Diagnosesysteme im Automobil

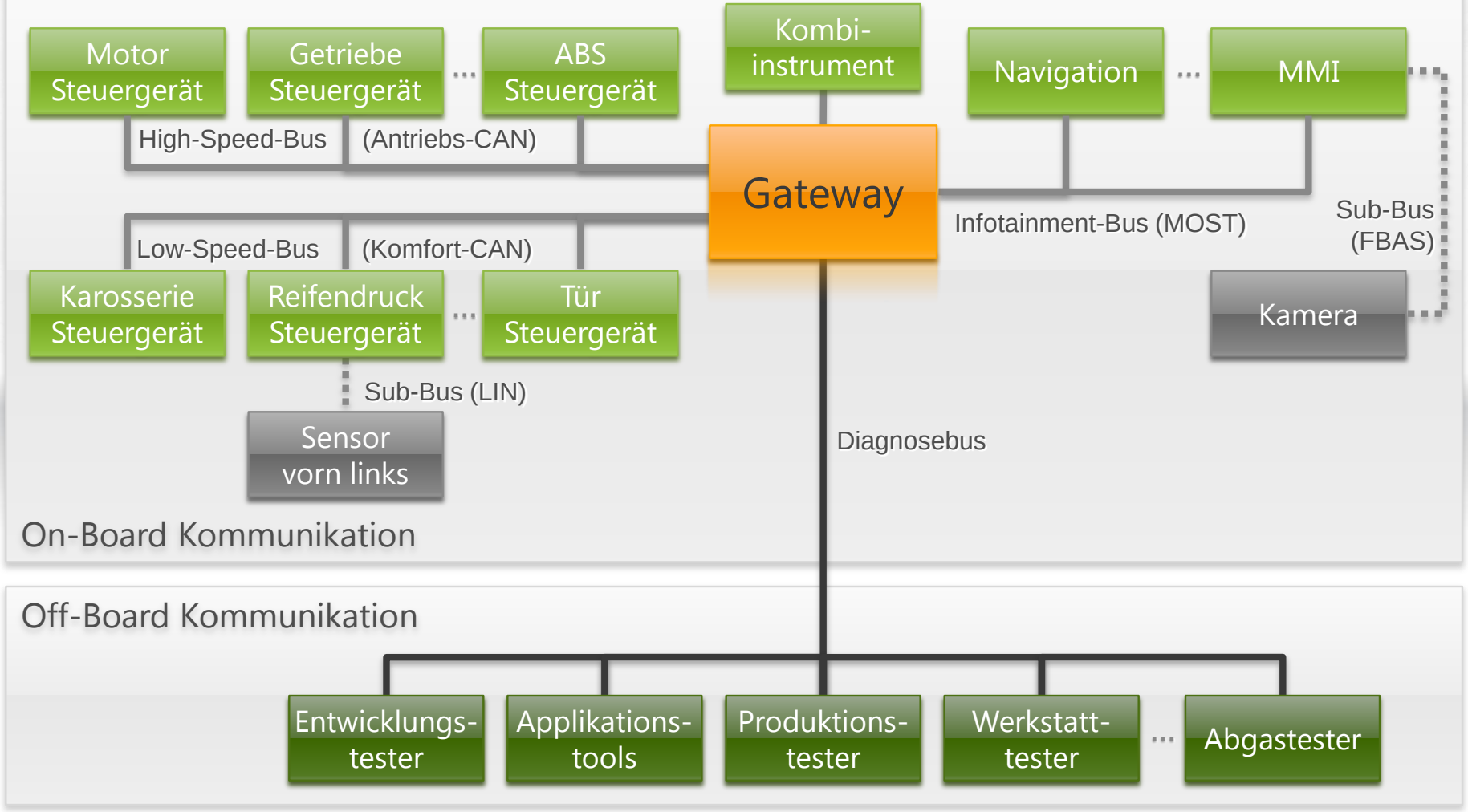
Transport- & Diagnoseprotokolle

Dr.-Ing. Jörg Supke
emotive GmbH, Stuttgart
Prof. Dr.-Ing. Werner Zimmermann
Hochschule Esslingen

emotive
Automotive Software Solutions

Inhalt

Kommunikation im Fahrzeug*



*vereinfacht

Open System Interconnection (OSI) Schichtenmodell (ISO 1978)

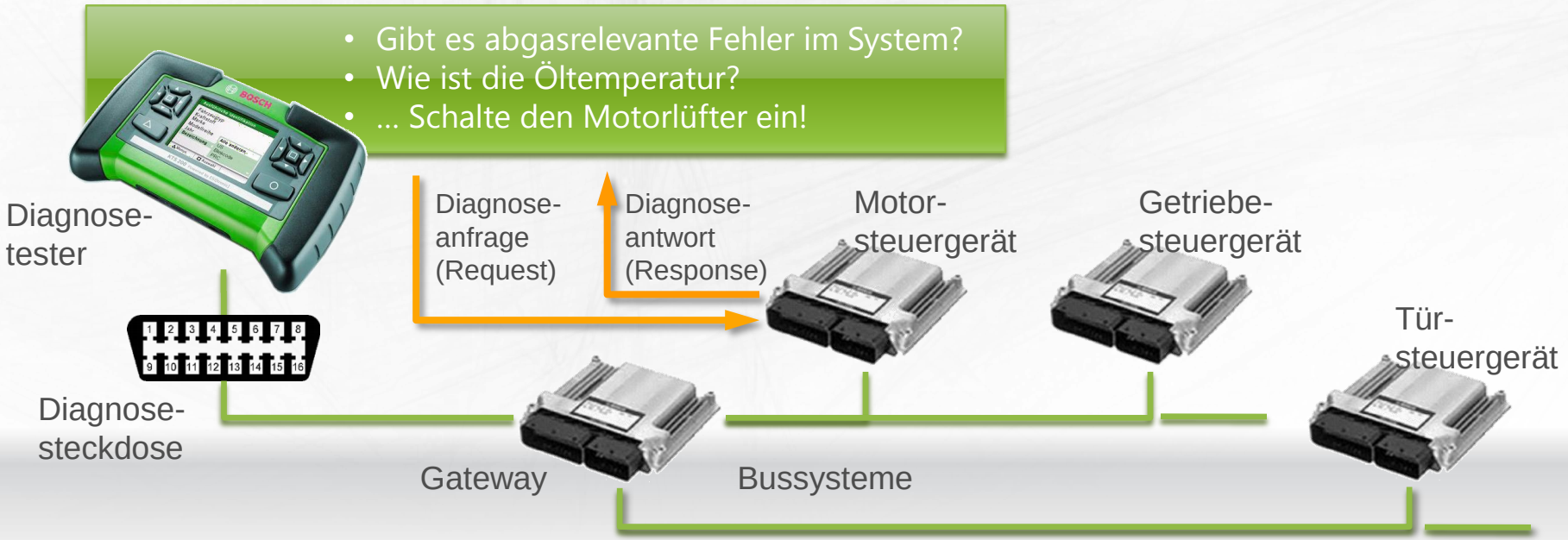
Eigentliche Anwendung (On-Board z.B. Motorsteuerung oder Off-Board z.B. Diagnosetester)

Schicht	Bezeichnung	Anwendung im Fahrzeug	
7	Application Layer (Anwendung)	Anwendungsprogramm, fertige Dienste, z.B. Fehlerspeicher lesen	Diagnoseprotokolle
6*	Presentation Layer (Darstellung)	Unterschiedliche Darstellung der Daten	
5*	Session Layer (Sitzungssteuerung)	Steuert Verbindungsprozesse, z.B. Authentifizierung, Synchronisation	
4	Transport Layer (Transport)	Segmentierung der Botschaften	Transportprotokolle
3*	Network Layer (Vermittlung)	Routing, Adressierung, Teilnehmererkennung, -überwachung	
2	Data Link Layer (Sicherung)	Botschaftsaufbau, Buszugriff, Fehlererkennung, Flussregelung	Bussysteme
1	Physical Layer (Bitübertragung)	Signalpegel, Bitkodierung	

Busleitungen und Steckverbinder (Mechanik)

* Werden für Anwendungen im Fahrzeug z. Z. nicht verwendet; Aufgaben werden von den anderen Schichten übernommen.

Off-Board-Kommunikation



- Kommunikation nur bei Bedarf
- Klare Rollenverteilung: **Diagnosetester fragt** an ⇨ ein oder mehrere **Steuergeräte antworten**
- Kommunikationsprinzip: **Request – Response**
- Diagnosetester weiß zu Beginn nicht, welche Steuergeräte im Fahrzeug verbaut sind ⇨ muss mit allen Steuergeräten kommunizieren können, darf aber nicht auf alles zugreifen
- Diagnose-Anfrage oder Antwort eventuell viel länger als eine Busbotschaft. Beispiel Fahrgestellnummer 17 Zeichen, Extremfall: Update der Steuergerätesoftware („Flashen“)

Protokollstapel (Protocol Stack)

Tester
(ISO: Client, ASAM: Master)

Diagnoseanwendung

Request

Response

Steuergerät(e)
(ISO: Server, ASAM: Slave)

Diagnose-Stack

ISO/OSI-Schichtenmodell

Diagnoseprotokoll ISO 14229, ISO 15765-3 ...	Application Layer (Schicht 7)	Diagnoseprotokoll ISO 14229, ISO 15765-3 ...
Transportprotokoll ISO 15765-2 (Segmentierung)	Transport Layer (Schicht 4)	Transportprotokoll ISO 15765-2 (Segmentierung)
Bussystem ISO 11898 (CAN) ...	Data Link Layer (Schicht 2) Physical Layer (Schicht 1)	Bussystem ISO 11898 (CAN) ...

Überblick der Standards

	Allgemeine Fahrzeugdiagnose			*Onboard Diagnose für abgasrelevante Systeme*
Application Layer	**KWP 2000** (inzwischen ersetzt durch UDS) ISO 14230-3	ISO/DIS 15765-3 *1 **KWP 2000 on CAN**	**UDS ISO 14229-1 ISO 15765-3** **UDS on CAN**	**OBD** **ISO 15031-5** (inhaltlich identisch zu SAE J1979)
Transport Layer	K-Line ISO 14230-2	**ISO 15765-2** (bei OBD nur für CAN)		
Data Link Layer Physical Layer (Bussystem)	K-Line ISO 14230-2	**ISO 11898 CAN**		
	ISO 14230-1 (prakt. identisch zu ISO 9141-2 K-Line)			Früher alternativ zu CAN: ISO 9141-2 K-Line, SAE J1850 PWM oder SAE J1850 VPWM

Historische Entwicklung in Europa

- K-Line Physical Layer (12V Pegel, UART, Bitraten 300 ... 9600 bit/s) als ISO 9141 genormt Ende der 80er Jahre, darüberliegende Schichten proprietär, z.B. KWP 1281 VW
- Diagnosebotschaften standardisiert für OBD in USA Anfang der 90er Jahre auf Basis SAE J1850 mit zwei verschiedenen Physical Layern, auf Intervention europäischer Hersteller später auch K-Line als ISO 9141-2 CARB. OBD verbindlich in Europa erst seit 2001 (Diesel seit 2004). Umstellung auf CAN verbindlich in USA seit 2007.
- Keyword Protokoll 2000 (KWP 2000) Standardisierung Mitte der 90er Jahre als ISO 14230 auf Basis K-Line
- Portierung von KWP 2000 auf CAN (*1 „KWP 2000 Diagnostics on CAN") Ende der 90er Jahre, enthalten in Normentwürfen für ISO 15765-3 und von verschiedenen Herstellern implementiert, aber niemals offiziell standardisiert! Gültiger Standard ISO 15765-3 (UDS on CAN) basiert auf UDS, nicht auf KWP 2000 und ist dazu nicht kompatibel!

