersteller festgelegt werden. Funktionale Adressen sind in SAE J2178 definierte 8 bit-Wertebereiche für einzelne Klassen von Steuergeräten, z. B. 10 … 17 h für Motorsteuergeräte, 18 … 1Fh für Getriebesteuergeräte, F0…FDh für Diagnosetester usw. Ein Steuergerät darf bei der Initialisierung und beim Empfang von Daten nur reagieren, wenn es seine eigene Adresse erkennt.

Dienste für die Datenübertragung (Communication Services) Das erste Byte der übertragenen Nutzdaten in einer Botschaft enthält den so genannten *Service Identifier* (SID), mit dem der Inhalt der Botschaft gekennzeichnet wird. Die folgenden Nutzdatenbytes enthalten dann die zugehörigen Parameter bzw. Antworten. Für jede Tester-Anfrage (Request) und für die zugehörige Steuergeräte-Antwort (Response) ist jeweils eine eigene SID definiert. Bei den Antworten gibt es in der Regel unterschiedliche SIDs für eine positive und für eine negative Antwort (Fehlermeldung).

Folgende Dienste müssen in der Protokollsoftware implementiert werden:

- **Start Communication Service Request** (SID 81 h): Verbindungsaufbau mit Übermittlung des Keywords
- **Stop Communication Service Request** (SID 82 h): Verbindungsabbau
- **Access Timing Parameter Request** (SID 83 h): Auslesen und Ändern der Timeout-Parameter der Datenübertragung. Das Ändern ist optional und muss von einem Steuergerät nicht unterstützt werden.
- **Send Data Service**: Senden eines beliebigen Diagnosedienstes. Der Datenblock mit bis zu 255 Datenbyte einschließlich SID wird vom Application Layer bereitgestellt bzw. beim Empfang an den Application Layer weitergereicht (s. h. Kap. 5).

Fehlerbehandlung Diagnosetester und Steuergerät müssen die empfangenen Botschaften auf Fehler (falsche Botschaftslänge, falsche Prüfsumme, Timeout Fehler) überprüfen:

- Steuergeräte ignorieren falsche Botschaften und senden keine Antwort (wird vom Tester über Timeout erkannt).
- Tester wiederholen eine Anfrage bei fehlerhaften oder ausbleibenden Antworten bis zu insgesamt 3 Mal.

Bei anderen Fehlern (falsches Header-Format, Fehler in den empfangenen Daten) sendet das Steuergerät eine negative Antwort.

Der Tester kann die Signale auf den Verbindungsleitungen überwachen und bei Fehlern die Übertragung wiederholen. Steuergeräte müssen derartige Fehler weder erkennen noch darauf reagieren.

2.2.4 Einschränkungen für emissionsrelevante Komponenten (OBD)

Abgasrelevante Systeme wie Motorsteuergeräte unterliegen einigen Einschränkungen:

- Target und Source Address Byte im Header müssen verwendet werden, das Längenbyte im Header entfällt, d. h. der Header besteht immer aus 3 Byte.
- Botschaften dürfen höchstens 7 Byte Nutzdaten haben.
- Tester-Anfragen sollen für die Zieldresse funktionale Adressen, Steuergeräteantworten müssen für die Zieladresse stets physikalische Adressen verwenden. Die Quelladresse ist immer eine physikalische Adresse.
- Nur die Default-Timeout-Werte dürfen verwendet werden. Abfragen oder Ändern der Timing Parameter ist nicht zulässig.
- Beide Initialisierungsarten sind zulässig, in jedem Fall aber erfolgt die Kommunikation anschließend mit der festen Bitrate von 10,4 kbit/s.

2.2.5 Schnittstelle zwischen Software und Kommunikations-Controller

Die K-Line-Schnittstelle wird praktisch immer über einen gewöhnlichen UART und nicht über einen speziellen Kommunikationscontroller realisiert. Übliche UARTs wickeln lediglich das Senden und Empfangen eines einzelnen Zeichens selbstständig ab, so dass praktisch das gesamte Protokoll inklusive der Timeout-Erzeugung und Überwachung in Software realisiert werden muss. Da die meisten Mikrocontroller-UARTs nicht über größere Sende- und Empfangspuffer verfügen, wird das Senden und Empfangen in der Regel zeichenweise interruptgesteuert abgewickelt. Kritisch dabei ist bei hoher CPU-Auslastung im Allgemeinen das Empfangen. Wenn der Diagnosetester mit den kürzest möglichen Abständen sendet, muss bei 10.4 kbit/s jede Millisekunde ein Zeichen aus dem UART ausgelesen und in einen im ungünstigsten Fall 260 Byte großen RAM-Puffer kopiert werden. Da die Checksumme erst am Ende der Botschaft überprüft werden kann, darf die Weiterverarbeitung der Botschaft eigentlich erst erfolgen, wenn alle Zeichen der Botschaft empfangen worden sind. Die Initialisierungssequenz macht spezielle Maßnahmen notwendig. Da sich UARTs in der Regel nicht auf die niedrige Bitrate von 5 bit/s für die 5 Baud-Initialisierung umprogrammieren lassen, muss auf der Steuergeräteseite die Empfangsleitung beim Warten auf eine 5 Baud-Initialisierungssequenz entweder im *Polling*-Verfahren oder über einen

Capture-Eingang überwacht werden. Für die Erkennung der schnellen Initialisierung lassen sich UARTs in der Regel umprogrammieren, da die Low-Phase von 25 ms ± 1 ms der Übertragung eines 0 h-Zeichens bei ca. 360 bit/s mit einer Toleranz von ± 4 % entspricht, so dass der UART auf diese Baudrate eingestellt werden und einen regulären Empfangsinterrupt auslösen kann. Die anschließende High-Phase kann dadurch überwacht werden, dass anschließend für 25 ms kein weiteres Zeichen mehr empfangen wird, weil jeder Low-Pegel in dieser Zeit vom UART als Startbit eines weiteren Zeichens erkannt werden würde. Ohne vorangegangene Initialisierung dürfen empfangene Zeichen ignoriert werden.

Dieselbe Problematik ergibt sich auf der Testerseite für das Senden der Initialisierungssequenz, insbesondere wenn dieser auf PC-Basis realisiert wird. Da dort in der Regel keine Capture-Compare-Ein-/Ausgänge vorhanden sind, die üblichen PC-Betriebssysteme zu einem relativ großen Zeit-Jitter führen und weder 5 bit/s noch 10.4 kbit/s Standard-Baudraten sind, sind bei enger Interpretation der Zeitvorgaben in der ISO-Spezifikation spezielle Geräte-Treiber und/oder zusätzliche Hardware nötig.

