Einführung in CAN und LIN

Vortrag Systemverständnis Fahrzeug

des Studiengangs Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Manuel Fritz

30.01.2024

Bearbeitungszeitraum	01.12.2023 - 01.02.2023
Kurs	STG-TINF23ITA
Ausbildungsfirma	Robert Bosch GmbH, Stuttgart-Feuerbach
Dozent	Prof. Dr. Axel Richter

Abstract

In diesem Handout, dass zu einem Vortrag im Fach Systemverständnis Fahrzeug des Kurses ITA23 an der DHBW Stuttgart gehört, wird der CAN-Bus und der LIN-Bus erklärt. Dazu gehört ein Überblick zu den Grundlagen eines Busnetzwerks, sowie eine Erklärung des CAN- und LIN-Bus. Dafür wird auf die Vor- und Nachteile beider eingegangen, sowie deren Anwendungszweck im Automobil erklärt.

Stand: 30. Januar 2024

Inhaltsverzeichnis

Abkürzungsverzeichnis

z.B.	zum Beispiel
CAN	Controller Area Network
LIN	Local Interconnect Network
ESP	Elektronisches Stabilitätsprogramm
ABS	Antiblockiersystem
Kfz	Kraftfahrzeug
EMV	Elektromagnetische Verträglichkeit
LSB	Least Significant Bit
LDF	LIN Description File
ISO	International Organization for Standardization

Abbildungsverzeichnis

1 Grundlagen eines Netzwerks

Ein Bussystem ist ein System, bei dem mehrere Teilnehmer über einen gemeinsamen Übertragungsweg verbunden sind und darüber Daten übertragen. Ein Teilnehmer kann Daten senden oder empfangen, ohne direkt in die Übertragung der anderen Teilnehmer einzugreifen.

1.1 Klassifizierung und Überblick von Bussystemen

Bussysteme können grundsätzlich in vier Klassen unterteilt werden, die jeweils einen Unterschiedlichen Anwendungszweck erfüllen.

Klasse	Übertragungsrate	Anwendung	Vertreter
A	Geringe Datenraten mit bis zu 10 kBit/s	Vernetzung von Aktoren und Sensoren	LIN
B	Mittlere Datenraten mit bis zu 125 kBit/s	Komplexe Mechanismen zur Fehlerbehandlung, Vernetzung von Steuergeräten im Komfortbereich	Low-Speed-CAN
C	hohe Datenraten mit bis zu 1 Mbit/s	Echtzeitanforderungen, Vernetzen von Steuergeräten im Antriebs- und Fahrwerksbereich	High-Speed-CAN
C+	sehr hohe Datenraten mit bis zu 10 Mbit/s		FlexRay
D	sehr hohe Datenraten mit über 10 Mbit/s	Vernetzung von Steuergeräten im Telematik- und Multimediabereich	MOST

Abbildung 1.1: Tabelle Busarten, überarbeitet, Quelle: [**S.82**; Rei11]

1.2 Einsatzgebiete im KFZ

Elektronische Anwendungen sind aus dem heutigen Automobil nicht mehr wegzudenken, weshalb es nötig ist, die unterschiedlichen Steuergeräte untereinander zu vernetzen, was die Notwendigkeit für ein Bussystem bringt. Grundsätzlich ist die Vernetzung von Autos in vier Hauptbereiche mit unterschiedlichen Anforderungen unterteilt. Die ersten zwei Hauptbereiche, der Antriebsstrang und das Chassis, beinhalten Echtzeitanwendungen, die hohe Anforderungen an die Leistungsfähigkeit benötigen, da die Zyklenzeiten des Busses oft im Bereich von wenigen Millisekunden liegen. Diese werden der Klasse C zugewiesen und es wird hauptsächlich High-Speed-CAN mit 500.000 Bitsendungen pro Sekunde (500k Baud) eingesetzt, da die Klasse C zusätzlich zu den recht hohen Datenraten auch sehr hohe Anforderungen an die Fehlertoleranz hat. Beispiele für Echtzeitanwendungen wären das Motormanagement, Antiblockiersystem (ABS) oder Elektronisches Stabilitätsprogramm (ESP), da diese Systeme unterschiedliche Daten aus verschiedenen Sensoren und Steuergeräten zuverlässig verarbeiten müssen. Ein weiterer Hauptbereich ist der Innenraum, der in den Bereich der Multiplex-Anwendung fällt. Bei einer Multiplex-Anwendung werden verschiedene Signale über eine Leitung übertragen, was vor allem für die Steuerung von Karosserieelementen (z.B. Anzeigen, Beleuchtung) und Komfortelementen (z.B. Sitzverstellung, Klimaregelung) nützlich ist. Diese werden meist zur Klasse B oder teilweise auch zur

Klasse A zugeordnet. Diese Busse besitzen im Vergleich zur Klasse C geringere Anforderungen an die Datenübertragungsrate und Fehlererkennung, da damit keine kritischen Anwendungen gesteuert werden. Der vierte Hauptbereich ist das Infotainment, was in der Telematik beinhaltet wird und von Bussen der Klasse D angesteuert wird.

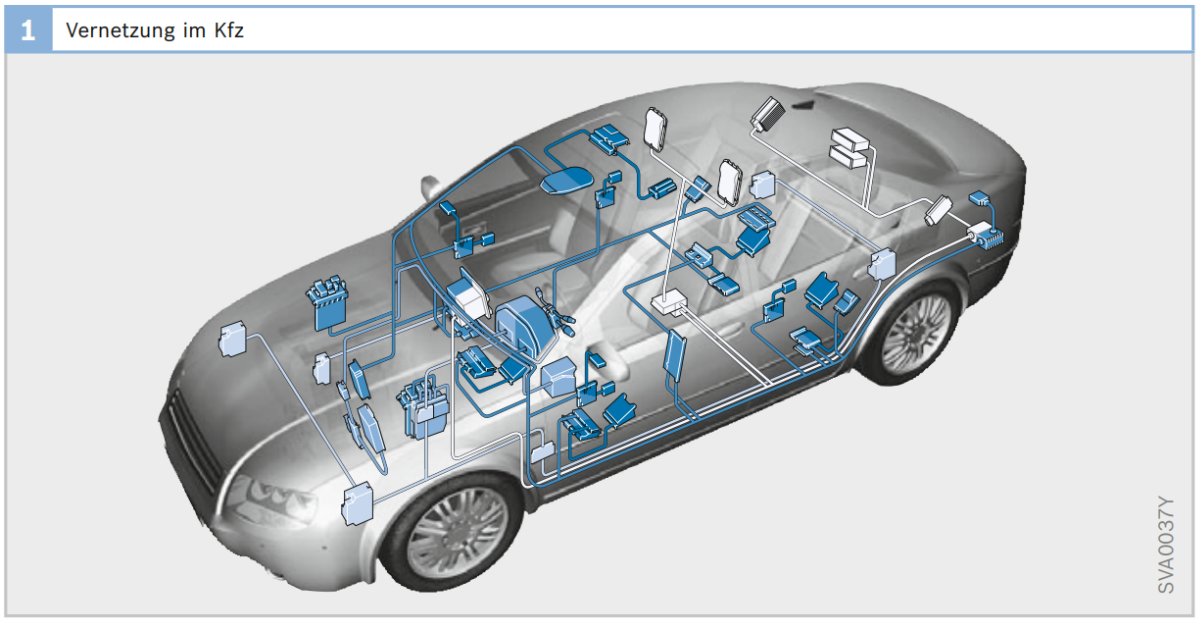


Abbildung 1.2: Grafik Vernetzung im Kfz, Quelle: [**S.85**; Rei11]

1.3 Anforderungen an ein Bussystem

1.3.1 Datenübertragungsrate

Die benötigte Datenübertragungsrate ist stark vom Einsatzweck abhängig. Ein Schalter zum Einschalten eines Lichts braucht wenige Bit und eine geringe Übertragungsrate, während die Übertragung einer Geschwindigkeit mit einer sehr hohen Genauigkeit und Updaterate eine deutlich höhere Datenübertragungsrate benötigt. Die Datenübertragungsrate wird meist in Bit/s oder einer Vielfachen davon angegeben (z.B. kBit/s, Mbit/s).