Quelle: Zimmermann

Testfragen – Einleitung

1. **Wer sendet den Diagnose Request?**
   a) Der Diagnosetester
   b) Das Steuergerät
2. **Zu welcher Schicht des ISO/OSI-Kommunikationsmodells gehört das Diagnoseprotokoll UDS?**
   a) Zur Schicht 2, dem Data Link Layer
   b) Zur Schicht 4, dem Transport Layer
   c) Zur Schicht 7, dem Application Layer
3. **Wie bezeichnen die ISO-Normen den Diagnosetester?**
   a) Als Client
   b) Als Server
   c) Als Master
4. **Wie bezeichnen die ASAM-Normen ein Steuergerät?**
   a) Als Master
   b) Als Slave
5. **Was ist der Inhalt der ISO 15031?**
   a) On-Board-Diagnose für abgasrelevante Systeme
   b) KWP 2000, das veraltete Protokoll für die allgemeine Fahrzeugdiagnose
   c) UDS, das aktuelle Protokoll für die allgemeine Fahrzeugdiagnose

Inhalt

Problemstellung

On-Board Kommunikation:

- Signale < max. Länge einer Data Link Layer Botschaft
- Ziel: **Direkte Verwendung der DLL-Botschaft** ohne Segmentierung für kurze Latenzzeit
- Zusammensetzen von mehreren Signalen zu einer DLL-Botschaft um Botschaftsoverhead zu reduzieren durch Interaktionsebenen (Application Layer), z.B. OSEK bzw. AUTOSAR COM

Off-Board Kommunikation:

- Kurze und Lange Datenblöcke bei Werkstattdiagnose, z.B. Lesen Drehzahl (2 Byte), Lesen Fahrgestellnummer (> 17 Bytes)
- Extrem lange Botschaften beim EOL/Flashen, z.B. komplette ECU Software (4 MB)
- Ziel: **Einfache Übertragung von Blöcken variabler Länge** mit hohem Datendurchsatz

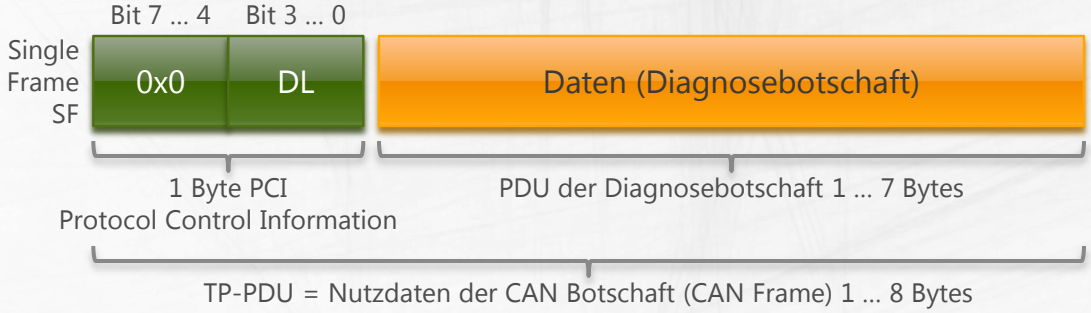
Aufgabe des Transport Layers

- Anpassung der Botschaftslänge von der Applikationsebene zur Busebene
- Aufspalten langer Diagnosebotschaften in mehrere kurze Busbotschaften beim Senden und Zusammensetzen beim Empfangen – **Segmentierung**
- **Flusskontrolle**, d.h. die Steuerung des zeitlichen Abstands der Botschaften, so dass der Empfänger nicht überlastet wird

Inhalt

Aufbau I

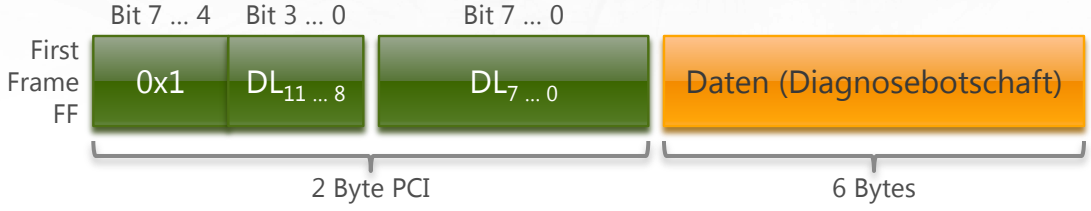
- Diagnosebotschaft < Busbotschaft: Unsegmentierte Single-Frame (SF) Botschaft



PCI = 1 Byte langer Header

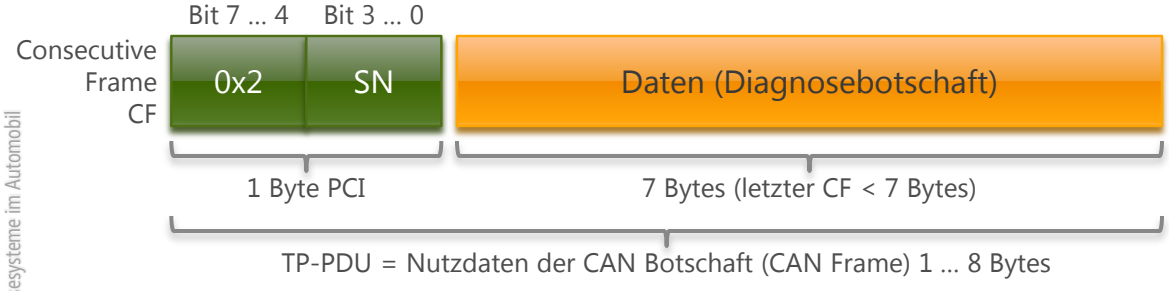
- 0 = SF, 1 = FF, 2 = CF, 3 = FC
- Bit 7...4 = 0 Kennzeichnet SF
- Bit 3...0 = DL Anzahl der folgenden Datenbytes

- Diagnosebotschaft >= Busbotschaft: Segmentierung über FF und CF Botschaften



Erste Botschaft mit 2 Byte PCI

- 0x01 Kennzeichnet FF
- DL Anzahl Bytes der gesamten Diagnosebotschaft (max. 4095)

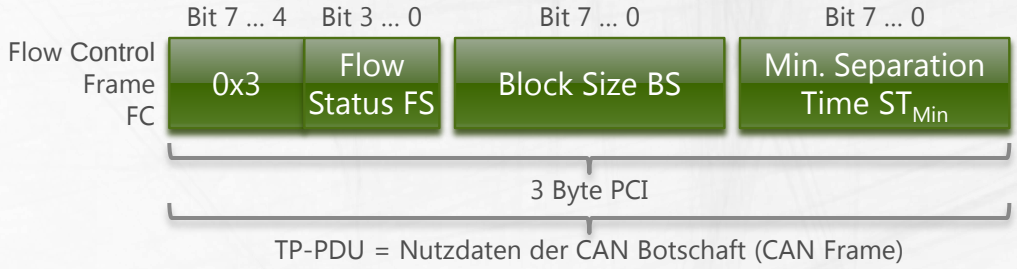


Folgebotschaften mit 1 Byte PCI

- 0x02 Kennzeichnet CF
- SN Sequenznummer: Laufende Nummer der CF modulo 16, beginnend bei 1 (FF: SN=0)

Aufbau II

- Flusssteuerung für segmentierte Botschaften vom Empfänger zum Sender



Flow Control mit 3 Byte PCI

- 0x03 Kennzeichnet FC
- FS Flow Status
  0 = ClearToSend (CTS)
  1 = Wait, 2 = Overflow
- BS Block Size: Zulässige Anzahl der folgenden CF, max. 255
- ST_{Min} Min. Separation Time: Mindestabstand der CF Frames in ms

- Sender sendet First Frame FF– Empfänger antwortet mit Flow Control Frame FC
- Sender sendet BS Consecutive Frames im Mindestabstand ST_{Min}
- Vorgang wiederholt sich ggf. mit neuem Flow Control Frames, bis Daten komplett sind

Timeout Bedingungen

- Für den Sender: Abstand zwischen der FF-Botschaft und der FC-Antwort: max. 1s
- Für den Empfänger: Abstand zwischen zwei CF-Botschaften: max. 1s
- Falls der Empfänger Zeit zur Verarbeitung benötigt, kann er mit einer Flow Control Botschaft mit Flow Status FS = Wait eine Pause erzwingen, bis er nach max. 1s mit einer weiteren Flow Control Botschaft zum Weitersenden oder erneut zum Warten auffordert
- Für Diagnosetester wird BS=0 und ST_{Min}=0 gefordert, d.h. Empfang auf der Testerseite effektiv ohne Flusssteuerung

Fehlererkennung

- Sender und Empfänger überwachen die segmentierte Übertragung durch die entsprechenden Timeouts, der Empfänger zusätzlich über die Sequenznummer
- Im Fehlerfall wird eine empfangene Botschaft ignoriert und die darüber liegende Schicht informiert. **In der Transportschicht erfolgt keine Fehlerbehandlung**, d.h. weder Sendewiederholung noch Fehlerbotschaft

