BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Beuth Hochschule für Technik Berlin
Fachbereich VII
Elektrotechnik und Feinwerktechnik

CAN-Bus

Ausarbeitung zum CAN-Bus
WS 09/10

Vorgelegt von: I. Bernsdorf (s7614510), T. Vogt (s760469),
L. Stojanovic (s760403), S. Szabadi (s760465)

Studiengang: Automatisierungstechnik und Elektronik
Abgabe: 06.11.2009

Betreuer:
Beuth Hochschule für Technik Berlin: Prof. Dr. Ing. Detlef Heinemann

Inhalt

1 CAN-Geschichte und Einsatzgebiete

Das Controller Area Network (CAN) wurde 1983 von der Robert Bosch GmbH entwickelt und gehört zu der Familie der Feldbusse. Es sind eine Vielzahl von unterschiedlichen Feldbusprotokollen kommerziell erhältlich. Jedoch ist kein anderes Protokoll so für den Einsatz im Automobilsektor geeignet wie der CAN-Bus. Aber auch in anderen Bereichen hat sich der CAN-Bus etabliert, wie z. B. in der Medizin-, der Automatisierungs- und der Schienenfahrzeugtechnik. In der Automatisierungstechnik wird häufig das CANopen Protokoll verwendet welches über eine ausgeprägt 7. OSI-Schicht verfügt, im Gegensatz zu dem CAN im Automotiv-Bereich. Dadurch werden nicht explizite Kenntnisse des CAN-Protokolls vorausgesetzt [4,5,11].

Zur Entwicklung leistungsstarker und kostengünstiger Controller-Lösungen kooperierte die Robert Bosch GmbH mit der Firma Intel und brachte im Jahre 1989 den ersten CAN-Controller als integrierten Schaltkreis auf den Markt. Dazu musste das CAN-Protokoll an das OSI-Schichten-Modell angepasst werden.
Der CAN-Bus unterscheidet sich von den bereits vorhandenen Vernetzungsprotokollen durch die ausgeprägte Fehlerbehandlung, die Priorisierung von Nachrichten, Echtzeitfähigkeit, die hohe Datenübertragungsrate und das Multimaster Prinzip.
Mit Hilfe von CAN konnte die Automobilindustrie massive Probleme mit dem immer stärkeren Einsatz von Elektronikmodulen, da die Vernetzung immer aufwendiger und fehleranfälliger wurde. Diese Vorteile sind auch auf die anderen genannten Anwendungsgebiete übertragbar [4,5,11].

1983 – Start der Entwicklung des CAN Bus durch die Robert Bosch GmbH
1985 – Fertigstellung der CAN-Spezifikation Version 1.0
1986 – Start der Standardisierung, ISO-Norm
1987 – Erste Prototypen von CAN Controller IC‘s
1989 – Erste CAN ICs sind käuflich zu erwerben
1991 – CAN Protokoll Version 2.0 Spezifikationen
1994 – Fertigstellung der ersten ISO-Norm
1994 – Erstes Serienfahrzeug mit CAN (Mercedes S-Klasse)
2008 – Geschätzte 65-67 Millionen produzierte Fahrzeuge mit durchschnittlich 10-15 verbauten CAN-Knoten

Die physikalischen Eigenschaften eines CAN Netzwerkes und der Formale Aufbau einer CAN-Nachricht und deren Pegel ist in den ISO-Normen 11898 festgehalten. Die Jahreszahl hinter den einzelnen ISO-Normen gibt das jeweilige Jahr der Verabschiedung an [4,5,11].

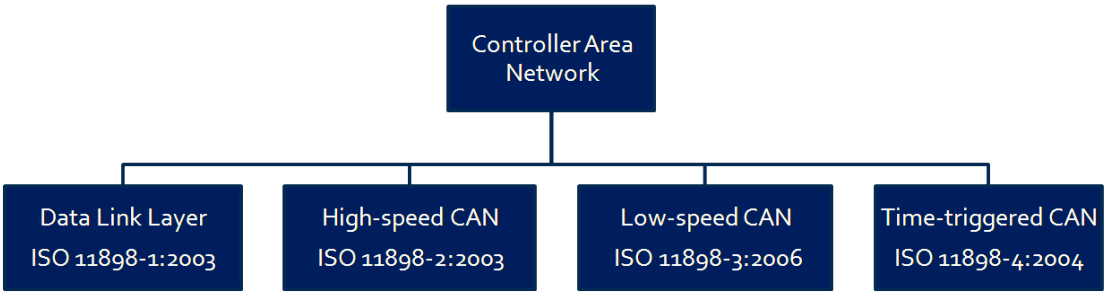


Abb. 1: Übersicht über die verschiedenen CAN ISO-Normen

2 OSI-Modell & Spezifikation

2.1 Die CAN - ISO Norm 11898-x

Der Kern des Standards besteht aus den ersten 3 Dokumenten, diese sind zuerst erschienen und enthalten die Grund-Implementierung des CAN 2.0. Hinzu kommen weitere Dokumente im CAN Standard. Diese beschreiben neuere Technologien, welche zum damaligen Zeitpunkt (erste Veröffentlichung) nicht vorgesehen waren. Nachfolgend sind die wesentlichen Merkmale der Norm Dokumente bis zum Standard 2.0 aufgezählt.

2.1.1 11898-1 (Data Link Layer and physical signalling) [1]

Das Dokument beschreibt im wesentlichem das CAN Protokoll. Bestehend aus dem OSI/ISO Schichten Modell, Data Link Layer (Medium Access Control) und Physical Layer (Physical Signalling). Ein weiteres Merkmal besteht im sogenannten Identifier (Header) einer CAN Nachricht.