1.3.2 Störsicherheit

Ein Ideales Bussystem ist komplett störsicher. Dies ist in der Praxis aber unmöglich zu erreichen, weshalb verschiedene Maßnahmen getroffen werden müssen, um eine Sichere Datenübertragung für z.B. die Motorsteuerung oder das ABS zu erreichen. Zur Erkennung von Übertragungsfehlern werden in das Netzwerkprotokoll verschiedene Sicherheitsmechanismen eingebaut. Ein beliebtes und einfaches und Mittel ist das Paritätsbit. Mit dem Paritätsbit wird für den zu übertragenen Datenblock (in Bit) geprüft, ob die Summe der Bits gerade oder ungerade ist. Dies wird auf Senderseite berechnet und der Nachricht angehängt, damit die Empfängerseite dies auch berechnen kann und mit dem empfangenen Wert vergleichen kann. Dasselbe Verfahren wird bei der Checksummenprüfung verwendet, wobei hier nicht nach der Parität geprüft wird, sondern aus den einzelnen Datenbits eine Prüfsumme berechnet wird. Diese Prüfsummenberechnung kann sich nach Protokoll und Anwendung unterscheiden.

1.3.3 Echtzeitfähigkeit

Die Echtzeitfähigkeit garantiert, dass die Berechnung und Übertragung von Daten innerhalb eines festgelegen Zeitintervalls stattfinden. Das erforderte Zeitintervall hängt auch wieder vom Anwendungszweck ab, weshalb es verschiedene Anforderungen an das Echtzeitverhalten gibt. Bei einer weichen Echtzeitanforderung hält ein System die vorgegebene Zeitvorgabe in der Regel ein, wobei diese gelegentlich auch überschritten werden kann, ohne dass dies gravierende Auswirkungen für das System hat. Im Gegensatz dazu steht die harte Echtzeitanforderung, bei der die vorgegebene Zeitvorgabe strikt eingehalten werden muss, da bei der Überschreitung dieser, die Daten nicht mehr verwendbar sind. Dies kann bei sicherheitsrelevanten Systemen, wie z.B. ABS oder der Motorsteuerung, zu schweren Problemen führen. Wenn ein Steuergerät Daten von einem anderen Steuergerät benötigt, muss das Bussystem die benötigte Datenübertragungsrate und das vorgeschriebene Zeitintervall einhalten, damit das Bussystem der gestellten Echtzeitanforderung genügt.

1.3.4 Netzknotenanzahl und Kopplung von Netzwerken

Ein Netzknoten ist ein Teilnehmer an der Buskommunikation und die maximale Netzknotenanzahl gibt die maximale Anzahl der einzubindenden Netzknoten an, welche für verschiedene Anwendungszwecke unterschiedlich ist.
Ein Gateway ist ein Rechner, der die Möglichkeit verschafft zwischen unterschiedlichen Protokollen zu kommunizieren. Dieser liest die Daten aus einem Protokoll ein, puffert diese und sendet sie mit einem anderen Protokoll auf einem anderen Subnetz.
Oft kommen auch mehrere gleiche Bussysteme für unterschiedliche Anwendungszwecke zum Einsatz, die miteinander kommunizieren müssen. Dies kann über mehrere verteilte Gateways geschehen oder über einen Zentralen Gateway. Der Trend geht allerdings immer mehr in Richtung eines zentralen Gateway, da man damit Kosten, durch die weniger verbauten Steuergeräte, sparen kann.

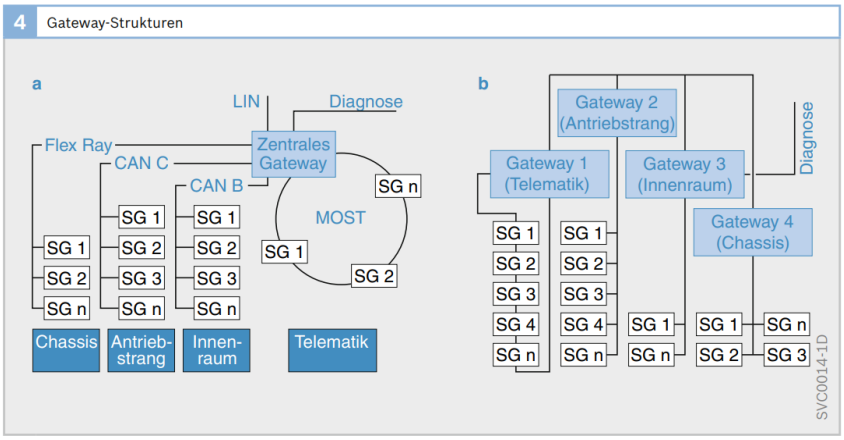


Abbildung 1.3: Grafik Gateway-Strukturen, Quelle: [**S.87**; Rei11]

1.4 Grundlagen

1.4.1 Topologien

Eine Topologie beschreibt den Aufbau eines Netzwerks. Hier werden nur die Bustopologie und die Sterntopologie beschrieben, da diese für das Bussystem in Automobilen hauptsächlich verwendet und kombiniert werden. Es gibt allerdings noch unzählige weitere Topologien.

1.4.1.1 Bustopologie

Die Bustopologie benötigt keine zentrale Steuereinheit, um eine Kommunikation durchzuführen. Es werden alle Netzwerkknoten an einen zentralen Bus angeschlossen, über den die Kommunikation stattfindet. Dies hat den Vorteil, dass weitere Netzknoten einfach und kostengünstig hinzugefügt werden können und ein Netzknoten ausfallen kann, ohne die Funktionstüchtigkeit der anderen Teilnehmer zu beeinträchtigen.

1.4.1.2 Sterntopologie

Bei der Sterntopologie gibt es ein zentrales Koppelelement, an dem mehrere Subnetze oder Steuergeräte angeschlossen sind. Dies schafft eine höhere Anpassungsfähigkeit beim Vernetzen, durch z.B. unterschiedlich eingesetzte Protokolle. Allerdings muss das Koppelelement alle Nachrichten bearbeiten und weiterleiten. Außerdem muss zum Erreichen einer gleichen Signallaufzeit die gleiche Leitungslänge verwendet werden.

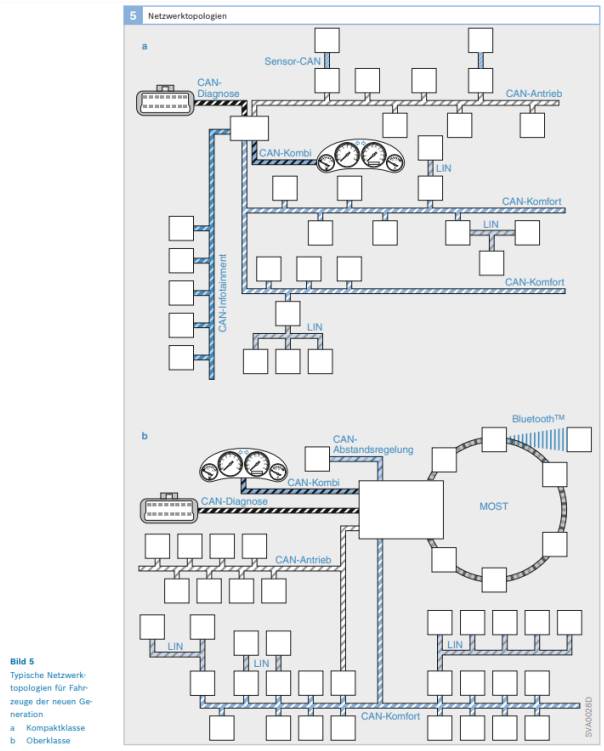


Bild 5
Typische Netzwerktopologien für Fahrzeuge der neuen Generation
a Kompaktklasse
b Oberklasse

Abbildung 1.4: Grafik Netzwerktopologien, Quelle: [**S.88**; Rei11]

1.4.2 Adressierung

Damit eine Nachricht und deren Informationen bei der Datenübertragung auf einem Bus identifiziert werden kann, erhält sie zusätzlich zum Datenpacket auch eine Information zur Datenübertragung. Diese wird genutzt, um sie zu identifizieren und dem richtigen Empfänger zukommen zu lassen. Es gibt mehrere Arten der Adressierung, die unterschiedliche Ansätze verfolgen. Bei der teilnehmerorientieren Adressierung, erhält die Nachricht als Identifikation eine Knotenadresse und alle anderen Knoten prüfen, ob die gesendete Nachricht die richtige Knotenadresse hat. Das nachrichtenorientierte Verfahren arbeitet mit der Adressierung von Nachrichten, wo die zu Knoten, die etwas empfangen, selbst entscheiden, ob sie die Nachricht verarbeiten wollen. Dadurch brauchen die Knoten keine Information über die Systemkonfiguration und können unabhängig voneinander arbeiten, zudem können Empfangsstationen ohne

Änderungen hinzugefügt werden, was eine sehr hohe Flexibilität bietet. Beim Übertragungsorientierten Verfahren, welches gerne mit den oben genannten Verfahren kombiniert wird, erfolgt die Adressierung mit Übertragungsmerkmalen wie z.B. einem Zeitfenster, in dem eine Nachricht gesendet wird.