Beispielablauf

Diagnosebotschaft mit 35 = 0x23 Bytes, Empfänger kann max. 24 Bytes puffern

Sender

FF: 0x10 0x23 6 Datenbytes

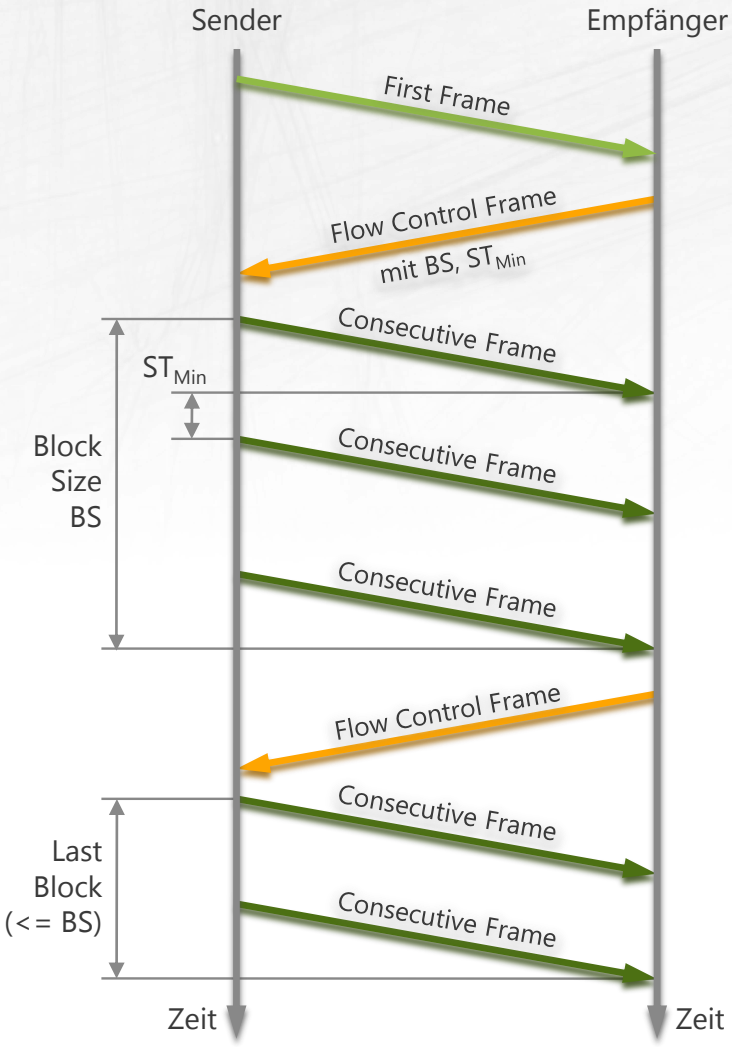
CF: 0x21 7 Datenbytes
SN = 1

CF: 0x22 7 Datenbytes
SN = 2

CF: 0x23 7 Datenbytes
SN = 3

CF: 0x24 7 Datenbytes
SN = 4

CF: 0x25 1 Datenbyte
SN = 5, CF mit < 8 Datenbytes zeigt Ende der segmentierten Übertragung



Empfänger

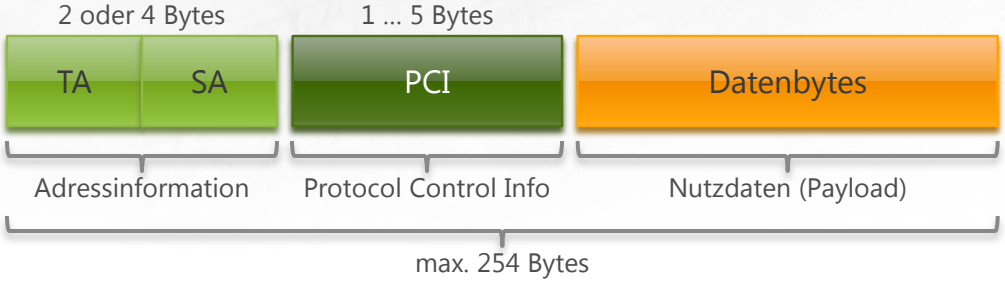
FC: 0x30 0x03 0x0A
CTS, BS = 3, ST_{Min} = 10 ms

FC: 0x30 0x03 0x0A
CTS, BS = 3, ST_{Min} = 10 ms

Inhalt

FlexRay nach AUTOSAR 3.0 (AUTOSAR FR TP)

- Spezifikation noch nicht stabil, ISO 10681-2 in Vorbereitung
- **Aufwärtskompatibel zu ISOTP für CAN** mit zusätzlichen Botschaften:
  - **Modifiziertes Botschaftsformat** mit Zieladresse (Target Address TA) und Quelladresse (Source Address SA). Bei CAN über CAN-ID gelöst. Bei FlexRay nötig, damit der selbe Static oder Dynamic Slot für verschiedene Diagnoseverbindungen genutzt werden kann.



  - Unsegmentierte Botschaft (**SF Extended**) mit bis zu 250 Datenbytes
  - Segmentierte Botschaften (**FF Extended** + CF) bis zu 4 GB (statt 4 kB)
  - Optionaler **Acknowledge Frame AF**: Empfangsbestätigung bzw. Aufforderung zur Sendewiederholung, da FlexRay im Data Link Layer im Gegensatz zu CAN keine automatische Empfangsbestätigung bzw. Sendewiederholung hat.
  - Segmentierung (ohne Flow Control und Acknowledge) auch als Multicast möglich

Testfragen – Transportprotokolle

1. Wo werden Transportprotokolle eingesetzt?

a) Bei der On-Board-Kommunikation zwischen Steuergeräten im Fahrbetrieb

b) Bei der Off-Board-Kommunikation zum Diagnosetester

2. Welcher Botschaftstyp wird bei der Übertragung kurzer Botschaften mit ISOTP eingesetzt?

a) First-Frame Botschaften

b) Single-Frame Botschaften

c) Flow-Control Botschaften

d) Consecutive-Frame Botschaften

3. Wer sendet Flow-Control Frames?

a) Immer der Diagnosetester

b) Immer das Steuergerät

c) Das Gerät, das eine segmentierte Botschaft senden will

d) Das Gerät, das eine segmentierte Botschaft empfangen soll

4. In welcher Reihenfolge sehen Sie Transportprotokoll-Botschaften am Beginn einer segmentierten Datenübertragung auf dem Bus?

a) SF - FC - CF - CF - ...

b) FC - FF - CF - CF - ...

c) FF - FC - CF - CF - ...

d) SF - FF - FC - CF - ...

5. Welche der folgenden Byte-Folgen könnte der Beginn einer First-Frame Botschaft sein?

a) 0x01 0x55 ...

b) 0x10 0x04 ...

c) 0x21 0x55 ...

d) 0x30 0x55 ...

6. Der ISO TP First-Frame eines segmentierten Datenblocks beginnt mit der Bytefolge 0x12 0x04 ... Wie viele Nutzdatenbyte enthält der gesamte Datenblock?

a) 6 Byte

b) 7 Byte

c) 204 Byte

d) 512 Byte

e) 516 Byte

7. Wie viele CAN-Botschaften muss ein Steuergerät versenden, wenn es einen Datenblock mit 32 Byte übertragen will und ISO TP verwendet?

a) 4 Botschaften

b) 5 Botschaften

c) 6 Botschaften

8. Was muss der Diagnosetester tun, wenn er als Antwort auf seine First-Frame Botschaft den folgenden Flow-Control Frame erhält: 0x30 0x04 0x08?

a) Er darf maximal 8 Consecutive-Frames mit je mindestens 4 ms Pause senden

b) Er darf maximal 4 Consecutive-Frames innerhalb von maximal 8 ms senden

c) Er muss mindestens 1 und maximal 4 Consecutive-Frames senden. Zwischen den Consecutive-Frames muss er eine Pause von mindestens 8 ms lassen

d) Er darf maximal 4 Single Frames senden

9. Wie lang darf ein Datenblock maximal sein, der mit dem ISO Transportprotokoll für CAN versendet werden soll?

a) 8 Byte

b) 256 Byte

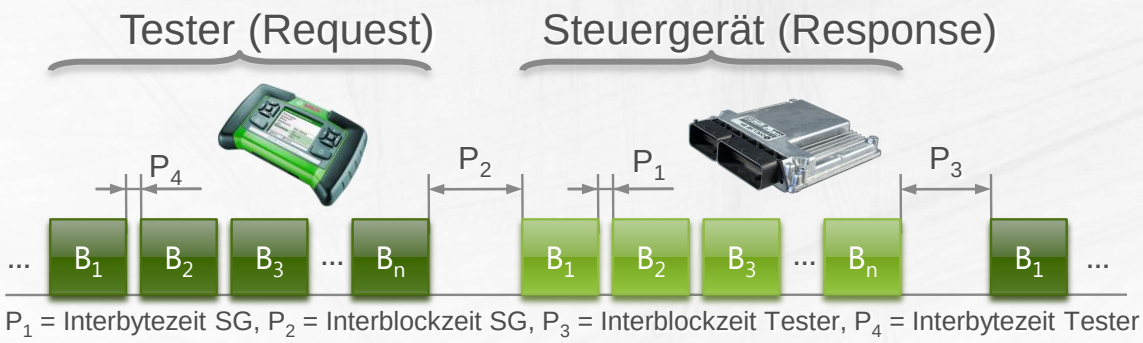
c) 4095 Byte

d) 64 KB

Inhalt

Allgemein – KWP 2000, UDS, OBD

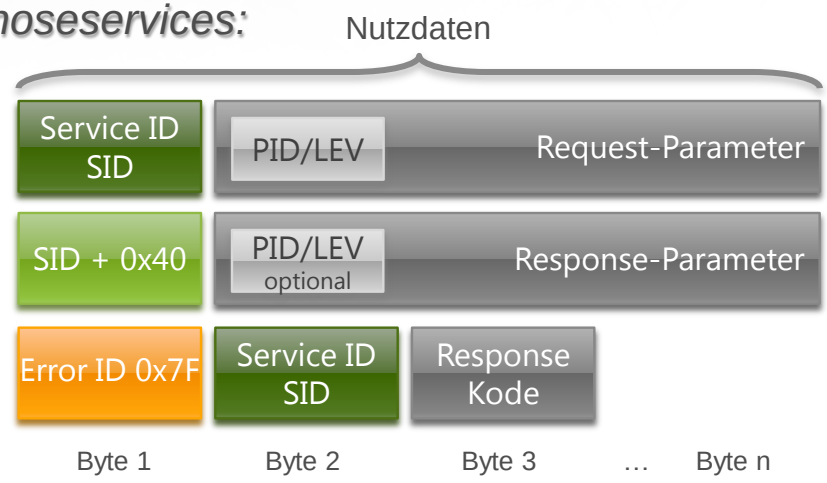
Diagnosekommunikation:



- Diagnosetester sendet Request an ein oder mehrere Steuergeräte
- Erhält vom SG eine positive oder negative Antwort
- SID (Service Identifier) kennzeichnet den Inhalt der Botschaft
- Die restlichen Bytes enthalten Parameter oder Daten (PID = Parameter Identifier oder LEV = Sub-Level-Function)

- Im Erfolgsfall gibt das SG eine positive Response zurück, dabei wird der SID um 0x40 erhöht und meist wird PID oder LEV wiederholt
- Im Fehlerfall gibt das SG eine negative Response mit dem festen Error ID 0x7F, dem wiederholten SID und einem Response-Kode zurück

Aufbau der Diagnoseservices:



Genormte Bereiche für Service IDs

Request	Positive Response	Beschreibung
0x00 ... 0x0F	0x40 ... 0x4F	OBD nach ISO 15031-5 Diagnose abgasrelevanter Systeme
0x10 ... 0x3E 0x83 ... 0x87	0x50 ... 0x7E 0xC3 ... 0xC7	UDS nach ISO 14229 KWP 2000 nach ISO 14230 Allgemeine Fahrzeugdiagnose
0x81 ... 0x82	0xC1 ... 0xC2	KWP 2000 auf K-Line nach ISO 14230
0xA0 ... 0xB9	0xE0 ... 0xF9	Reserviert für Fahrzeughersteller (OEM)
0xBA ... 0xBE	0xFA ... 0xFE	Reserviert für Steuergerätehersteller
Andere Bereiche		Reserviert