- CAN 2.0 A Low-Speed (11 Bit), Base frame format
- CAN 2.0 B High-Speed (29 Bit), Extended frame format

Physical Layer:

- Verbindungsstecker (**M**edium **D**ependent **I**nterface)
- Sende/-Empfänger Charakteristik (**P**hysical **M**edium **A**ttachment)
- Bit/- Codierung/- Dekodierung/- Timing/- Synchronisation (**P**hysica**l** **S**ignalling)

Data Link Layer:

- Fehlererkennung, Fehlersignalisierung, Bus-Zugriffskontrolle, Empfangsbestätigung (**M**edium **A**ccess **C**ontrol)
- Akzeptanzfilterung, Overload Meldungen, Recovery Management (**L**ogic **L**ink **L**ayer)

OSI Schichten Modell:

- Definitionen der Schnittstellen zwischen Layer 1 und Layer 2

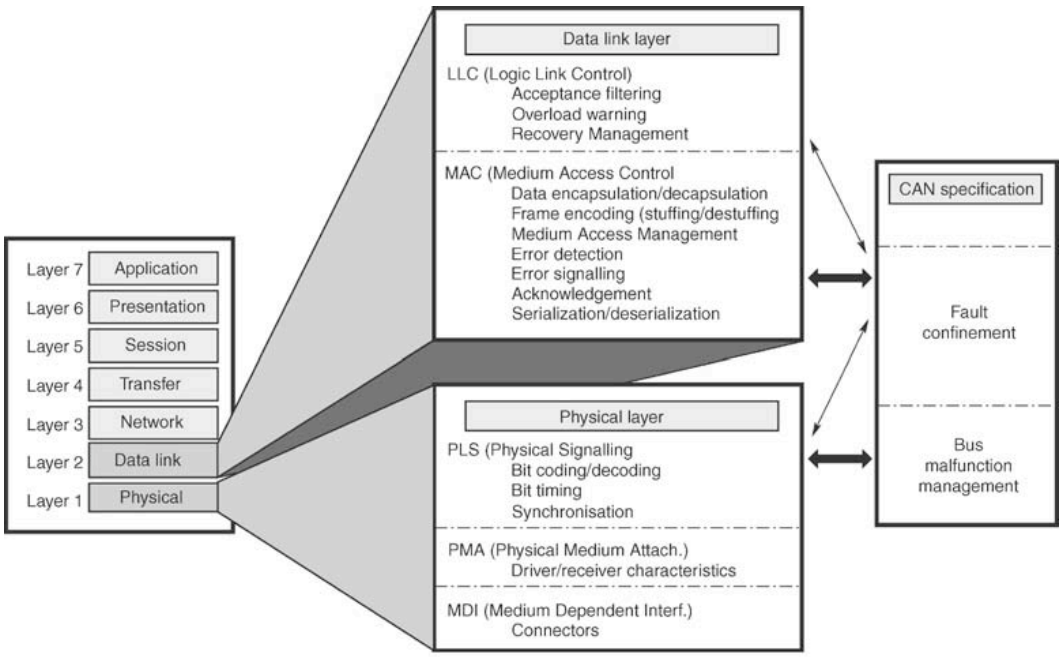


Abb. 2: OSI Schichten Modell [5]

2.1.2 11898-2 (High-Speed CAN) [2] und 11898-3 (Low-Speed CAN) [3]

Hier werden jeweils unterschiedliche Erweiterungen im Bereich des Physical Layers (PMA) beschrieben.

2.2 Übertragungsgeschwindigkeiten

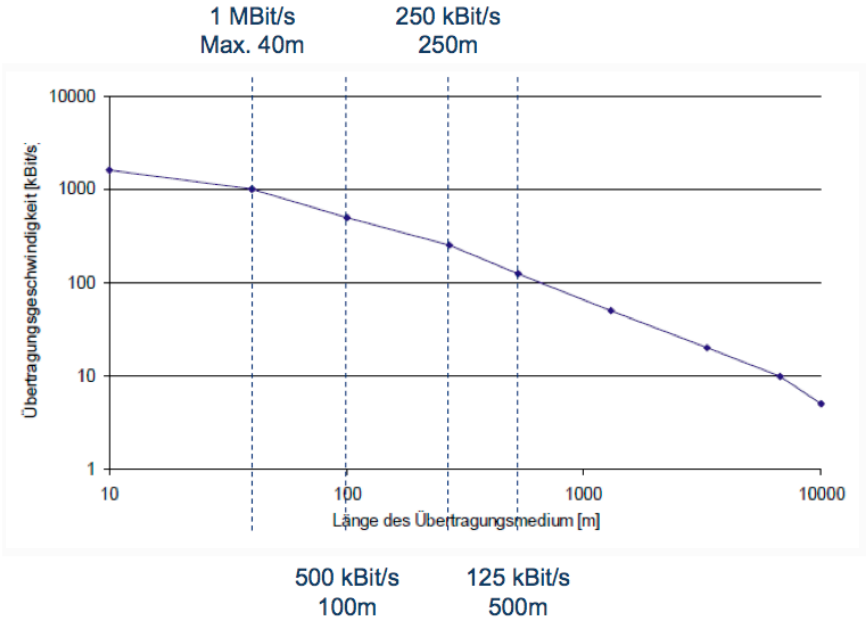


Abb. 3: Übertragungsgeschwindigkeit in Abhängigkeit der Übertragungsstrecke

Im Abb. 3 sind die max. Übertragungsgeschwindigkeiten bei entsprechenden Leitungslängen für die CAN Bus Datenübertragung dargestellt.
Die dargestellten Maximalwerte Bit/s kommen durch die signalbedingte Ausbreitung auf der Busleitung zustande. Bei hoher Übertragungsgeschwindigkeit ist die Bitzeit kürzer und umgekehrt betrachtet länger.
Von daher darf die Zeit, die ein Signal am Bus anliegt nicht kürzer sein als die Zeit welche das Signal braucht um sich auszubreiten (sonst treten Übertragungsfehler auf). Um diese Bedingung einzuhalten müssen mit zunehmender Leitungslänge die Bit/s entsprechend herabgesetzt werden [9].
Da das Übertragungsmedium in der ISO 11898-1 Physical Layer nicht vorgegeben ist, also das Übertragungsmedium aus optisch, elektrisch oder aus elektromagnetisch modulierten Signalen bestehen kann, hängt die zu erreichende Übertragungsgeschwindigkeit ebenso vom dem jeweiligen Medium ab.
Z.B. bei der elektrischen Übertragung, kommt mit zunehmender Leitungslänge die kapazitive Wirkung der Leitung hinzu, dadurch werden Signalflanken immer flacher und schwerer zu erkennen (Fehler können auftreten).