2.2.6 Ältere K-Line-Varianten

Vor Veröffentlichung des KWP 2000-Protokolls wurden andere K-Line-Protokolle verwendet, die alle bezüglich der Bitübertragung und Verbindungsaufnahme auf ISO 9141 und der 5 Baud-Initialisierung beruhen, danach aber über ein spezifisches Keyword auf ein proprietäres Protokoll umschalteten. Weit verbreitet war das Keyword KW 71-Protokoll, das z. B. bei BMW eingesetzt wurde. Dieses Protokoll wurde für Bitraten von 1200 bit/s oder 9600 bit/s implementiert. Es definierte abhängig vom jeweiligen Fahrzeughersteller unterschiedliche Botschaften und Services. Während der Kommunikation fand ein laufendes Handshaking zwischen Diagnosetester und Steuergerät statt, indem das jeweils empfangene Zeichen in invertierter Form als Echo zurückgesendet wurde. Das Protokoll ist inzwischen ebenso veraltet wie die Keyword-Varianten KW 81 und 82 (Opel) oder KW 1281 (VW/Audi).

2.2.7 Zusammenfassung K-Line – Layer 1 und 2

- Bus- oder Punkt-zu-Punkt-Verbindung zwischen Diagnosetester und Steuergerät(en).
- Zeichenorientiertes, UART-basiertes Übertragungsprotokoll mit bidirektionaler Ein-Draht-Leitung (K-Line), Bitrate bis zu 10,4 kbit/s, Signalpegel U_B (Batteriespannung).
- Kommunikation zwischen Diagnosetester und genau einem Steuergerät. Tester und Steuergerät werden über eindeutige, feste Adressen identifiziert.
- Kommunikation wird vom Diagnosetester aufgebaut, Steuergerät gibt Bitrate und Protokolloptionen vor, Diagnosetester muss sich entsprechend einstellen.
- Kommunikation nach dem Request – Response – Verfahren, d. h. Tester stellt Anfrage (Request), Steuergerät antwortet (Response).

Tab. 2.5 Charakteristische Eigenschaften von SAE J1850 Bussystemen

	SAE J1850 PWM	SAE J1850 VPWM
Hauptvertreter	Ford	General Motors, Chrysler
Bit-Codierung	Pulsbreitenmodulation (PWM)	Variable Pulsbreitenmodulation (VPWM)
Bitrate	41,6 kbit/s	10,4 kbit/s (Mittelwert)
Datenleitung	Zwei-Draht (Twisted Pair)	Ein-Draht (Single Wire)
Signalpegel	5 V – Differenzsignal Low < 2,2 V, High > 2,8 V max. 6,25 V	U_{batt} unipolar Low < 3,5 V, High > 4,5 V Max. 20 V
Nutzdaten	0…8 Byte je Botschaft	
Botschaftslänge	max. 101 bit (inkl. Header u. Trailer)	
Buszugriff	CSMA/CA	

- Botschaften mit 1 … 255 Datenbytes und 2 … 5 Byte Header/Trailer.
- Abstand zwischen den Bytes einer Botschaft (Inter Character Break) max. 20 ms, Abstand zwischen Tester-Anfrage (Request) und Steuergeräte-Antwort (Response) 25 … 50 ms. Abstand bis zur nächsten Tester-Anfrage min. 55 ms.
- Übertragungsdauer einer Botschaft mit 255 Datenbytes bei 10,4 kbit/s:
- Best Case: 250 ms, Worst Case: 5,5 s, d. h. Nutzdatenrate < 1 KB/s
- Überwachung durch Prüfsumme, Formatüberwachung und Timeout. Bei vom Tester erkannten Übertragungsfehlern max. 3 Übertragungsversuche. Bei vom Steuergerät erkannten Übertragungsfehlern Fehlermeldung an den Tester.
- Realisierung des Protokolls nahezu vollständig in Software. UART für Zeichenübertragung.

2.3 SAE J1850

SAE J1850 ist ein veraltetes, bitstrom-orientiertes Class A/B-Protokoll für die On- und Off-Board-Kommunikation, das vor allem von amerikanischen PKW-Herstellern eingesetzt wird [4, 5]. Bei genauer Betrachtung handelt es sich dabei eigentlich sogar um zwei auf der physikalischen und der Bitübertragungsschicht zueinander inkompatible Bussysteme, die aber einen gemeinsamen Data Link Layer verwenden (Tab. 2.5).

Im Gegensatz zu K-Line, CAN, LIN und FlexRay, die alle mit *Non Return to Zero* (*NRZ*) Bitcodierung arbeiten, bei der das Bussignal während einer Bitdauer konstant auf *Low* oder *High* bleibt, verwendet J1850 Impulssignale zur Bitcodierung.

In der PWM-Variante beginnt jedes Bit mit einem *Low-High*-Übergang (Abb. 2.26). Abhängig davon, ob als Datenbit eine 1 oder eine 0 übertragen werden soll, bleibt das Signal dann für 1/3 bzw. 2/3 der Bitdauer auf *High*, bevor es für den Rest der Bitdauer wieder auf *Low* wechselt. Die Bitdauer ist konstant. Bei 41,6 kbit/s ist $T_{bit} = 24\ \mu s$.

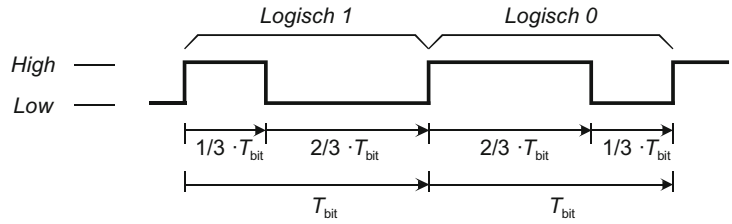

Abb. 2.26 Bitcodierung bei J1850 PWM

Tab. 2.6 Bitcodierung bei J1850 VPWM (Kombination von Dauer und Pegel)

Signalpegel	Signaldauer	Entspricht Datenbit
Low	$T_{\text{bit,short}}$	0
High	$T_{\text{bit,long}}$	0
High	$T_{\text{bit,short}}$	1
Low	$T_{\text{bit,long}}$	1

In der VPWM-Variante beginnt jedes Bit mit einem Übergang und endet mit dem nächsten Übergang. Der Abstand zwischen den Übergängen wird zwischen den beiden festen Werten $T_{\text{bit,short}} = 64\,\mu\text{s}$ und $T_{\text{bit,long}} = 2 \cdot T_{\text{bit,short}} = 128\,\mu\text{s}$ umgeschaltet. Für jedes der zu übertragenden Datenbits 0 bzw. 1 stehen dabei je zwei unterschiedliche Impulssignale zur Verfügung (Tab. 2.6).

Dabei wird immer diejenige Kombination zur Übertragung des nächsten Datenbits gewählt, die zu einem Wechsel des Signalpegels, d. h. einem Übergang auf der Leitung führt (Abb. 2.27). Die Bitrate ist damit nicht konstant, sondern hängt von der Reihenfolge der Datenbits ab. Im Mittel beträgt sie

$$\frac{1}{(T_{\text{bit,long}} + T_{\text{bit,short}})/2} = \frac{1}{96\,\mu\text{s}} = 10{,}4\,\text{kbit/s}\,.$$

In beiden Fällen werden spezielle Bits (Start of Frame SOF, End of Data EOD, End of Frame EOF) zur Unterscheidung mit abweichender Impulsdauer codiert.