	Negative Response	Beschreibung
	0x7F	Fehlermeldungen

Typische Error Response-Codes

Reponse-Kode	Beschreibung
0x10	General reject
0x11, 0x12, 0x7E, 0x7F	Service or Subfunction not supported (in active Session)
0x13	Message length or format incorrect
0x31	Out of range
0x21	Busy – Repeat request
0x78	Busy – Response pending
0x22	Conditions not correct
0x24	Request sequence error
0x33	Security access denied
0x35	Invalid key
0x36	Exceed attempts

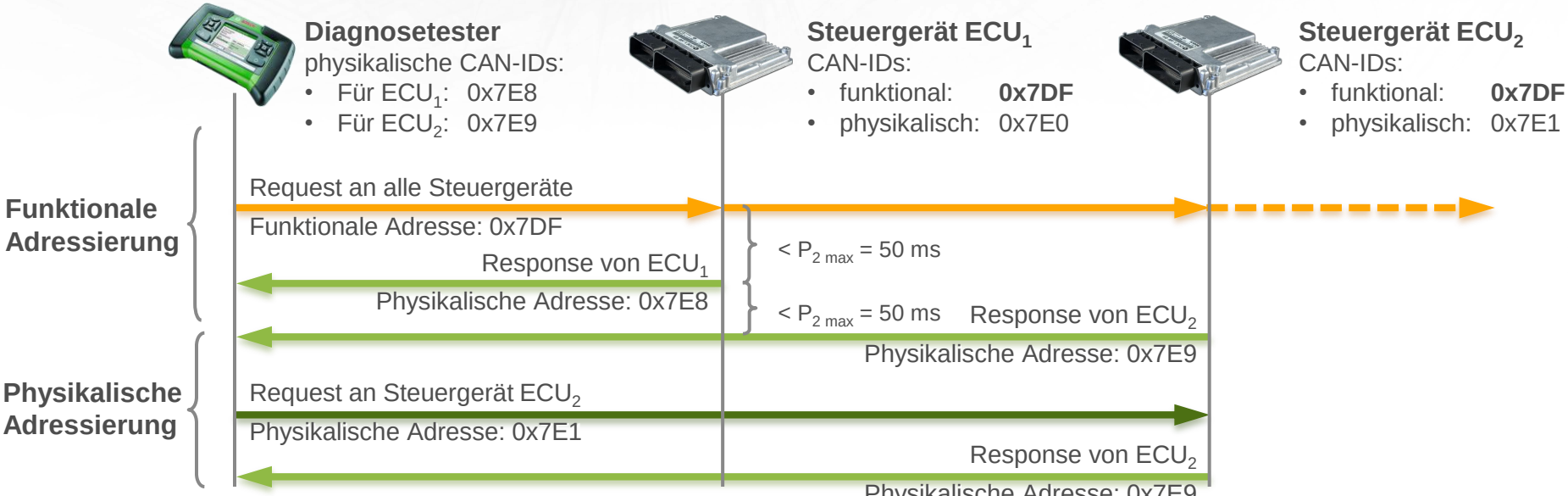
Adressierungsarten

Physikalische Adressierung

- Für jede Kommunikationsverbindung zwischen dem Diagnosetester und jedem einzelnen Steuergerät wird ein eindeutiges Paar von Adressen (z.B.: CAN IDs) festgelegt, jeweils eine Adresse für den Request und eine für die Response

Funktionale Adressierung (gilt nur für Steuergeräte und nicht für den Tester)

- Da der Diagnosetester u. U. nicht weiß, welche Steuergeräte im Fahrzeug verbaut sind, kann er einen Request auch gleichzeitig an alle Steuergeräte mit einer für alle Steuergeräte gemeinsamen funktionalen Adresse senden
- Auf die funktionale Adressierung antworten in der Regel mehrere Steuergeräte. Sie können dabei auch ihre physikalische Adresse zurückgeben.
- Steuergeräte können daher zwei Adressen haben: Eine physikalische und eine optional funktionale Adresse



Adressierungsarten (Adressen)

- Für OBD-relevante Geräte sind die Adressen in ISO 15765-4 fest vorgegeben:

	11 Bit CAN-ID	29 Bit CAN-ID
Funktionale Adressierung Request: Tester ⇨ alle Steuergeräte	0x7DF	0x18DB33F1
Physikalische Adressierung Response: Steuergerät$_X$ ⇨ Tester	0x7E8 + XX	0x18DAF1XX
Physikalische Adressierung Response: Tester ⇨ Steuergerät$_X$	0x7E0 + XX	0x18DAXXF1

XX = 0x00 für Steuergerät$_1$, 0x01 für Steuergerät$_2$ usw.

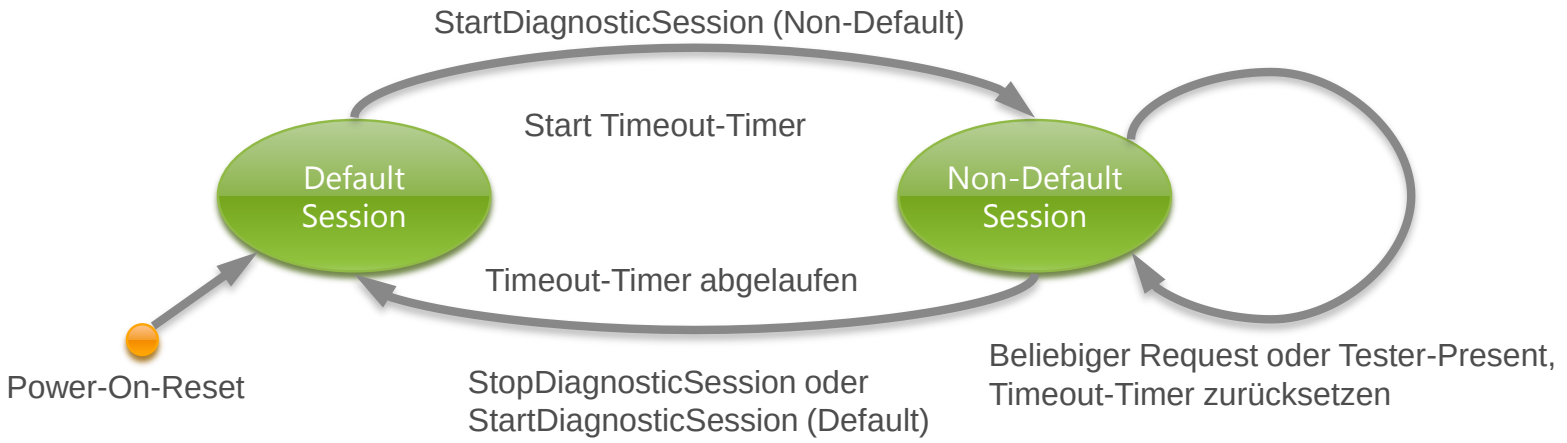
  - Geräteadressierung bei KWP 2000 (K-Line, 8bit Adressen, werden dort als Target- und Source-Adresse im Header der Diagnosebotschaft übertragen):
    - physikalische Adresse des Diagnosetesters: 0xF1
    - funktionale Adresse aller OBD Steuergeräte: 0x33
  - Erweiterte Adressierung: ISO 15765-2 erlaubt die Verwendung einer erweiterten Adressierung. Dabei wird das 1. Datenbyte der CAN-Botschaft zusätzlich zur Adressierung verwendet. Da dadurch das Nutzdatenfeld weiter verkleinert wird, versucht man aber, diese Adressierung zu vermeiden.

Beachte: Timeout bei funktionaler Adressierung

- Bei physikalischer Adressierung kann der Tester sofort nach Erhalt der Response den nächsten Request an ein Steuergerät senden
- Bei funktionaler Adressierung muss er nach jeder Response den Timeout P_2 = 50ms neu abwarten, da er nicht weiß, wie viele Steuergeräte antworten werden

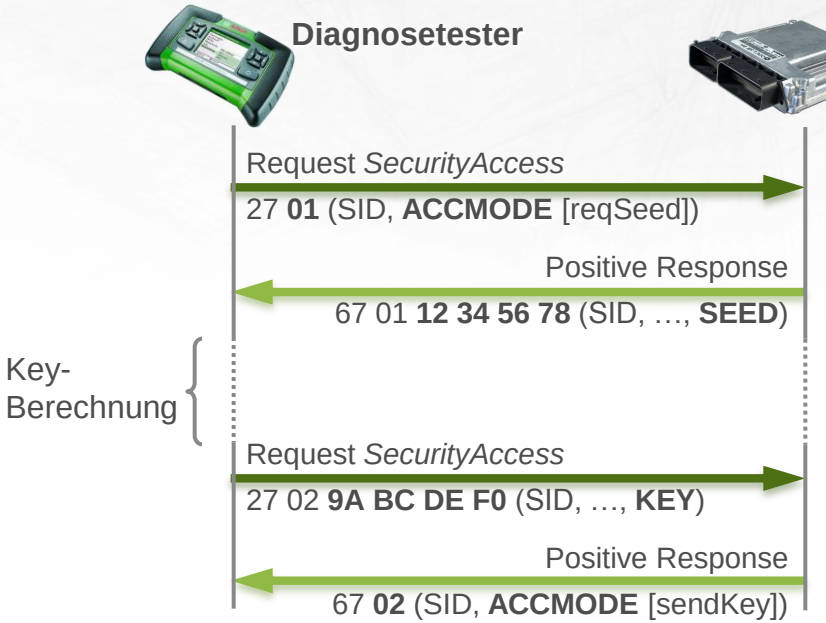
Sicherheitskonzept - Diagnosesitzungen

- In der *Default Diagnostic Session* (Normalzustand) ist nur ein Teil der Diagnosedienste, Messwerte und Steuergerätedaten verfügbar
- Über *Start Diagnostic Session* kann in einen speziellen Diagnosemodus umgeschaltet werden:
  - *Programming Session*: Flash-Programmierung
  - *Extended Diagnostic Session*: Justierung der Leerlaufdrehzahl usw.
  - OEM oder Steuergerätehersteller spezifische Sessions (Developer Session etc.)
- Bei der Umschaltung in eine erweiterte Diagnostic-Session muß ggf. ein *Security Access* durchgeführt werden.
- Beim Umschalten können außerdem ggf. die Kommunikationsparameter verändert werden (Baudrate, Timeout etc.)
- Die Umschaltung kann außerdem an gewisse Bedingungen (z.B. Motor aus) geknüpft sein
- Eine *Non Default Session* wird automatisch beendet, wenn der Tester nicht regelmäßig *Tester-Presents* oder andere Botschaften sendet (z.B.: Timeout P_{3max} = 5 s)



Security Access

- Nach der Umschaltung in eine erweiterte Diagnostic-Session muß in der Regel ein *Security Access* durchgeführt werden
- Der *Security Access* basiert auf einem „Seed & Key" Verfahren
- Ist das Steuergerät bereits in einer höheren Diagnostic-Session, sendet es auf den Request *SecurityAccess* eine Null-Seed zurück (Bytefolge mit Nullen)



„Seed & Key" Verfahren:

1. Der Tester fordert vom SG eine SEED an (Die Seed ist eine Zufallszahl)
2. Durch einen geheimen Algorithmus berechnet der Tester aus der SEED einen Key, den er zum SG zurücksendet
3. Das SG berechnet mit dem selben Algorithmus ebenfalls den Key und vergleicht ihn mit dem vom Tester
4. Bei Gleichheit gibt das SG die Diagnostic-Session frei und bestätigt dies durch eine positive Response