1.4.3 Zugriffsarten auf einen Bus

Um eine Nachricht auf einem Bus zu senden, muss der Knoten auf den Bus zugreifen können. Die Regelung davon, geschieht entweder mit einem vorhersehbaren Verfahren (z.B. bestimmte Zeitpunkte für jeden Knoten) oder mit einem zufälligen Verfahren, bei dem jeder Knoten bei einem freien Bus probiert eine Nachricht zu senden. Desweitern kann man zwischen Time Division Multiple Access (in diesem Referat nicht behandelt), Master-Slave und Multimaster unterschieden werden.

1.4.3.1 Master-Slave

Beim Master-Slave Verfahren gibt ein Master den Zeitpunkt und die Häufigkeit der einzelnen Sendungen der Slaves an. Der Slave antwortet nur, wenn er von einem Master angesprochen wird. Manche Master-Slave Protokolle erlauben es allerdings, dass sich der Slave beim Master mit einer Sendungsbitte meldet.

1.4.3.2 Multimaster

Im Gegensatz zum Master-Slave Verfahren sind beim Multimaster Verfahren alle Knoten Master und dürfen alle selbstständig und unkontrolliert auf einen Bus zugreifen. Deshalb muss es eine Möglichkeit zur Kollisionserkennung geben, die dann z.B. eine Nachricht priorisiert und die andere Nachricht verzögert.

1.4.4 Steuermechanismen

Wann eine Nachricht gesendet wird, kann anhand von zwei Verfahren determiniert werden, die nach unterschiedlichen Prinzipien arbeiten.

1.4.4.1 Ereignissteuerung

Eine Nachricht wird bei der Ereignissteuerung dann übertragen, wenn dies durch ein Ereignis gefordert wird. Dies bietet die Vorteile, dass man eine große Reaktionsfähigkeit auf asynchrone Ereignisse hat, einfach neue Knoten nachrüsten kann. Außerdem wird das Netzwerk nicht mit unnötigen Übertragungen belastet. Allerdings ist diese Art der Ereignissteuerung nicht deterministisch, weshalb man nicht nachweisen kann, wann eine Nachricht gesendet wurde.

1.4.4.2 Zeitsteuerung

Bei der Zeitsteuerung ist dies allerdings ohne Probleme möglich, da die Nachrichten nach einem Netzwerkplan, ohne Kollision, hintereinander abgearbeitet werden. Damit ist die Zeitliche Aktualität einer Nachricht immer bestimmt und eine Nachricht kann nicht unterdrückt werden. Hier muss die Kapazität für eine Netzerweiterung allerdings angepasst werden und es gibt nur eine geringe Reaktionsfähigkeit auf asynchrone Ereignisse. Die Echtzeitanforderung kann trotz Zeitsteuerung bei einem schnellen System auch erreicht werden.

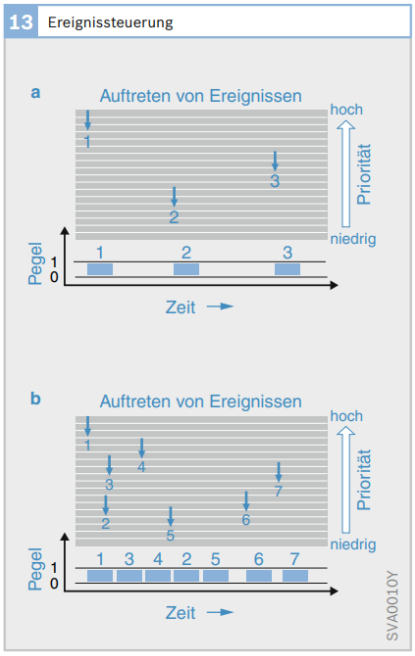


Abbildung 1.5: Grafik Ereignissteuerung, Quelle: [**S.78**; Rei11]

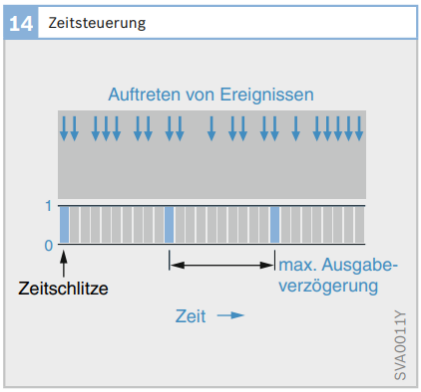


Abbildung 1.6: Grafik Zeitsteuerung, Quelle: [**S.80**; Rei11]

1.4.5 Quantisierung

Da in einem Kraftfahrzeug viele verschiedene Signale verarbeitet werden, müssen diese in das richtige Format gebraucht werden. Schalterzustände können zum Beispiel (z.B.) einfach durch ein Bit dargestellt werden, während z.B. Temperaturen oder Spannungen erst noch quantisiert werden müssen. Beim Quantisieren wandelt man die Werte von kontinuierlichen und unendlichen Werten in diskrete Werte um. Allerdings muss, um die Wertegleichheit der Systeme aufrecht zu erhalten, in allen Systemen gleich zwischen binär und physikalischer Form umgerechnet werden.

2 CAN-Bus

Der Begriff CAN steht für Controller Area Network. Mit der zunehmenden Elektronisierung im Automobilbereich kam die Notwendigkeit auf, verschiedene Steuergeräte zu vernetzen. Mit dem CAN-Bus wurde von der Robert Bosch GmbH ein Bussystem entwickelt, dass erstmals 1990 in Serie im Mercedes-Benz W140 eingesetzt wurde. Durch seine vielen Vorteile hat sich diese Art des Busses schnell durchgesetzt und wird seitdem flächendeckend in der Automobilbranche verwendet. Auch heute ist CAN noch der Standard für die Vernetzung von Steuergeräten im Kraftfahrzeugbereich. Deshalb wurde es auch von der International Organization for Standardization (ISO) und der SAE (Firma für Standardisierung im Automobilbereich) für den Datenaustausch in Kraftfahrzeugen standardisiert. Es gibt mehrere Versionen des CAN-Bus. Dazu zählt CAN2.0, Single Wire CAN, Low Power CAN, CAN FD und CAN XL. Mit TTCAN wurde auch ein zeitbasiertes Protokoll eingeführt. Can 2.0 kann außerdem in Lowspeed CAN (CAN-B) und in Highspeed CAN (CAN-C) unterschieden werden. CAN-B wird hauptsächlich im Komfort- und Karosseriebereich eingesetzt, da dort die Geschwindigkeit von 5kBit/s bis 125kBit/s meist ausreicht. Für Geschwindigkeiten von 125kBit/s bis 1Mbit/s wird ein Highspeed-CAN-Bus verwendet. Dies wird größtenteils für Echtzeitanforderungen beim Antriebsstrang eingesetzt.

2.1 Übertragungssysteme

2.1.1 Logische Buszustände und Kodierung

Es gibt zwei Zustände, mit denen die Informationsbits übertragen werden. Der dominante Zustand stellt eine binäre 0 dar, während der rezessive Zustand eine binäre 1 darstellt. Der dominante Zustand ist in der Lage, den rezessiven Zustand zu überschreiben. Die Kodierung beim CAN-Bus geschieht nach dem NRZ-Verfahren (Non-Return-To-Zero), welches beschreibt, dass zwischen zwei gleichwertigen Übertragungszuständen nicht zwangsweise ein Nullzustand herrschen muss. Zum Senden und Empfangen, werden die logischen Zustände in Binärwerte gerechnet und den CAN-Controller gesendet.