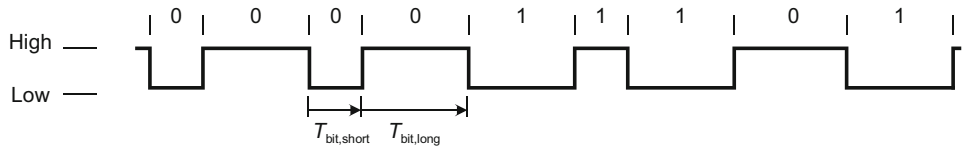

Abb. 2.27 Beispiel für die Übertragung der Datenbitfolge *000011101…* bei VPWM

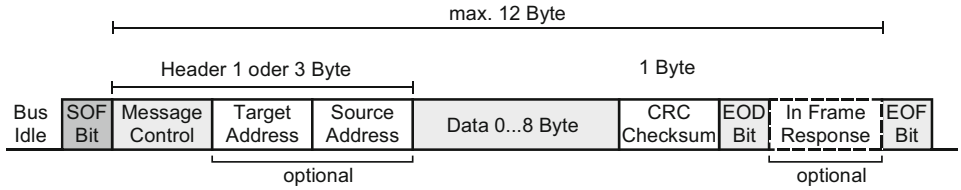

Abb. 2.28 SAE J1850 Botschaftsformat

Abbildung 2.28 zeigt das gemeinsame Botschaftsformat. Die Bits im *Message Control Byte* zeigen an, welche Priorität die Botschaft hat (0…7 mit 0 als höchster Priorität), ob ein 1 Byte (Chrysler) oder ein 3 Byte Header (GM, Ford) verwendet wird und ob eine *In Frame Response* (siehe unten) erwartet wird. Falls mehrere Sender gleichzeitig auf den Bus zugreifen, „gewinnt" wie bei CAN die Botschaft mit der höchsten Priorität. *Target* und *Source Address Byte* sind bei reiner Punkt-zu-Punkt-Verbindung optional (angezeigt durch ein Bit im *Message Control Byte*). Wie bei KWP 2000 kann der Empfänger (*Target*) funktional oder physikalisch adressiert werden (ebenfalls angezeigt durch ein Bit im *Message Control Byte*). Die Senderadresse (*Source*) ist immer eine physikalische Adresse (siehe Abschn. 5.1.3). Zur Fehlererkennung dient eine 8 bit CRC-Prüfsumme.

Alternativ zum beschriebenen Header mit *Message Control Byte* und den (optionalen) *Target* und *Source Address Bytes* existiert eine Header-Form mit einem reinen 1 Byte-Header, bei dem der Header wie bei CAN als Message Identifier dient. Diese Form, bei der die Empfänger eine inhaltsbezogene Akzeptanzfilterung durchführen müssen, wird vorzugsweise für die On-Board-Kommunikation zwischen Steuergeräten im Fahrzeug eingesetzt.

Eine Besonderheit stellt die optionale *In Frame Response* (*IFR*) dar. Die *IFR* ermöglicht es dem Empfänger, ohne selbst eine eigene Botschaft mit Header usw. senden zu müssen, unmittelbar auf die empfangene Botschaft zu antworten. Die Antwort kann innerhalb der *IFR* optional ebenfalls durch eine CRC-Prüfsumme gesichert werden. Die *IFR* darf ein oder mehrere Bytes umfassen. Durch eine weitere Arbitrierung innerhalb der *IFR* ist es auch möglich, dass mehrere Empfänger mit jeweils 1 Byte auf die Botschaft antworten.

Die Gesamtlänge einer Botschaft für den Header, die Nutzdaten, die CRC-Prüfsumme und die *In Frame Response* (ohne die SOF, EOD und EOF Bits) ist beschränkt auf max. 12 Byte, d. h. bei dem in der Praxis üblichen 3 Byte Header und 8 Datenbytes kann die *In Frame Response* max. 1 Byte lang sein.

Das Format der Dienste für die On-Board-Kommunikation ist in SAE J2178 festgelegt, die hier nicht besprochen werden soll. Für die Off-Board-Kommunikation (OBD-Diagnose) werden die in SAE J1979 definierten Dienste verwendet, die inhaltlich identisch mit ISO 15031 sind und in Abschn. 5.3 erläutert werden.

Aufgrund der ungewöhnlichen Bitcodierung und der bitweisen Arbitrierung ist das SAE J1850-Protokoll trotz der relativ niedrigen Bitrate zu komplex, um es rein softwaretechnisch mit Hilfe der Standardperipherie eines üblichen Mikrocontrollers sinnvoll zu

implementieren. Daher müssen ähnlich wie bei CAN zusätzlich zu den Bustransceivern spezielle J1850-Kommunikationscontroller als separate Bausteine oder auf dem Mikrocontroller integriert verwendet werden.

2.4 Sensor-Aktor-Bussysteme

Neben dem Drang zu immer höheren Übertragungsraten für Steuer- und Regelanwendungen, die zur Entwicklung von FlexRay geführt haben, gibt es auch im Bereich der Low-Cost-Bussysteme offensichtlich noch Einsatzgebiete, für die LIN (siehe Abschn. 3.2) nicht schnell genug, Low-Speed-CAN dagegen zu teuer zu sein scheint [6]. Jedenfalls haben sich in den letzten Jahren mehrere Industriekonsortien gebildet, die versuchen, neue Bussysteme zur Anbindung von Sensoren und Aktoren an Steuergeräte zu definieren und im Rahmen von SAE oder ISO zu standardisieren. Dabei sind praktisch alle großen Automobilelektronikhersteller aktiv und wirken in diesen Konsortien in wechselnder Rollenverteilung mit, obwohl oder weil die Standards teilweise miteinander konkurrieren. Da der Markt in diesem Bereich noch zu unübersichtlich ist, um beurteilen zu können, welche dieser Standards langfristige Bedeutung haben werden, soll hier nur ein kurzer Überblick gegeben werden.

2.4.1 SENT – Single Edge Nibble Transmission nach SAE J2716

Beim SENT-Bus, der seit 2007 als SAE J2716 standardisiert ist, handelt es sich um eine unidirektionale Verbindung, um Messwerte von einem intelligenten Sensor zu einem Steuergerät zu übertragen. SENT ist als Ersatz für Analog- oder PWM-Schnittstellen gedacht und benötigt wie diese mit Versorgungsspannung, Signalleitung und Masse mindestens drei Leitungsanschlüsse am Sensor. Ziel des SENT-Protokolls ist es, bei der Übertragung eine Auflösung von 12 bit zu gewährleisten, was bei Analog- oder einfachen PWM-Signalen schwierig ist.