Fehlerspeicher

Inhalt des Fehlerspeichers (Ereignisspeicher):

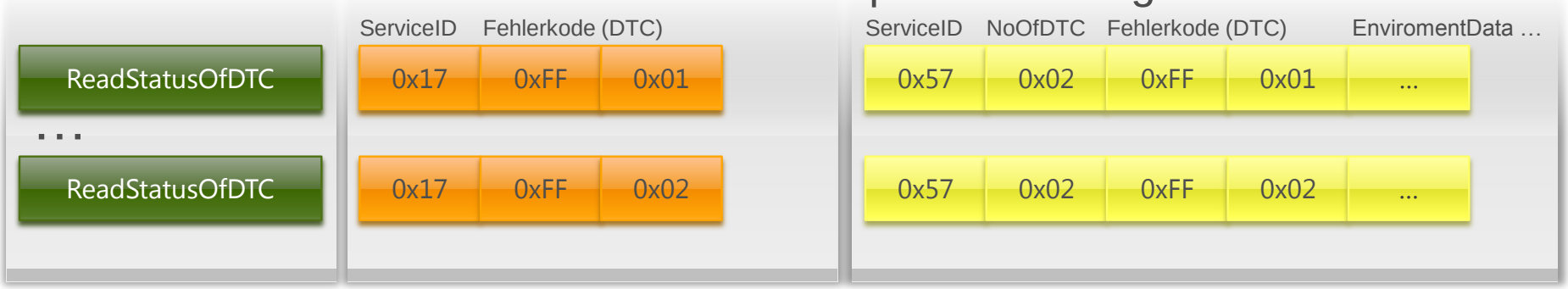
- Anzahl der Ereignisse
- StatusOfDTC
- Fehlerkode (DTC = Diagnostic Trouble Code)
- Umgebungsdaten

Auslesen des Fehlerspeichers (ISOTP Sequenz):

1. Lesen des Fehlerspeichestatus

StatusOfDTC 0x02 = requestIdentifed2ByteHexDTCAndStatus, 0x03 = requestSupported2ByteHexDTCAndStatus; GroupOFDTC 0xFFFF = allGroups

2. Iteratives Lesen der einzelnen Fehlerspeichereinträge



Diagnoseservice Request (Tester) Response (ECU)

Testfragen – Diagnoseprotokolle

1. Zu welcher Ebene des ISO/OSI-Schichtenmodells gehören die Diagnoseprotokolle?

a) Ebene 7
b) Ebene 4
c) Ebene 2

2. Was ist die der SID?

a) Die eindeutige Adresse eines Steuergerätes
b) Eine eindeutige Kennziffer, die den Zweck einer Diagnoseanfrage beschreibt.
c) Eine eindeutige Kennziffer, die einen Parameter eines Diagnosedienstes beschreibt.

3. Zu welchem Diagnoseprotokoll gehören die SIDs 0x01 bis 0x0F?

a) OBD
b) KWP 2000
c) UDS

4. Wie bezeichnet man das Kommunikationsmodell zwischen Diagnosetester und Steuergerät?

a) Frage- und Antwortspiel
b) Client-Server-Modell
c) Request-Response-Verfahren

5. Wer sendet seine OBD-Diagnosebotschaften mit funktionaler Adressierung?

a) Der Diagnosetester an ein einzelnes Steuergerät
b) Der Diagnosetester an eine Gruppe von Steuergeräten
c) Ein einzelnes Steuergerät an den Diagnosetester
d) Eine Gruppe von Steuergeräten bei den Antworten an den Diagnosetester

6. Woher weiß der Diagnosetester, von welchem Steuergerät er eine bestimmte Response erhalten hat?

a) Jede Response enthält als erstes Datenbyte die Adresse des antwortenden Steuergerätes
b) Jedes Steuergerät muss dem Diagnosetester die Response auf einer anderen CAN ID senden
c) Der Diagnosetester kann das überhaupt nicht feststellen

7. Woher weiß der Diagnosetester bei funktionaler Adressierung, dass alle Steuergeräte geantwortet haben?

a) Der Diagnosetester weiß immer, wie viele Steuergeräte im Fahrzeug verbaut sind. Da grundsätzlich jedes Steuergerät positiv oder negativ antwortet, muss der Diagnosetester die Antworten nur mitzählen.
b) Das ist egal. Der Diagnosetester nimmt ohnehin immer nur die allererste Antwort zur Kenntnis.
c) Der Diagnosetester wartet nach dem Request immer für eine vordefinierte Zeit (Timeout) auf Antworten.

8. Wozu dient das Seed & Key Verfahren?

a) Zum Zurücksetzen des Kilometerzählers.
b) Damit wird die Nummer des Werkstattmitarbeiters im Steuergerät gespeichert, um spätere Reklamationen einfacher zuordnen zu können.
c) Zum Freischalten kritischer Diagnosedienste.

9. Was muss der Fahrzeughersteller beim Seed & Key Verfahren unbedingt geheim halten?

a) Den Algorithmus zur Berechnung des Keys aus dem Seed
b) Den Initialisierungswert (Seed) für die Berechnung des Schlüssels
c) Den Schlüssel (Key)

10. Welche Diagnosesitzung ist nach dem Einschalten eines Steuergerätes aktiv?

a) Die Programming-Session
b) Die Extended-Diagnostic-Session
c) Die Default-Session

Inhalt

Geschichte On Board Diagnose

- 1970 US-Kongress beschließt Clean Air Act – Gründung der Environmental Protection Agency (EPA)
- 1982 GM implementiert erste interne OBD: ALCL (Assembly Line Communications Link) später umbenannt in ALDL (Assembly Line Diagnostics Link)
- 1987 Einführung der OBD für Neufahrzeuge ab Bj. 1988 durch kalifornische Umweltbehörde (CARB und EPA) zur Abgasüberwachung …

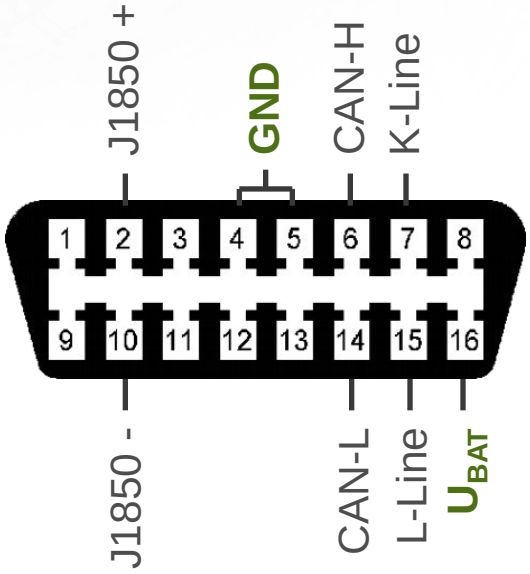
Allgemeines zur On Board Diagnose

Aufgaben:

- Ständige Überwachung aller abgasrelevanten Komponenten
- Erfassen Emissionserhöhungen während der Laufzeit
- Gewährleistung niedriger Abgasemissionen
- Schutz von Komponenten, z.B. Katalysator bei Fehlzündungen
- Meldung von abgasrelevanten Fehlern an den Fahrer
- Speichern von Fehlern
- Bereitstellung einer Diagnoseschnittstelle

OBD-Diagnosestecker:

- ISO 15031-3 oder SAE J1962
- Praktisch sind nur die mechanischen Abmessungen sowie die Pins für GND und U_{BAT} verlässlich
- 1 m Umkreis vom Fahrer – oft unter Amaturenbrett oder Aschebecher



Allgemeines zur Diagnoseschnittselle

- Beschrieben in ISO 15031-5 / SAE J1979 (OBD) und ISO 15765-4 (CAN für OBD)
- Vorgaben für das Transportprotokoll ISO TP und das CAN Bussystem
  - CAN-Bitraten 250 kbit/s oder 500 kbit/s, 11bit oder 29bit CAN IDs, müssen vom Tester automatisch erkannt werden, Tester darf beim Empfang effektiv keine Flußsteuerung erzwingen (darf nur ISO TP Flow Control Frame mit BS = 0, S_{Tmin} = 0 senden ⇨ Block: 30 00 00)
  - Diagnose-Requests verwenden nur die funktionale Adresse 0x7DF
  - Steuergeräte, die eine OBD-Anfrage nicht unterstützen, sollen nicht antworten (keine negative Response, um Bussystem zu entlasten), d.h. Tester muss immer Timeout abwarten
- Zulässige Diagnose-Requests, sgn. „Test Modes“:

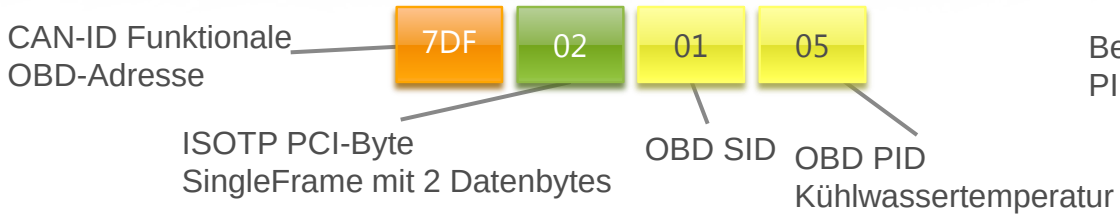
Funktionsgruppe	SID	Beschreibung
Meß- und Fahrzeugdaten	0x01	Abfrage von Messdaten
	0x09	Abfrage von Fahrzeuginformationen wie Fahrgestellnummer
Fehlerspeicher	0x03	Fehlercodes lesen (DTC Diagnostic Trouble Codes)
	0x02	Umgebungsbedingungen der Fehler lesen (Freeze Frames)
	0x07	OBD-Fehlercodes für „vorläufig defekte“ Fehler lesen
	0x04	Fehlerspeicher löschen (DTCs, Freeze Frames und MIL)
Test von abgasrelevanten Komponenten	0x06	Selbsttest-Ergebnisse für vollständige Testzyklen lesen
	0x07	Selbsttest-Ergebnisse aus dem letzten Fahrzyklus lesen
	0x08	Komponententest aktivieren

Beispiel: Messdatenlesen

- SID = 0x01 (Abfrage von Messdaten); Auswahl des Messwerts erfolgt über PID:

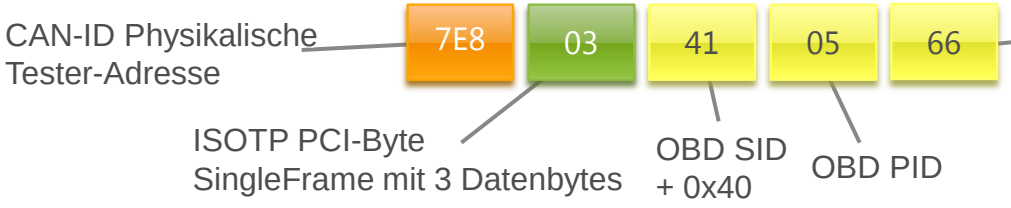
PID	Bedeutung	Datengröße	Wertebereich [min … max]
0x01	Anzahl Fehler im Fehlerspeicher u.a.	8 bit	siehe unten
0x04	Motorlast	8 bit	0 … 100 %
0x05	Kühlwassertemperatur	8 bit	-40 … +215 °C
0x0C	Motordrehzahl	16 bit	0 … 16383,75 1/min
…	…	…	…

- Welche PIDs der jeweilige SID unterstützt, kann mit dem PID 0x00 abgefragt werden (Antwort: Bitfeld)
- Request zum Lesen der Kühlwassertemperatur (PID = 0x05) vom Tester zum Steuergerät



Bei Bedarf kann auch eine Liste von PIDs mit einem Request abgefragt werden.