2.3 SAE Klassen

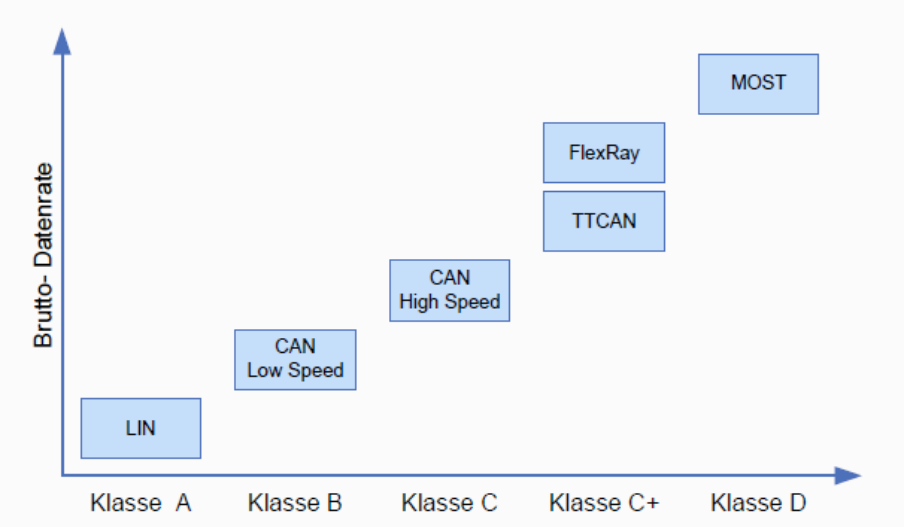


Abb. 4: SAE Klassen Diagramm [8]

Die **S**ociety of **A**utomotive **E**ngineers hat sich die mühe gemacht und die verschiedenen verwendeten Bus Protokolle verglichen und in fünf Kategorien von A bis D eingeteilt. Datenrate gegenüber der Echtzeitfähigkeit (Klasse D) nicht Echtzeitfähig (Klasse A). So kann leichter eine Auswahl eines Netzwerkprotokolls gegenüber den entsprechenden Anforderungen ausgewählt werden [10].

3 Busankopplung/Vernetzung von CAN-Knoten

In der Praxis werden CAN Netzwerke am häufigsten mit einer zweiadrigen Symmetrischen Leitung (Twisted Pair) aufgebaut. Entweder in geschirmter oder ungeschirmter Ausführung. Abb. 5 zeigt ein Beispiel von drei CAN Teilnehmern in Linien Topologie. An beiden Leitungsenden sind Abschlusswiderstände vorgesehen, um mögliche Leitungs-Reflexionen zu vermeiden, in der Regel liegen diese bei 120Ω (Wellenwiderstand der Leitung).

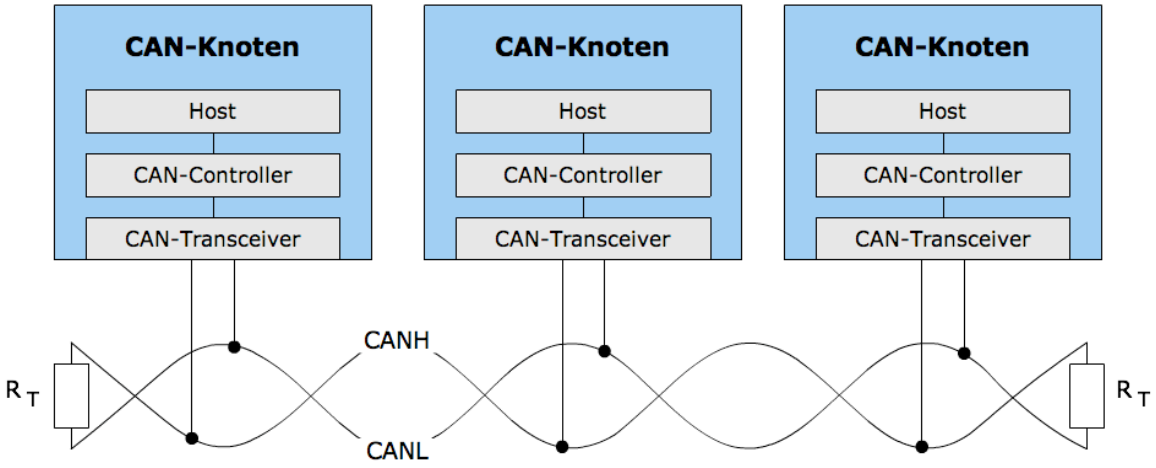


Abb. 5: Vernetzung von CAN-Knoten über Twisted-Pair-Leitung (R_T = 120Ω) [8]

Alle CAN Knoten sind über CANH und CANL angeschlossen. Die Kommunikation findet über das Multi-Master Prinzip statt, jeder Knoten kann mit jedem anderen Knoten kommunizieren.
Die Binären Signale werden im NRZ (Non Return to Zero) Verfahren kodiert und mittels Differenzsignalen übertragen. Diese Art der Signalübertragung bietet eine hohe Störfestigkeit gegenüber einwirkenden Störungen auf die Busleitung. Für beide CAN Varianten Lowspeed und Highspeed gelten unterschiedliche Spezifikationen [8].