Eine SENT-Botschaft besteht aus einem Synchronisierimpuls fester Länge (168 µs) sowie einer 4 bit Status-Information, sechs 4 bit-Datenwerten, d. h. zwei 12 bit Messwerten, und einer 4 bit CRC-Prüfsumme. Jeder 4 bit Wert $n = 0, 1, \ldots, 15$ (*Nibble*) wird als ein Spannungsimpuls übertragen, bei dem der Abstand der negativen Signalflanken gemäß $T_{\text{Impuls}} = 36\ \mu\text{s} + n \cdot 3\ \mu\text{s}$ variiert wird (Abb. 2.29). Für die Übertragung von zwei 12 bit Messwerten werden damit maximal $T_{\text{Botschaft}} = 816\ \mu\text{s}$ benötigt. Die Nutzdatenrate liegt unter 3,7 KB/s, etwa das Dreifache des bei LIN möglichen Wertes. Da der Empfänger die Dauer des Synchronisierimpulses dynamisch ausmessen kann, dürfen auf der Sensorseite einfache RC-Taktgeneratoren verwendet werden.

Das erste Bit der Status-Information wird üblicherweise zur Fehlersignalisierung eingesetzt. Die restlichen Bits werden bei einfachen Sensoren herstellerspezifisch verwendet werden, um z. B. den Messbereich des Sensors zu codieren. Alternativ besteht mit dem sogenannten *Slow Serial Channel* die Möglichkeit, längere Zusatzinformationen zu über-

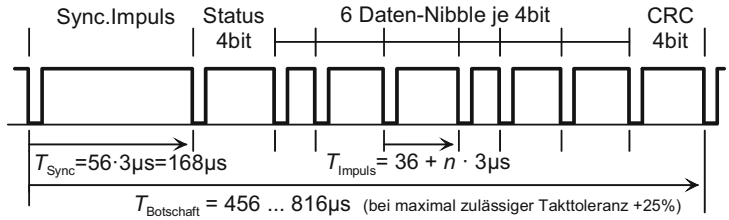

Abb. 2.29 SENT-Botschaft

tragen. Die Zusatzinformation wird über insgesamt 18 SENT-Botschaften verteilt, wobei jede Botschaft innerhalb des Status-Felds nur 2 bit dieser Zusatzinformation enthält, die auf der Empfangsseite dann wieder zur vollständigen Information zusammengefasst werden müssen. Innerhalb der Zusatzinformationen kennzeichnet ein 4 bzw. 8 bit langer *Identifier* den Inhalt der folgenden Daten. Vorgesehen sind *Identifier* um die SENT-Protokollversion, Sensortyp, Sensorhersteller, Seriennummern, Abgleichdaten des Sensors oder genauere Informationen zu aufgetretenen Fehlern (*Diagnostic Error Code*) zu übertragen.

2.4.2 PSI 5 – Peripheral Sensor Interface 5

In direkter Konkurrenz zu SENT steht PSI 5, das seine historischen Wurzeln im Airbag-Bereich hat und bei Beschleunigungssensoren verschiedene Vorgängerschnittstellen ablöst. Mittelfristig soll PSI 5 aber auch in anderen Bereichen der Sensorik im Fahrzeug verwendet werden. Seit Version 2.0 der Spezifikation gibt es daher neben dem Bereich *Airbag* auch Profile für die Anwendungsbereiche Fahrwerk (*Chassis and Safety Control*) und Antriebsstrang (*Powertrain*).

Im Gegensatz zu SENT benötigt der Sensor bei PSI 5 nur zwei Leitungsanschlüsse, weil das Signal durch Spannungs- bzw. Strommodulation über den Versorgungsspannungsanschluss übertragen wird (Abb. 2.30). Die PSI 5-Kommunikation ist in der Grundform wie bei SENT unidirektional. Vom Sensor zum Steuergerät besteht eine Punkt-zu-Punkt-Verbindung, wobei der Sensor seine Daten periodisch selbstständig sendet (asynchroner Betrieb). Alternativ ist aber auch eine Hintereinanderschaltung von Sensoren (*Daisy Chain*) oder ein Linienbus von einem Steuergerät zu mehreren Sensoren möglich, bei dem die Sensoren auf Anforderung des Steuergerätes senden (synchroner Betrieb). Im synchronen Betrieb ist auch eine eingeschränkte bidirektionale Kommunikation möglich.

Das Steuergerät versorgt den Sensor mit einer konstanten Spannung. Die Datenübertragung vom Sensor zum Steuergerät erfolgt, indem der Sensor seine Stromaufnahme um etwa $\Delta I = 26$ mA (13 mA in einer *Low-Power*-Variante) verändert (Abb. 2.30). Die Bitdarstellung erfolgt im Manchester-Code. Eine positive Flanke in Bitmitte überträgt eine „0“, eine negative Flanke eine „1“. An der Bitgrenze wird immer dann eine zusätzliche Signalflanke erzeugt, wenn das folgende Bit denselben Wert hat wie das vorherige Bit. Die Bitrate

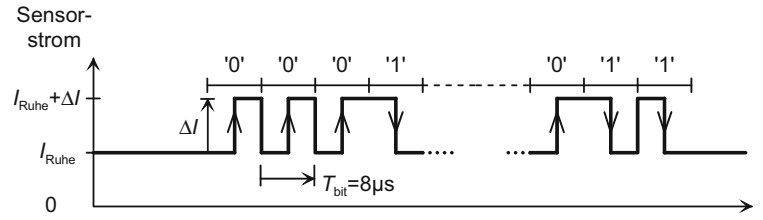

Abb. 2.30 PSI 5-Botschaft vom Sensor zum Steuergerät bei f_{bit} = 125 kbit/s

ist üblicherweise 125 kbit/s, darf aus Kompatibilitätsgründen zu einem älteren Protokoll aber auch auf 189 kbit/s eingestellt werden.

Eine vollständige Sensorbotschaft besteht aus 2 Startbits, einem Datenfeld mit n Bit und alternativ 1 Paritätsbit oder einer 3 bit CRC-Prüfsumme (Abb. 2.31a). Seit V2.x enthält das Datenfeld mindestens einen Sensorwert (10 … 24 bit), optional einen zweiten Sensorwert (0 … 12 bit), eine Statusinformation (0 …2 bit) sowie diverse Steuerbits (0 … 6 bit), wobei die maximal zulässige Gesamtlänge von 28 bit nicht überschritten werden darf. In der Vorgängerversion V1.3 war das Datenfeld noch auf einen Sensorwert mit n = 8, 10 oder 16 bit oder zwei Sensorwerte mit je 10 bzw. 12 bit beschränkt, d. h. maximal 24 bit. Bei der Messwertübertragung wird nur ein Teil des Wertebereichs genutzt. Bei n = 10 beispielsweise werden Messwerte nur im Bereich − 480 … + 480 übertragen, der restliche Bereich ist für Statusinformationen reserviert. Der Wert + 500 etwa dient zur Anzeige eines allgemeinen Sensordefekts.

