Transportprotokolle

4

Alle aktuellen Bussysteme definieren Data Link Layer Botschaften mit begrenzter Nutzdatenlänge. So kann CAN nach ISO 11898 (Abschn. 3.1) maximal 8 Nutzdatenbytes, FlexRay nach ISO 17458 (Abschn. 3.3) maximal 254 Byte Daten je Botschaft übertragen. Die Übertragung größerer Datenblöcke sowie Adressierungsverfahren, die die Weiterleitung von Botschaften über Gateways zwischen verschiedenen Netzen erlauben, sind zunächst nicht spezifiziert. Für Diagnoseanwendungen oder zur Flash-Programmierung von Steuergeräten sind derartige Übertragungen aber notwendig.

Daher ist für CAN bzw. FlexRay ein Protokoll für die Transportschicht notwendig, d. h. für die Ebenen 3 und 4 des OSI-Schichtenmodells. Dessen Aufgaben sind:

- Übertragung von Datenblöcken, die größer sind als die *natürliche* Blockgröße des Data Link Layers (*Segmentierung, Desegmentierung*)
- Flusskontrolle (*Flow Control*), d. h. Steuerung des zeitlichen Abstands der Datenblöcke so, dass der Empfänger nicht überlastet wird, und zeitliche Überwachung der gesamten Kommunikation (*Timeout*)
- Weiterleiten von Botschaften in fremde Netze, d. h. in ein Netz mit einem anderen Adressraum oder einer anderen Adressierungsmethode als das lokale Netz.

Leider haben sich im Laufe der Zeit mehrere Lösungen etabliert. Neben dem in ISO 15765-2 standardisierten Transportprotokoll, das bei KWP 2000 on CAN und UDS on CAN verwendet wird, werden bei VW/Audi die auf einer frühen OSEK COM Version beruhenden Protokolle TP 1.6 bzw. TP 2.0 eingesetzt. Auch das bei Nutzfahrzeugen übliche SAE J1939 enthält eine eigene Transportschicht. Für FlexRay hat AUTOSAR ein inzwischen in ISO 10681-2 genormtes Transportprotokoll vorgeschlagen, das sich an ISO 15765-2 orientiert, aber mehrfach verändert wurde und nicht mehr wirklich aufwärts kompatibel ist. Für CAN FD ist zu erwarten, dass das bisherige CAN-Transportprotokoll ISO 15765-2 ebenfalls modifiziert werden muss.

W. Zimmermann und R. Schmidgall, *Bussysteme in der Fahrzeugtechnik*, ATZ/MTZ-Fachbuch, 153
DOI 10.1007/978-3-658-02419-2_4, © Springer Fachmedien Wiesbaden 2014

4.1 Transportprotokoll ISO TP für CAN nach ISO 15765-2

Wie in Abschn. 3.1 dargestellt, können in einer CAN-Botschaft maximal 8 Datenbytes übertragen werden. Die Nachricht wird durch einen *Message Identifier* gekennzeichnet. Eine direkte Adressierung von Sender und Empfänger erfolgt nicht. Um das ursprünglich nur für K-Line-Verbindungen definierte KWP 2000-Diagnoseprotokoll, bei dem größere Datenblöcke sowie eine Sender- und Empfängeradressierung notwendig sind, auch über CAN-Verbindungen implementieren zu können, wurde in ISO 15765-2 ein entsprechendes Transportprotokoll (TP) definiert, das sich aber nicht nur für Diagnose- sondern auch für andere Aufgaben verwenden lässt. Das Protokoll beschreibt:

- Die Abbildung von 4 Transportprotokoll-Botschaftstypen auf CAN-Botschaften.
- Die Aufteilung (*Segmentierung*) von größeren Datenblöcken bis zu 4095 Byte auf einzelne CAN-Botschaften, die auf der Empfängerseite wieder zusammengesetzt werden (*Desegmentierung*).
- Ein Verfahren zur Flusssteuerung zwischen Sender und Empfänger.
- Die Abbildung der KWP 2000-Sende-Empfänger-Adressen auf CAN-Message-IDs.
- Dienste (Services) zur Kommunikation zwischen Application und Transport Layer.

4.1.1 Botschaftsaufbau

Aus Sicht der Anwendung wird zwischen Adressierungsinformation (AI) und den eigentlichen Daten (*SDU…Service Data Unit*) unterschieden. Die Adressierungsinformation (*Normal Addressing*) wird bei der Übertragung auf den CAN-Identifier abgebildet (Abb. 4.1). Bei der so genannten erweiterten oder gemischten Adressierung (Extended und Mixed Addressing), die bei Netzen mit Gateways verwendet wird, enthält zusätzlich das erste Nutzdatenbyte der CAN-Botschaft eine weitere Adressinformation. Als erstes bzw. nächstes Byte der CAN-Botschaft fügt das Transportprotokoll ein zusätzliches Steuerbyte (*PCI … Protocol Control Information*) hinzu. Dieses zeigt an, wie die Nutzdaten zu interpretieren sind. In den restlichen Bytes der insgesamt max. 8 Byte langen CAN-Botschaft folgen die Nutzdatenbytes der Anwendung.

Falls alle Anwendungsdaten in die verbleibenden Datenbytes der CAN-Botschaft passen, wird ein sogenannter *Single Frame SF* gesendet. Dabei besteht die *Protocol Control Information PCI* aus einem Byte, wobei Bit 7…4 auf 0 gesetzt werden, Bit 3…0 enthalten die Anzahl der folgenden Nutzdatenbytes (*DL … Single Frame Data Length*).

Falls die Anwendungsdaten nicht in eine einzelne CAN-Botschaft passen, müssen die Anwendungsdaten auf mehrere CAN-Botschaften verteilt werden (Segmentierung, *Multi Frame* Botschaften). Zuerst wird ein sogenannter *First Frame FF* gesendet. Dabei besteht PCI aus zwei Bytes. Die oberen 4 bit des ersten PCI-Bytes enthalten den Wert 1 h, die unteren 4 bit des ersten PCI-Bytes und das gesamte zweite PCI-Byte, insgesamt 12 bit, enthalten die Länge der Nutzdaten (*FF_DL … First Frame Data Length*). Trotz des etwas verwirren-

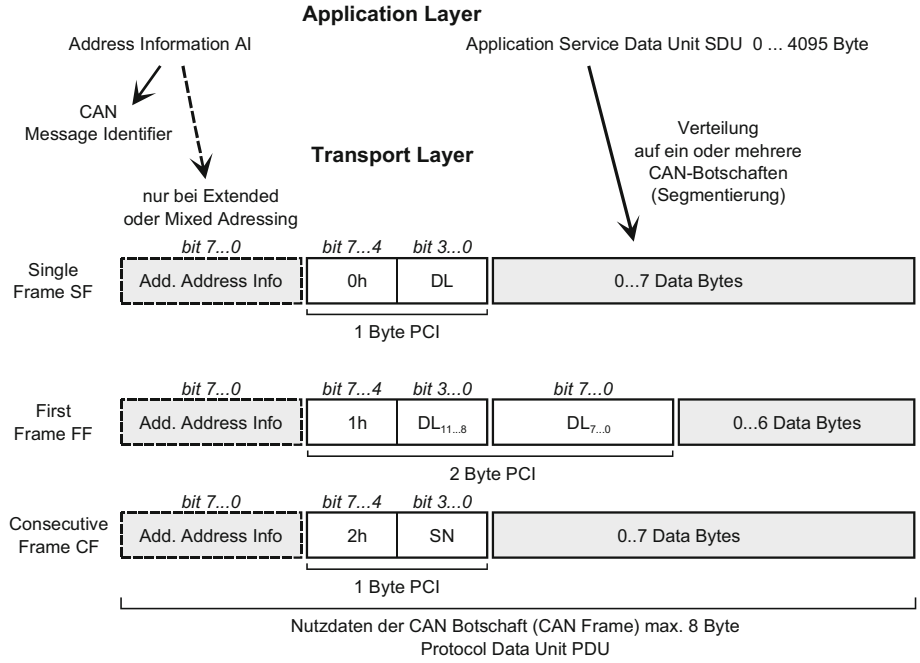


Abb. 4.1 Zuordnung zwischen Anwendungsdaten und CAN-Botschaften

den Namens *First Frame Data Length* ist dabei die Gesamtlänge der Anwendungsdaten und nicht nur die Länge der im *First Frame* selbst enthaltenen Daten gemeint. Anschließend folgen sogenannte *Consecutive Frames CF*, bis die gesamten Anwendungsdaten versendet worden sind. Bei einem *Consecutive Frame* besteht PCI aus einem Byte, dessen obere 4 bit den Wert 2 h und die unteren 4 bit die sogenannte Sequenznummer SN enthalten. Die zu einem Anwendungsdatensatz gehörenden CAN-Botschaften werden in aufsteigender Reihenfolge durchnummeriert, wobei der *First Frame* die Sequenznummer 0, der erste *Consecutive Frame* die Nummer 1, der nächste Frame die Nummer 2 usw. erhält.

Da das Feld für die Sequenznummer innerhalb des PCI Bytes nur 4 bit lang ist, wird die Sequenznummer Modulo 16 übertragen.

Durch das bei jeder CAN-Botschaft übertragene, mindestens 1 Byte lange PCI-Feld reduziert sich die Nutzdatenrate auf im günstigsten Fall 7/8 = 87 % der in Abschn. 3.1 für CAN angegebenen Maximalwerte, d. h. ca. 26 KB/s (CAN mit einer Bitrate von 500 kBit/s und 29 bit Identifiern).

4.1.2 Flusssteuerung, Zeitüberwachung und Fehlerbehandlung

Single Frame Botschaften werden vom Sender spontan gesendet. Da der CAN Data Link Layer die Fehlersicherung und Übertragungswiederholung selbstständig übernimmt, erfolgt auf der Ebene des Transportprotokolls keine Bestätigung des Empfängers. Der Mindestabstand zwischen aufeinanderfolgenden *Single Frame* Botschaften ist nicht spezifiziert.

Für segmentierte *Multi Frame* Botschaften ist der in Abb. 4.2 dargestellte Ablauf vorgesehen. Der Sender sendet die erste Datenbotschaft (*First Frame*) und wartet dann auf eine *Flow Control FC* Botschaft vom Empfänger (Abb. 4.3). In der *Flow Control* Botschaft teilt der Empfänger dem Sender mit, welche Anzahl BS (*Block Size*, BS = 1 … 255) von weiteren CAN-Botschaften (*Consecutive Frames*) er unmittelbar aufeinander empfangen kann und welche Mindestzeit ST_{min} (*Separation Time*) der Sender zwischen den einzelnen Botschaften pausieren muss. Nachdem der Sender die BS weiteren Botschaften gesendet hat, wartet er auf eine erneute *Flow Control* Botschaft vom Empfänger, bevor er weiter sendet. Dieser Ablauf wiederholt sich, bis alle Anwendungsdaten übertragen sind. Falls der Empfänger BS = 0 signalisiert, darf der Sender beliebig viele *Consecutive Frames* senden, ohne nochmals auf eine *Flow Control* Botschaft warten zu müssen. Statt einer *Flow Control* Botschaft mit FS = 0 h (*Flow State Clear To Send*) kann der Empfänger auch FS = 1 h (*Wait*) senden. Damit wird der Sender aufgefordert, mit dem Senden solange zu warten, bis er eine weitere *Flow Control* Botschaft erhält. Diese kann ihn dann zur Fortsetzung des Sendens oder zum weiteren Warten auffordern. Um ein endloses Warten zu verhindern, ist die Anzahl der aufeinanderfolgenden Wait-Anforderungen begrenzt. Ist die Datenlänge der segmentierten Botschaft für den Empfänger zu groß, meldet er dies mit FS = 2 h (*Overflow*).

Der Sender führt beim Warten auf eine *Flow Control* Botschaft, der Empfänger beim Warten auf den nächsten *Consecutive Frame* eine Timeout-Überwachung (Defaultwert 1 s) durch. Im Fall eines Timeouts oder beim Empfang einer Botschaft mit einer falschen Sequenznummer erfolgt eine Fehlermeldung an den eigenen Application Layer, aber keine Fehlermeldung zur Gegenseite, d. h. falls Sender und Empfänger Fehlermeldungen austauschen sollen, muss dies auf der Ebene der Anwendung geschehen. Die fehlerhaft empfangenen Daten selbst werden vom Empfänger ignoriert.

Im Unterschied zu üblichen CAN-Botschaften für Steuer- und Regelaufgaben, die meist von mehreren Steuergeräten empfangen und weiterverarbeitet werden (1:n Kommunikation), ist dies hier nur bei *Single Frame* Botschaften möglich. Segmentierte *Multi Frame* Botschaften dagegen müssen in der Praxis über den CAN Identifier genau ein einziges Steuergerät als Empfänger ansprechen (1 : 1 Kommunikation), da die *Flow Control* Botschaften zur Flusssteuerung nur von einem einzigen Empfänger zurückgesendet werden dürfen (siehe auch Abschn. 4.1.5).