2.1.2 Leitungsarten

2.1.2.1 Zweidrahtleitung

Prinzipiell kann für den CAN-Bus jedes beliebige Übertragungsmedium verwendet werden, auf dem es möglich ist, dominante und rezessive Zustände zu übertragen. Meist wird eine verdrillte Zweidrahtleitung verwendet, die galvanisch gekoppelt ist. Diese ist kostengünstig und nicht so schwer wie eine stark isolierte Leitung. Dadurch, dass die Magnetfelder der zwei Leitungen gegenläufig sind, gleichen sich die Störimpulse der Leitungen aus. Es wird auf die zwei Leitungen, CAN_L (CAN-Low Leitung) und CAN_H (CAN-High Leitung) eine symmetrische und differenzierte Datenübertragung durchgeführt. Dabei werden die Bits unter der Verwendung von unterschiedlichen Spannungen auf die Leitungen übertragen. Dadurch reduziert sich die Empfindlichkeit gegenüber Gleichtaktstörungen (Störungen auf beiden Leitungen), da sich diese rechnerisch ausfiltern lassen. Dies ist allerdings nur der Fall, wenn sich die Störung auf beide Leitungen gleich auswirkt. Ist dies nicht der Fall, hat das System ein Differenzproblem. Außerdem kann man das eigene Abstrahlverhalten bei hohen Baudraten durch zusätzliche Schirmung der Leitung erreichen.

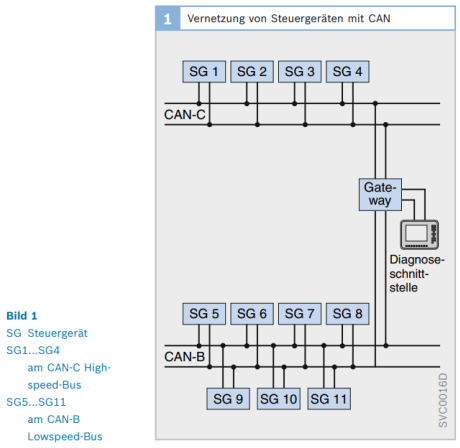


Abbildung 2.1: Grafik Vernetzung von Steuergeräten, Quelle: [**S.92**; Rei11]

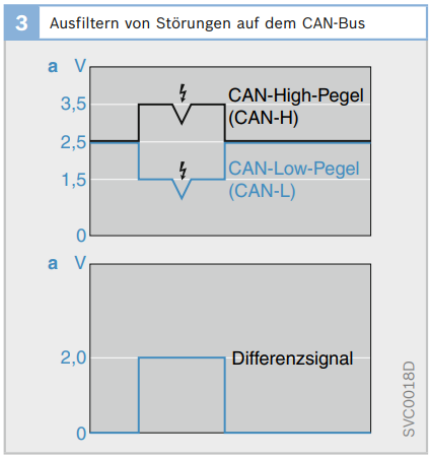


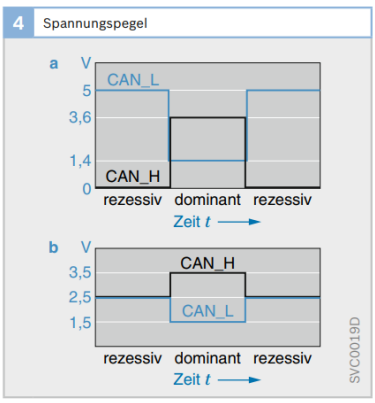
Abbildung 2.2: Grafik Vernetzung von CAN Steuergeräten, Quelle: [**S.94**; Rei11]

2.1.2.2 Eindrahtleitung

Zum Einsparen einer zweiten Leitung ist es möglich die Kommunikation über eine Leitung stattfinden zu lassen. Dadurch müssen allerdings alle Busteilnehmer über eine gemeinsame Masse verfügen, weshalb dies nur bei räumlich begrenzter Ausdehnung möglich ist. Zudem ist das das Ausfiltern von Störimpulsen nicht möglich und die Eindrahtleitung anfälliger gegenüber Störeinstrahlung. Dadurch muss ein höherer Pegelhub gewählt werden, was im Gegenzug die eigene Störabstrahlung erhöht, weshalb die Flankensteilheit verringert werden muss. Dadurch wird die Übertragungsgeschwindigkeit begrenzt, weshalb eine Eindrahtleitung nur bei Lowspeed CAN im Bereich der Karosserie und Komfortelektronik eingesetzt wird. Die Möglichkeit des Betriebs über eine Leitung bedeutet allerdings auch, dass ein Zweidrahtsystem, wenn auch nur eingeschränkt, auch bei Ausfall einer der Leitungen funktionsfähig bleibt.

2.1.3 Spannungspegel

Die Spannungspegel auf den Leitungen werden vom CAN-Transceiver anhand der ihm übergebenen Bitströme gesetzt. Highspeed-CAN und Lowspeed-CAN verwenden jeweils unterschiedliche Spannungspegel. Die einzelnen Spannungen können anhand von Abbildung 2.3 abgelesen werden.



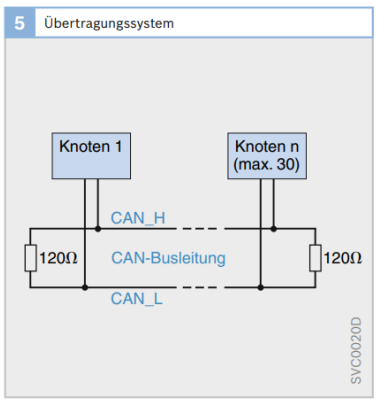


Abbildung 2.3: Grafik Spannungspegel CAN, Quelle: [**S.94**; Rei11]

2.1.3.1 Reflexionsfreier Abschluss

Damit die Reflexion am Ende einer Leitung gedämpft wird, werden die Busleitungen an beiden Enden mit einem Widerstand von 120Ω abgesichert. Oft werden auch zwei 60Ω Widerstände verwendet um die Reflexion kontrolliert, mit einem Kondensator mit einer Kapazität von wenigen nF, auf Masse abzuleiten.

2.1.4 Grenzwerte

Damit die Auswertung der gesendeten Bits erfolgreich ist, muss das Signal am Abtastzeitpunkt bei jedem Knoten innerhalb einer Bitzeit ungestört vorhanden sein. Verzögerungen ergeben sich aus Signallaufzeiten, weshalb die zulässige Übertragungsrate eines Busses von der Gesamtlänge des Bussystems abhängig ist. Eine Anzahl von bis zu 30 Netzknoten ist ohne Probleme realisierbar.

Länge	Geschwindigkeit
Genau 40m	1Mbit/s
40-100m	500kBit/s
100-250m	250kBit/s
250-500m	125kBit/s
500-1000m	40kBit/s

Abbildung 2.4: Tabelle Geschwindigkeiten Highspeed-CAN, anhand Quelle: [**S.95**; Rei11]

2.2 CAN-Protokoll

2.2.1 Protokollschichten

Zusammenhängende Aufgaben werden beim CAN-Protokoll in thematischen Schichten beschrieben. Diese können in das OSI-Modell eingeordnet werden, worauf in diesem Vortrag allerdings nicht weiter eingegangen wird. Die oberste Schicht ist die Application-Layer (Anwendungsschicht), in der sich Informationen in Datenstrukturen befinden, die von der Anwendung verwendet werden. Der Object-Layer (Objektschicht) ist für die Verwaltung von Nachrichten zuständig. Dieser entscheidet, welche Nachricht zu welchem Zeitpunkt versendet werden soll, und führt die Akzeptanzprüfung durch. Darunter folgt der Transport-Layer (Transportschicht). Dieser formt die Nachrichten, die er vom Object-Layer erhält, um, sodass diese zum Senden bereit sind und zum Physical-Layer übertragen werden können. Beim Empfangen präsentiert der Transport-Layer die Nachricht dem Object-Layer. Zudem ist der Transport-Layer für die Arbitrierung und Fehlererkennung zuständig. Die Physical-Layer (Physikalische Schicht) beschreibt die physikalische Komponente des Netzwerks.

2.2.2 Adressierung

Der CAN-Bus arbeitet mit der inhaltsbezogenen Adressierung, bei der eine einzelne Nachricht einen Identifier bekommt. Dieser Identifier besteht aus 11 Bit bei einem Standard-Format-Identifier und 29 Bit beim Extended-Format-Identifier. Mit dem 29 Bit Identifier ist es möglich, über 536 Millionen (2^{11}) unterschiedliche CAN-Botschaften in einem System zu übertragen, während mit dem 11 Bit Identifier nur circa 2000 (2^9) Nachrichten übertragen werden können. Im weiteren Verlauf betrachten wir allerdings nur den 11 Bit Identifier.