Bei vergleichbaren Bedingungen wie bei SENT, d. h. n = 24 bit und CRC-Prüfsumme, dauert eine PSI 5-Botschaft bei 125 kbit/s nur 232 µs, während SENT im ungünstigsten Fall bis zu 816 µs benötigt.

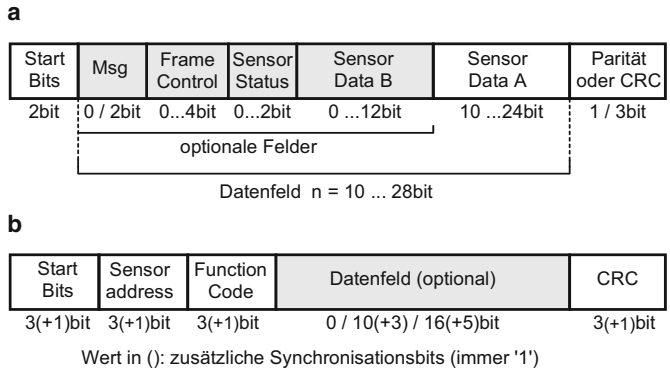

Abb. 2.31 PSI 5-Botschaftsformate. **a** Sensor → Steuergerät, **b** Steuergerät → Sensor

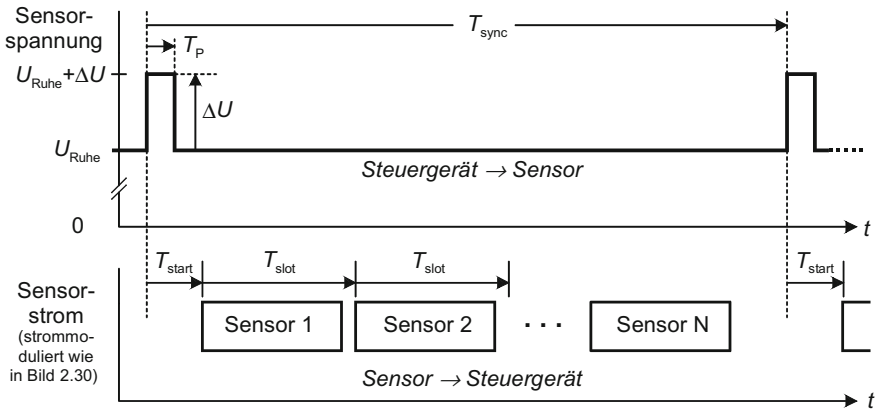

Abb. 2.32 PSI 5-Botschaften im synchronen Betrieb

Wie der *Serial Channel* bei SENT bietet auch PSI5 die Möglichkeit, außer den eigentlichen Messdaten weitere Informationen vom Sensor zum Steuergerät über insgesamt 18 PSI5-Botschaften verteilt zu übertragen. Je Botschaft werden dann im sonst optionalen Message Feld 2 bit der Zusatzinformation gesendet. Die gesamte Zusatzinformation besteht wiederum aus einem 4 bzw. 8 bit *Identifier*, einer 6 bit CRC-Prüfsumme, einigen Steuerbits sowie den eigentlichen Daten (12 bzw. 16 bit).

Im asynchronen Betrieb bestimmt der Sensor selbst, wann und wie oft er Daten sendet. Im alternativen synchronen Betrieb sendet der Sensor immer nur, wenn das Steuergerät die Versorgungsspannung für den Sensor für ca. 30 µs um mindestens 2,5 V (bei V1.3 waren es 3,5 V) anhebt (Abb. 2.32). Auf diese Weise ist auch ein Busbetrieb möglich, bei dem mehrere Sensoren in vordefinierten Zeitschlitzen (*Slots*) hintereinander senden. Die Reihenfolge der Sensoren, die Anzahl und Dauer der Zeitschlitze und der Abstand T_{sync} der Synchronisationsimpulse können für jede Anwendung individuell festgelegt werden.

Über die Synchronisationsimpulse ist auch eine langsame Datenübertragung vom Steuergerät zu den Sensoren möglich, d. h. PSI 5 kann damit bidirektional arbeiten. Dabei werden die Synchronisationsimpulse moduliert. Beim sogenannten *Tooth Gap* Verfahren überträgt der eigentliche Synchronisationsimpuls eine logische „1“, während ein fehlender Synchronisationsimpuls als logisch „0“ interpretiert wird. Dieses Verfahren ist allerdings nur bei konstanten Werten von T_{Sync} möglich. Um auch variable Abstände der Synchronisationsimpulse zu erlauben, wenn eine Sensorabfrage beispielsweise drehzahlabhängig erfolgen soll, ist seit Protokollversion V2.0 alternativ auch eine Pulsdauermodulation (*Pulse-Width-Modulation PWM*) der Synchronisationsimpulse möglich. Dabei überträgt ein kurzer Impuls von ca. $T_P = 30$ µs eine logische „0“, ein langer Impuls von ca. 60 µs eine logische „1“.

Das Botschaftsformat für die Steuergeräte-Sensor-Kommunikation enthält 3 Startbits, ein Adress- und Befehlsfeld sowie ein Datenfeld variabler Länge, eine 3bit CRC Prüfsumme

sowie ein Antwortfeld für den Sensor (Abb. 2.31b). Über das Adressfeld kann ein einzelner Sensor ausgewählt werden, dem über das *Function Code* Feld ein Befehl übermittelt werden soll. Die Antwort des Sensors erfolgt dann in den Synchronisationsimpuls-Zyklen, die auf das letzte CRC-Bit der Steuergerätebotschaft folgen. Damit die Sensoren den Beginn einer Steuergerätebotschaft eindeutig erkennen können, müssen zunächst vor Beginn der eigentlichen Botschaft mindestens 5 Synchronisationsimpulse mit „0" oder 31 Impulse mit „1" vorliegen. Außerdem wird nach jedem dritten Bit in die Steuergerätebotschaft ein „1" Bit eingefügt.

Da unterschiedliche Sensortypen existieren und das Protokoll eine Reihe von Parametern und Optionen bietet, ist ein Initialisierungsmechanismus vorgesehen. Dabei sendet ein Sensor nach dem Einschalten der Versorgungsspannung zunächst wiederholt eine Initialisierungsbotschaft, die Informationen über den Sensor und das verwendete Protokoll enthält, bevor die normale Messdatenübertragung beginnt.