3.1 Aufbau eines CAN-Knotens

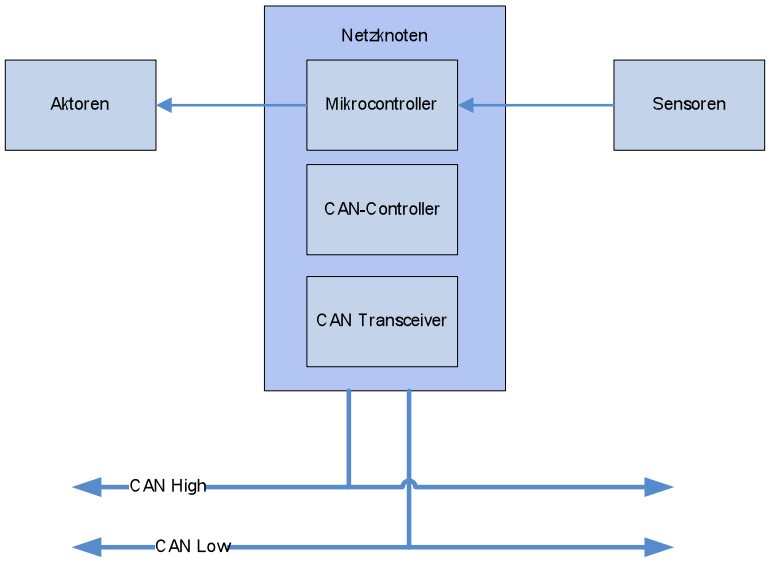


Abb. 6: CAN-Knoten Strukturübersicht

Die Abb. 6 zeigt ein einfachen Strukturüberblick eines CAN Knotens. Drei Elementare Bausteine sind zu erkennen der Mikrokontroller, CAN-Controller und CAN Transceiver
Die gesamte Funktionalität (Sensor/ Aktor) von einem CAN-Knoten ist im Mikrokontroller realisiert. Entweder ist er direkt mit einem CAN-Controller verbunden oder der Mikrokontroller enthält zusätzlich diesen. Die Aufgabe vom CAN-Controller ist unter anderem das Codieren und Decodieren der Sendedaten. Der Transceiver ist abhängig von dem verwendeten Übertragungsmedium (Physical Layer).
Es gibt in der Praxis zwei Sorten von CAN-Controllern. Den Basic CAN- und den Full-CAN-Controller. In den folgenden Abschnitten werden diese dargestellt und erläutert.

3.2 Basic CAN- und Full-CAN-Controller

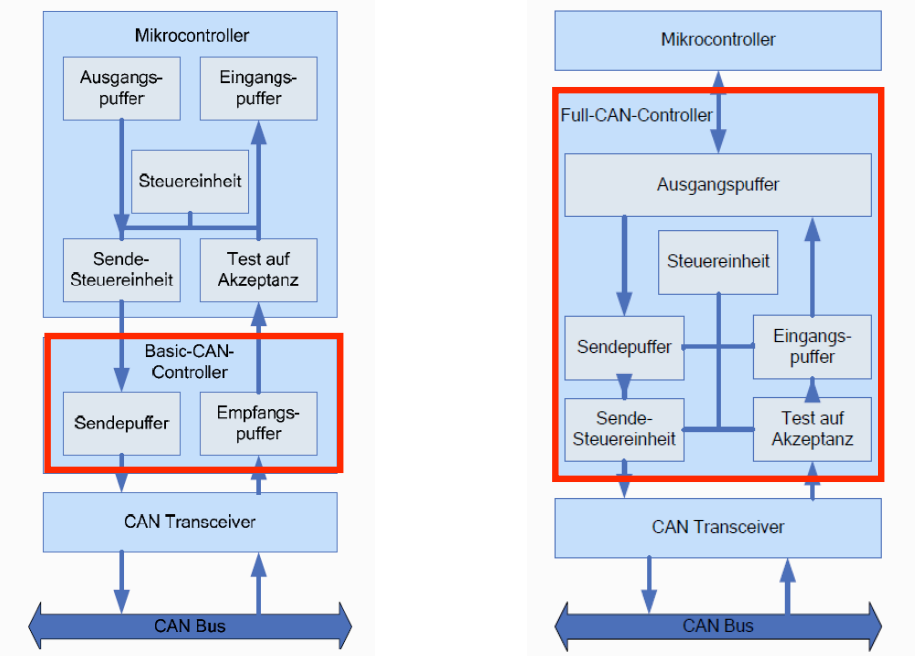


Abb. 7: Vergleich Basic- (links) und Full-CAN-Controller (rechts) [10]

Vorteil:
- günstig
- Gut für niedrige Datenraten geeignet

Nachteil:
- Nachrichten können verloren gehen
- Filterung (Test auf Akzeptanz) benötigt Rechenleistung

Der Basic CAN Kontroller wird in Netzwerken mit niedriger Datenrate eingesetzt.

Vorteil:
- Gut für hohe Datenraten geeignet
- Akzeptanzfilterung im CAN-Controller
- Benötigt kaum Rechenleistung

Der Full CAN Kontroller wird im Bereich der Highspeed Netze eingesetzt.

3.3 CAN-Transceiver

Die Anpassung der Daten an das Übertragungsmedium erfolgt dann mit einem Transceiver. Für die unterschiedlichen CAN Varianten gibt es auch unterschiedliche Transceiver.
Die Abbildung 4 zeigt den prinzipiellen Aufbau eines High-Speed-CAN-Transceivers. Bei einem rezessiven Pegel sperren beide Transistoren und somit liegt an CAN High und Low dasselbe Potenzial (Vcc * 0,5) an. Wird ein dominantes Signal übertragen, dann schalten beide Transistoren durch. Durch einen 120 Ω Abschlusswiderstand stellt sich bei Highspeed-CAN Netzwerken eine Spannungsdifferenz von 2 V ein (siehe Kapitel 5.3).
Bei einem elektrischen CAN ist die maximale Teilnehmeranzahl abhängig von der Übertragungsrate. Bei einer Geschwindigkeit von 1 MBit/s können bis zu 32 Knoten angeschlossen werden. Bei einem Lowspeed-CAN, mit einer Übertragungsgeschwindigkeit von 125 KBit/s, sind immerhin 64 Teilnehmer möglich.