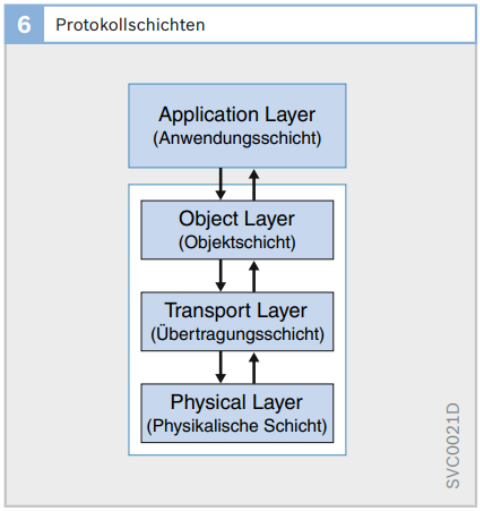


Abbildung 2.5: Grafik Protokollschichten, Quelle: [**S.96**; Rei11]

2.2.3 Steuerung des Zugriffs

2.2.3.1 Priorisierung

Um eine reibungslose Kommunikation mit dem Multimaster Prinzip zu ermöglichen, müssen manche Botschaften vor anderen priorisiert werden. Dies wird über den Identifier gelöst, wobei ein Identifier mit niedriger Binärzahl für eine hohe Priorität und ein Identifier mit hoher Binärzahl für eine niedrige Priorität steht. Die Vergabe der Priorität wird anhand von der Änderungsgeschwindigkeit oder der Sicherheitsbedeutung einer Nachricht bestimmt.

2.2.3.2 Arbitrierungsphase

Befindet sich der Bus im rezessiven Zustand (der Bus ist frei), kann jede Station mit dem Senden beginnen. Die übermittelte Nachricht startet mit einem dominanten Bit, in dessen Anschluss der Identifier übertragen wird. Sobald mehrere Stationen gleichzeitig anfangen auf den Bus zu senden, setzt sich die Nachricht mit der höchsten Priorität durch, wodurch es zu keinem Zeit- und Datenverlust kommt. Jede Station, die ein rezessives Bit sendet, wird durch ein dominantes Bit, dass von einer anderen Station gesendet wird, überschrieben. Die Sendestationen prüfen mit einem Logischen UND, ob der Pegelstatus, mit dem von ihnen gesetzten übereinstimmt. Ist dies nicht der Fall, bricht die Botschaft ihre Nachricht ab, weshalb sich die Botschaft mit der höchsten Priorität durchsetzt. Alle anderen Sender warten nun, bis der Bus wieder frei ist und beginnen erneut mit dem Senden. Um Konflikte des Überschreibens vorzubeugen, dürfen zwei Nachrichten niemals denselben Identifier besitzen. Bei CAN2.0A muss die Nachricht mit der höchsten Priorität 130 Bitzeiten warten, was bei einer Datenübertragungsrate von $500kBit/s$ etwa eine Verzögerung von $260\mu s$ ergibt. Wird die Busauslastung allerdings höher, erhöht sich der zeitliche Versatz von Nachrichten mit niedriger Priorität stark. Man kann nicht sicher sein, ob die Nachricht überhaupt ankommt, weshalb das Bussystem auf die einzelnen Gegebenheiten abgestimmt sein muss.

2.2.4 Botschaftsformat

2.2.4.1 Arten der Telegrammformate

Zum einen gibt es das Data-Frame (Datentelegramm), in dem Nachrichten, die von einer sendenden Station bereitgestellt werden, gesendet werden. Ein störungsfreies Netzwerk kommt ausschließlich mit dieser Telegrammart aus. Mit dem Remote-Frame (Datenanforderungstelegramm) kann eine Station

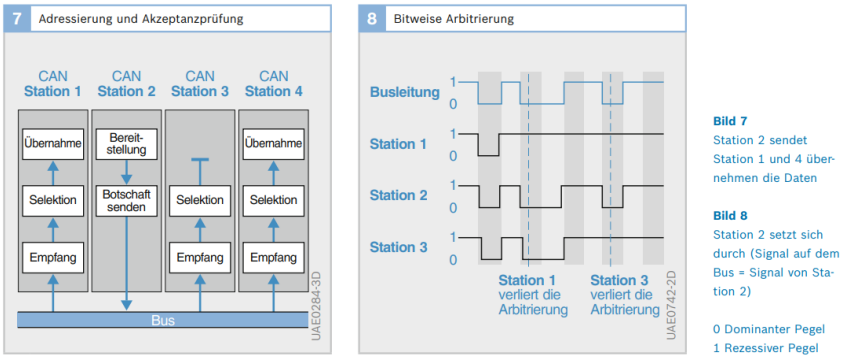


Abbildung 2.6: Grafik Adressierung, Akzeptanzprüfung, Bitweise Arbitrierung, Quelle: [**S.97**; Rei11]

benötigte Daten von einer Datenquelle anfordern. Die Datenquelle sendet dann mit entsprechenden Daten. Bei CAN wird dies nur recht selten eingesetzt, da die Datenquelle selbst in bestimmten Zeitabständen die Daten unaufgefordert sendet. Ein Error-Frame (Fehlertelegramm) wird gesendet, wenn eine Station einen Fehler erkennt, um den anderen Stationen diesen Fehler mitzuteilen. Zuletzt gibt es noch das Overload-Frame (Überlastungstelegramm), bei dem eine Station den anderen mitteilt, dass er zum jetzigen Zeitpunkt keine weiteren Frames verarbeiten kann, da er überlastet ist. Es gibt mit CAN2.0A und CAN2.0B zwei Möglichkeiten einen Frame aufzubauen, wobei im weiteren Verlauf nur CAN2.0A betrachtet wird.

2.2.4.2 Frames

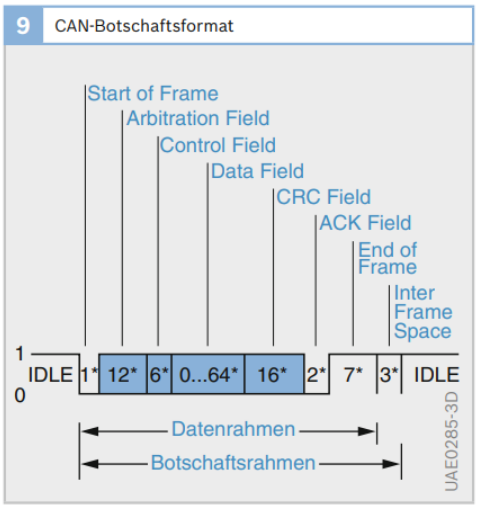


Abbildung 2.7: Grafik CAN-Botschaftsformat, Quelle: [**S.98**; Rei11]

Start-Of-Frame Ein Frame wird mit dem Start-Of-Frame eingeleitet, indem ein dominantes Bit als Start der Übertragung dient. Hierdurch werden alle Knoten synchronisiert und CAN braucht keine Uhr, um die Knoten zu synchronisieren.

Arbitration Field Danach beginnt das Arbitration Field. Bei CAN2.0A wird der 11Bit Identifier mit einem Kontrollbit und einem RTR-Bit (Remote Transmission Request Bit) gesendet. Das RTR-Bit sagt aus, ob das Frame ein Data- oder Remote-Frame ist. Ein Data-Frame wird durch ein dominantes RTR-Bit und ein Remote-Frame durch ein rezessives Bit dargestellt. Wenn eine Station sendet, während die andere

Station eine Nachricht anfragt, also beide denselben Nachrichtenidentifier senden, wird der Konflikt über das RTR-Bit gelöst und die Sendestation sendet die Daten, während die Requeststation diese empfängt.

Control Field (Kontrollfeld) Das Kontrollfeld besteht aus einem IDE-Bit (Identifier Extension Bit), welches bei CAN2.0A dominant und bei CAN2.0B rezessiv gesendet wird. Dieses Bit ist gefolgt von einem rezessiven Bit, dass für die Zukunft reserviert ist. Außerdem werden im Control Field noch vier Bits gesendet, die die Anzahl der Datenbytes im Datenfeld beschreibt, um eine Datenprüfung durchführen zu können.

Data Field (Datenfeld) Das Datenfeld enthält Daten zwischen 0 und 8 Bytes. Es ist auch möglich, mehrere Botschaften in einer Nachricht und in einem Datenfeld zu übertragen. Die kürzeste mögliche Nachricht besitzt 0 Bytes und trägt somit keine Daten, und wird deshalb gerne zum Synchronisieren verteilter Prozesse verwendet. Der gesendete Frame hätte in diesem Fall eine Größe von nur 44 Bit.