- Response des Motor-Steuergerätes zum Tester



$$\text{Temperatur} = \frac{0x66}{256} \cdot [215 - (-40)]°C + (-40°C) = 62°C$$

Auszug aus ISO 15031-5

Table 127 — Request current powertrain diagnostic data request message

Data Byte	Parameter Name	Cvt	Hex Value	Mnemonic
#1	**Request current powertrain diagnostic data request SID**	**M**	**01**	**SIDRQ**
#2	PID#1 (see Annex B)	M	xx	PID
#3	PID#2 (see Annex B)	U	xx	PID
#4	PID#3 (see Annex B)	U	xx	PID
#5	PID#4 (see Annex B)	U	xx	PID
#6	PID#5 (see Annex B)	U	xx	PID
#7	PID#6 (see Annex B)	U	xx	PID
U = User Optional — the parameter may be present or not				

Request für Diagnoseservice
SID = 0x01 „Messwerte lesen“
PID = 0x02 “Kühlwassertemeratur (ECT)“

Table 128 — Request current powertrain diagnostic data response message

Data Byte	Parameter Name	Cvt	Hex Value	Mnemonic
#1	**Request current powertrain diagnostic data response SID**	**M**	**41**	**SIDPR**
	data record of 1st supported PID = [			PIDREC_
#2	PID#1	M	xx	PID
#3	data A,	M	xx	DATA_A
#4	data B,	C1	xx	DATA_B
#5	data C,	C1	xx	DATA_C
#6	data D]	C1	xx	DATA_D
:	:	:	:	:
	data record of mth supported PID = [			PIDREC_
#n-4	PID#m	C2	xx	PID
#n-3	data A,	C2	xx	DATA_A
#n-2	data B,	C3	xx	DATA_B
#n-1	data C,	C3	xx	DATA_C
#n	data D]	C3	xx	DATA_D
C1 = Conditional — “data B - D” depend on selected PID value				
C2 = Conditional — parameter is only present if supported by the ECU				
C3 = Conditional — parameters and values for “data B - D” depend on selected PID number and are only included if PID is supported by the ECU				

Response

Definition und Skalierung
PID 0x05 “Engine Coolant Temperature”

Table B.6 — PID $05 definition

PID (hex)	Description	Data byte	Min. value	Max. value	Scaling/bit	External test equipment SI (Metric) / English display
05	Engine Coolant Temperature	A	-40 °C	+215 °C	1 °C with -40 °C offset	ECT: xxx °C (xxx °F)
	ECT shall display engine coolant temperature derived from an engine coolant temperature sensor or a cylinder head temperature sensor. Many diesels do not use either sensor and may substitute Engine Oil Temperature instead.					

Auszug aus ISO 15031-5

Table 185 — Request vehicle information request message

Message direction:	External test equipment → All ECUs		
Message Type:	Request		
Data Byte	**Description (all values are in hexadecimal)**	**Byte Value (Hex)**	**Mnemonic**
#1	Request vehicle information request SID	09	SIDRQ
#2	InfoType: 02 - VIN (Vehicle Identification Number)	02	INFTYP

Request für Diagnoseservice
SID = 0x09 „RequestVehicleInformation“
PID = 0x02 “VIN lesen“

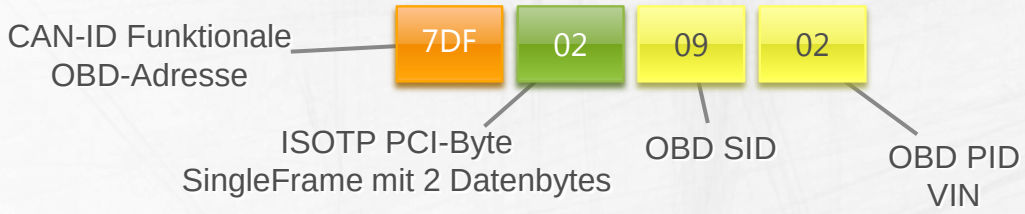
Table 186 — Request vehicle information response message

Message direction:	ECU #1 → External test equipment		
Message Type:	Response		
Data Byte	**Description (all values are in hexadecimal)**	**Byte Value (Hex)**	**Mnemonic**
#1	Request vehicle information response SID	49	SIDPR
#2	InfoType: 02 - VIN (Vehicle Information Number)	02	INFTYP
#3	Number of data items: 01	01	NODI
#4	1st ASCII character of VIN: ‘1’	31	VIN
#5	2nd ASCII character of VIN: ‘G’	47	VIN
#6	3rd ASCII character of VIN: ‘1’	31	VIN
#7	4th ASCII character of VIN: ‘J’	4A	VIN
#8	5th ASCII character of VIN: ‘C’	43	VIN
#9	6th ASCII character of VIN: ‘5’	35	VIN
#10	7th ASCII character of VIN: ‘4’	34	VIN
#11	8th ASCII character of VIN: ‘4’	34	VIN
#12	9th ASCII character of VIN: ‘4’	34	VIN
#13	10th ASCII character of VIN: ‘R’	52	VIN
#14	11th ASCII character of VIN: ‘7’	37	VIN
#15	12th ASCII character of VIN: ‘2’	32	VIN
#16	13th ASCII character of VIN: ‘5’	35	VIN
#17	14th ASCII character of VIN: ‘2’	32	VIN
#18	15th ASCII character of VIN: ‘3’	33	VIN
#19	16th ASCII character of VIN: ‘6’	36	VIN
#20	17th ASCII character of VIN: ‘7’	37	VIN

Response

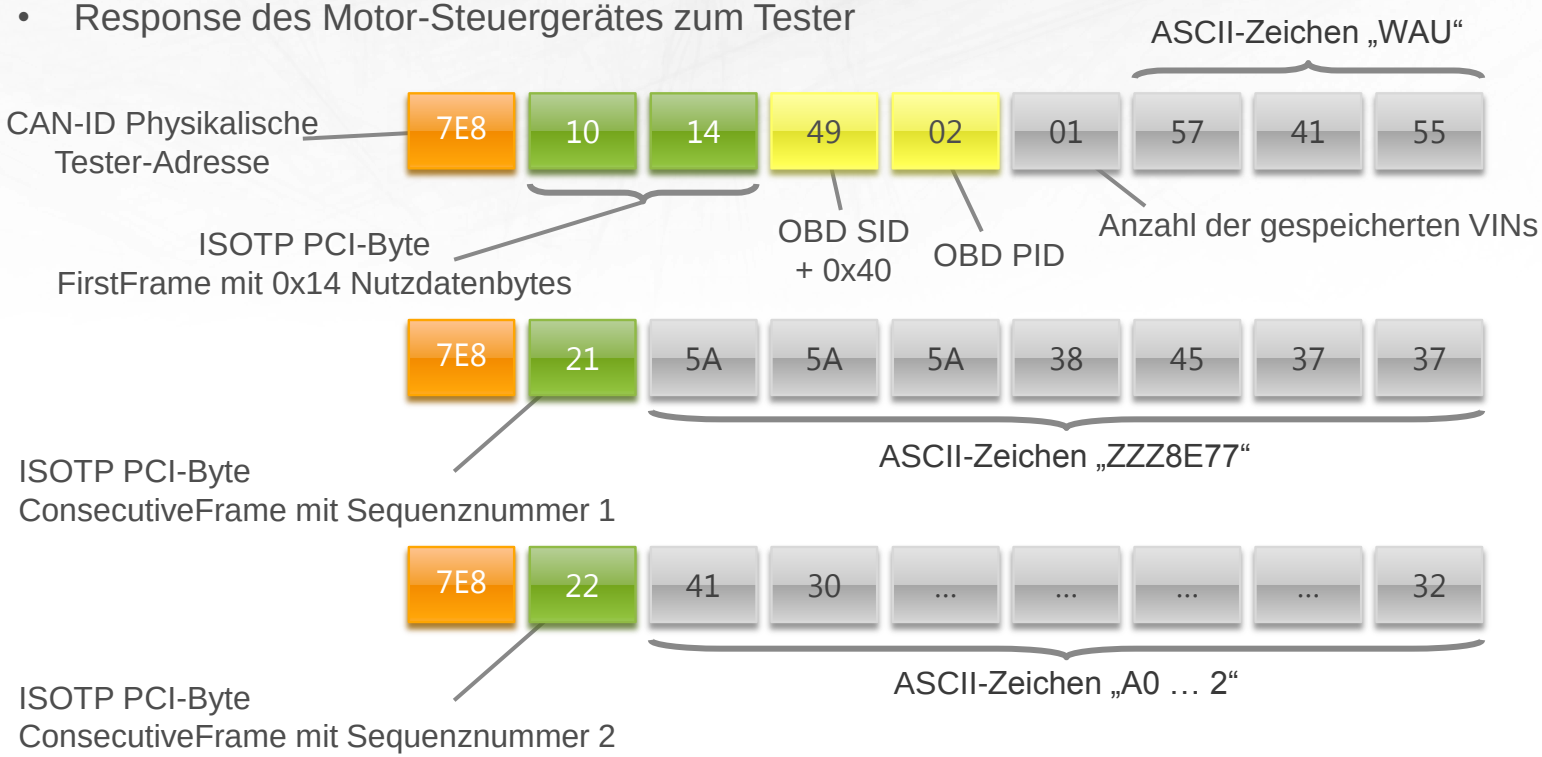
Beispiel: Lesen Fahrgestellnummer

- Request zum Lesen der Vehicle Identification Number vom Tester zum Steuergerät (SID: 0x09, PID: 0x02)



VIN = Anzahl der Datensätze + 17 ASCII Zeichen

- Response des Motor-Steuergerätes zum Tester



Beispiel: Fehlerspeicher

1. Abfrage, ob und wie viele Fehler gespeichert sind

SID	PID
01	01

SID	PID	$Data_1$	$Data_2$	$Data_3$	$Data_4$
41	01	83	33	FF	63

$Data_1$: Bit 0 … 6 = Anzahl der Fehler, Bit 7 = MIL On
$Data_{2 \dots 4}$: Informationen über aktuelle Überwachungen – Monitoring (Fehlzündungen, Einspritzanlage, Katalysator etc.)

2. Abfrage der OBD-Fehlercodes – Diagnostic Trouble Codes

SID
03

SID	DTC_1 (High, Low)		DTC_2 (High, Low)		DTC_3 (High, Low)	
43	01	19	XX	XX	XX	XX

3. Lesen der gespeicherten Messdaten (Freeze Frames) zu einem gespeicherten Fehler

SID	PID	FRNO
02	02	XX

SID	PID	FRNO	$Data_1$	$Data_2$	$Data_3$
42	02	XX	XX	XX	...