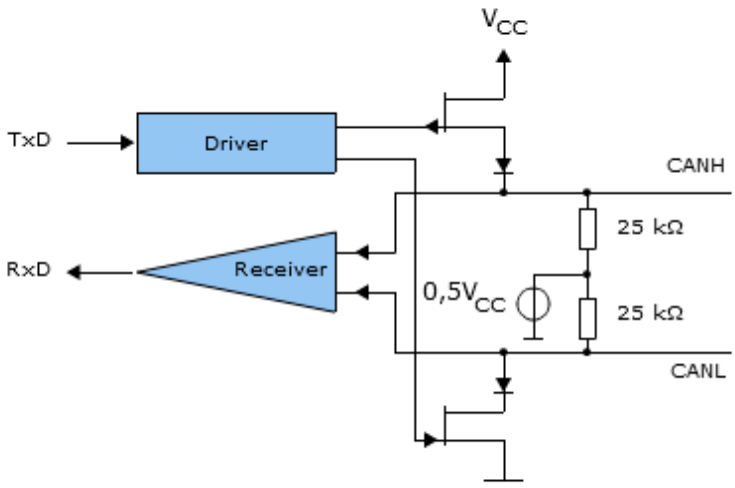


Abb. 8: Aufbau eines CAN-Transceivers [8]

4 CAN-Bus Pegel

Bevor wir weitergehen zu den Signalpegeln muss an dieser Stelle kurz erwähnt werden, dass beim CAN-Bus wie auch bei anderen Bussystemen die binären Signalzustände nicht als High und Low bezeichnet werden sondern als Rezessiv und Dominant.

4.1 Erzeugung dominanter und rezessiver Pegel

Grund für dominante und rezessive Pegel ist z.B. das die Signalübertragung sowohl aus elektrisch, elektromagnetisch modulierten und optischen Signalen realisiert werden kann. Für die jeweiligen Zustände Low und High kann es so keine einheitliche Definition geben. In Abb. 9 wird anhand eines einfachen Beispiels verdeutlicht wie diese Zustände für elektrische Bus-Netze realisiert werden (Wired-AND).

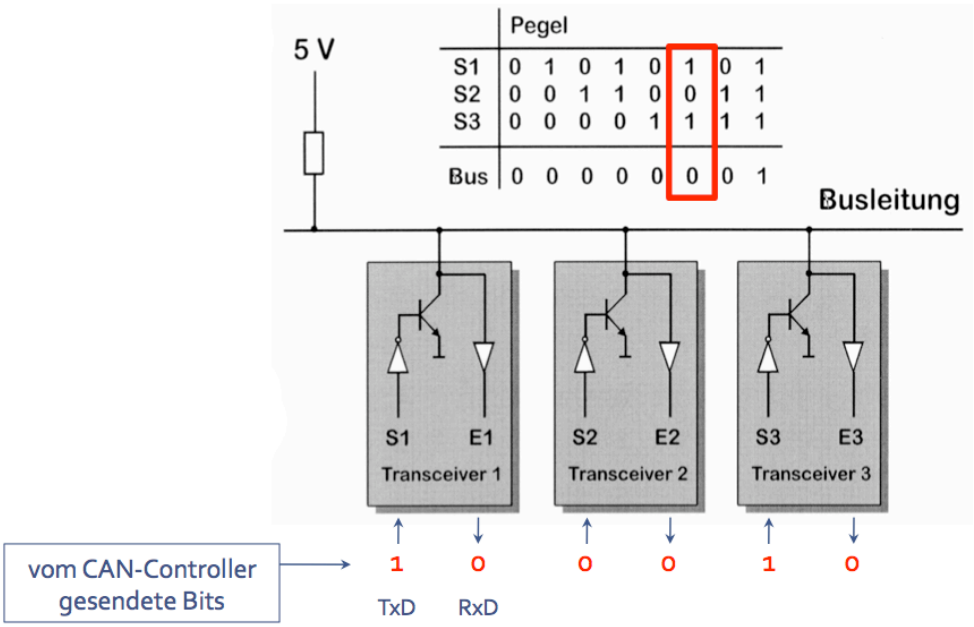


Abb. 9: Wired-AND, Rezessiv und Dominant [8]

- Rezessiv entspricht einer logischen „1“
- Dominant entspricht einer logischen „0“

Sobald ein Kollektorausgang den logischen Pegel „0“ annimmt (durch Schalten des Transistors) so setzt sich dieser auf dem Bus durch.

Im CAN 2.0 muss zwischen dem Low-Speed und High-Speed unterschieden werden. Wie der Name beider Norm Dokumente schon sagt, sind hier unterschiedliche Geschwindigkeiten standardisiert. Dadurch resultieren für beide Varianten unterschiedliche Physical Layer Definitionen und Spezifikationen.
Die Buspegel weisen bei Rezessiven oder Dominanten Signalen entsprechende Spannungsdifferenz Signale auf.

4.2 CAN-Low-Speed

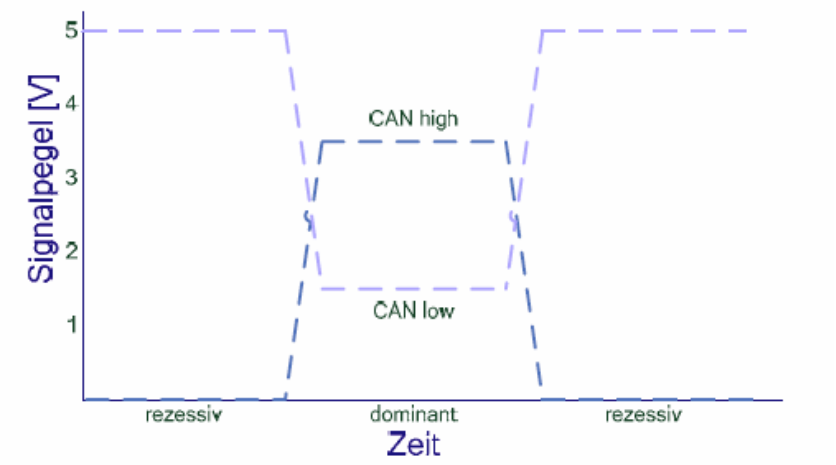


Abb. 10: Buspegel nach ISO 11898-3 [3]

Diese CAN Variante kann als Eindraht oder Zweidraht Bus aufgebaut werden. Bei der Eindraht Variante müssen alle Teilnehmer über eine gemeinsame Masse verfügen. Die max. zu erreichende Übertragungsrate liegt bei 125 kbit/s.
Vorteil, geringere Kosten durch Eindrahtvernetzung, im KFZ leicht zu realisieren und häufig eingesetzt. Es besteht weiter die Möglichkeit, dass bei einer Zweidrahtvernetzung bei Ausfall eine der beiden Datenleitungen die Datenübertragen über diese eine weiter realisiert werden kann. Busabschlusswiderstände können ebenfalls bei einer Zweidrahtvariante bei geringer Leitungslänge vernachlässigt werden.
Nachteil, anfällig gegenüber Störeinflüssen [8][10].