CRC Field (Sicherungsfeld) Im Cyclic-Redundancy-Check-Field (zyklisches Redundanzprüfungsfeld) wird eine, vom Data-Field berechnete, 15 Bit Prüfsumme gesendet und mit einem 16. rezessiven Bit abgeschlossen. Das CRC-Field dient zur Erkennung von auftretenden Übertragungsstörungen.

ACK Field (Quittierungsfeld) Mit dem ACK-Field (Acknowledgement- oder Bestätigungsfeld) wird von einem Knoten, direkt im Anschluss an das Data Field, der erfolgreiche Empfang einer Nachricht bestätigt. Dieser Knoten muss die Daten nicht empfangen, sondern teilt dem Sender lediglich mit, dass die Nachricht korrekt gesendet wurde. Das ACK-Field wird vom Sender rezessiv gesendet und bei erfolgreichem Empfang eines Empfängers dominant überschrieben.

End of Frame (End-Kennung) Die End-Kennung kennzeichnet das Ende einer Botschaft mit sieben rezessiven Bits.

Inter Frame Space (Rahmsubsubsubsection) Die Rahmenpause besteht aus drei rezessiven Bits und signalisiert einen wieder freien Bus. Dies wird allerdings von Error- und Overload-Frames nicht beachtet, sodass diese direkt senden können. Dadurch wird eine direkte Signalisierung von Fehlern und Problemen ermöglicht.

2.2.5 Störungserkennung

Es kann nie von einer störungsfreien Datenübertragung ausgegangen werden. Allerdings ist eine sichere und zuverlässige Datenübertragung für sicherheitsrelevante Bauteile im Automobil vonnöten, weshalb es die Möglichkeit geben muss, die Datenübermittlung auf Korrektheit zu überprüfen. Dafür bietet CAN mehrere Prüfmechanismen an. Bei der zyklische Redundanzprüfung berechnet ein Algorithmus eine Prüfsumme, die mit der Nachricht übermittelt wird. Der Empfänger rechnet diese ebenfalls aus und vergleicht diese mit der gesendeten. So kann entschieden werden, ob die Nachricht erfolgreich übermittelt wurde. Mit dem Frame-Check (Rahmenformat-Überprüfung) prüfen alle Busteilnehmer, ob die gesendeten oder empfangenen Datenrahmen die vorgegebenen Datenrahmengrößen einhalten. Mit dem ACK-Check bestätigt eine andere Station den korrekten Empfang einer Nachricht mit einem ACK-Bit. Beim Monitoring überwacht der Sender fortlaufend den Buspegel und ist somit in der Lage, Übertragungsfehler eigenständig zu identifizieren. Die Bitstuffing Prüfung wird verwendet, um zu schauen, ob zwischen Start-of-Frame und dem Ende des CRC-Field maximal fünf aufeinanderfolgende Bits den identischen Zustand besitzen. Der Sender fügt nach fünf identischen Bits ein Bit in entgegengesetztem Zustand ein, um mit der Bitstuffing Prüfung konform zu sein. Diese Zusatzbits werden, beim sogenannten Destuffing, vom Empfänger beim Empfang wieder gelöscht. Damit ist erkennbar, ob eine Leitungsstörung vorliegt, und das Synchronisieren zwischen Knoten lässt sich durch einen garantierten wechseln der Zustände einfacher lösen.

2.2.5.1 Error Frames

Wird ein Fehler von einem Knoten erkannt, sendet dieser einen Error-Frame, der aus 6 Dominanten Bits besteht. Dadurch wird erreicht, dass Bitstuffing erkannt wird und alle anderen Knoten auch einen Error Frame mit 6 dominanten Bits senden. Ein Error-Frame wird mit 8 rezessiven Bits abgeschlossen.

2.2.5.2 Fehlererkennung bei Ausfällen

Defekte Stationen können den Busverkehr stark belasten und stören, wenn diese häufig fehlerhafte Botschaften senden oder korrekte Botschaften durch das Senden eines Error-Frames unterbrechen. Deshalb ist es mithilfe einer statistischen Fehlerauswertung möglich, den Fehler zu lokalisieren. Die Station erkennt die Wahrscheinlichkeit ihrer eigenen Fehlfunktion daran, wie häufig sie Botschaften abbricht, bevor andere Stationen ein Error-Frame senden. Beim Erkennen der eigenen Fehlfunktion bricht die Station ihre Kommunikation selbstständig ab.

2.3 Hardware

2.3.1 CAN-Controller

Der CAN-Controller erzeugt aus zu übertragenden Daten einen Frame mit allen für das CAN-Protokoll erforderlichen Felder.

2.3.1.1 Basic-CAN

Mit einem Basic-CAN Controller ist es nur möglich die Grundfunktionen eines CAN-Protokolls zur Erzeugung des Bitstroms zu realisieren. Dadurch ist dieser kostengünstiger und braucht eine kleinere Chipfläche. Allerdings steht nur ein Zwischenpuffer zur Verfügung, auf den ein Rechner (z.B. ein Mikroprozessor) zugreift und der Rechner muss die Daten aus dem Puffer gelesen haben, bevor der CAN-Controller neue Daten empfangen kann. Auch die Akzeptanzprüfung muss vom Rechner durchgeführt werden, weshalb dieser die Rechenkapazitäten für die CAN-Verwaltung zur Verfügung stellen muss. Deshalb wird Basic-CAN meist nur für niedrige Übertragungsraten oder für hohe Übertragungsraten mit wenigen Nachrichten verwendet.

2.3.1.2 Full-CAN

Ein Full-CAN Controller bearbeitet alle Akzeptanztests und regelt die Sende- und Empfangssteuerung. Dieser bekommt die zu akzeptierenden Nachrichten beim Initialisieren mitgeteilt und leitet nur noch diese bestimmten Nachrichten an den angeschlossenen Rechner weiter. Diese Art von CAN-Controllern ist meist schon auf Chips integriert, da dies günstigere Herstellungskosten mit sich bringt.

2.3.1.3 Bauteile ohne lokalen Rechner

Es gibt auch CAN-Bauteile ohne lokalen Rechner, die Daten nur über Ports ein und auslesen können. Dies ist geeignet für Sensoren und Aktoren, da diese Art der Anbindung an CAN günstiger ist. Allerdings wird hierfür ein Master-Knoten benötigt, um sie zu steuern.

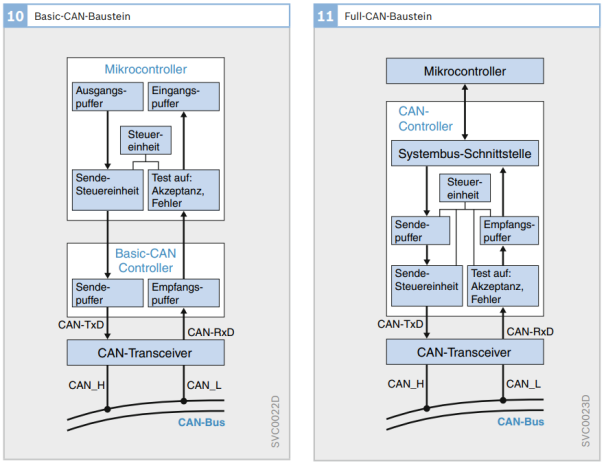


Abbildung 2.8: Grafik Basic-CAN und Full-CAN Baustein, Quelle: [**S.102**; Rei11]

2.3.2 Transceiver

Ein Transceiver regelt aus dem vom CAN-Controller erzeugten Bitstrom die richtigen Spannungspegel, die für den CAN-Bus notwendig sind. Bei einem Zweidrahtleitungssystem sind dies CAN_H, CAN_L und die Referenzspannung U_{ref}.

2.3.3 Sleep Mode

Der CAN-Bus muss auch bei ausgeschalteter Zündung betriebsbereit sein. Um das Bordnetz möglichst wenig zu belasten, können CAN-Bauteile in einen Sleep-Mode verfallen. Im Sleep-Mode wird der Sendeteil eines Transceivers ausgeschaltet, um den Stromverbrauch zu senken, während der Empfängerteil aktiv bleibt, um den Bus zu beobachten, ob eine Nachricht, wie z.B. eine Wake-Up Botschaft übertragen wird.