Auf der physikalischen Ebene und bei der Bustopologie dagegen sind einige Kombinationen grundsätzlich unverträglich. Seit Protokollversion 2.x werden daher Profile für verschiedene Anwendungsgebiete definiert. So erlaubt das Profil für Airbag-Anwendungen beispielsweise alternativ Linienbus- oder Daisy-Chain-Topologien, während im Powertrain-Bereich nur Linienbusse empfohlen werden. Für die Übertragung vom Steuergerät zum Sensor ist bei Airbags nur die Tooth-Gap-Methode erlaubt, während in anderen Anwendungen auch die PWM-Methode zugelassen wird. Airbag-Systeme sollen im synchronen Modus möglichst mit einer festen Periodendauer von 500 µs arbeiten, während bei Powertrain auch eine variable Periodendauer in Abhängigkeit der Motordrehzahl eingesetzt werden darf. Da reale Sensoren und Steuergeräte kaum alle in der Protokollspezifikation vorgesehenen Möglichkeiten unterstützen werden, muss im Einzelfall sehr genau geprüft werden, ob die vorgesehenen PSI5-Komponenten tatsächlich vollständig untereinander kompatibel sind.

2.4.3 ASRB 2.0 – Automotive Safety Restraint Bus (ISO 22898)

Im Gegensatz zu SENT und PSI 5 erlaubt das vom Safe-by-Wire Plus Konsortium definierte ASRB 2.0 Protokoll eine vollwertige bidirektionale Kommunikation und eignet sich daher nicht nur für Sensoren sondern auch für Aktoren, die von einem Steuergerät angesteuert werden müssen. Wie PSI 5 stammt es aus dem Airbag-Bereich, kann aber prinzipiell für beliebige Sensoren und Aktoren eingesetzt werden. Die Bustopologie ist sehr flexibel. Neben dem klassischen Linienbus sind bei ASRB auch Daisy-Chain, Ring- und Baumstrukturen möglich.

ASRB ist ein echtes Master-Slave-Bussystem, bei dem über eine Zwei-Draht-Verbindung gleichzeitig Versorgungsspannung und Signal übertragen werden. Im Gegensatz zu PSI 5 wird dabei der Spannungspegel moduliert (Abb. 2.33). Für die erste Hälfte des Bittaktes legt das *Master*-Steuergerät den Bus niederohmig auf eine Spannung von ca. 11 V. Diese *Power* Phase dient der Spannungsversorgung der Sensoren und Aktoren (*Slaves*). In der zweiten

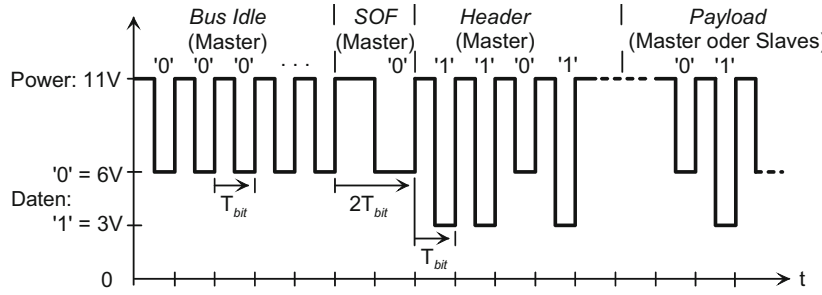

Abb. 2.33 Bitübertragung bei ASRB 2.0

Hälfte des Bittaktes, der *Data* Phase, erfolgt die Bitübertragung. Dabei treibt der *Master* den Bus hochohmig mit 6 V. In dieser Phase kann ein Sender den Bus entweder auf 6 V belassen und damit ein „0" Datenbit senden, oder ihn niederohmig auf 3 V herunterziehen und damit ein „1" Datenbit senden. Sender darf dabei prinzipiell jeder Busteilnehmer sein. Mögliche, stets vom *Master*-Steuergerät vorgegebene Bitraten sind 20, 40, 80 und 160 kbit/s.

Bereits im Ruhezustand sendet das *Master*-Steuergerät ständig „0" Datenbits. Der Beginn einer Botschaft wird durch das *Start of Frame (SOF)* Muster angezeigt, bei dem die Bitdauer vom Master verdoppelt wird (Abb. 2.33). Danach folgt der eigentliche Botschaftsheader, der ebenfalls ausschließlich vom *Master* gesendet wird. In der zweiten Hälfte der Botschaft senden je nach Zweck der Botschaft entweder wiederum der *Master* oder einer oder mehrere der *Slaves*. Das Botschaftsschema ähnelt also dem Verfahren bei LIN mit einer *Header*-Phase, die immer vom *Master* gesendet wird, und einer Payload-Phase, in der beliebige Busteilnehmer senden können.

Es gibt zwei Botschaftsarten, die vom *Master* über den Wert des *SOF*-Bits signalisiert werden (Abb. 2.34). Bei Botschaften zur Abfrage von Sensordaten, sogenannten *S-Frames*, beginnt das *Master*-Steuergerät die Botschaft mit SOF = 0 und zwei weiteren *Header*-Bits *MSA* und *SEL*. Daraufhin sendet der erste Sensor seine Daten und eine CRC-Prüfsumme, anschließend der zweite Sensor usw. Wie viele Sensoren abgefragt werden, welcher Sensor in welchem Zeitschlitz (*Slot*) antwortet und wie viele Bits die Sensordaten bzw. die CRC-Prüfsumme umfassen, wird in der Entwicklungsphase festgelegt und muss dem *Master* und den *Slaves* bekannt sein. *S-Frames* werden in der Regel vom *Master* periodisch gesendet. Schnelle Sensoren liefern bei jeder Abfrage einen neuen Messwert, langsame Sensoren können sich paarweise einen Zeitschlitz teilen und nur bei jeder zweiten Abfrage antworten (*Slot Multiplexing*). Welcher der beiden Partner eines Paars antworten soll, gibt der *Master* im *Header* über das *Select Bit SEL* vor. Bei einer mit den SENT und PSI 5-Beispielen vergleichbaren Konfiguration (Abfrage von 2 Sensor-Messwerten mit je 12 bit und 3 bit CRC sowie einer Bitrate von 80 kbit/s) dauert die Übertragung der Messwerte ca. 425 µs.

Optional kann der *Master* auch einen beliebigen Sensor gezielt abfragen. Dazu setzt er im *Header* das *Multiple Sharing Bit MSA* = 1 und sendet statt eines Slaves im ersten Zeitschlitz selbst ein 6 bit langes Sensor-Adresswort. In den restlichen Zeitschlitzen antworten