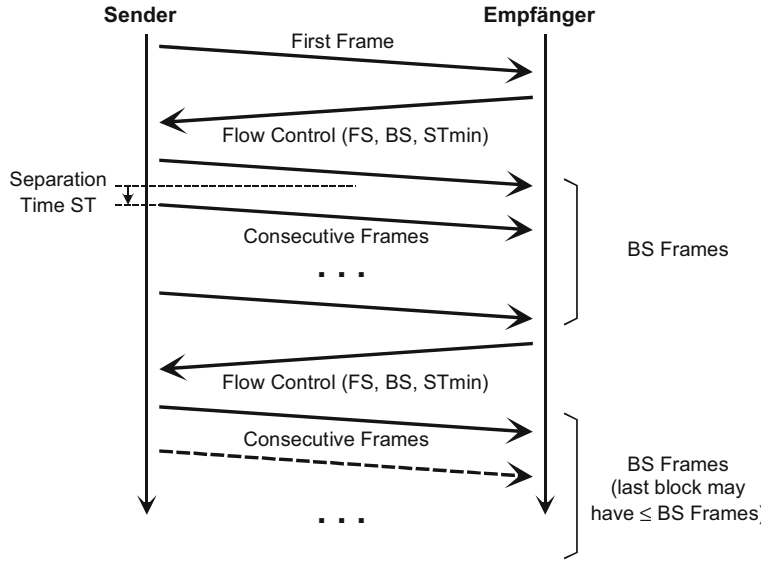


Abb. 4.2 Flusssteuerung bei Multi Frame Botschaften

4.1.3 Dienste für die Anwendungsschicht (Application Layer Services)

ISO 15765-2 spezifiziert zwar keine Programmier-API, definiert aber zumindest die in einer derartigen API mindestens zu implementierende Funktionalität (Abb. 4.4). Leider fehlen dabei präzise Angaben über die Handhabung des Sende- bzw. Empfangspuffers im Zusammenspiel zwischen Application Layer und Transport Layer:

- *Service Data.request*: Aufforderung zum Senden eines Datenblocks von bis zu 4095 Bytes an eine vorgegebene Adresse. Die Anwendung erhält eine Bestätigung bzw. Fehlermeldung *Data.confirm*, wenn der Block vollständig gesendet wurde, so dass der Sendepuffer für die nächste Botschaft wieder verwendet bzw. freigegeben werden kann. Durch geeignetes, in der Norm aber nicht definiertes Zusammenspiel zwischen Application und

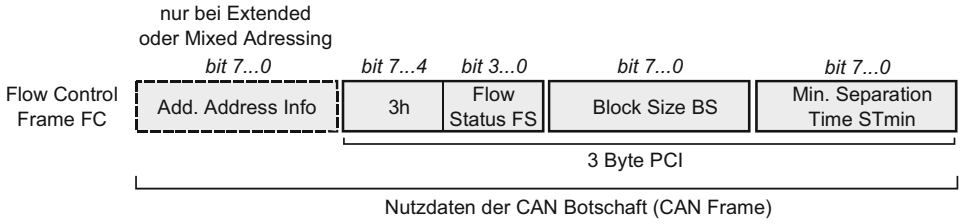


Abb. 4.3 Flow Control CAN-Botschaft (FS = CTS = 0 h Clear To Send)

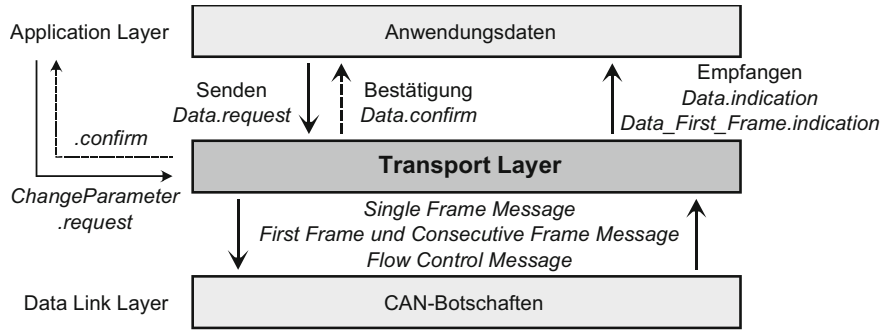


Abb. 4.4 Schnittstelle zwischen den Schichten

Transport Layer ist es auch möglich, Botschaften auf der Senderseite inkrementell zu erstellen und zu senden.

- *Service Data.indication*: Die Anwendung erhält eine Mitteilung, dass ein vollständiger Datenblock von bis zu 4095 Byte empfangen wurde. Zusätzlich erhält die Anwendung eine Information, wenn der erste Datenblock einer *Multi Frame* Botschaft empfangen wurde (*Data_First-Frame.indication*). Mit dem *First Frame* ist die Länge der zu erwartenden Botschaft bereits bekannt, so dass der Transport Layer oder der Application Layer einen ausreichend dimensionierten Empfangspuffer bereitstellen kann. Falls bei Datenblöcken im KB-Bereich kein ausreichend großer Empfangspuffer verfügbar ist, signalisiert der Empfänger dem Sender über die *Flow Control* Botschaften die tatsächliche Puffergröße. Er muss die Daten dann verarbeiten, sobald der Puffer gefüllt ist. Der dafür notwendige Dienst, mit der der Transport Layer dem Application Layer mitteilen kann, dass der Empfangspuffer voll ist, wird in der Norm nicht spezifiziert. Problematisch bei der inkrementellen Verarbeitung auf der Empfangsseite ist, dass die Botschaft zu diesem Zeitpunkt noch nicht vollständig ist, so dass bei einem später folgenden Übertragungsfehler, z. B. durch einen Fehler auf der Senderseite, ein unvollständiger, aber schon weiter verarbeiteter Datensatz vorliegt. Dieses Problem, das zum Beispiel auftritt, wenn bei einem Flash-Programmiervorgang die Übertragung der zu programmierenden Daten abgebrochen wird, obwohl bereits ein Teil des Flash-ROMs neu programmiert wurde, muss durch geeignete Maßnahmen im Application Layer gelöst werden.
- *Service ChangeParameter.request*: Die Anwendung kann die Parameter für BS und ST im Bereich von 1…255 verändern und erhält eine Bestätigung bzw. Fehlermeldung *ChangeParameter.confirm* vom Transport Layer. Die Gegenseite erfährt von dieser Änderung erst durch die nächste Flow Control Botschaft.

Die Timeout-Werte sind nach ISO 15765-2 konstant 1 s. Die Anzahl der maximal akzeptablen, aufeinander folgenden Flow Control Wait-Botschaften ist eine Applikationskonstante für den Sender und den Empfänger. Sie wird zwischen Sender und Empfänger nicht automatisch ausgehandelt.

Die Entscheidung, ob ein Steuergerät gleichzeitig senden und empfangen kann (Voll-Duplex) oder nicht (Halb-Duplex), bleibt der Implementierung überlassen. In jedem Fall soll ein Steuergerät aber in der Lage sein, gleichzeitig mit mehreren anderen Steuergeräten, d. h. Geräten mit unterschiedlichen Adressen, zu kommunizieren. Wobei ein Empfänger den Sender aber gegebenenfalls über *Flow Control Wait* Botschaften bremsen kann.

4.1.4 Protokoll-Erweiterungen

Die Norm definiert, wie dargestellt, 4 Botschaftstypen (High Nibble des PCI-Bytes 0 h…3 h). Die verbleibenden Werte sind für zukünftige Erweiterungen reserviert. Botschaften mit unbekanntem PCI-Wert oder anderen Fehlern wie einer falschen Sequenznummer sollen vom Empfänger ignoriert werden.

4.1.5 Adressierung bei KWP 2000/UDS – Zuordnung von CAN Identifiern

Beim Diagnoseprotokoll KWP 2000 werden im Application Layer der Sender (*Source Address*) und der Empfänger (*Target Address*) mit je einer 1 Byte großen Adresse gekennzeichnet. ISO 15765-2 und -3 geben Empfehlungen, wie diese logische Adressierung auf CAN Message Identifier abgebildet werden soll. Für die Diagnose abgasrelevanter Steuergeräte werden die Identifier in ISO 15765-4 fest vorgegeben, zu weiteren Einzelheiten siehe Abschn. 5.1.3.

Weiterhin enthält der Anhang der Norm einen Vorschlag, wie die CAN Message Identifier festgelegt werden sollen, wenn das ISO 15765-2 Transportprotokoll in einem CAN-Netz verwendet wird, in dem Steuergeräte zusätzlich mit Botschaften nach dem SAE J1939-Standard kommunizieren, wie er bei Nutzfahrzeugen üblich ist.

4.1.6 Bandbreite des ISO TP für CAN

Während in der On-Board-Kommunikation vor allem die Latenzzeiten (Abschn. 3.1.7 und 3.3.6) für die Datenübertragung wichtig sind, interessiert bei der Flash-Programmierung (Abschn. 9.4), dem Software-Update von Steuergeräten, vor allem die Bandbreite des Bussystems. Weil dabei große Datenblöcke übertragen werden müssen, kommt grundsätzlich ein Transportprotokoll zum Einsatz. Dadurch verringert sich die effektive Bandbreite des Bussystems gegenüber der maximalen Nutzdatenrate des reinen *Data Link Layers* [1].

Die Übertragungsdauer T_{Frame} einer einzelnen CAN-Botschaft hängt nach Gl. 3.2 neben der Bitrate von der Anzahl der Nutzdaten, der Länge des *Message Identifiers* und der Zahl der notwendigen *Stuff-Bits* ab. Für die *Stuff-Bits* können nur Minimal- und Maximalwert angegeben werden, da die tatsächliche Anzahl von den Werten des *CAN Identifiers*

und der Datenbytes der jeweiligen Botschaft abhängt. Bei der Übertragung großer Datenmengen kann angenommen werden, dass (eventuell mit Ausnahme der letzten Botschaft) alle CAN Botschaften $n_{\text{Data,DLL}} = 8$ Datenbyte enthalten und dass die mittlere Anzahl der Stuff-Bits zwischen dem Minimal- und dem Maximalwert nach Gl. 3.3 liegt. Bei einer Bitrate von 500 kbit/s ergibt sich bei 11 bit *Message Identifiern* dann für die mittlere Übertragungsdauer einer einzelnen Botschaft $T^{*}_{\text{Frame}} = 246\,\mu\text{s}$ und auf Ebene des CAN *Data Link Layers* eine mittlere Nutzdatenrate

$$f^{*}_{\text{Data,DLL}} = n_{\text{Data,DLL}} / T^{*}_{\text{Frame}} = 31,8\,\text{KB/s} \tag{4.1}$$

Dabei wird vorausgesetzt, dass der Sender CAN Botschaften schnellstmöglich sendet, d. h. im nach CAN Spezifikation zulässigen Minimalabstand von 3 Bit-Takten. In der Praxis benötigen CAN Kommunikationscontroller bzw. die zugehörige Ansteuersoftware oft länger. Außerdem wurde angenommen, dass keine anderen CAN Botschaften den Bus blockieren. Diese Voraussetzung ist in einem typischen Flash-Programmierszenario realisierbar, da die Diagnoseprotokolle KWP 2000 und UDS Diagnosedienste bieten, mit denen die normale Steuergerätekommunikation während des Programmiervorgangs abgeschaltet werden kann (siehe Kap. 5).

Durch den Overhead des Transportprotokolls verringert sich die effektive Datenrate gegenüber Gl. 4.1. Würde man eine unsegmentierte Übertragung mit ISO TP *Single Frame SF* Botschaften (Abb. 4.1) verwenden, könnte man mit jeder Botschaft nur $n_{\text{Data,SF}} = 7$ Byte Nutzdaten senden, da jeweils zusätzlich das *PCI Byte* des Transportprotokolls mit übertragen werden muss. Die Nutzdatenrate aus Sicht der höheren Protokollschichten verringert sich somit auf

$$f_{\text{Data,unseg}} = n_{\text{Data,SF}} / n_{\text{Data,DLL}} \cdot f^{*}_{\text{Data,DLL}} = 7/8 \cdot f^{*}_{\text{Data,DLL}}. \tag{4.2}$$

Dabei wurde angenommen, dass für die Adressierung ausschließlich der CAN *Message Identifier* verwendet wird (*Normal Addressing*). Setzt man dagegen das *Mixed* oder *Extended Addressing* ein, verliert man ein weiteres Nutzdatenbyte je Botschaft.

Sinnvoll ist das Transportprotokoll aber eigentlich nur, wenn man statt der unsegmentierten eine segmentierte Übertragung mit einer Folge von *First Frame FF* und *Consecutive Frame CF* Botschaften verwendet (Abb. 4.2). Der FF kann nur 6, die CF-Botschaften wieder 7 Nutzdatenbyte enthalten. Damit lassen sich Datenblöcke bis zu $n_{\text{Datablock}} \leq 4095$ Byte übertragen. Die notwendige Anzahl von Botschaften ist

$$N_{\text{FF}} = 1 \quad \text{und} \quad N_{\text{CF}} = \lceil (n_{\text{Datablock}} - 6\,\text{Byte}) / 7\,\text{Byte} \rceil \tag{4.3}$$

Zusätzlich sind noch ein oder mehrere *Flow Control FC* Botschaften notwendig. Deren Anzahl hängt von der Puffergröße des Empfängers ab, die dieser dem Sender über den *Block Size BS* Parameter der FC Botschaft mitteilt:

$$\begin{aligned} &\text{bei unbeschränktem Puffer BS} = 0: \quad N_{\text{FC}} = 1 \\ &\text{bei endlicher Puffergröße BS} > 0: \quad N_{\text{FC}} = \lceil N_{\text{CF}} / BS \rceil \end{aligned} \tag{4.4}$$

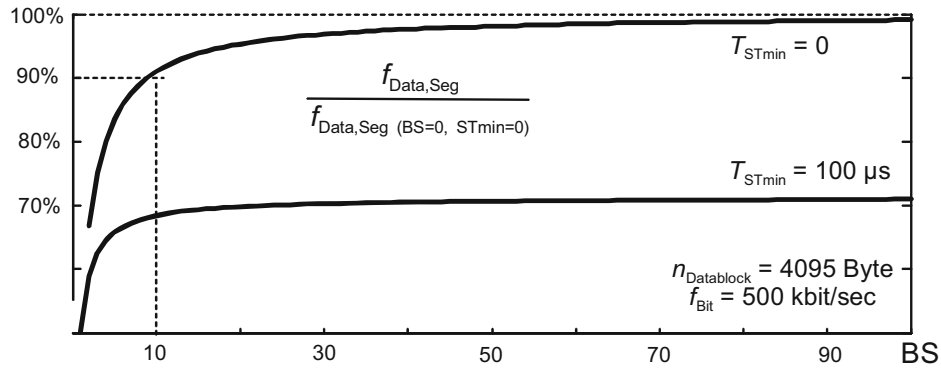


Abb. 4.5 Abhängigkeit der effektiven Datenrate von den *Flow Control* Parametern

Vernachlässigt man, dass die FC Botschaften und eventuell auch die letzte CF Botschaft etwas kürzer sind als die übrigen Botschaften, kann man die Nutzdatenrate für die segmentierte Übertragung auf der Ebene des Transportprotokolls abschätzen:

$$f_{\text{Data,seg}} \approx \frac{n_{\text{Datablock}}}{(N_{\text{FF}} + N_{\text{CF}} + N_{\text{FC}}) \cdot T^*_{\text{Frame}}} \tag{4.5}$$

Bei einem Datenblock von $n_{\text{Datablock}} = 4095$ Byte und einem unbeschränkten Puffer sind insgesamt 587 CAN Botschaften notwendig. Mit einem Bittakt von 500 kBit/s ergibt sich eine effektive Datenrate von 27,7 KB/s, d. h. das ISO Transportprotokoll erreicht im besten Fall etwa 87 % des Wertes des CAN *Data Link Layers* nach Gl. 4.1. Bei kleinen Puffergrößen BS auf der Empfängerseite bricht die Datenrate deutlich ein, für BS > 10 ist der Verlust mit weniger als 10 % dagegen noch akzeptabel (Abb. 4.5).