4.3 CAN High-Speed

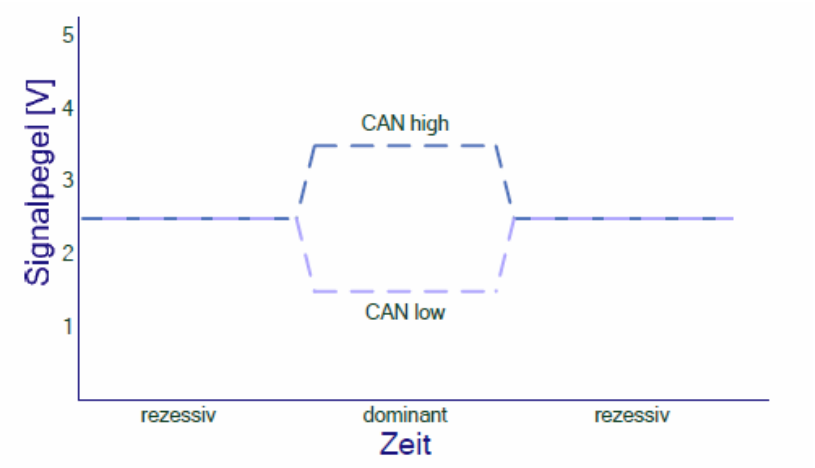


Abb. 11: Buspegel nach ISO 11898-2 [2]

Der High-Speed CAN muss mit einer Zweidraht Leitung aufgebaut werden und über entsprechende Busabschlusswiderstände an den Leitungsenden verfügen 120 Ohm (Wellenwiderstand der Leitung). In Abhängigkeit der Leitungslänge kann eine Datenrate von bis zu max. 1Mbit/s erreicht werden [8].

5 CAN Frames

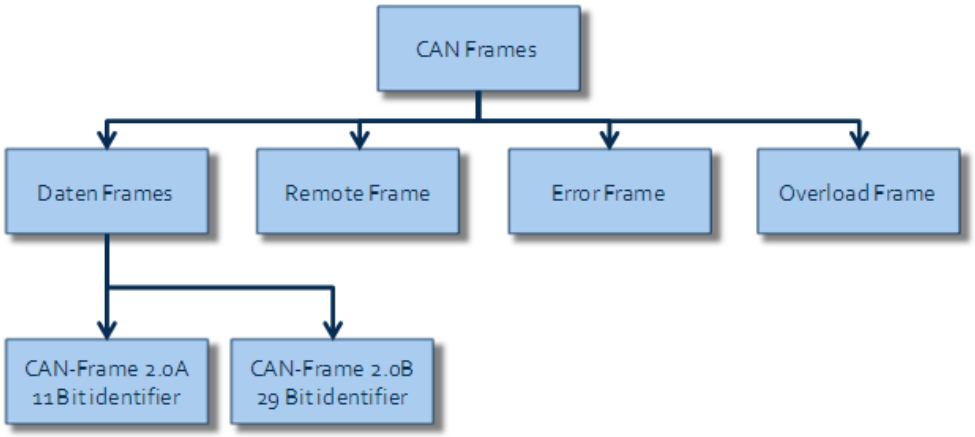


Abb. 12: Übersicht über die unterschiedlichen CAN-Frames

Die Datenübertragung von CAN erfolgt in Form von Botschaftsrahmen, auch Frames genannt. Es wird zwischen vier verschiedenen Frames unterschieden. Mit einem Remote-Frame können Nachrichten von anderen CAN-Knoten angefordert werden. Ein solcher Remote-Frame enthält dabei keine Nutzdaten. Für den Fall einer Fehlerdetektion versendet ein CAN-Knoten einen Error-Frame an alle anderen Netzwerkteilnehmer. Mit einem Overload-Frame kann ein Knoten allen anderen Teilnehmern eine Verzögerung mitteilen und zur gleichen Zeit keine neuen Nachrichten empfangen. Die Overload- und Error-Frames können nicht durch den Software Entwickler erzeugt werden, die Erzeugung dieser Nachricht erfolgt durch den verwendeten CAN-Controller [4].
Die vierte Form von Frames stellen schließlich Daten-Frames dar, welche Nutznachrichten, wie z. B. ein Sensorsignal, enthalten. Daten-Frames bestehen je aus einem Arbitrationsfeld, Kontrollfeld, Datenfeld, Sicherungsfeld und Quittierungsfeld [4].

Abb. 13: Aufbau einer CAN-Nachricht [10]

Man unterteilt Daten-Frames weiterhin in zwei Unterarten, die sich hauptsächlich in der Länge ihres Arbitrationsfeldes und ihres Kontrollfeldes unterscheiden. So verfügen CAN 2.0A Nachrichten über einen 11 Bit-Identifier und ein reserviertes Bit im Kontrollfeld, für eventuelle Erweiterungen des CAN-Standards. CAN 2.0B Nachrichten dagegen besitzen einen 29

Bit-Identifier der in einen 11 Bit und einen 18-Bit-Teil unterteilt ist. Für eventuelle Erweiterungen verfügen Nachrichten dieses Formats über zwei reservierte Bits. Die meisten CAN-Controller sind in der Lage beide Nachrichtenformate zu empfangen und zu senden [4].