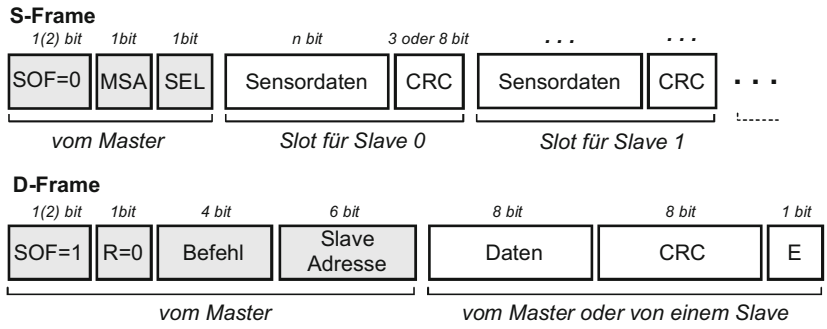

Abb. 2.34 Botschaftsaufbau (*SOF*-Bit wird mit doppelter Bitdauer gesendet)

die Sensoren wie üblich mit ihren Messdaten. Bei der nächsten Abfrage setzt der *Master* im *Header* wieder *MSA* = 0, worauf der vorher adressierte Sensor im ersten Zeitschlitz seine Messdaten einfügt. Alle übrigen Sensoren antworten bei beiden Botschaften in ihren jeweiligen Zeitschlitzen wie gewöhnlich. *D-Frame* Botschaften dienen zum Lesen oder Schreiben von Daten. Der *Slave* wird über eine eindeutige 6 bit Adresse im *Header* der Botschaft adressiert. Dabei kann lediglich ein einzelnes Datenbyte übertragen werden, das aufwendig über eine 8 bit CRC-Prüfsumme gesichert wird. Ob gelesen oder geschrieben werden soll, wird über das 4 bit lange Befehlsfeld signalisiert. Um den Datenwert lesen oder schreiben zu können, muss der *Master* mit einem *Write Pointer* Befehl zunächst die Speicheradresse im *Slave* auswählen und anschließend in einer zweiten Botschaft mit einem *Read Memory* oder *Write Memory* Befehl den Datenwert lesen bzw. schreiben. Falls mehr als 256 Byte adressiert werden sollen, muss zuvor über eine weitere Botschaft mit dem *Write Page* Befehl der Adressbereich (*Page*) ausgewählt werden. Die Sensor- bzw. Aktor-Konfiguration ist in einem definierten Adressbereich abgelegt und kann vom *Master*-Steuergerät ausgelesen bzw. neu programmiert werden. Am Ende der Botschaft wird das *Error Bit* E = 0 gesendet. Übertragungsfehler können von einem Busteilnehmer signalisiert werden, indem er es mit E = 1 überschreibt. Für das Zünden von Airbags sowie die sofortige Alarmierung des Masters durch einen Crash-Sensor existieren eine Reihe von speziellen Mechanismen.

Mit dem Aufkommen von PSI5 und SENT ist es um ASRB ruhiger geworden. Der zugehörige ISO 22896 Standard ist seit 2006 unverändert.

2.4.4 DSI – Distributed Systems Interface

DSI ist der Vorschlag eines weiteren Firmenkonsortiums, das mit Rückhaltesystemen auf dieselben Anwendungen zielt wie ASRB und PSI5. DSI verbindet ein Master-Steuergerät mit bis zu 15 Slave-Geräten. In der Protokollversion DSI 2.5 sendet der Master ein spannungsmoduliertes Signal, bei dem die Bitinformation im Tastverhältnis enthalten ist. Das

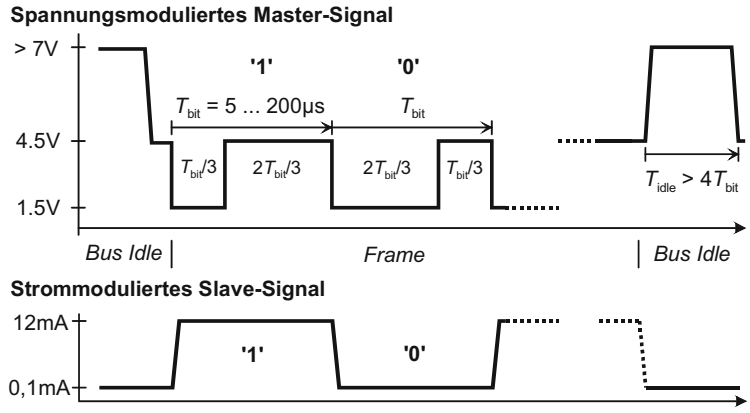

Abb. 2.35 Bitübertragung bei DSI 2.5

Spannungssignal dient gleichzeitig zur Versorgung der Slaves (Abb. 2.35). Die Kommunikation ist bidirektional. Ein Slave antwortet, indem er im vom Master vorgegebenen Bittakt seine Stromaufnahme zwischen 0,1 und 12 mA moduliert.

Das Standard-Botschaftsformat des Masters enthält ein Datenbyte, die 4 bit lange Slave-Adresse und einen 4 bit langen Befehlscode sowie eine 4 bit CRC-Prüfsumme (Abb. 2.36).

Der Slave antwortet zeitversetzt, d. h. wenn der Master bereits die nächste Befehlsbotschaft sendet. Die Antwort enthält zwei Datenbyte und wieder eine 4 bit CRC-Prüfsumme. Durch einen speziellen Befehl kann der Master das Format der Botschaften umschalten. Das Standard-Format wird als *Standard Long Word* bezeichnet. Beim *Short Word Format* entfällt jeweils das erste Byte, beim *Enhanced Format* ist die Daten- bzw. CRC-Länge konfigurierbar.

Mit DSI 3 wurde 2011 ein massiv modifiziertes Nachfolgeprotokoll vorgestellt, das sich als Alternative zu PSI 5 positioniert. Das Grundkonzept der Spannungs- und Strommodulation wird beibehalten, doch ändern sich die Signalpegel und das Zeitverhalten. Es werden zwei Geräteklassen und zwei Übertragungsmodi unterschieden. Im einfachsten Fall (*Sensor Class*) sind nur Sensoren an das Bussystem angeschlossen, die periodisch Messdaten zum Steuergerät (*Master*) übertragen. Dieser *Periodic Data Collection Mode* entspricht annähernd dem synchronen Betrieb von PSI 5. Im Ruhezustand versorgt das DSI 3-