2.4 Erweiterungen von CAN und Protokolle auf CAN-Basis

Dass der CAN-Bus ein sehr beliebtes Übertragungsmedium ist, sieht man an den vielzähligen Abwandlungen vom originalen CAN und dem großen Einsatzbereich außerhalb der Automobilbranche.

2.4.1 CAN FD

Da eine Bandbreitenerweiterung von CAN2.0 erwünscht wurde, entwickelte die Robert Bosch GmbH 2012 den CAN FD (CAN Flexible Data Rate) Standard. Ein CAN-FD System kann sowohl CAN-FD Nachrichten als auch normale CAN-Nachrichten korrekt interpretieren. Allerdings kann ein normales CAN2.0 System keine CAN FD Nachrichten interpretieren. Damit nun CAN-FD Steuergeräte mit CAN-FD Steuergeräten mit bis zu 1Mbit/s Daten austauschen können, werden die CAN2.0 Steuergeräte temporär vom Bus genommen und die CAN FD Steuergeräte umgeschaltet, sodass sie bis zu 64 Datenbytes verarbeiten können. Dies wird durch ein zusätzliches Umschaltbit umgesetzt.

2.4.2 CAN XL

Als Forderungen nach einer erneuten Bandbreitenerhöhung beim CAN-Bus laut wurden, wurde CAN XL entwickelt, um ein Datenfeld von 2048 Bytes und eine Datenrate von bis zu 10MBit übertragen zu können. Dies ermöglicht Ethernet Pakete, die immer mehr Einzug in das moderne Fahrzeug finden, zu tunneln. CAN XL ist auch kompatibel mit CAN FD und CAN2.0, allerdings schaltet CAN XL, bei der Datenübertragung, auf eine andere physikalische Ebene mit unterschiedlichen Datenraten, Spannungen und optional auch einer unterschiedlichen Pulsweitenmodulation um.

2.4.3 Andere CAN-Protokolle

2.4.3.1 CAN TP2.0

Mit dem CAN-Transportprotokoll 2.0 wurde von VW ein Transportprotokoll eingeführt, dass speziell zu Diagnosezwecken entwickelt wurde. Es wird möglich mehr als 8 Daten Bytes in mehreren CAN-Nachrichten zu übertragen, während derselbe Identifier verwendet wird. Dadurch müssen mehrere zusammengehörige Datenpakete nicht mehr einzeln aufgefordert werden, was eine effizientere Diagnosekommunikation erlaubt.

2.4.3.2 Bosch MCNet

Mit dem Bosch MCNet (Mobile Communication Network) wurde ein Transportprotokoll auf physikalischer Ebene und Sicherungsschicht entwickelt, welches speziell für Multimedia-Anwendungen angepasst war. MCNet kann nur Befehle und Informationen transportieren, allerdings keine Video- und Audio-Dateien. Dadurch setzte es sich nie durch, zeigt aber trotzdem das Potential und die Weiterentwicklungsfähigkeit, die CAN aufweist.

2.4.3.3 CANopen

In der Automobilindustrie gibt es, anders als in der Automatisierungstechnik, fast keine Standardisierung für die Anwendungsschicht bei CAN. CANopen standardisiert die Anwendungsschicht und wird hauptsächlich für standardisierte Anwendungsprofile in Fahrzeugen genutzt. Beispiele hierfür wären Kommunalfahrzeuge mit Anbaugeräten oder der Umbau von Serienfahrzeugen zu Sonderfahrzeugen.

Für mehr Informationen zu weiteren CAN-Protokollen, CAN FD und CAN XL Quelle [Rau22].

3 LIN-Bus

Der Local Interconnect Network (LIN)-Bus wurde entwickelt, um eine kostengünstigere Alternative zum Low-Speed-CAN zu schaffen. Es wurde von einem Konsortium mehrerer Automobilhersteller entwickelt und 2001 erstmals im Mercedes Benz SL eingesetzt. LIN dient der Vernetzung einzelner Teilbereiche eines Fahrzeugs als Subnetz zu CAN und hat deswegen eine sehr geringe Datenrate und nur wenige Busteilnehmer (maximal 16). Es besteht eine lineare Busstruktur und die einzelnen Knoten sind durch eine Eindrahtleitung verbunden. Eine vorgeschriebene Topologie existiert nicht, wobei es meist als lineare Busstruktur umgesetzt wird.

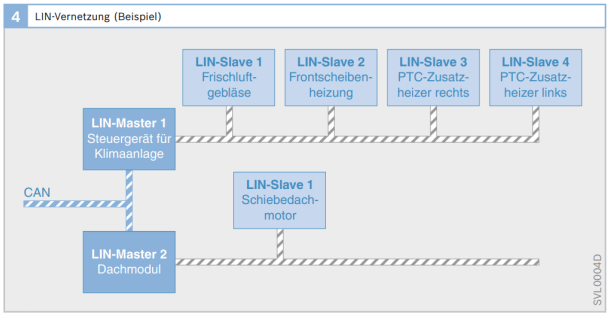


Abbildung 3.1: Grafik Vernetzung bei LIN, Quelle: [**S.111**; Rei11]

3.1 Verwendungszweck

LIN wird hauptsächlich verwendet, um Sensoren und Aktoren in der Karosserieelektronik zu vernetzen, die nicht die Bandbreite von CAN benötigen. Dafür werden oft einzelne Elemente eines Autos mit einem eigenen Bus vernetzt. Ein Beispiel dafür wäre ein Türmodul, welches mehrere Sensoren und Aktoren, wie die Verrieglung, Außenspiegelverstellung und Tasten für die Sitzverstellung, besitzt.

3.2 Übertragungssystem

3.2.1 Aufbau und Pegel

Als Übertragungsmedium wird bei LIN eine unabgeschirmte Eindrahtleitung verwendet, die zwei Pegel annehmen kann. Der dominante Pegel (logisch 0) hat meist 0V, während der rezessive Pegel (logisch 1) üblicherweise der Batteriespannung entspricht. Der rezessive Pegel ist am Master mit $1k\Omega$ und am Slave mit $30k\Omega$ abgesichert. Durch die Vielzahl an unterschiedlichen Anwendungszwecken und so auch Vernetzungen, akzeptiert LIN einen sehr breiten Toleranzbereich beim Empfangen der Daten, damit auch mit Störsignalen noch gültige Signale empfangen werden können.

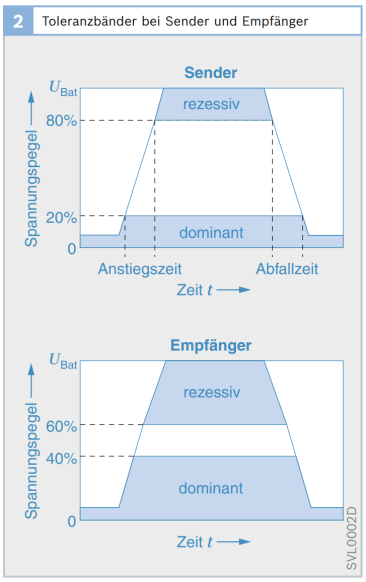


Abbildung 3.2: Grafik Toleranzbänder, Quelle: [**S.107**; Rei11]

3.2.2 Buszugriff

Der LIN-Bus verwendet das Master-Slave Verfahren, wobei Nachrichten entweder zwischen einem (Point-to-Point), mehreren (Multicast) oder allen (Broadcast) Slaves ausgetauscht werden. Es gibt für den Master mehrere Möglichkeiten eine Kommunikation zwischen Knoten zu initiieren. Die Erste ist, dass der Master der Slave nach Daten fragt (z.B. Messwerte oder Schalterzustände), eine Weitere wäre, dass der Master dem Slave eine Anweisung sendet (z.B. Verschließen der Tür). Außerdem kann der Master eine Kommunikation zwischen zwei Slaves initiieren.

3.2.3 Datenrate

Die maximale Datenrate liegt bei 20kBit/s, da dies ein guter Kompromiss aus der Forderung nach einer hohen Flankensteilheit (Dauer, die der Spannungspegel zum Ansteigen braucht) zur Synchronisierung der Slaves und einer geringen Flankensteilheit zur Verbesserung des EMV-Verhaltens ist. Als empfohlene Standardübertragungsrate wird meist 2,4kBit/s, 9,6kBit/s und 19,2kBit/s gewählt, sowie eine minimale Übertragungsrate von 1kBit/s um Timeouts zu vermeiden. Es ist eine Flankensteilheit mit 1 *bis* $3\frac{V}{\mu s}$ angegeben.