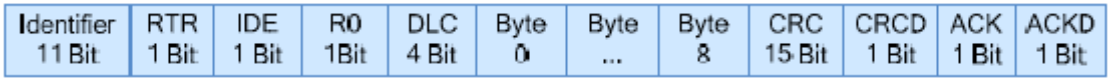


Abb. 14: CAN-Nachricht nach Standard 2.0A [10]

Abb. 15: CAN-Nachricht nach Standard 2.0B [10]

5.1 Startbit (Start of Frame)

Die Übertragung einer CAN-Nachricht beginnt immer mit dem Senden eines dominanten Startbits und dient der Synchronisation aller CAN-Knoten [4].

5.2 Arbitrationsfeld (Arbitration Field)

Das Arbitrationsfeld enthält den Identifier einer bestimmten Nachricht. In einem Netzwerk sollte ein solcher Identifier je nur einmal vergeben werden, da eine mehrmalige Verwendung zu Kommunikationsfehlern führen kann. Bei CAN 2.0A Nachrichten umfasst das Arbitrationsfeld einen 11 Bit breiten Identifier sowie das Remote-Bit, welches bei Daten-Frames immer dominant übertragen wird. Als Konsequenz haben Nachrichten mit Nutzdaten immer Vorrang vor Remote anfragen. Die Länge des Arbitrationsfeld einer CAN 2.0B Nachricht umfasst 31 Bits. Dabei beinhalten 29 Bits den Identifier, während durch die verbleibenden zwei Bits, die immer rezessiv gesendet werden, sichergestellt wird, dass sich die Nachrichten Formate stets unterscheiden und damit bevorzugt CAN 2.0A Botschaften übertragen werden [4].

5.3 Kontrollfeld (Control Field)

Das Kontrollfeld beinhaltet Informationen über die Anzahl der nachfolgenden Datenbits sowie über die Signalisierung von Remote Anforderungen. Botschaften des CAN 2.0A Standards enthalten dazu noch das Identifier Extension Bits (IDE), welches rezessiv gesendet wird und Einfluss auf die Priorisierung hat. Ein CAN 2.0A Frame hat somit immer Vorrang vor einem CAN 2.0B Frame. Durch diese Erweiterung des CAN 2.0A Standards steht nur noch ein Bit für eventuelle Erweiterungen zur Verfügung.
Die Struktur einer CAN 2.0B Nachricht ist ähnlich zu der von CAN 2.0A Standards. Das IDE-Bit ist hier jedoch ein Teil des 29 Bit breiten Identifiers und für zukünftige Erweiterungen stehen zwei reservierte Bits zur Verfügung [4].

5.4 Datenfeld (Data Field)

Für die Übertragung von Daten, wie z. B. eines intelligenten Sensors stehen maximal acht Bytes zur Verfügung. Theoretisch sind auch kürzere Nachrichten möglich und diese verkürzen die Zeit der Busbelegung. Zur Synchronisation von Teilnehmern ist sogar ein Versenden von CAN-Nachrichten ohne Dateninhalt möglich [4].

5.5 Sicherungsfeld (Cyclic Redundancy Check Field)

Insgesamt besteht das Sicherungsfeld aus einer 15 Bit „Prüfsumme“ und einem Delimiter (CRCD) der stets rezessiv versendet wird. Die Prüfsumme wird dazu zyklisch über vorhergehenden Bits gebildet und dient der Erkennung von fehlerhaft übertragenden Bits [4].

5.6 Quittierungsfeld (Acknowledgement Field)

Das Quittierungsfeld besteht aus zwei Bits. Zum Einem aus einem Acknowledgement Bit (ACK) und zum Anderen aus einem Acknowledgement Delimiter (ACKD). Beide werden rezessiv übertragen. Im Falle einer fehlerfreien Übertragung wird der Erhalt von dem empfangenden CAN-Knoten bestätigt. Dabei hat eine Akzeptanzfilterung von eingehenden Nachrichten keinen Einfluss auf die Bestätigung des fehlerfreien Empfangs der Botschaft [4].

5.7 Ende Kennung (End of Frame)

Das Ende einer CAN-Nachricht wird den anderen Knoten durch sieben aufeinanderfolgende rezessive Bits signalisiert. Mit der Übertragung von sieben gleichen Signalpegeln wird bewusst die Bit Stuffing Regel verletzt [4].

5.8 Rahmenpause (Inter Frame Space)

Nach dem End of Frame werden drei weitere rezessive Bits übertragen, so dass insgesamt mit dem End of Frame zehn rezessive Bits übermittelt werden. Diese Sequenz gilt für Remote und Daten Frames. Nachrichten die bei dem Auftreten eines Fehlers gesendet werden können diese Bitfolge allerdings überschreiben. Zu diesen Nachrichten gehören die aktiven Error-, passiven Error- und Overload-Frames. Welches der beiden Error Frames die Inter Frame Space überschreitet, ist abhängig vom Fehlerzustand des sendenden CAN-Knotens (siehe Kapitel 11.2) [4].

5.8.1 Error Frame

Mit einem Error Frame kann ein Netzwerkteilnehmer den restlichen Teilnehmern einen Übertragungsfehler signalisieren. Dafür gibt es zwei unterschiedliche Nachrichtentypen, diese sind Abhängig von dem Fehlerzustand des Knotens (siehe Kapitel 12.2). Befindet sich ein Teilnehmer im Error Active Zustand, so darf dieser Knoten ein Active Error Frame senden.
Ein Active Error Nachricht besteht aus insgesamt drei Teilen. Im primären Teil werden sechs dominante Bits übertragen, welche die Stuff-Bit Regel verletzt und überschreiben damit den korrupten Teil der Nachricht. Diese Nachricht kann dann nach dem Error Frame nochmals übertragen werden, sofern der Bus nicht von einer Nachricht mit einer höheren Priorität belegt ist. Danach können noch bis zu sechs dominanten Bits übertragen werden und bilden den sekundären Teil des Error Frames. Der sekundäre Teil dient zur globalen Fehlersignalisierung. Abschließend wird ein Error Delimter übertragen, dieser setzt sich aus sechs rezessiven Bits zusammen.
Wenn ein CAN-Knoten gestört ist und sich nicht mehr im Error Active Mode befindet, so kann dieser sofern sich der Knoten nun im Error Passive Mode befindet, ein Passive Error Nachricht senden. Die Passive Error Nachricht unterscheidet sich nur im primären Teil von der Error Active Nachricht. Die sechs Bit des primären Teils werden somit rezessiv übertragen und können die Datenübertragung auf dem Bus somit nicht stören. Mit der Passive Error Nachricht wird nur die Übertragung des fehlerhaften Knoten gestoppt.