Bisher wurde angenommen, dass der Empfänger über den Parameter *Separation Time* ST_{min} in der FC Botschaft keine Pause zwischen den einzelnen CF Botschaften eingefordert hat. Für $ST_{\text{min}} > 0$ verlängert sich die Übertragung der CF Botschaften. Als Anzahl N_{ST} der einzufügenden Pausen ergibt sich

$$\begin{aligned} &\text{für BS} = 0: \quad N_{\text{ST}} = N_{\text{CF}} - 1 \\ &\text{für BS} > 0: \quad N_{\text{ST}} = N_{\text{CF}} - 1 - \lceil N_{\text{CF}}/BS - 1 \rceil \end{aligned} \tag{4.6}$$

Damit reduziert sich die Datenrate auf

$$f_{\text{Data,seg}} \approx \frac{n_{\text{Datablock}}}{(N_{\text{FF}} + N_{\text{CF}} + N_{\text{FC}}) \cdot T^*_{\text{Frame}} + N_{\text{ST}} + T_{\text{STmin}}} \tag{4.7}$$

Bereits bei $T_{\text{STmin}} = 100$ µs bricht die Datenrate massiv ein (Abb. 4.5).

4.2 Transportprotokoll für FlexRay nach ISO 10681-2

Das bisher einzige Transportprotokoll für FlexRay-Bussysteme wurde innerhalb der AUTOSAR-Initiative entwickelt (siehe Kap. 8). Nachdem es ursprünglich vollständig aufwärts kompatibel zu ISO 15765-2 war, wurde es mittlerweile mehrfach modifiziert, um die Unterschiede zwischen CAN und FlexRay besser zu berücksichtigen. FlexRay sichert die Daten bei der Übertragung zwar mit einer CRC-Prüfsumme, liefert aber im Gegensatz zu CAN weder eine Empfangsbestätigung noch wird die Übertragung im Fehlerfall automatisch wiederholt. Diese Mechanismen wurden daher innerhalb des FlexRay Transportprotokolls nachgebildet. Inzwischen befindet sich das Protokoll als ISO 10681-2 im Standardisierungsprozess. Im Folgenden sollen vor allem die Unterschiede des aktuellen Standardvorschlags zu ISO TP für CAN beschrieben werden. Die früheren AUTOSAR-Varianten wurden in der zweiten und dritten Auflage dieses Buches erläutert.

4.2.1 Botschaftsaufbau und Adressierung

Im Gegensatz zu CAN, wo die Sender- und Empfängeradressierung auf den CAN-Message-Identifier abgebildet wird, muss die Adressinformation bei FlexRay innerhalb der Botschaft übertragen werden, da es nicht sinnvoll ist, jeder Verbindung einen separaten FlexRay-Zeitschlitz zuzuordnen. Danach folgen die *Protocol Control Information* (PCI) und anschließend die Nutzdaten.

Die Empfängeradresse (*Target Address TA*) und die Senderadresse (*Source Address SA*) werden dabei jeweils als zwei Byte lange Werte übertragen (Abb. 4.6). Die Mehr-Byte-Werte werden im Big-Endian-Format dargestellt, d. h. mit dem höchstwertigen Byte zuerst. Die Adressen liegen am Anfang des FlexRay-Nutzdatenfelds, d. h. in derjenigen Position, die bei FlexRay im dynamischen Segment als eine Art optionale *Message ID* fungiert, falls das *Payload Preamble Bit* im FlexRay Botschaftsheader gesetzt ist (siehe Abschn. 3.3.2).

Diese *Message ID* kann durch den Kommunikationscontroller automatisch ausgewertet werden kann. Auf diese Weise kann ein FlexRay Zeitschlitz für Übertragungen von einem Sender zu mehreren Empfängern verwendet werden, wobei die Empfänger über das Adressfeld ähnlich wie bei CAN eine schnelle, hardwaregestützte Akzeptanzfilterung durchführen können. Speziell für Gateways, die mit sehr vielen Geräten kommunizieren,

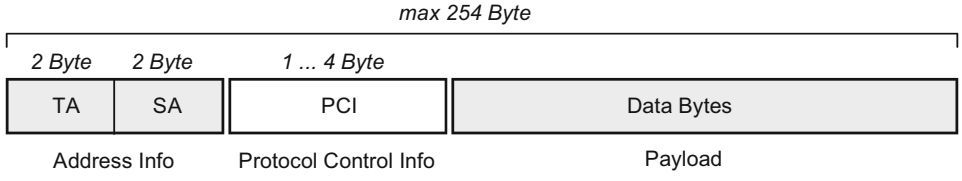


Abb. 4.6 PDU (Protocol Data Unit)-Format der FlexRay-TP-Botschaften

Tab. 4.1 Format des PCI-Felds

Frame-Typ	*1. Byte*		*2. Byte*	*3. Byte*	*4. Byte*
	Bit 7…4	*Bit 3…0*			
Start Frame STFU	4h	ACK=0	FPL	ML	
Start Frame STFA	4h	ACK=1	FPL	ML	
Consecutive Frame CF1	5h	SN	FPL		
Consecutive Frame CF2	6h	SN	FPL		
ConsecutiveFrame CFEOB	7h	SN	FPL		
Flow Control FC_CTS	8h	FS = 3	BC	BfS	
Flow Control FC_ACK	8h	FS = 4	RET/ACK	BP	
Flow Control FC	8h	FS	FS=5 Wait, FS=6 Abort, FS=7 Overflow		
Last Frame LF	9h	0	FPL	ML	

FPL…Frame Payload Length
ML …Message Length
SN …Sequence Number
BfS…Buffer Size
ACK..Acknowledge
BfS…Buffer Size
FS…Flow Status
BC…Bandwidth Control
BP…Byte Position
RET…Retry

kann dadurch die Anzahl der FlexRay Zeitschlitze reduziert werden, die z. B. für die Diagnosekommunikation, insbesondere für Softwareupdates (*Flashen*) von Steuergeräten reserviert werden müssen [1].

4.2.2 Verbindungsarten und Übertragungsablauf

Die von CAN bekannten *Single*, *Consecutive* und *Flow Control Frames* wurden für FlexRay modifiziert, der *First Frame* entfällt (Tab. 4.1). Es sind segmentierte und unsegmentierte Übertragungen ohne und mit Bestätigung möglich, wobei auch der Fall abgedeckt wird, dass die Gesamtlänge der Daten zu Beginn einer segmentierten Übertragung noch nicht bekannt ist. Der Ablauf ist wie folgt:

- Jede Übertragung beginnt mit einem *Start Frame*. Durch das ACK Bit wird signalisiert, ob der Sender eine Empfangsbestätigung (*Acknowledge*) wünscht (STFA mit ACK = 1) oder nicht (*Unacknowledged*, STFU mit ACK = 0). Segmentierte Übertragungen, die bei CAN zur Unterscheidung mit einem *First Frame* beginnen mussten, können anhand der Längenangaben von unsegmentierten Übertragungen unterschieden werden. Im FPL-(*Frame Payload Length*)-Feld steht die Länge der innerhalb des *Start Frames* über-

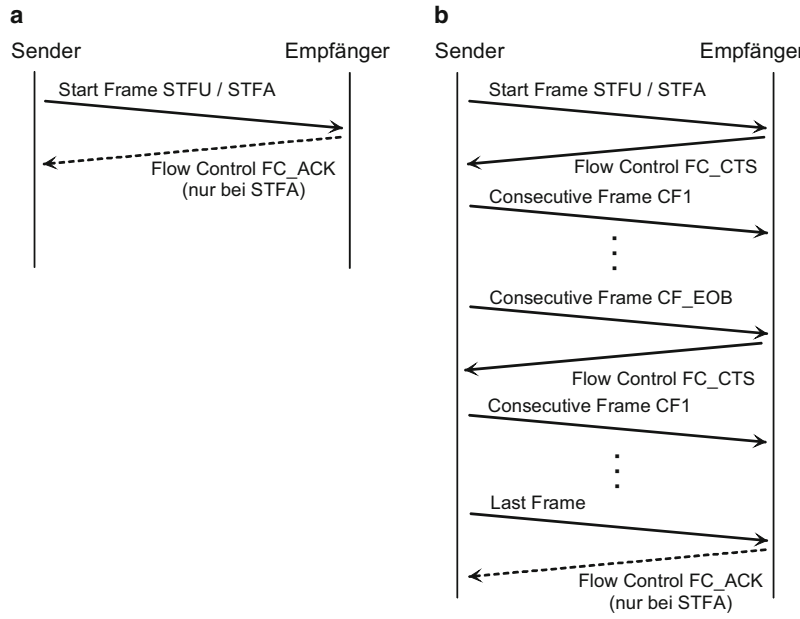


Abb. 4.7 Botschaftssequenzen bei ISO 10681-2 bei fehlerfreier Übertragung, **a** Unsegmented unacknowledged/acknowledged, **b** Segmented unacknowledged/acknowledged

tragenen Nutzdaten in Byte, im ML-(*Message Length*)-Feld die Länge der Nutzdaten der gesamten Botschaft. Sind beide Werte gleich, ist keine Segmentierung notwendig.

- Falls eine Bestätigung angefordert wurde, antwortet der Empfänger mit einem *Flow Control Frame* FC_ACK (Abb. 4.7a), der mit RET/ACK = 0 den erfolgreichen Empfang bestätigt. Mit RET/ACK = 1 kann der Empfänger den Sender auffordern, die Übertragung zu wiederholen. Dabei signalisiert er im BP Feld, ab welchem Byte die Daten fehlerhaft waren.
- Ist im *First Frame* der ML Wert größer als FPL, folgen auf den *First Frame* weitere Daten in *Consecutive Frames* (Abb. 4.7b). Die Maximallänge ist 64 kByte. Auf den *First Frame* muss der Empfänger zuerst mit einem *Flow Control Frame* FC_CTS antworten, in dem er gegebenenfalls mit dem Parameter BC eine Bandbreitensteuerung fordert (siehe Abschn. 4.2.3). Die Blockgröße, hier *Buffer Size* BfS genannt, gibt die maximale Anzahl von Bytes an, die in den folgenden *Consecutive Frames* übertragen werden darf, bevor der Sender auf einen weiteren *Flow Control Frame* warten muss. Dass der Sender einen weiteren *Flow Control Frame* erwartet, signalisiert er, indem er am Ende eines Blocks einen *End of Block Consecutive Frame* CF_EOB statt eines normalen *Consecutive Frames* sendet. Die Sequenznummer SN der *Consecutive Frames* hat dieselbe Aufgabe wie bei CAN.
- Zuletzt überträgt der Sender einen *Last Frame*. Falls dieser noch Nutzdaten enthält, wird deren Anzahl im FPL Feld angezeigt, während in ML nochmals die Datenlänge der ge-

Tab. 4.2 Bandwidth Control Parameter BC

Bit 7 … 3	Bit 2 … 0
Maximum Number of PDUs per Cycle MNPC	Separation Cycle Exponent

samten segmentierten Botschaft übertragen wird. Dies dient als Prüfmöglichkeit, zumal es auch möglich ist, im *Start Frame* zunächst ML = 0 zu senden, falls die Gesamtlänge zu Beginn der Übertragung noch nicht bekannt ist. Wenn der Sender mit dem *Start Frame* STFA eine Bestätigung angefordert hat, folgt diese am Ende der gesamten Sequenz als *Flow Control Frame* FC_ACK.

- ISO definiert zwei Typen von *Consecutive Frames*. Die segmentierte Übertragung beginnt stets mit dem Frame-Typ CF1. Wird eine Sendewiederholung vom Empfänger gefordert, so wechselt der Frame-Typ auf CF2. Bei einer weiteren Sendewiederholung wechselt der Frame-Typ zurück zu CF1 usw.
- Ähnlich wie bei CAN kann der Empfänger den Sender mit einem *Flow Control Frame* mit FS = 5 zum Warten auffordern, mit FS = 7 einen Pufferüberlauf melden oder mit FS = 6 die Übertragung abbrechen. Entsprechend sind wieder Zeitüberwachungen und maximale Übertragungswiederholungen definiert.

Die Werte für die PCI-Kennung sind so festgelegt, dass sie nicht mit den Werten nach ISO 15765-2 überlappen. Es ist daher, wie in den ersten AUTOSAR Versionen vorgesehen, auch weiterhin möglich, alternativ die CAN-kompatiblen PCI-Formate zu verwenden, wenn auf die FlexRay-spezifischen Erweiterungen verzichtet werden kann.

4.2.3 Bandbreitensteuerung

Durch die in Abschn. 4.2.1 beschriebene Adressierungsmethodik ist es möglich, die Bandbreite für die Kommunikation zwischen zwei Teilnehmern dynamisch zu verändern. Reserviert man in einem FlexRay-Zyklus mehrere Zeitschlitze für Transportprotokollbotschaften, so kann man diese Zeitschlitze bündeln, um z. B. bei der Flash-Programmierung große Datenmengen mit hoher Bandbreite an ein oder mehrere Steuergerät zu übertragen. Allerdings besteht die Gefahr, dass der Pufferspeicherbereich des Empfängers nicht ausreicht, wenn der Zeitabstand zwischen den einzelnen Botschaften zu kurz wird. Daher verwendet das FlexRay-Transportprotokoll wiederum eine Flusssteuerung über die *Flow Control Frames FC_CTS*, die aber gegenüber dem CAN-Transportprotokoll modifiziert wurde (Tab. 4.1). Neben seiner Pufferlänge BfS sendet der Empfänger den sogenannten *Bandwidth Control* BC Parameter, der eine ähnliche Rolle spielt wie die *Separation Time* bei CAN (Tab. 4.2).

Wird im BC Byte der Wert MNPC = 0 gesetzt, darf der Sender die maximale mögliche Bandbreite nutzen. Andernfalls gibt MNPC an, wie viele *Consecutive Frames* der Empfänger innerhalb ein und desselben FlexRay-Zyklus verarbeiten kann. Der Wert SCexp

definiert, welcher Mindestabstand zwischen den Frames liegen muss. Die geforderte Anzahl von FlexRay-Zyklen berechnet sich dabei zu $SC = 2^{SCexp} - 1$. Durch geeignete Wahl dieser Werte lassen sich je nach Anforderungen sowohl eine kontinuierliche Busauslastung als auch ein Burst-Betrieb realisieren.