3.3 LIN-Protokoll

3.3.1 Frame

Beim LIN-Protokoll werden die Daten in einen Frame eingebettet. Das zu sendende Paket startet mit dem Header (Botschaftskopf), in dem die Synchronisierung und die Nachrichtenidentifikation stattfindet. Dieses Paket wird von einer Response (Nachrichtenfeld) beantwortet.

3.3.2 Synchronisierung

Zu Beginn einer Nachrichtenübertragung werden alle Clients synchronisiert. Dies wird benötigt, damit man eine große Spezifikation des Timings zu erreichen und somit kostengünstige Hardware ohne Quar-

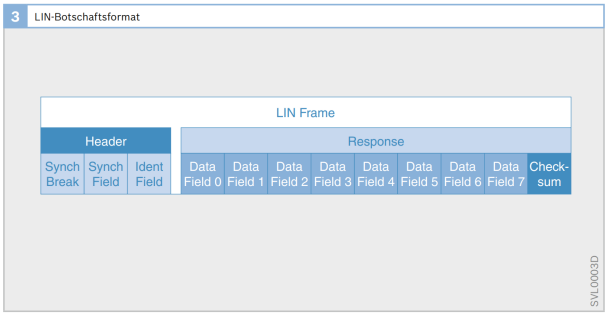


Abbildung 3.3: Grafik Frame, Quelle: [**S.108**; Rei11]

zoszillatoren einsetzen kann. Der Master darf nicht mehr als ±0,5% vom Takt abweichen, während der Slave am Anfang der Synchronisation bis zu 15% abweichen darf, wenn er am Ende der Synchronisation nicht mehr als 2% vom Master abweicht. Für die Synchronisierung sendet der Master erst die Syncbreak (Synchronisierungspause), in dem 13 aufeinanderfolgende dominante Bits gesendet werden und ein rezessives Bit gesendet wird. Danach wird mit dem Syncfield (Synchronisierungsfeld) die Bitfolge 01010101 vom Master gesendet, damit die Slaves ihre Zeitbasis mit dem des Masters abgleichen und anpassen können. Somit sind alle Slaves mit dem Master synchronisiert.

3.3.3 Identifier

Das dritte Byte des Headers ist der Identifier der Nachricht. Diese ist parallel zum CAN-Bus auch eine inhaltsbasierte Adressierung und gibt Aufschluss über den Inhalt der Botschaft. Für die Akzeptanzfilterung prüfen alle am Bus angeschlossenen Knoten, ob sie die Botschaft empfangen, verarbeiten oder ignorieren wollen. Der Identifier besteht aus 6 Bit, wodurch 64 mögliche Identifier möglich sind. Allerdings sind 4 davon für systemrelevante Zwecke reserviert. Abgeschlossen wird der Identifier mit 2 Bit für die Prüfsumme des Identifier, damit es nicht zu einem Übertragungsfehler oder einer fehlerhaften Botschaftszuordnung kommt.

3.3.4 Response/Datenfeld

Im Datenfeld findet die eigentliche Übertragung der Daten statt. Der Slave erkennt aus dem Identifier, ob er angesprochen wird und sendet daraufhin die Antwort im Datenfeld zurück. Die Datenübertragung startet mit einem Least Significant Bit (LSB). Jedes einzelne Byte, dass übertragen werden soll, wird mit einem Startbit eingeleitet und mit einem Stoppbit abgeschlossen. Dies dient zur Neusynchronisation der Knoten, um Datenübertragungsfehler zu vermeiden. Auch bei LIN wird die Datenübertragung durch das Errechnen, Senden und Vergleichen von Prüfsummen abgesichert.

3.3.5 LDF-File

Das LIN Description File dient dazu, Spezifikationen von Netzteilnehmern, Frames und Identifierer in einer Konfiguration festzuhalten. Es wird dadurch das komplette LIN-Netzwerk konfiguriert, weshalb es auch Aufschluss über die gemeinsamen Schnittstellen zwischen den Komponenten in einem Automobil gibt. Zum Beispiel die Schnittstelle zwischen dem Auto und einem Automodul (z.B. eine Tür), dass von einem Zulieferer geliefert wird. Aus diesem LDF-File können außerdem automatisch Daten in C generiert werden, die zur Implementierung von LIN in den Steuergeräten verwendet werden.

3.3.6 Message Scheduling

Die Nachrichten werden bei LIN gemäß der im LDF-File vorhandenen Scheduling-Tabelle, festgelegten Reihenfolge und des Zeitrahmens übermittelt. Häufig benötigte Informationen können öfters in der Liste eingetragen werden und werden somit mehrfach abgearbeitet. Der Master arbeitet diese Liste von Anfang bis Ende ab und fängt danach wieder von vorne an. Damit ist das Übertragungsraster für jede Nachricht bekannt, dass gewährleistet, dass jede Übertragung vom Master initiiert wird. Die Scheduling-Tabelle kann sich allerdings je nach Betriebszustand des Autos (z.B. Zündung an oder aus) verändern. Außerdem können beim weiterentwickelten LIN2.0 Bus Daten auch nach Bedarf vom Master mit einem speziellen Identifier (Sporadic Frames) angefordert werden und Slaves können mit einem speziellen Identifier (Event Triggered Frames) Ereignisse melden.

3.3.7 Sleep Mode

Auch beim LIN-Bus existiert einen Sleep-Modus, um den Stromverbrauch zu senken. Dieser kann entweder durch einen speziellen Identifier (60) als Kommando vom Master initiiert werden oder die Slaves gehen selbstständig in den Sleep Mode nach einer Übertragungslücke von 4 Sekunden. Zum Aufwecken wird vom Master ein Wake-Up-Signal in Form von 128 Datenbytes geschickt und alle Slaves müssen nach 4-64 Bitzeiten initialisiert sein und auf den Master reagieren können.

4 Fazit

Zusammenfassend kann man sagen, dass beide Bussysteme auf dem Markt eine Berechtigung haben. CAN sticht vor allem wegen der schnellen Datenraten, Zuverlässigkeit, einfacher Erweiterbarkeit und der recht hohen Fehlersicherheit auf. Außerdem werden wichtige Botschaften priorisiert, was eine reibungslose Kommunikation bei Echtzeitanwendungen gewährleistet. Dies kann allerdings dazu führen, dass es bei einem falsch konzipierten System zum Untergang von Nachrichten geringer Priorität kommen kann. Im Gegensatz zu CAN, wird LIN hauptsächlich für unkritische Anwendungsbereiche verwendet, da LIN eine deutliche geringere Datenrate, Fehlerprävention und Fehlerüberprüfung hat. Dadurch kann ein LIN-Bus allerdings sehr kostengünstig und mit wenig Aufwand in einem Fahrzeug verwendet werden.

Da man in einem Automobil unterschiedliche Anwendungsbereiche abdecken muss, werden unterschiedliche Bussysteme, wie LIN, CAN, MOST und Flexray, verwendet und durch Gateways miteinander vernetzt. Dadurch vereint man die Vorteile aller Bussysteme und sorgt für eine möglichst kostengünstige und effiziente Vernetzung in den verschiedenen Anwendungsbereichen im Automobil.

Literatur

[Bor23] Kai Borgeest. *Elektronik in der Fahrzeugtechnik : Hardware, Software, Systeme und Projektmanagement*. 5. Auflage. ATZ/MTZ-Fachbuch. Wiesbaden: Springer Vieweg, 2023. ISBN: 9783658414832. URL: https://doi.org/10.1007/978-3-658-41483-2.

[Rau22] Mathias Rausch. *Kommunikationssysteme im Automobil : LIN, CAN, CAN FD, CAN XL, FlexRay, Automotive Ethernet*. Hanser eLibrary. München: Hanser, 2022. ISBN: 9783446474574.

[Rei10] Konrad Reif. *Batterien, Bordnetze und Vernetzung*. SpringerLink Bücher. Description based upon print version of record. Wiesbaden: Vieweg+Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden, 2010. ISBN: 9783834897138. URL: https://doi.org/10.1007/978-3-8348-9713-8.

[Rei11] Konrad Reif. *Bosch Autoelektrik und Autoelektronik : Bordnetze, Sensoren und elektronische Systeme*. 6., überarbeitete und erweiterte Auflage. SpringerLink Bücher. Wiesbaden: Vieweg+Teubner, 2011. ISBN: 9783834899026. URL: https://doi.org/10.1007/978-3-8348-9902-6.

Anhang