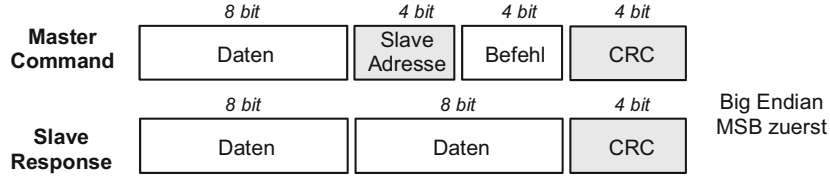

Abb. 2.36 Botschaftsformat beim Distributed Systems Interface DSI 2.5

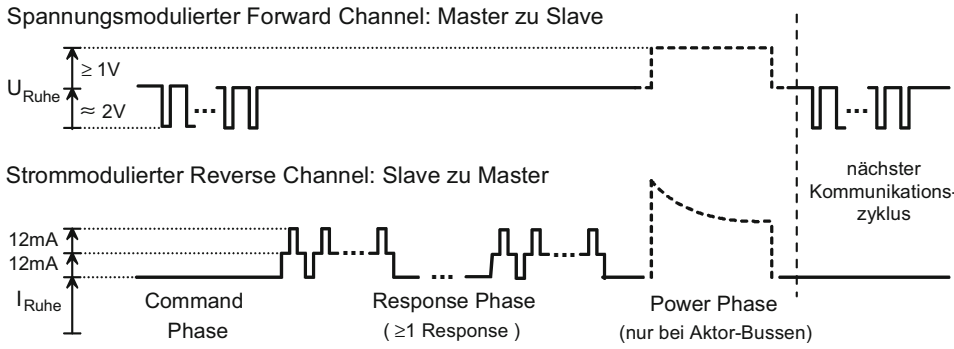

Abb. 2.37 Bitübertragung bei DSI 3

Steuergerät die Sensoren mit einer konstanten Spannung und die Sensoren halten ihre Ruhestromaufnahme konstant (Abb. 2.37). Zur Synchronisation reduziert das Steuergerät periodisch kurzzeitig die Spannung um etwa 2 V. Daraufhin senden die Sensoren zeitlich hintereinander ihre Messdaten. Die Bitcodierung erfolgt wiederum durch Strommodulation, wobei der jeweilige Sensor seine Stromaufnahme um 12 bzw. 24 mA erhöht. Um Übertragungsfehler zu erkennen, dürfen nicht alle möglichen Kombinationen der drei Strompegel verwendet werden, so dass im Schnitt etwa 4 bit je 3 Taktschritte übertragen werden. Der Bittakt beträgt typ. 125 kbit/s, d. h. $T_{\text{bit}} = 8\,\mu\text{s}$. Das Format der Messdatenbotschaft (*Periodic Data*) ist in weiten Grenzen konfigurierbar (Abb. 2.38). Die Zyklusdauer ist 100 µs bis 5 ms.

Statt der einfachen Synchronisationsimpulse kann das Steuergerät auch langsam selbst Daten an die Sensoren schicken (*Background Diagnostic*), dies ist vergleichbar mit dem *Serial Channel* von PSI 5 und SENT. Für die Bitübertragung des Steuergeräts wird eine Manchester-Codierung (vergl. Abb. 2.30) verwendet, wobei DSI 3 wie bei den einfachen Synchronisationsimpulsen die Sensorspannung um mindestens 2 V moduliert (Abb. 2.37). Die vom Steuergerät gesendete *Command* Botschaft wird dabei über mehrere Perioden verteilt (konfigurierbar). Damit die Sensoren neben den Messdaten auch explizit auf die *Command Botschaften* mit *Response Botschaften* antworten können, müssen dafür Zeitschlitze der *Response Phase* reserviert werden.

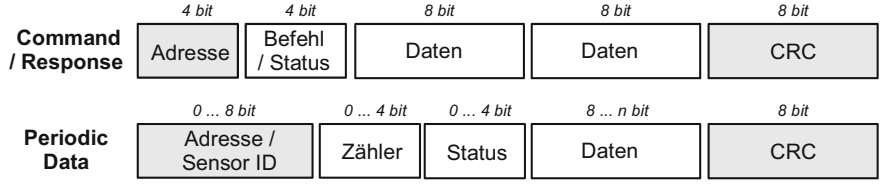

Abb. 2.38 Botschaftsformat bei DSI 3

Sind am Bus nicht nur Sensoren sondern auch Aktoren angeschlossen, ist die Latenz bei der Übertragung von *Command Botschaften* oft kritisch und die Stromaufnahme von Aktoren ist erheblich höher als bei einfachen Sensoren. Für solche *Power Class* Anwendungen ist der reine *Command Response* Übertragungsmodus vorgesehen. Das Steuergerät sendet durch Spannungsmodulation eine vollständige *Command Botschaft*, worauf ein oder mehrere Sensoren mit *Response Botschaften* im Format nach Abb. 2.38 antworten. Das frei konfigurierbare *Periodic Data Format* kann in diesem Modus nicht verwendet werden. Zusätzlich wird je Zyklus eine *Power Phase* eingefügt, in der das Steuergerät die Versorgungsspannung um mindestens 1 V anhebt. Die Aktoren können in dieser Phase ihre internen Pufferkondensatoren aufladen, dabei muss das Steuergerät bis zu 400 mA liefern. Aufgrund der hohen, nicht konstanten Stromaufnahme ist in dieser Phase keine Datenübertragung möglich.

2.5 Normen und Standards zu Kap. 2

K-Line	ISO 9141 Road Vehicles – Diagnostic Systems – Requirements for interchange of digital information, 1989, www.iso.org, ISO 9141-2 Road Vehicles – Diagnostic Systems – Part 2: CARB Requirements for interchange of digital information, 1994 und 1996, www.iso.org ISO 9141-3 Road Vehicles – Diagnostic Systems – Part 3: Verification of the communication between vehicle and OBD II scan tool, 1998, www.iso.org ISO 14230 siehe Kap. 5
SAE J1850	SAE J1850 Class B Data Communications Network Interface, 2006, www.sae.org
Sensor-Aktor-Busse	SAE J2716 SENT – Single-Edge Nibble Transmission for Automotive Applications, 2010, www.sae.org PSI 5 Peripheral Sensor Interface for Automotive Applications, Technical Specification V1.3, 2008, www.psi5.org PSI 5 Peripheral Sensor Interface for Automotive Applications, Technical Specification V2.0, 2011, www.psi5.org, Grundstandard und domänenspezifische Dokumente für Airbag, Vehicle Dynamics und Powertrain PSI 5 Peripheral Sensor Interface for Automotive Applications, Technical Specification V2.1, 2012, www.psi5.org, Grundstandard sowie domänenspezifische Dokumente Safe-by-Wire Plus: Automotive Safety Restraints Bus Specification (ASRB) V2.0, 2004, www.nxp.com/acrobat_download/other/automotive/safe_by_wire_plus.pdf ISO 22896 Road Vehicles – Deployment and sensor bus for occupant safety systems (entspricht ASRB V2.0), 2006, www.iso.org DSI (Distributed Systems Interface Standard) Bus Standard V2.5, 2009, www.dsiconsortium.org DSI3 Bus Standard Rev. 1.0, 2011, www.dsiconsortium.org

Literatur

[1] A. S. Tanenbaum: Computer Networks. Prentice Hall, 5. Auflage, 2010

[2] G. Schnell, B. Wiedemann: Bussysteme in der Automatisierungs- und Prozesstechnik. Springer-Vieweg Verlag, 8. Auflage, 2011

[3] O. Strobel (Hrsg.): Communication in Transportation Systems. IGI Global EBook, http://www.igi-global.com, 2013

[4] R. K. Jurgen (Hrsg.): Automotive Electronics Handbook. McGraw Hill Verlag, 2. Auflage, 1999

[5] R. K. Jurgen (Hrsg.): Multiplexing and Networking. SAE Verlag, 1999

[6] D. Paret: Multiplexed Networks for Embedded Systems: CAN, LIN, FlexRay, Safe by Wire. Wiley & Sons Verlag, 1. Auflage, 2007