4.2.4 Fehlerbehandlung und Implementierungshinweise

Die Behandlung von Fehlern wie falscher Datenlänge, Sequenznummer oder Überschreitung von Zeitschranken erfolgt bei unbestätigten Übertragungen ähnlich wie bei ISO TP, indem der Empfänger die fehlerhafte Botschaft einfach ignoriert. Im Fall der bestätigten Übertragung erhält der Sender eine Rückmeldung über die *Flow Control* FC_ACK Botschaft und kann die Übertragung wiederholen.

ISO 10681-2 fordert, dass Sender und Empfänger in der Lage sein müssen, gleichzeitig mehrere Verbindungen abzuwickeln, solange sich diese in der Kombination von Sender- und Empfängeradresse unterscheiden. Die Norm definiert allerdings keine Prioritäten für die parallel aktiven Verbindungen. Eine effiziente Implementierungsmethode für die Bandbreitenzuteilung ist das *Round Robin Verfahren* (Abb. 4.8). Dabei werden die einzelnen Sendepuffer für die Botschaften als Warteschlangen (*Queue*) reihum bedient. Für die Pufferverwaltung schlägt die Norm zwei Steuergrößen vor. Jede Warteschlange besitzt eine Füllstandsanzeige (*Filling Level*), die anzeigt, wie viele Transportprotokoll-Botschaften je Verbindung anstehen. Ein weiterer Zähler (*TX Pending Counter*) dokumentiert, wie viele Botschaften von der Transportschicht (in der Norm als *Communication Layer* bezeichnet) an den *Data Link Layer* übergeben wurden, aber noch nicht versendet sind. Steht dieser Zähler zu Beginn eines FlexRay-Zyklus nicht auf Null, so wurden im vorhergehenden Zyklus nicht alle Botschaften dieser Warteschlange versendet. Dies kann hauptsächlich im dynamischen Segment vorkommen, wenn eine Botschaft am Ende des FlexRay-Zyklus durch Botschaften in früheren Zeitschlitzen soweit verzögert wird, dass sie nicht mehr im laufenden Zyklus versendet werden kann (siehe Abschn. 3.3.6). In diesem Fall muss die Transportschicht warten, bis der *TX Pending Counter* wieder den Wert 0 hat, da sonst möglicherweise *Consecutive Frames* in der falschen Reihenfolge versendet werden.

Wie die Schnittstelle zwischen Transportschicht und der übergeordneten Anwendung gestaltet werden kann und wie das Transportprotokoll in den gesamten AUTOSAR Protokollstapel eingebunden ist, wird im Kap. 8 beschrieben.

4.2.5 Bandbreite des FlexRay Transportprotokolls

Bei FlexRay hängt die Nutzdatenrate auf Ebene des *Data Link Layers* davon ab, mit welcher Periodendauer T_{Cycle} der Kommunikationszyklus (*Cycle*) arbeitet, wie viele Botschaften je

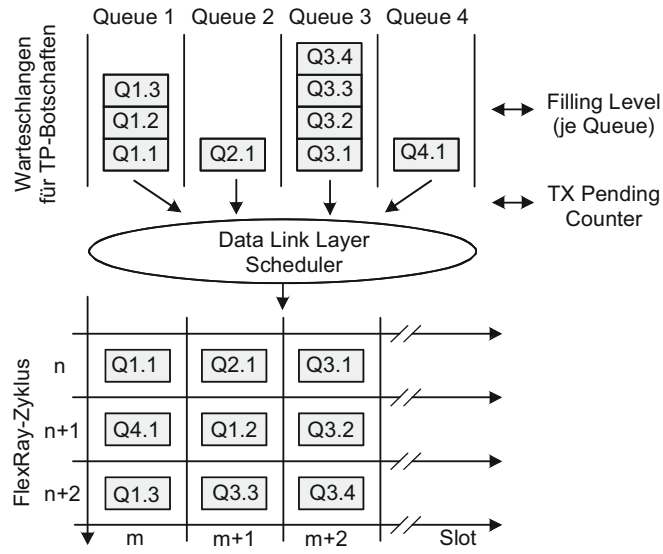


Abb. 4.8 Round Robin Scheduling von Transportprotokoll-Botschaften

Zyklus $N_{\text{FramesPerCycle}}$ für eine bestimmte Übertragung reserviert sind und wie viele Datenbytes $n_{\text{Data,DLL}}$ in einer Botschaft gesendet werden können:

$$f_{\text{Data,DLL}} = N_{\text{FramesPerCycle}} \cdot n_{\text{Data,DLL}} / T_{\text{Cycle}} \tag{4.8}$$

Die Anzahl $n_{\text{Data,DLL}}$ der Datenbytes einer Botschaft bestimmt umgekehrt die Dauer der zugehörigen *Slots* und damit zusammen mit der Aufteilung des Kommunikationszyklus in ein statisches und ein dynamisches Segment die Obergrenze für $N_{\text{FramesPerCycle}}$. Die gegenseitigen Abhängigkeiten wurden im Abschn. 3.3.6 dargestellt. Würde man beispielsweise den maximal möglichen Wert $n_{\text{Data,DLL}} = 254$ Byte je Botschaft wählen, wäre diese bei einer Bitrate von 10 Mbit/s nach Tab. 3.9 etwa 270 µs lang. In ein 2 ms langes dynamisches Segment passen dann $N_{\text{FramePerCycle}} = 7$ Botschaften je Zyklus, falls das Segment vollständig für diese Übertragung genutzt werden kann. Bei einer Zykluszeit *von* 5 ms ergibt sich dann eine Nutzdatenrate von 347 KB/s. Soll mit *Cycle Multiplexing* gearbeitet werden, ist für T_{Cycle} nicht die Periodendauer des Basis-Kommunikationszyklus sondern die effektive Multiplex-Periodendauer einzusetzen. Bereits beim *Multiplexen* mit zwei Zyklen würde sich die Datenrate halbieren.

Beim FlexRay-Transportprotokoll verliert man bei unsegmentierter Übertragung insgesamt 8 Byte je Botschaft für das Adress- und PCI-Feld des Transportprotokolls (*Start Frame* nach Abb. 4.6 und Tab. 4.1). Wegen des im Beispiel mit 254 Byte sehr großen FlexRay-Datenfelds verringert sich die Nutzdatenrate lediglich um 3 %:

$$f_{\text{Data,unseg}} = (n_{\text{Data,DLL}} - 8\,\text{Byte}) / n_{\text{Data,DLL}} \cdot f_{\text{Data,DLL}} = (254 - 8)/254 \cdot f_{\text{Data,DLL}} \tag{4.9}$$

So lange Botschaften sind üblicherweise nur im dynamischen Segment möglich. Da im statischen Segment alle *Slots* gleich lang sein müssen, legt man dieses in der Regel nur für kurze Botschaften mit beispielsweise $n_{\text{Data,DLL}} = 16$ Byte aus. Dort ist das Transportprotokoll mit seinem *Overhead* von bis zu 8 Byte nicht mehr effektiv.

Auch die segmentierte Übertragung (Abb. 4.7) beginnt bei FlexRay mit einem *Start Frame SF*. In den folgenden *Consecutive Frames CF* gehen wegen des kürzeren PCI-Feldes 6 Byte verloren. In der letzten Botschaft, dem *Last Frame LF*, sind es wieder 8 Byte. Die Anzahl der notwendigen Botschaften für einen größeren Datenblock ist

$$\begin{aligned} N_{\text{FF}} &= 1 \\ N_{\text{LF}} &= 1 \\ N_{\text{CF}} &= \lceil (n_{\text{Datablock}} - 2 \cdot (n_{\text{Data,DLL}} - 8)\,\text{Byte}) / (n_{\text{Data,DLL}} - 6)\,\text{Byte} \rceil \end{aligned} \tag{4.10}$$

Vernachlässigt man zunächst die Flusssteuerung, so kann man die Anzahl der notwendigen Kommunikationszyklen für die Übertragung abschätzen zu

$$N_{\text{Cycles}} \approx \lceil \frac{N_{\text{FF}} + N_{\text{CF}} + N_{\text{LF}}}{N_{\text{Framespercyle}}} \rceil \tag{4.11}$$

Als grober Wert für die Datenrate ohne Flusssteuerung ergibt sich dann

$$f_{\text{Data,seg}} \approx \frac{n_{\text{Datablock}}}{N_{\text{Cycles}} \cdot T_{\text{Cycle}}} \tag{4.12}$$

Um einen Datenblock von 64 KB zu übertragen, sind in der obigen FlexRay-Konfiguration 263 Botschaften notwendig, die in 38 Zyklen mit etwa 340 KB/s gesendet werden.

Berücksichtigt man die Flusssteuerung, hängt die Anzahl der notwendigen *Flow Control FC* Botschaften wieder von der Puffergröße des Empfängers ab. Da die FC Botschaften nur 8 Byte lang sind, ist ihre Übertragungsdauer mit $T_{\text{Frame,FC}} \approx 20\,\mu\text{s}$ gegenüber der im Beispiel 254 Byte langen CF Datenbotschaften mit $T_{\text{Frame,CF}} \approx 270\,\mu\text{s}$ fast vernachlässigbar. Viel kritischer ist die Frage, wo die *Slots* für die FC Botschaften innerhalb des Kommunikationszyklus relativ zu den *Slots* für die Datenbotschaften platziert werden und wie man die Blockgröße des Empfängers relativ zur Anzahl der Datenbotschaften je Zyklus wählt. Passen die Festlegungen nicht zusammen, z. B. eine Blockgröße des Empfängers von 4 Botschaften bei 7 möglichen Botschaften je Kommunikationszyklus, so verliert man durch das Warten auf die FC Botschaften unter Umständen vollständige Kommunikationszyklen. Wenn die einzelnen Steuergeräte dann auch noch unterschiedliche Blockgrößen oder Mindestabstände zwischen den Datenbotschaften anfordern und die Flash-Programmierung mit der für den normalen Fahrbetrieb optimierten Zyklus-Konfiguration erfolgen soll, ist die reale Datenrate oft erheblich schlechter als der theoretische Wert nach Gl. 4.12. Eine detaillierte Beschreibung dieser Probleme und möglicher Abhilfemaßnahmen findet sich in [1].

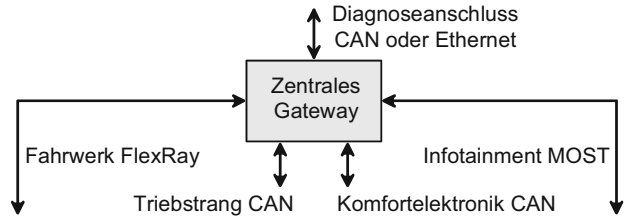


Abb. 4.9 Busstruktur aktueller Fahrzeuge

In der Regel erfolgt die Flash-Programmierung über den normalen Diagnosezugang des Fahrzeugs. Von den internen Bussystemen ist der Diagnosezugang meist durch ein zentrales *Gateway*-Steuergerät entkoppelt, an dem die internen Bussysteme des Fahrzeugs zusammenlaufen (Abb. 4.9). In einer solchen Kette bestimmt das Bussystem mit der niedersten Datenrate die effektive Bandbreite. Die Übertragung zu einem Steuergerät am internen FlexRay-Bus erfolgt also bestenfalls mit der Datenrate des CAN-Busses, wenn ein gewöhnlicher CAN-Diagnoseanschluss verwendet wird. Umgekehrt beschleunigt ein Ethernet-Diagnoseanschluss die Datenübertragung zu einem Steuergerät am internen CAN-Datenbus nicht. Da viele dieser Busse unterschiedliche *Frame* Formate und Bitraten haben, können die ankommenden Botschaften nicht einfach durchgeleitet, sondern müssen im *Gateway* zwischengespeichert und umformatiert werden. Weil im Normalbetrieb oft sehr viele Botschaften verarbeitet werden und Gateways für die Flash-Programmierung oft nicht umkonfiguriert werden können, obwohl dabei die normale Kommunikation in der Regel abgeschaltet wird, stellen kleine Datenpuffer, d. h. kleine Blockgrößen für die Transportprotokolle, einen Flaschenhals dar. Die Verarbeitungszeit im Gateway ist für die Übertragung der eigentlichen Datenbotschaften (FF, CF) unkritisch [1], wirkt aber bei der Flusssteuerung wie eine vergrößerte *Separation Time* (siehe Abb. 4.5).

4.3 Transportprotokoll TP 2.0 für CAN

Im Gegensatz zum ISO-Transportprotokoll handelt es sich hierbei um ein verbindungsorientiertes Protokoll. Dabei werden dynamisch exklusive Verbindungen zwischen zwei CAN-Teilnehmern, sogenannte *Kanäle*, eingerichtet und nach Beendigung des Datenaustausches wieder aufgelöst. Die gesamte Kommunikation gliedert sich in drei Phasen:

- **Verbindungsaufbau** (Eröffnung eines Transportkanals)
- **Datenübertragung** innerhalb des Transportkanals
- **Verbindungsabbau** (Schließen des Transportkanals).

Für den Verbindungsaufbau, aber auch für einige andere Dienste, besteht darüber hinaus die Möglichkeit zur 1:n Kommunikation (Broadcast). Im Gegensatz zu ISO-TP, bei dem die CAN Identifier statisch konfiguriert sind, werden die CAN Identifier für die Kanäle dynamisch vergeben.

TP 2.0 betreibt ein sehr aufwendiges Verfahren zur Flusskontrolle. Dabei wird jede vollständig empfangene Botschaft oder ein Block von mehreren Botschaften vom Empfänger quittiert (Handshake). Weitere wesentliche Merkmale des Transportprotokolls TP 2.0 sind:

- Die Abbildung von 10 verschiedenen Transportprotokoll-Botschaftstypen auf CAN-Botschaften.
- Die Aufteilung (Segmentierung) von beliebig großen Datenblöcken auf einzelne CAN-Botschaften, die auf der Empfängerseite wieder zusammengesetzt werden (Desegmentierung).
- Automatische Sendewiederholung bei Timeout-Fehlern.

4.3.1 Adressierung und CAN Message Identifier

Jedes Steuergerät besitzt eine fahrzeugweit eindeutige logische Adresse, die nach einem festen Schema einem CAN Message Identifier zugeordnet ist. Zusätzlich gibt es logische Adressen, unter denen mehrere Steuergeräte, z. B. alle Steuergeräte des Antriebsstrangs, zu einer Steuergerätegruppe zusammengefasst werden. Auch derartige Gruppenadressen werden jeweils einem CAN-Identifier zugeordnet. Die derartig festgelegten CAN-Identifier werden für den Verbindungsaufbau sowie andere Broadcast-Dienste verwendet und als *Eröffnungs-ID* bezeichnet:

Eröffnungs-ID = *Basis-ID* + *Logische Adresse des Gerätes*

Eindeutige Geräteadresse bzw. Broadcast-Adresse für alle Steuergeräte am Antriebsstrang-Bus, am Infotainment-Bus, am Komfort-Bus oder alle Steuergeräte im Fahrzeug

Beim Aufbau einer dynamischen Verbindung (*Kanal*) handeln Sender und Empfänger individuelle CAN-Identifier aus, die sogenannten *Kanal IDs*, welche dann für die anschließende Datenübertragung bis zum Abbau der jeweiligen Verbindung verwendet werden.