FRNO: Messdaten, Angabe über PIDs wie bei SID 0x01

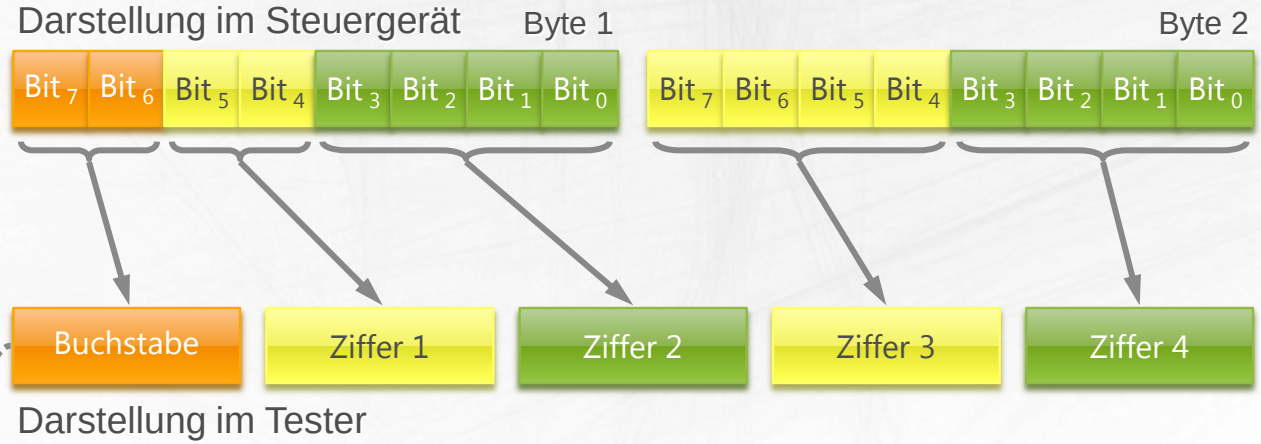
$Data_{1 \dots n}$: PID + Messwert, Skalierung wie bei SID 0x01

SID
04

SID
44

Request (Tester) Response (ECU)

Fehlerkodes nach ISO 15031-6



Beispiel:

DTC 0x0119 ⇨ P 0 1 1 9

Antriebsstrang, DTC nach ISO/SAE, Einspritzanlage, Wassertemperatursensor, Wackelkontakt

Buchstabe

Bit	L
00	P
01	C
10	B
11	U

P = Powertrain (Antrieb)

C = Chassis (Fahrwerk)

B = Body (Karosserie)

U = Network (Bussystem)

1. Ziffer

0 = DTC nach ISO 15031-6

1,2 = DTC des OEM

2. Ziffer

1,2 = Einspritzanlage

3 = Zündung

4 = Abgas/Luft-system

usw.

3. und 4. Ziffer

Komponente und Fehlerart

Testfragen – OBD

1. Welches Bussystem schreiben amerikanische Zulassungs-behörden für neue PKW's seit 2008 für die OBD-Diagnose vor?

a) K-Line
b) SAE J1850
c) CAN
d) LIN
e) FlexRay
f) MOST

2. Was heißt MIL?

a) Mobile Internet Link
b) Malfunction Indicator Lamp
c) Modular Integration Layer

3. Welche Diagnosedienste werden von OBD angeboten?

a) Neuprogrammierung von Steuergeräten zur Anpassung an neue Abgasnormen
b) Nachjustieren der Leerlaufdrehzahl, der Abgasrückführung usw.
c) Abfrage von Meßdaten, Fahrgestellnummern usw.

4. Mit einem CAN-Monitor beobachten Sie die folgende Diagnosebotschaft: „41 04 80" (alle Werte hexadezimal, ohne CAN ID und ISOTP PCI). Um was handelt es sich?

a) Diagnose Request
b) Diagnose Response

5. Mit einem CAN-Monitor beobachten Sie die folgende Diagnosebotschaft: „41 04 80" (alle Werte hexadezimal, ohne CAN ID und ISOTP PCI). Zu welchem Diagnosedienst gehört diese Botschaft?

a) Lesen von Messwerten
b) Lesen der Fahrgestellnummer
c) Lesen von Fehlercodes
d) Löschen des Fehlerspeichers

6. Mit einem CAN-Monitor beobachten Sie die folgende Diagnosebotschaft: „41 04 80" (alle Werte hexadezimal, ohne CAN ID und ISOTP PCI). Welcher dezimale Wert wird hier geliefert?

a) 80°C Kühlwassertemperatur
b) 128 °C Kühlwassertemperatur
c) 50 % Motorleistung
d) 80 PS Motorleistung
e) 128 PS Motorleistung
f) 800 1/min Motordrehzahl
g) 1280 1/min Motordrehzahl

7. Die aktuelle Motordrehzahl sei 900 1/min. Welchen hexadezimalen Wert muss das Steuergerät zurücksenden?

a) 0x0384
b) 0x0E10
c) 0x10E0
d) 0x8403
e) keinen der obigen Werte

Inhalt

UDS nach ISO 14229 und ISO 15765-3

- UDS = **Unified Diagnostic Services** (für die allgemeine Fahrzeugdiagnose)
- Im Gegensatz zu OBD
  - keine Vorgaben zur Wahl der CAN Identifier und Baudrate
  - nur SID und LEV (= PID) festgelegt, Daten durch OEM frei definierbar (z.B. Fehlercodes)
  - Funktionen, z.B. Zugriff auf Fehlerspeicher, teilweise äquivalent zu OBD, aber andere SID und andere Parameter
- Botschaftsaufbau wie bei OBD
  - Auswahl des Dienstes durch **SID** (1. Byte)
  - Auswahl von Unterfunktionen durch das **Subfunction Level** Byte $LEV_{Bit\ 6...0}$ (entspricht dem PID bei OBD, LEV wird nicht bei allen UDS Diensten verwendet)
  - Optional Verzicht auf *Positive Response* Botschaften: Wenn im Request $LEV_{Bit\ 7} = 1$ gesetzt, verzichtet der Tester auf eine positive Antwort des Steuergeräts – Reduzierung der Buslast (z.B. beim Dienst *Tester Present*). Negative Antworten werden immer gesendet.
- **Diagnosedienste** (Liste siehe nächste Seite)
  - Zugriff auf den Fehlerspeicher und Stellgliedansteuerung ähnlich OBD, aber Aufbau des Fehlerspeichers, Fehlercodes usw. nicht standardisiert
  - Lesen und Schreiben von Messwerten und anderen Steuergerätewerten über vordefinierte PIDs, bei UDS als *Identifier* bezeichnet, z. T. durch UDS standardisiert, z.B. für die *Vehicle Identification Number (VIN)*, in der Mehrzahl aber OEM-spezifisch
  - Alternativ Lesen und Schreiben direkt über Speicheradressen
    - Vorteil: Zugriff auf beliebige statt nur vordefinierte Werte
    - Nachteil gegenüber Identifiern: Adressen in der Regel abhängig von der ECU-Softwareversion

Diagnosedienste* UDS und KWP 2000

Funktionsgruppe	SID UDS	SID KWP	Default Session	Bezeichnung UDS	Bezeichnung KWP 2000	Beschreibung
Diagnostic and Communication Management	0x10		X	DiagnosticSessionControl	StartDiagnosticSession	Steuerung der Diagnosesitzung
		0x20			StopDiagnosticSession	
	0x3E		X	TesterPresent		
	0x27			SecurityAccess		
	0x11		X	ECUReset		
		0x81	X		StartComm (Verbindungsaufbau)	Steuerung der Buskommunikation
		0x82	X		StopComm (Verbindungsabbau)	
	0x28			CommunicationControl	DisableNormalMessageTransm.	
		0x29			EnableNormalMessageTransm.	
	0x85			ControlDTCSetting		
	0x83			AccessTimingParameters		
	0x87	0x84		LinkControl	NetworkConfiguration	
	0x86		(X)	ResponseOnEventService		
Data Transmission		0x21			ReadDataByLocalIdentifier	Lesen und Schreiben von Messwerten und Steuergerätedaten
	0x22		(X)	ReadDataByIdentifier	ReadDataByCommonIdentifier	
		0x1A	X		ReadECUIdentification	
	0x23		(X)	ReadMemoryByAddress		
		0x3B			WriteDataByLocalIdentifier	
	0x2E		(X)	WriteDataByIdentifier	WriteDataByCommonIdentifier	
	0x3D		(X)	WriteMemoryByAddress		
	0x2A	0x26		ReadDataByPeriodicIdentifier	SetDataRates	
	0x2C		X	DynamicallyDefineDataIdentifier		

*Auszug

Diagnosedienste* UDS und KWP 2000

Funktionsgruppe	SID UDS	SID KWP	Default Session	Bezeichnung UDS	Bezeichnung KWP 2000	Beschreibung
Stored Data Transmission	0x14		X	ClearDiagnosticInformation		Lesen und Schreiben des Fehlerspeichers
	0x19	0x13	X	ReadDTCInformation	ReadDiagnosticTroubleCode	
		0x12	X		ReadFreezeFrameData	
		0x18	X		ReadDTCsByStatus	
		0x17	X		ReadStatusofDTCs	
Input/Output Control		0x30			InputOutputControlByLocalId	
	0x2F			InputOutputControlByIdentifier	InputOutputControlByCommonId	
Remote Activation of Routine	0x31		(X)	RoutineControl	StartRoutineByLocalIdentifier	Ausführen von Routinen in der ECU, z.B. Stellgliedtests oder Programmierroutinen
		0x32	(X)		StopRoutineByLocalIdentifier	
		0x38	(X)		StartRoutineByAddress	
		0x39	(X)		StopRoutineByAddress	
		0x33	(X)		RequestRoutineResultsByLocalId	
		0x3A	(X)		RequestRoutineResultsByAddress	
Upload / Download	0x34			RequestDownload		Lesen und Schreiben größerer Datenblöcke Haupteinsatzgebiet: Flash-Programmierung
	0x35			RequestUpload		
	0x36			TransferData		
	0x37			RequestTransferExit		

*Auszug

Sonstiges

Kommunikationsparameter

- Beim Umschalten in eine erweiterte Diagnose-Session oder danach können Parameter des Transportprotokolls bzw. des Bussystems umgeschaltet werden, z.B.:
  - Baudrate
  - Timeout-Werte (über *AccessTimingParameters)*
- Um die verfügbare Busbandbreite bei der Flash-Programmierung zu vergrößern, schaltet man in den anderen Steuergeräten meist mit *CommunicationControl* die *NormalCommunication* ab, d.h. sperrt das Senden der regulären Busbotschaften.
- Bei Stellgliedtests oder bei der Flash-Programmierung wird in der Regel mit *ControlDTCSetting* die Fehlerspeicherung deaktiviert