4.3.2 Broadcast-Botschaften

Broadcast-Botschaften werden grundsätzlich mit der *Eröffnungs-ID* als CAN Message Identifier gesendet und haben alle denselben schematischen Aufbau (Abb. 4.10). Alle Botschaften besitzen nur sieben Datenbytes, das achte Byte wird nicht übertragen. Im ersten Datenbyte wird die logische Adresse des Ziel-Steuergerätes gesendet. Auf diese Weise ist die Weiterleitung der Botschaften über Gateways möglich. Der *Opcode* im zweiten Byte kennzeichnet den Typ der Botschaft, anschließend folgt ein als *Service-ID SID* bezeichnetes Byte, das einen der im Application Layer definierten Dienste auswählt, sowie dessen Parameter. Bei den Diensten kann es sich sowohl um herstellerspezifische Funktionen als

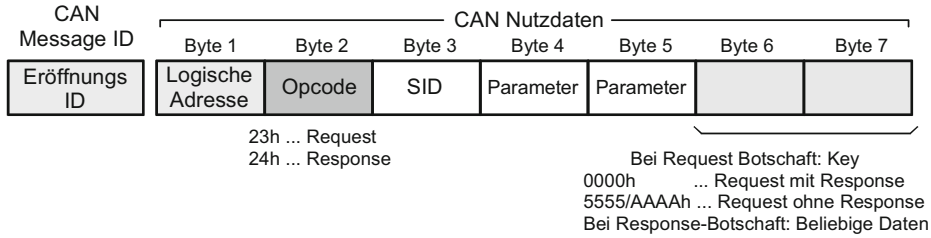


Abb. 4.10 Broadcast-Botschaften bei TP 2.0

auch um die durch die KWP 2000- bzw. UDS-Normen definierten Funktionen handeln, die in Kap. 5 beschrieben werden.

TP 2.0 unterscheidet Broadcast-Botschaften, bei denen eine Antwort (Response) vom Empfänger erwartet wird, und solche, bei denen keine Antwort notwendig ist.

Bei einer *Broadcast Request* Botschaft wird das Opcode-Byte auf den Wert 23 h gesetzt, bei der *Broadcast Response* Antwortbotschaft auf 24 h. Falls eine Antwort vom Empfänger erwartet wird, setzt der Sender im *Broadcast Request* die Bytes 6 und 7, den sogenannten *Key*, auf 0000 h. Der Sender sendet die *Request* Botschaft insgesamt fünf Mal innerhalb von 100 ms, der Empfänger führt den zugehörigen Dienst aus, sobald er die Botschaft einmal erkannt hat, und muss innerhalb von maximal 500 ms antworten. Die Reaktion bei Ausbleiben der Antwort ist nicht spezifiziert.

Broadcast Request Botschaften, bei denen der Sender keine Antwort erwartet, werden ebenfalls innerhalb von 100 ms fünf Mal gesendet, wobei in den Bytes 6 und 7 als *Key* alternierend 5555 h und AAAAh gesendet wird. Der Empfänger führt den Dienst erst dann aus, wenn er die Botschaft mit jedem der beiden *Keys* mindestens einmal innerhalb der 100 ms empfangen hat.

Bei Diensten, welche die Empfänger der Botschaft in einen besonderen Systemzustand versetzen, müssen die *Request* Botschaften gegebenenfalls anschließend in größerem Abstand, z. B. 1 s, wiederholt werden, um diesen Zustand aufrecht zu erhalten. Erkennt ein Steuergerät während dieser sogenannten Retrigger-Phase einen Timeout, fällt es aus dem Sondermodus wieder in den Normalmodus zurück.

4.3.3 Dynamischer Kanalaufbau und Verbindungsmanagement

Für die eigentliche Datenübertragung zwischen zwei Geräten müssen zunächst eine logische Verbindung, ein sogenannter Kanal, aufgebaut und die für die spätere Übertragung verwendeten CAN-Identifier ausgehandelt werden. Der dynamische Kanalaufbau erfolgt wiederum mit dem *Eröffnungs-ID* als CAN Message Identifier und stellt einen Spezialfall der Broadcast-Botschaften dar (Abb. 4.11).

Die beiden CAN-ID Felder RX ID und TX ID lassen nur Platz für 11 bit-CAN Identifier, 29 bit-Identifier werden gegenwärtig nicht unterstützt. Die unteren 8 Bit des Identifiers

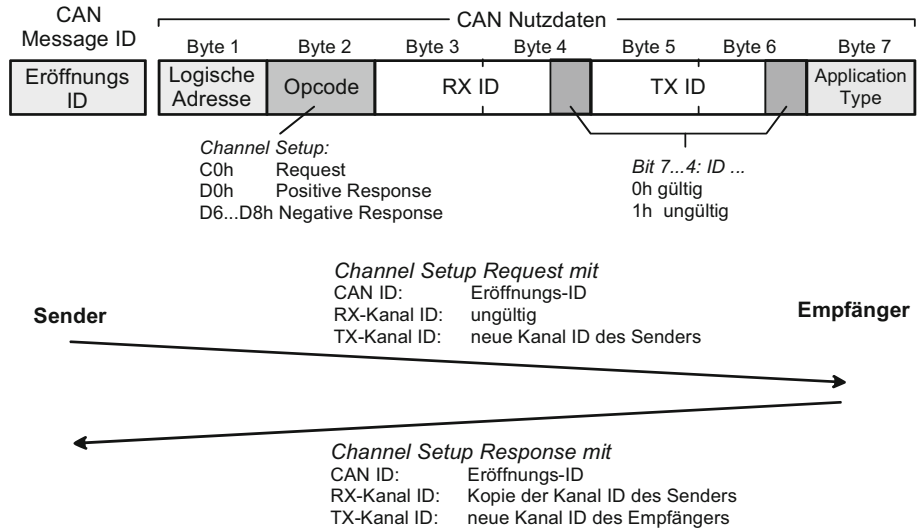


Abb. 4.11 Botschaften für den dynamischen Kanalaufbau bei TP 2.0

werden in Byte 3 bzw. Byte 5 gesendet, die oberen 3 Bit in der unteren Hälfte der Bytes 4 bzw. 6. Die oberen 4 bit dieser beiden Bytes zeigen mit dem Wert 0 h an, dass die CAN-Identifier gültig sind, mit dem Wert 1 h, dass der jeweilige CAN-Identifier ungültig ist. Dies ist für den Handshake-Ablauf beim Verbindungsaufbau notwendig:

- Der Sender sendet einen *Channel Setup Request* (Opcode C0 h) mit derjenigen CAN-ID im Feld TX ID, mit der er zukünftig Daten empfangen will. Das Feld RX ID markiert er als ungültig.
- Der Sender antwortet, falls er die Kanaleröffnung akzeptiert, mit einer positiven *Channel Setup Response* (Opcode D0 h), wobei er im Feld TX ID jetzt diejenige CAN-ID zurücksendet, mit der er zukünftig Daten empfangen will. Außerdem wiederholt er im Feld RX ID die CAN-ID, die ihm der Sender gerade als seine zukünftige *Kanal-ID* mitgeteilt hat. Das Vertauschen der Werte in den RX ID und TX ID Feldern erscheint zunächst verwirrend. Es ist aber leicht zu verstehen, wenn man sich vergegenwärtigt, dass im TX ID Feld stets diejenige CAN ID steht, an die der jeweilige Empfänger seine Antworten oder eigenen Botschaften an den anderen Partner senden (TX … transmit) muss.
- Falls der Empfänger die Verbindung ablehnt, sendet er die *Channel Setup Response* mit einem der Opcodes D6 h … D8 h, um eine dauerhafte oder zeitweise Ablehnung der Verbindung zu signalisieren. Der Timeout beim Verbindungsaufbau beträgt 50 ms. Der Sender darf den Verbindungsversuch bis zu 10mal wiederholen. Im Feld *Application Type* wird definiert, welche Anwendungsfunktion des Steuergerätes angesprochen werden soll. Der Wert 01 h spezifiziert bei VW/Audi beispielsweise die KWP 2000-Diagnose.

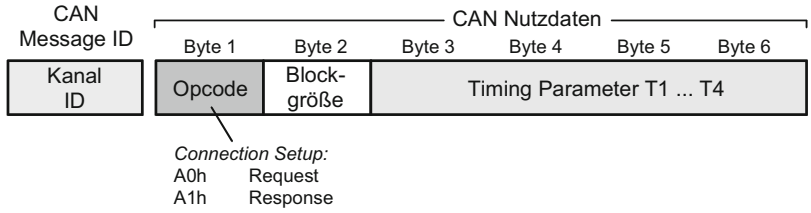


Abb. 4.12 Parametrierung von Verbindungskanälen bei TP 2.0

Nach erfolgreichem Verbindungsaufbau werden alle weiteren Botschaften bis zum Verbindungsabbau dann mit den beiden ausgehandelten *Kanal-IDs* übertragen.

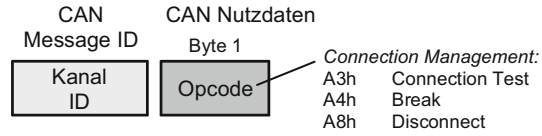
Als nächstes werden die Verbindungsparameter für den neuen Kanal ausgehandelt. Dazu sendet der Sender eine *Connection Setup Request* Botschaft mit den Verbindungsparametern des Senders (Abb. 4.12).

Das Byte 2 enthält dabei die Blockgröße, d. h. die Anzahl der CAN-Botschaften, die hintereinander empfangen werden sollen, bevor der Empfang durch eine ACK-Botschaft (siehe unten) bestätigt werden muss. Zulässige Blockgrößen sind 1…15. Die Bytes 3 bis 6 enthalten diverse Timeout-Parameter, z. B. die maximal zulässige Zeit T1 zwischen der letzten CAN-Botschaft eines Blocks und der zugehörigen Bestätigung durch die ACK-Botschaft, oder die minimal notwendige Zeit T3 zwischen zwei aufeinander folgenden CAN-Botschaften. Die Zeiten werden mit 6 bit Auflösung sowie einem 2 bit Skalierungsfaktor übertragen, der angibt, ob der Wert mit 100 µs, 1 ms, 10 ms oder 100 ms multipliziert werden muss. Die Zeiten T2 und T4 sind für Erweiterungen vorgesehen und werden im Normalfall als FFh übertragen. Der Empfänger antwortet in der *Connection Setup Response* Botschaft mit den entsprechenden Werten für seine Seite. Anschließend ist der Kanal bereit für die eigentliche Datenübertragung, die im folgenden Abschnitt beschrieben wird.

Der abschließende Abbau der Verbindung erfolgt durch eine *Disconnect* Botschaft (Abb. 4.13). Der Abbau wird durch den Empfänger bestätigt, in dem er seinerseits mit einer *Disconnect* Botschaft antwortet. Anschließend dürfen keine Botschaften mehr mit den dieser Verbindung zugeordneten *Kanal-IDs* gesendet und empfangen werden.

In komplexen Fahrzeugen werden häufig Gateways zur Anbindung des Diagnosetesters sowie zur Verbindung mehrerer Bussysteme eingesetzt. Da diese Gateways in der Regel nur Botschaften weiterleiten, die Botschaftsinhalte selbst aber nicht analysieren, sind sie auch nicht in der Lage, die für einen Kanal ausgehandelten Timeout-Zeiten zu überwachen. Sie besitzen daher ein festes Zeitraster für die Timeout-Überwachung. Dies kann unter Um-

Abb. 4.13 Weitere Botschaften für Verbindungsabbau und Verbindungssteuerung



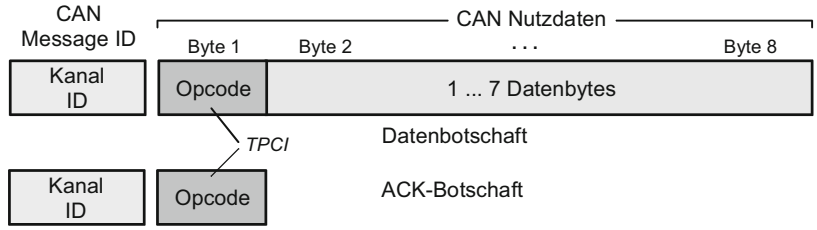


Abb. 4.14 Datenbotschaft und ACK-Botschaft bei TP 2.0

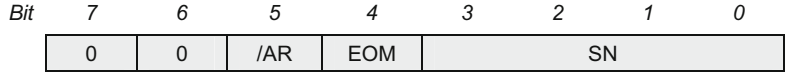


Abb. 4.15 Aufbau des Transport Control Bytes TPCI des Senders

ständen kleiner sein als die ausgehandelten Zeiten für den dynamischen Kanal. Es ist somit möglich, dass ein Gateway aufgrund dieser festen Zeiten einen Timeout für einen Kanal detektiert, obwohl die beiden Kommunikationspartner sich zeitlich korrekt verhalten. Für diesen Fall existieren *Connection Test* Botschaften, die im Zeitraster des Gateways gesendet werden. Der Empfänger antwortet mit einer *Connection Setup Response* Botschaft. Das Gateway erkennt ein aus seiner Sicht korrektes Timing und lässt den Kanal weiterhin geöffnet.

4.3.4 Datenübertragung

Die Datenübertragung erfolgt blockweise, wobei der Empfänger den Empfang jedes Blocks bestätigen muss. Ein Block besteht aus einer oder mehreren aufeinanderfolgenden CAN-Botschaften. Die maximale Blockgröße wurde beim Aufbau des Kanals mit den *Connection Setup* Botschaften ausgehandelt. Die Bestätigung durch den Empfänger erfolgt durch eine ACK-Botschaft, bevor der Sender den nächsten Block senden darf. Die Anzahl der aufeinanderfolgenden Blöcke ist beliebig.

Eine einzelne Datenbotschaft kann dabei maximal 7 Datenbyte enthalten (Abb. 4.14). Das erste Byte, das sogenannte *Transport Control Byte TPCI* des Senders, dient der Steuerung der Übertragung und hat den in Abb. 4.15 dargestellten Aufbau. Mit *Acknowledge Request/AR* = 0 signalisiert der Sender, dass er als nächstes eine ACK-Botschaft als Bestätigung erwartet, mit *End of Message EOM* = 1, dass es sich um die letzte Botschaft eines Blockes handelt. Die Sequenznummer SN ist eine fortlaufende Zahl, mit der aufeinander folgende Datenbotschaften durchnummeriert werden. Da nur 4 bit zur Verfügung stehen, wird die Sequenznummer mod 16 übertragen. Beide Kommunikationspartner besitzen einen unabhängigen SN-Zähler, der jeweils mit 0 beginnt und kontinuierlich bei jeder gesendeten Botschaft inkrementiert wird.

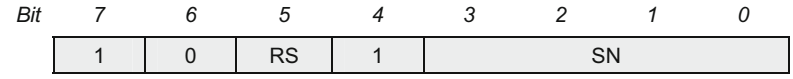


Abb. 4.16 Aufbau des Transport Control Bytes TPCI des Empfängers

Die ACK-Botschaft des Empfängers besteht nur aus dem TPCI-Byte (Abb. 4.16). SN enthält die Sequenznummer der zu quittierenden Datenbotschaft plus 1. Mit dem *Receive Status Bit RS* = 1 (*Receiver Ready*) bestätigt der Empfänger, dass er für den Empfang des nächsten Blocks bereit ist. Mit RS = 0 (*Receiver Not Ready*) wird die Übertragung weiterer *Frames* verhindert. Innerhalb der Zeit T1 muss der Empfänger dann eine weitere ACK-Botschaft senden. Ist dabei wiederum RS = 0 gesetzt, so wird die Überwachungszeit T1 neu gestartet. Wird keine weitere ACK-Botschaft gesendet, wiederholt der Sender der Datenbotschaft den zuletzt gesendeten Block. Nach einer bestimmten Anzahl solcher Versuche bricht der Sender die Übertragung ab und schließt den Kanal.

Falls der Empfänger eine Datenübertragung vorzeitig beenden will, sendet er eine BREAK-Botschaft (Abb. 4.13). Der Sender antwortet darauf mit einer Datenbotschaft mit EOM = 1. Die seit dem letzten ACK übertragenen Daten werden verworfen, der Kanal bleibt jedoch geöffnet. Der Sender wartet eine bestimmte Zeit und startet dann die Übertragung erneut. Nach einer bestimmten Anzahl von erfolglosen Versuchen bricht er die Übertragung ab und schließt den Kanal.

Für den Fall, dass auf der Ebene des Transportprotokolls mehrere CAN-Botschaften gleichzeitig zur Übertragung anstehen, soll die Protokollsoftware die Botschaften in der in Tab. 4.3 dargestellten Reihenfolge senden.

4.4 Transportprotokoll TP 1.6 für CAN

Das Transportprotokoll 1.6 (TP 1.6) ist der Vorläufer des im vorigen Kapitel beschriebenen Protokolls TP 2.0 und verwendet weitgehend dieselben Konzepte und Botschaften. Auch hier werden exklusive Verbindungen (Kanäle) zwischen zwei CAN-Teilnehmern dynamisch eingerichtet und nach Beendigung des Datenaustausches wieder aufgelöst. Die

Tab. 4.3 Botschaftsprioritäten

Höchste Priorität 1	Connection Setup Response Connection Test
	BREAK-Botschaft
	ACK-Botschaft
Niedrigste Priorität 4	Datenbotschaft Connection Setup Request Disconnect

Kanaleröffnung erfolgt mit fest konfigurierten CAN-Identifiern, den *Eröffnungs-IDs*. Die CAN-Identifier für die Datenübertragung, die *Kanal-IDs* werden beim Verbindungsaufbau ausgehandelt. Analog zu TP 2.0 betreibt auch TP 1.6 ein aufwendiges Verfahren zur Flusskontrolle. Dabei kann der Sender für jedes vollständig gesendete Telegramm oder einen Block von mehreren Telegrammen eine Quittung (Handshake) vom Empfänger anfordern. Nach jedem vollständig übertragenen Datenblock wechselt die Senderichtung. Das bedeutet, die Gegenstelle (ehemals der Empfänger) wird jetzt zum Sender. Dieser kann nun selbst Daten senden oder das Senderecht wieder zurückgeben. Damit können bidirektionale, gleichberechtigte Kanäle zwischen zwei Busteilnehmern aufgebaut werden.
Die wesentlichen Unterschiede zu TP 2.0 sind:

- keine Broadcast-Botschaften (TP 2.0: Opcode 23 h und 24 h),
- keine Botschaft zum Verbindungstest (TP 2.0: Opcode A3 h),
- keine Botschaft zur Unterbrechung der Datenübertragung (TP 2.0: Opcode A4 h).

4.4.1 Botschaftsaufbau

Wie bei TP 2.0 besitzt auch hier jedes Steuergerät eine fahrzeugweit eindeutige logische Adresse, die einem festen Anfrage- bzw. Antwortkanal zugeordnet ist. Der zugehörige CAN-Identifier wird wiederum als *Eröffnungs-ID* bezeichnet. Über diesen Kanal können Botschaften zur Anforderung eines dynamischen Kanals ausgetauscht werden.

Darüber hinaus besitzt jeder Busteilnehmer ein bis vier feste Adressen, die für die Kommunikation über einen dynamischen Kanal verwendet werden können. Für den Antriebsstrang sind die CAN-Identifier ab 740 h reserviert, für den Komfort-Bus ab 300 h und für den Infotainment-Bus ab 4E0 h.

Für den *Eröffnungs-ID* gilt:

Eröffnungs-ID = *Basis-ID* + *Logische Adresse des Gerätes*

Eindeutige Geräteadresse bzw. Broadcast-Adresse für alle Steuergeräte bzw. den Diagnosetester am Antriebsstrang-Bus, am Infotainment-Bus bzw. am Komfort-Bus

4.4.2 Dynamischer Kanalaufbau

Für die eigentliche Datenübertragung zwischen zwei Geräten muss, wie bei TP 2.0 im Abschn. 4.3.3 beschrieben, zunächst eine logische Verbindung, ein sogenannter Kanal, aufgebaut werden. Dabei werden die für die spätere Übertragung verwendeten CAN-Identifier ausgehandelt. Der dynamische Kanalaufbau erfolgt mit der *Eröffnungs-ID* als CAN-Identifier. Der Botschaftsaufbau bei TP 1.6 unterscheidet sich vom Aufbau bei TP 2.0 (Abb. 4.17).

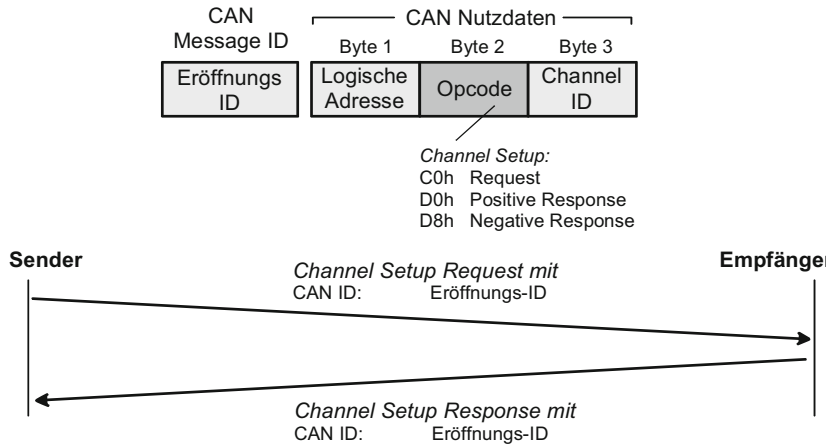


Abb. 4.17 Botschaften für den dynamischen Kanalaufbau bei TP 1.6

Das Steuergerät, das den Kanal aufbauen will, sendet in der *Channel Setup Request* Botschaft die zugehörige *Kanal-ID*, d. h. den CAN-Identifier, mit dem es zukünftig Datenbotschaften empfangen will. Das andere Steuergerät antwortet mit der *Kanal-ID*, mit der es seinerseits Datenbotschaften empfangen will.

4.4.3 Datenübertragung und Datenrichtungswechsel

Die Datenübertragung erfolgt mit den bereits im Abschn. 4.3.4 beschriebenen Datenbotschaften, die vom Empfänger jeweils durch ACK-Botschaften bestätigt werden, mit den jeweiligen *Kanal-IDs* als CAN-Identifier.

TP 1.6 fordert dabei, dass nach jedem abgeschlossenen Datenblock die Datenflussrichtung wechselt. Das Steuergerät, das die Verbindung aufgebaut hat, beginnt mit dem Senden eines Datenblocks. Sobald der Datenblock vollständig gesendet und in der letzten Botschaft des Datenblocks EOM = 1 gesetzt wurde, wird der bisherige Sender zum Empfänger und umgekehrt. Der neue Sender kann nun seinerseits einen Datenblock senden, in dessen letzter Botschaft wieder EOM = 1 gesetzt wird. Oder er gibt durch eine Datenbotschaft mit 0 Datenbytes und gesetztem EOM = 1 das Senderecht sofort wieder ab. Bei jedem Richtungswechsel wird SN wieder auf 0 gesetzt.

TP 1.6 unterscheidet darüber hinaus zwischen einem *schnellen* und einem *langsamen* Richtungswechsel für die Acknowledge-Botschaft. Beim *langsamen* Datenrichtungswechsel wird zunächst ein ACK gesendet, bevor der *neue* Sender seinerseits mit der Datenübertragung beginnt. Beim schnellen Datenrichtungswechsel wird kein ACK angefordert und der *neue* Sender beginnt sofort mit der Datenübertragung (Abb. 4.18).

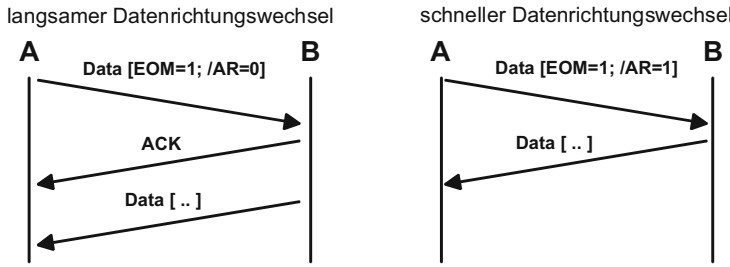


Abb. 4.18 Langsamer und schneller Datenrichtungswechsel bei TP 1.6

4.5 Transportprotokoll SAE J1939/21 für CAN

Das von der amerikanischen Society of Automotive Engineers SAE spezifizierte, auf CAN basierende Protokoll SAE J1939 wird hauptsächlich im Nutzfahrzeugbereich verwendet. Dabei werden in mehreren Dokumenten die verschiedenen Ebenen des ISO/OSI-Referenzmodells beschrieben (Tab. 4.4). Bei der Definition wurde versucht, Strukturen des früher in amerikanischen Nutzfahrzeugen stark verbreiteten Busprotokolls nach SAE J1708 (zeichenorientiertes Busprotokoll mit 9600 bit/s und einer physikalischen Schnittstelle ähnlich RS485) mit der zugehörigen Applikationsschicht nach SAE J1587 auf CAN zu übertragen.

Tab. 4.4 Protokollstapel nach SAE J1939 (*vereinfacht*)

Dokument	*Inhalt*	*ISO/OSI Referenzmodel*	
SAE J1939/73 SAE J1939/71	Beschreibung der Dateninhalte für Off- und On-Board-Kommunikation	Application	7
		Presentation	6
		Session	5
SAE J1939/21	Transportprotokoll	Transport	4
SAE J1939/31	Spezifikation einer Bridge	Network	3
SAE J1939/21	Übertragung auf Basis von CAN 2.0B	Data Link	2
SAE J1939/14 SAE J1939/11, /15	Bitrate, Busankopplung, Verkabelung und Steckverbinder	Physical	1

Grundlage der Datenübertragung nach SAE J1939 bildet die Spezifikation CAN 2.0B, auf der auch ISO 11898 basiert. Dabei werden ausschließlich 29-Bit CAN-Identifier verwendet. Die elektrischen und mechanischen Eigenschaften der Busankopplung mit einer festen Bitrate von 250 kBit/s bzw. 500 kBit/s werden in SAE J1939/11, J1939/14 und J1939/15 definiert. Data Link Layer (Schicht 2) und Transport Layer (Schicht 4) werden leider nicht eindeutig unterschieden, sondern im Dokument SAE J1939/21 zusammengefasst.

Auf Anwendungsebene existiert sowohl eine Spezifikation für die Datenübertragung im Fahrbetrieb (On-Board-Kommunikation nach SAE J1939/71) als auch für die Diagnose (Off-Board-Kommunikation nach SAE J1939/73 und OBD nach SAE J1939/3).

Neben den in Tab. 4.4 dargestellten Protokollschichten existieren noch ein Schichtenübergreifendes Netzwerkmanagement (SAE J1939/81) sowie verschiedene Normergänzungen für Spezialanwendungen wie Landmaschinen (SAE J1939/2) oder Industriegeneratoren (SAE J1939/75).

4.5.1 Übertragungsarten, Adressierung und CAN Message Identifier

Alle Steuergeräte bei J1939 erhalten eine eindeutige, 8 bit große logische Steuergeräteadresse, wobei der Wert FFh als Broadcast-Adresse reserviert ist. Die Steuergeräteadressen sind im Allgemeinen statisch definiert. Die Netzwerkmanagement-Spezifikation SAE J1939/81 schlägt aber auch ein Verfahren vor, um die Steuergeräteadresse dynamisch zu konfigurieren.

SAE J1939 verwendet 29 bit CAN-Identifier. Diese sind in unterschiedliche Felder aufgeteilt, mit denen die Informationen zur Adressierung und zum Botschaftsinhalt codiert werden (Abb. 4.19). Viele dieser Felder haben ihren Ursprung im Vorläuferprotokoll J1708/J1587. Über die obersten 3 bit des CAN-Identifiers wird die Priorität der Botschaft codiert. Die unteren 8 bit des CAN-Identifiers geben die logische Adresse des Senders (*Source Address*) an. Den größten Teil des CAN-Identifiers nimmt die so genannte *Parameter Group Number PGN* ein. Deren wichtigster Teil ist das 8 bit große *PDU Format PF* Feld, das zwischen einer verbindungsorientierten und einer nachrichtenorientierten Datenübertragung unterscheidet. Bei der *verbindungsorientierten* Übertragung (*PDU 1 Format*) ist das Ziel der Botschaft ein einzelnes Steuergerät, dessen 8 bit große Zieladresse (*Destination Address*) dann innerhalb des PGN-Felds übertragen wird. Bei der *nachrichtenorientierten* Übertragung (*PDU 2 Format*) dagegen wird ein 8 bit Wert (*Group Extension* GE) übertragen, der den Inhalt der Nutzdaten auf der Anwendungsebene kennzeichnet. Das reservierte Bit und das *Data Page Bit* sind für zukünftige Erweiterungen vorgesehen und derzeit für Anwendungen von SAE J1939 bei Straßenfahrzeugen mit 0 belegt.