Spezielle Möglichkeiten

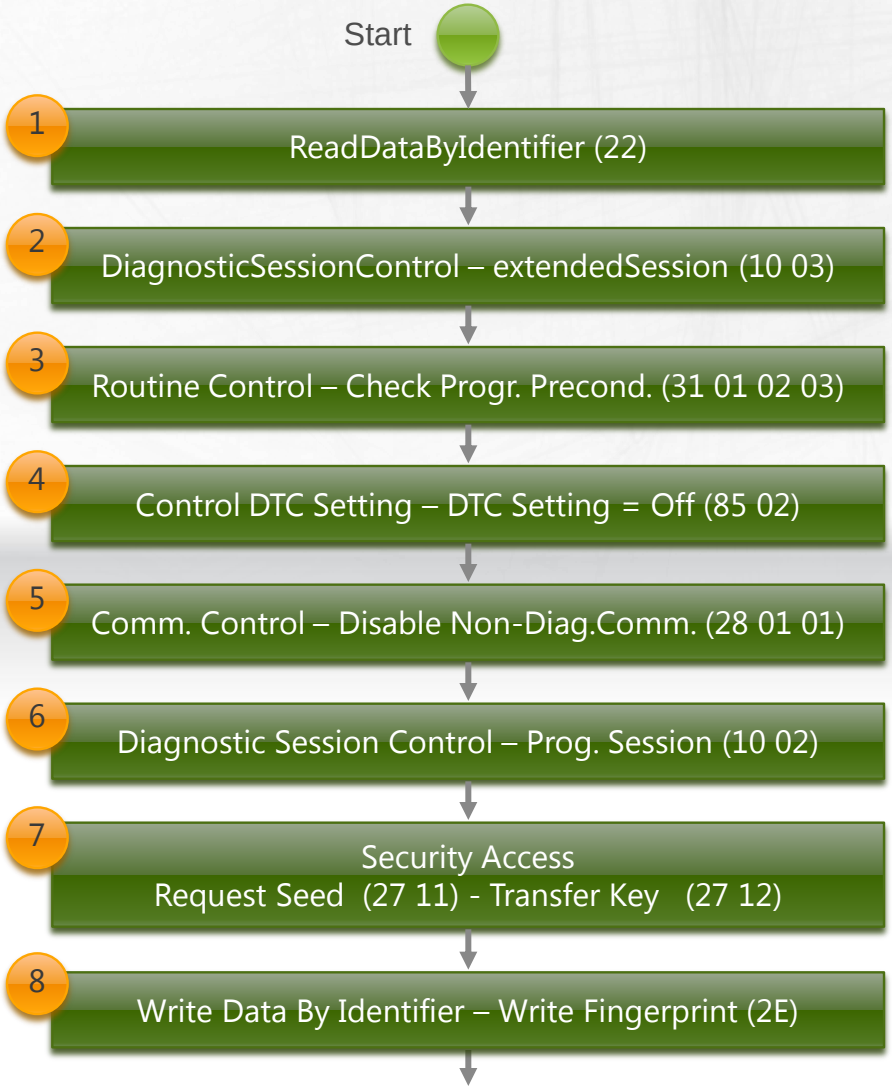
Anwendung vorzugsweise bei der Datenaufzeichnung an Prüfständen bzw. während der Applikation

- Periodisches Senden von Steuergerätewerten nach einmaligem Request *ReadDataByPeriodicIdentifier* (statt Polling wie bei OBD bzw. KWP 2000). Die Sendefrequenz kann in drei vom Gerätehersteller festgelegten Stufen gewählt werden. Für die periodischen Botschaften sind optionale Formate ohne PCI und SID zugelassen, um den Bus zu entlasten.
- Ereignisgesteuertes Senden von Steuergerätewerten nach einmaligem Request *ResponseOnEventService*. Ereignisse sind z.B. Zeitgeberinterrupts, Fehlerspeichereinträge (Vorbild SAE J1939), Werteänderung oder Schwellwertüberschreitung.

Praktische Umsetzung

- Komplexer Standard, z.T. aufwendige Dienste (*ResponseOnEvent*, *ReadScalingData*) und Redundan-zen (*WriteMemoryByAddress*, *TransferData*) ⇨ herstellerspezifische Untermengen

Beispiel: Flashprogrammierung I



Auslesen der Steuergeräteidentifikation, um die Hard- und Softwarekompatibilität zu prüfen

Umschalten in eine erweiterte Diagnosesitzung, da nachfolgende Dienste in Default Session unzulässig

Vorbedingungen prüfen, z.B. Motor aus

Abschalten der Fehlerspeicherung

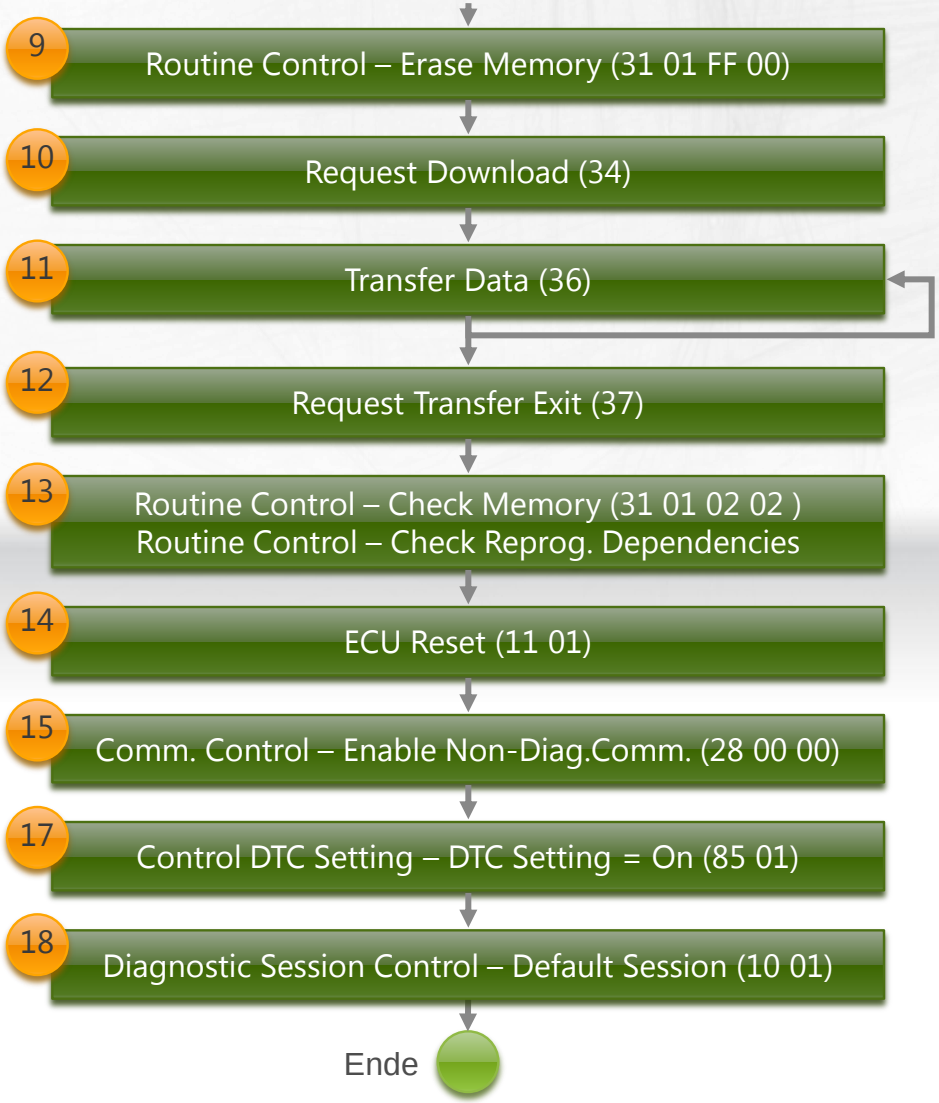
Abschalten der normalen Gerätekommunikation

Umschalten in die eigentliche Programmiersitzung

Login Prozedur: Seed anfordern – Key an ECU senden

Kennung des Testers schreiben, damit Programmiervorgang dokumentiert ist

Beispiel: Flashprogrammierung II



Flash-Speicher löschen

Steuergerät für Download vorbereiten

Innerhalb einer Schleife: Blockweise Übertragen der Datenblöcke zum Steuergerät

Download abschließen – *Response Pending* (Oft wird erst die eigentliche Flashprogrammierung durchgeführt)

Programmierung verifizieren und Konsistenz sicherstellen

Neustart des Steuergerätes

Wiedereinschalten der normalen Kommunikation

Fehlerspeicherung wieder einschalten

Ende der Programmiersitzung ⇨ Default Session

Inhalt

Unterschiede zu UDS

- Die direkte Adaption des K-Line-Protokolls KWP 2000 auf CAN (**ISOTP**)
- Wurde nie offiziell standardisiert, war aber in einer Vorabversion ISO/DIS 15765-3 publiziert und wurde (und wird noch) von vielen Herstellern eingesetzt
- Unterschiede zwischen UDS und KWP 2000:
  - Expliziter Verbindungsauf- und -abbau (bei K-Line-Systemen, Dienste SID = 0x81, 0x82)
  - Kurze und lange Identifier für Steuergerätedaten und -funktionen (Local Identifier, Common Identifier)
  - Viele KWP 2000-Dienste wurden bei UDS zu Unterfunktionen eines Dienstes zusammengefasst, so dass die Anzahl der SIDs geringer wurde
  - Parameter der meisten Dienste geändert oder (etwas) genauer spezifiziert, insbesondere das Handling der Diagnosesitzungen
  - Ereignisgesteuerte Kommunikation nicht möglich, periodische Datenübertragung nur eingeschränkt

Testfragen – UDS & KWP 2000

1. **In welchem Standard wird UDS definiert?**
   a) ISO 9141
   b) ISO 11898
   c) ISO 14229 / ISO 15765
   d) ISO 14230
   e) SAE J1708
   f) SAE J1839

2. **Was ist bei OBD und UDS gleich?**
   a) Das Request-Response Kommunikationskonzept und der Botschaftsaufbau mit Service und Parameter IDs
   b) Die Werte der SIDs und PIDs
   c) Die Dienste zum Lesen und Schreiben von Steuergerätedaten

3. **Welche UDS-Funktionsgruppe dient zum Lesen und Schreiben von Steuergerätedaten?**
   a) Diagnostic and Communication Management
   b) Data Transmission
   c) Stored Data Transmission
   d) Upload/Download

4. **Wie groß sind die Identifier für den Zugriff auf Steuergerätewerte bei UDS?**
   a) 8 bit
   b) 16 bit
   c) 32 bit

5. **Mit welchem Dienst kann man jede beliebige Steuergerätespeicherzelle lesen (vorausgesetzt, der Security Access war erfolgreich)?**
   a) Read Data by Identifier
   b) Read Memory by Address
   c) Request Download
   d) Read DTC Information

6. **Mit welchem UDS-Diagnosedienst kann man das Löschen des Flash-Speichers auslösen?**
   a) Write Memory by Address
   b) Write Data by Identifier
   c) Input/Output Control
   d) Routine Control

Vielen Dank für Ihre Aufmerksamkeit

Sprechen Sie mit uns!

Wir helfen Ihnen gern.

www.emotive.de