Die Werte des *PDU Format Bytes PF* und damit der Inhalt der Nutzdaten einer Botschaft wird im Wesentlichen durch die Normen für die Applikationsschicht J1939/71 und J1939/73 festgelegt. Einige wenige Werte im Bereich E8 h … EFh sind für spezielle Aufgaben wie die im folgenden Abschnitt beschriebene Transportschicht für die segmentierte

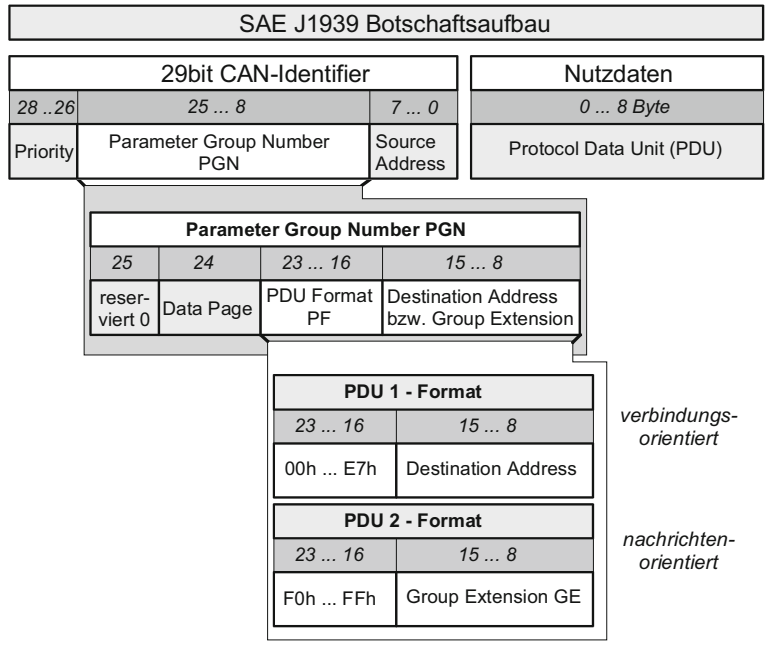


Abb. 4.19 Aufbau von SAE J1939-Botschaften und Struktur des CAN-Identifiers

Datenübertragung vorgesehen. Diese wird eingesetzt, wenn der zu übertragende Datenblock mehr als 8 Datenbytes umfasst und daher nicht mehr in eine einzelne CAN-Botschaft passt.

Als Beispiel für eine *nachrichtenorientierte* Botschaft soll die *Engine Temperature #1* Botschaft betrachtet werden, die periodisch jede Sekunde vom Motorsteuergerät gesendet wird und Temperaturinformationen enthält. Wie in SAE J1939/71 definiert, verwendet die Botschaft das *PDU 2 Format* PF = 254 = FEh und die *Group Extension* GE = 238 = EEh und enthält 8 Datenbytes mit den in Abb. 4.20 dargestellten Werten.

Die *Torque Speed Control #1* Botschaft dagegen, mit der das Motorsteuergerät bzw. die Motorbremse von übergeordneten Fahrzeugsystemen ferngesteuert werden, ist eine *verbindungsorientierte* Botschaft mit PF = 0. Sie wird alle 10 ms gezielt an das Motorsteuergerät bzw. alle 50 ms an die Motorbremse gesendet (Abb. 4.21) und enthält nur 4 Datenbytes. Im

Byte 1	*Byte 2*	*Byte 3, 4*	*Byte 5, 6*	*Byte 7*	*Byte 8*
Kühlwasser-temperatur	Kraftstoff-temperatur	Motoröl-temperatur	Turbolader-temperatur	Ladeluftkühl-ertemp.	LLK-Thermo-statventil
- 40 ...+210 °C 1 K/bit		- 273 ... +1735 °C 0,03125 K/bit		- 40 ... +210 °C 1 K/bit	0 ... 100 % 0,4 %/bit

Abb. 4.20 Aufbau der Engine Temperature #1 Botschaft (PF = FEh, GE = EEh)

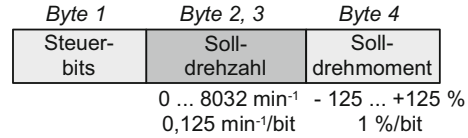


Abb. 4.21 Aufbau der Torque Speed Control #1 Botschaft (PF = 00 h)

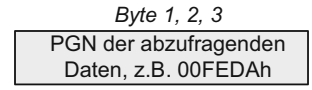


Abb. 4.22 Abfrage von Daten mit einer Request Botschaft (PF = EAh)

ersten Byte werden eine Reihe von Steuerbits zusammengefasst, mit denen festgelegt wird, ob der Motor drehzahl- bzw. drehmomentgeregelt betrieben, der Drehzahlregler für Leerlaufbetrieb, Fahrbetrieb oder Nebenabtriebe umgeschaltet werden soll. Außerdem wird die Priorität des externen Eingriffs gegenüber dem Leerlaufdrehzahlregler der Motorsteuerung bzw. dem Gaspedal festgelegt.

Botschaften, die mehr als 8 Datenbytes umfassen, erfordern die im folgenden Abschnitt beschriebene segmentierte Datenübertragung. Ein Beispiel ist die Abfrage der Software-Identifikations- und Versionsnummern. Dazu sendet das abfragende Gerät eine verbindungsorientierte *Request Botschaft* mit PF = 234 = EAh (Abb. 4.22) an das Steuergerät, dessen Identifikation ausgelesen werden soll. Die Botschaft enthält die 3 Byte lange PGN (PF und GE) der abzufragenden Informationen, im Beispiel die Werte PF = 254 = FEh, GE = 218 = DAh für die *Software Identification*. Die Antwort des Steuergerätes sind nachrichtenorientiert übertragene, segmentierte Botschaften mit einem Textstring variabler Länge im Datenteil (Abb. 4.23a, Details im folgenden Abschnitt).

Falls das Steuergerät die angeforderten Daten nicht liefern kann, antwortet es mit einer ACK-Botschaft (PF = 232 = E8 h), die im ersten Datenbyte den Wert 1 (*Not Acknowledge*)

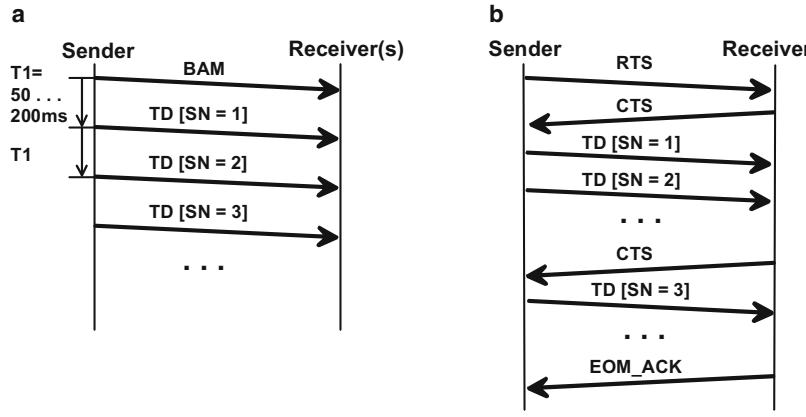


Abb. 4.23 Alternative Abläufe bei der segmentierten Datenübertragung, **a** nachrichtenorientierte Übertragung, **b** verbindungsorientierte Übertragung

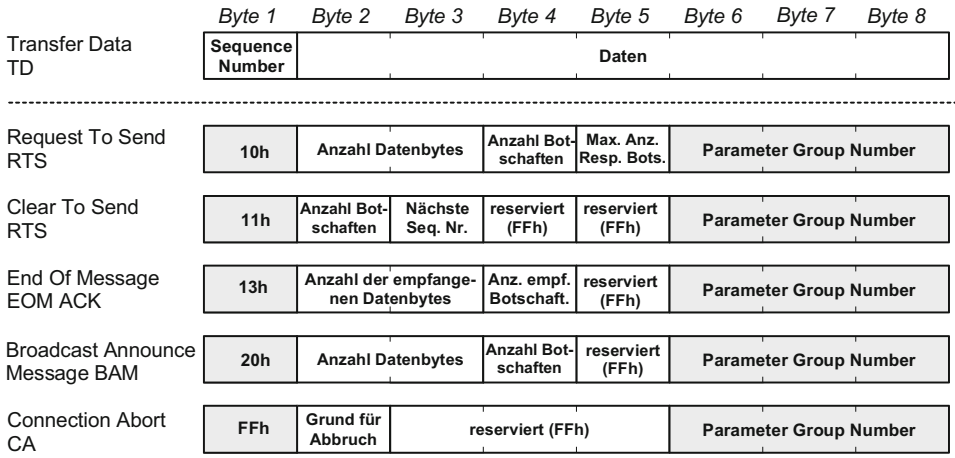


Abb. 4.24 Nutzdatenbereich der Botschaften für die segmentierte Übertragung

enthält. In einigen Fällen ist eine positive Bestätigung (*Acknowledge*) mit 0 im ersten Datenbyte erforderlich. Die letzten 3 Byte einer ACK-Botschaft enthalten die PGN, auf die sich die ACK-Botschaft bezieht.

Für herstellerdefinierte Botschaften sind PF = 239 = EFh und PF = 255 = FFh vorgesehen.

4.5.2 Segmentierte Datenübertragung (Multi Packet)

Wenn die segmentierte Datenübertragung an mehrere Empfänger gerichtet ist und daher *nachrichtenorientiert* erfolgt, kündigt der Sender den Empfängern die segmentierte Datenübertragung durch eine *Broadcast Announce Message BAM* an (Abb. 4.23a). Nach einer Pause von jeweils 50 … 200 ms sendet er dann in den *Transfer Data TD* Botschaften die Nutzdaten. Dabei wird im ersten Byte eine bei 1 beginnende, mit jeder Datenbotschaft hochgezählte Sequenznummer SN übertragen, die übrigen 7 Datenbytes der CAN-Botschaft enthalten Nutzdaten. Die letzte Datenbotschaft wird gegebenenfalls mit FFh-Bytes aufgefüllt, so dass die CAN-Botschaften stets die volle Länge von 8 Datenbytes haben. Mit der BAM wurde die Gesamtzahl der Datenbytes sowie die Anzahl der zu erwartenden Botschaften übertragen (Abb. 4.24). Somit kann jeder Teilnehmer das Ende des Datenblocks erkennen, ohne dass dies besonders gekennzeichnet wird. Da die Sequenznummer nicht überlaufen darf, können auf diese Weise Datenblöcke mit maximal $255 \cdot 7 = 1785$ Byte übertragen werden. Wegen der großen Pausen von min. 50 ms zwischen den einzelnen CAN-Botschaften liegt die Nutzdatenrate bei dieser Art der Übertragung unter 160 B/s.

Erfolgt die segmentierte Datenübertragung dagegen *verbindungsorientiert*, d. h. mit einem einzigen Empfänger, so ist ein ähnlicher Handshake-Mechanismus vorgesehen wie

bei ISO 15765-2 (Abb. 4.23b). Der Sender sendet eine *Request To Send RTS* Botschaft an den Empfänger, in der er die Länge des Datenblocks (Gesamtzahl der Bytes und Anzahl der Segmente) ankündigt (Abb. 4.24). Der Empfänger antwortet mit einer *Clear To Send CTS* Botschaft, in der er mitteilt, wie viele CAN-Botschaften er ohne Pause hintereinander empfangen kann und welche Sequenznummer er als nächstes erwartet. Nachdem der Sender diese Anzahl von CAN-Botschaften gesendet hat, wartet er auf die nächste CTS Botschaft vom Empfänger, bevor er weitere Datenbotschaften sendet. Die letzte Datenbotschaft wird wieder mit FFh-Bytes aufgefüllt. Anschließend bestätigt der Empfänger mit einer *End Of Message Acknowledge EOM ACK* Botschaft den Empfang des gesamten Datenblocks. Will der Empfänger den Verbindungswunsch nicht annehmen oder die Übertragung vorzeitig abbrechen, so sendet er statt CTS eine *Connection Abort CA* Botschaft, die auch den Grund des aufgetretenen Problems enthält. Bei der verbindungsorientierten Datenübertragung dürfen die Datenbotschaften ohne Mindestabstand gesendet werden, so dass die volle CAN-Bandbreite ausgenutzt werden kann. Für die Maximalabstände existieren wiederum Zeitschranken im Bereich zwischen 750 ms und 1250 ms.

Für die *Transfer Data TD* Botschaft ist das PDU Format Byte PF = 235 = EBh reserviert. Die als *Connection Management* bezeichneten RTS, CTS, EOM, BAM und CA Botschaften verwenden PF = 236 = ECh. Zwischen ihnen wird mit unterschiedlichen Werten des ersten Nutzdatenbytes unterschieden. Zusätzlich wird bei diesen Botschaften in den letzten 3 Datenbytes die Parameter Group Number PGN des zugehörigen Nutzdatenblocks übertragen.

Empfänger müssen in der Lage sein, gleichzeitig mindestens eine verbindungsorientierte und eine nachrichtenorientierte Datenübertragung zu verarbeiten. Sender können gleichzeitig eine nachrichtenorientierte und gegebenenfalls mehrere verbindungsorientierte Übertragungen zu unterschiedlichen Empfängern handhaben.

4.6 Transportprotokoll DoIP nach ISO 13400

Mit der zunehmenden Nutzung von Fahrerassistenz- und Infotainmentsystemen steigt der Softwareumfang in Fahrzeugen und damit die Download-Zeit bei der End-of-Line-Programmierung (Flash Programmierung) in der Fertigung und in der Werkstatt immer weiter. Verwendet man dazu die übliche CAN Diagnoseschnittstelle mit einer maximalen Datenrate im Bereich um 30 KByte/s, so dauert die Übertragung von 100 MByte, wie sie bereits bei einem geringfügigen Update der Kartensoftware des Navigationssystems notwendig würde, knapp eine Stunde. Aus diesem Grund verwenden einige Oberklassefahrzeuge neuerdings eine zusätzliche Ethernet-Schnittstelle und führen die Übertragung mit dem Standard-Internetprotokoll TCP/IP durch (Abb. 4.25).

Die 10/100 Mbit/s Ethernet-Verbindung könnte zunächst freie Anschlüsse des OBD-Diagnosesteckers nutzen (Abb. 4.26). Mittelfristig soll eine neue, besser für die hohen Bitraten geeignete Steckverbindung eingeführt werden. Zusätzlich zu Ethernet ist ein Aktivierungssignal vorgesehen, mit dem die Verbindung freigeschaltet wird.

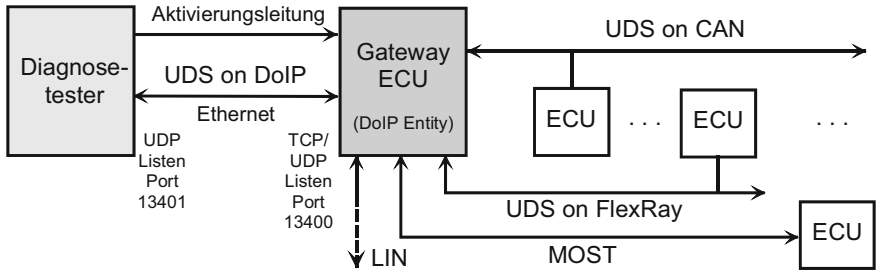
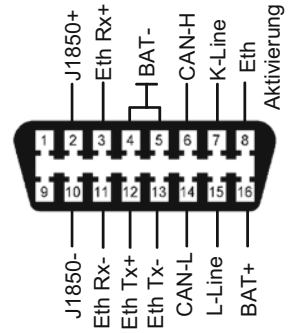


Abb. 4.25 Ethernet – TCP/IP – Schnittstelle zum Fahrzeug

Abb. 4.26 Erweiterung des OBD-Diagnosesteckers für Ethernet



Um existierende Diagnoseprotokolle wie UDS (siehe Kap. 5) sowohl auf der Steuergeräteseite als auch im Diagnosetester weiter nutzen zu können und die Anfragen des Diagnosetesters vom Gateway-Steuergerät direkt zu den internen CAN, FlexRay und LIN Bussystemen durchleiten zu können, wurde mit *Diagnostics over IP* (DoIP) ein neues Transportprotokoll entwickelt. DoIP, das als ISO 13400 standardisiert wird, erlaubt es, gewöhnliche UDS-Botschaften in TCP/IP-Botschaften zu verpacken und diese über Ethernet oder WLAN zu übertragen (Abb. 4.27). Der Ethernet-Protokollstapel hat mit TCP und UDP, IP v4 bzw. v6 sowie dem Hilfsdienst DHCP den üblichen Umfang [2].

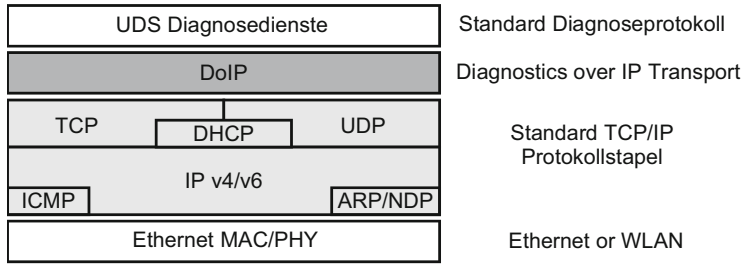


Abb. 4.27 Protokollstapel mit DoIP-Transportprotokoll

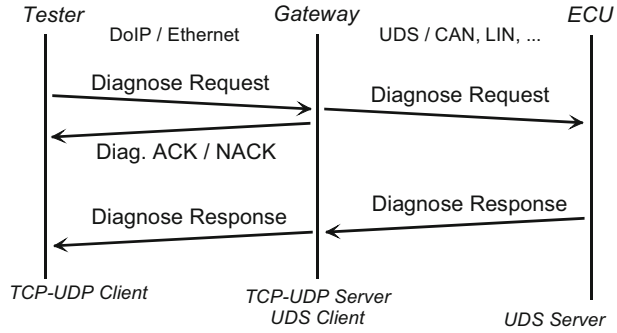


Abb. 4.28 DoIP Botschaftsablauf zwischen Tester, Gateway und Steuergerät

Gegenüber dem Diagnosetester agiert das Gateway als TCP- bzw. UDP-Server mit vordefinierten IP-Ports. Weitere Ports können dynamisch vergeben werden. Die IP-Adressen werden über DHCP automatisch zugeordnet, wobei es feste Defaultwerte gibt, falls kein DHCP-Server vorhanden ist. Zusätzlich zu den diagnoseüblichen Request-Response-Botschaften zwischen Tester und Steuergerät tauschen das Gateway und der Tester auf der Ethernet-Seite Bestätigungsbotschaften (*Acknowledge ACK*) aus (Abb. 4.28). Damit muss das Gateway die erfolgreiche Weiterleitung einer Diagnosebotschaft zum internen Fahrzeugnetz bestätigen. Im Fehlerfall enthält diese Meldung (*Not Acknowledge* NACK) einen entsprechenden Fehlercode.

Die DoIP-Botschaften (Abb. 4.29 und Tab. 4.5) verwenden einen Header mit Steuerinformationen, mit dem verschiedene Nutzdatentypen (*Payload*) unterschieden werden. Enthält der Nutzdatenblock eine UDS Diagnosebotschaft (UDS Request bzw. Response) bzw. eine ACK/NACK-Bestätigungsmeldung, so werden die UDS *Source* bzw. *Target* Adressen von Steuergerät und Diagnosetester vorangestellt. Im Rest des Nutzdatenblocks folgen der SID *Service Identifier* und die notwendigen Parameter und Daten (vgl. Abschn. 5.1.3 und 5.2) bzw. bei Bestätigungsmeldungen ein 1 Byte langer Fehlercode. Diese Botschaften verwenden alle das TCP-Protokoll.

Bevor Diagnosebotschaften ausgetauscht werden, muss zunächst eine Art Verbindungsaufbau zwischen Diagnosetester und Fahrzeug-Gateway erfolgen. Sobald das Gateway nach dem Anschluss der Ethernet-Verbindung über DHCP mit einer gültigen IP-Adresse konfiguriert wurde, sendet es im Abstand von 500 ms über UDP eine *Vehicle Announce-*

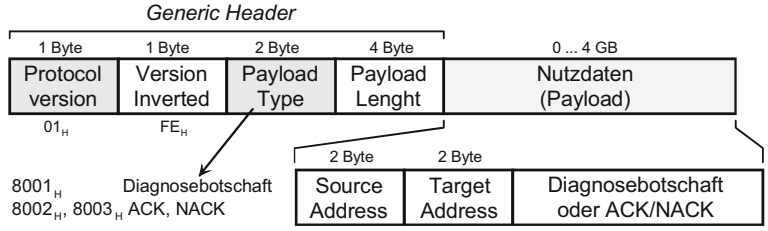


Abb. 4.29 DoIP Botschaftsformat (weitere Botschaften siehe Tab. 4.5)

Tab. 4.5 DoIP-Botschaftstypen

Bezeichnung	Payload Type	IP Protokoll	Anwendung
Generic DoIP Header NACK	0000_H		Meldung des Gateways bei Fehlern im DoIP-Header, enthält Fehlercode
Vehicle Identification Request	0001_H … 0003_H	UDP	Abfrage der Fahrzeugkennungen durch den Tester. Enthält keine Nutzdaten. Zusätzliche Untervarianten, falls der Tester die VIN und/oder die eindeutigen Gerätekennziffern kennt und gezielt ein Gateway ansprechen will.
Vehicle Announcement Message bzw. Vehicle Identification Response	0004_H	UDP	Ankündigung des Fahrzeugs, dass ein DoIP- Gateway vorhanden ist. Enthält die Fahrgestellnummer VIN (17 Byte), die UDS-Diagnoseadresse des Gateways (2 Byte) und zwei je 6 Byte lange eindeutige Gerätekennziffern
Routing Activation Request/Response	0005_H 0006_H	TCP	Aktivieren der Weiterleitung von Diagnosebotschaften von/zum internen Fahrzeugnetz
Alive Check Request/Response	0007_H 0008_H	TCP	Aufrechterhalten der TCP/IP-Verbindung, falls über längere Zeit keine Diagnosebotschaften ausgetauscht werden
DoIP Entity Status Request/Response	4001_H 4006_H	UDP	Abfrage zum Typ des DoIP Gerätes (Gateway oder normales Steuergerät) und zur Anzahl der möglichen Verbindungen (TCP/IP-Sockets)
Diagnostic Power Mode Info Request/Response	4003_H 4004_H	UDP	Abfrage, ob die Fahrzeugnetzwerke aktiv sind (Zündung ein usw.)
Diagnostic Message Request Diag. ACK/NACK	8001_H 8002_H 8003_H	TCP	Austausch von Diagnosebotschaften, Einzelheiten siehe Text

ment Message. Diese teilt dem Diagnosetester die Fahrgestellnummer (*Vehicle Identification Number VIN*), die UDS-Diagnoseadresse des Gateways sowie zwei Hardwarekennziffern mit, z. B. die MAC-Adresse. Durch die Hardwarekennziffern soll der Tester unterscheiden können, ob im Fahrzeug außer dem Gateway weitere Geräte direkt mit dem Ethernet-Zugang verbunden sind. Falls der Tester die *Vehicle Announcement* Informationen verpasst, kann er das Fahrzeug durch Senden einer *Vehicle Identification Request Message* jederzeit zur Wiederholung der Botschaften auffordern.

Im Anschluss an die Fahrzeugidentifikation fordert der Diagnosetester das Gateway durch einen *Routing Activation Request* auf, nachfolgende Diagnosebotschaften tatsächlich von und zum internen Fahrzeugnetz weiterzuleiten. Der *Routing Activation Request*

enthält dabei die UDS-Diagnoseadresse des Gateways. Dadurch kann in zukünftigen Fahrzeugen mit möglicherweise mehreren Gateways gezielt ein einzelnes Gateway ausgewählt werden. Über weitere Parameter dieser Botschaft soll zukünftig voraussichtlich auch eine Login/Authentifizierungsprozedur abgewickelt werden, um den Fahrzeugzugang freizuschalten. Während einer bestehenden Verbindung kann der Diagnosetester über verschiedene Botschaften, die hier nicht im Detail beschrieben werden sollen, Statusinformationen mit dem Gateway austauschen.

4.7 Transportprotokoll für CAN FD

Für die CAN-Weiterentwicklung CAN FD muss eine Erweiterung des bisherigen Transportprotokolls nach ISO 15765-2 (Abschn. 4.1) definiert werden. Da CAN FD bis zu 64 Byte in einer einzelnen CAN-Botschaft übertragen kann, ist das bisher nur 4 bit lange Datenlänge-Feld DL im Single Frame SF des Transportprotokolls nicht mehr ausreichend (Abb. 4.1). Weil das modifizierte Transportprotokoll aufwärtskompatibel bleiben soll, wird wohl ähnlich wie bei FlexRay (Tab. 4.1) ein modifizierter Single Frame mit einem mindestens 6 bit langen DL-Feld eingeführt.

Bei den übrigen Transportprotokoll-Botschaften besteht aus Sicht von CAN FD kein Änderungsbedarf. Allerdings kann darüber nachgedacht werden, auch eine Variante des First Frames FF einzuführen, falls angestrebt wird, die Blockgröße für segmentierte CAN-Übertragungen von bisher 4 KB in Richtung der beim FlexRay-Transportprotokoll möglichen 64 KB-Blöcke zu erweitern.

4.8 Normen und Standards zu Kapitel 4

ISO 15765 (ISO TP)	Road vehicles – Diagnostics on CAN, siehe Kap. 5
AUTOSAR	siehe Kap. 8
ISO 10681	Road vehicles – Communication on FlexRay, siehe Kap. 5
VW TP	VW CAN-Transportprotokoll TP 2.0, Version 1.1, 2002, internes Dokument VW CAN-Transportprotokoll, Version 1.6.1, 2000, internes Dokument

SAE J1939	SAE J1939 Serial Control and Communications Heavy Duty Vehicle Network, 2012, www.sae.org SAE J1939/1 On-Highway Equipment Control and Communications Network, 2011, www.sae.org SAE J1939/2 Agricultural and Forestry Off-Road Machinery Control and Communications Network, 2013, www.sae.org SAE J1939/11 Physical Layer 250 kbit/s, Shielded Twisted Pair, 2012 SAE J1939/14 Physical Layer 500 kbit/s, 2011, www.sae.org SAE J1939/15 Reduced Physical Layer 250 kbit/s, Unshielded Twisted Pair, 2008 SAE J1939/13 Off-Board Diagnostic Connector, 2011, www.sae.org SAE J1939/21 Data Link Layer, 2010, www.sae.org SAE J1939/31 Network Layer, 2010, www.sae.org SAE J1939/81 Network Management, 2011, www.sae.org SAE J1939/71 Vehicle Application Layer, 2012, www.sae.org SAE J1939/73 Application Layer - Diagnostics, 2010, www.sae.org SAE J1939/3 On-Board Diagnostics Implementation Guide, 2008 SAE J1939/74 Application Layer – Configurable Messaging, 2010 SAE J1939/75 Application Layer – Generator Sets and Industrial, 2011
SAE J1587	SAE J1587 Electronic Data Interchange Between Microcomputer Systems in Heavy-Duty Vehicle Applications, 2013, www.sae.org
SAE J1708	SAE J1708 Serial Data Communications Between Microcomputer Systems in Heavy-Duty Vehicle Applications, 2010, www.sae.org
ISO 13400 DoIP	ISO 13400 Road vehicles – Diagnostic communication over Internet Protocol (DoIP), www.iso.org Part 1: General information and use case definition, 2011 Part 2: Transport protocol and network layer services, 2012 Part 3: Wired vehicle interface based on IEEE 802.3, 2011 Weitere Literaturangaben zu Internet Protokollen siehe Kap. 8, zu Ethernet siehe Kap. 3

Literatur

[1] R. Schmidgall: Automotive Embedded Systems Software Reprogramming, 2012, PhD-Thesis, School of Engineering and Design, Brunel University London, http://bura.brunel.ac.uk/handle/2438/7070

[2] R. Stevens: TCP/IP Illustrated. Addison-Wesley, 3 Bände, 2002