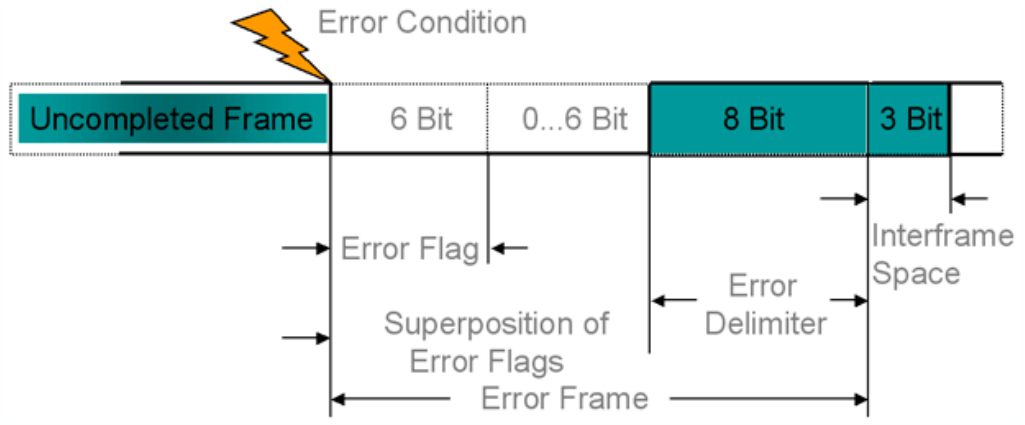


Abb. 16: Aufbau eines Active Error Flag Frames [12]

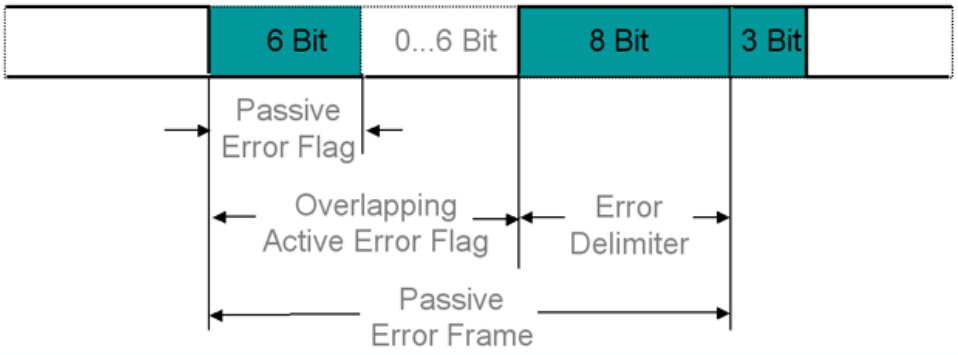


Abb. 17: Aufbau eines Passive Error Flag Frames [12]

5.8.2 Overload Frame

Ein Overload Frame wird dann gesendet, wenn ein an der CAN-Kommunikation teilnehmender Knoten es nicht rechtzeitig schafft die Daten zu verarbeiten. Die diesen Fall sendet der betreffende Knoten einen Overload Frame, dieses besteht wie die Error Frames aus insgesamt drei Teilen. Der primäre Teil dieses Frames, ist identisch mit dem des Active Error Frames. Jeder Overload Frame wird mit dem Overload Delimiter abgeschlossen, dieser besteht aus acht dominanten Bits und anschließend wird der Bus wieder freigegeben. Sollte die Verarbeitung der Daten noch nicht abgeschlossen sein, so kann der Knoten ein weiteres mal einen Overload Frame versenden. Das versenden von Overload Frames hat keinen Einfluss auf den Fehlerzähler (siehe Kapitel 11.2).

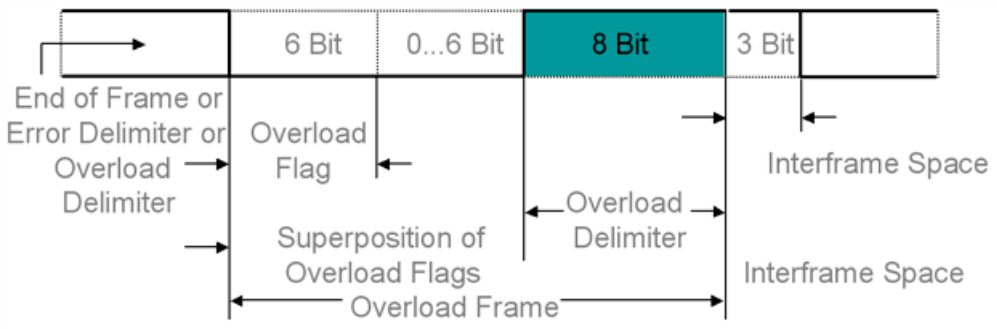


Abb. 18: Aufbau eines Overload Frames [12]

6 Zeichencodierung

Beim CAN Bus werden die Zeichen NRZ codiert (Non-Return-to-Zero). Im nachfolgenden Abschnitt wird der NRZ Code genauer beschreiben. Um die Nachteile des NRZ Codes zu veranschaulichen, wird zusätzlich der Manchester Code beschrieben.

6.1 NRZ-Kodierung (Non-Return-to-Zero)

Die NRZ-Kodierung ist ein Verfahren zur Leitungskodierung. Dabei können zwei unterschiedlich hohe Spannungspegel oder Phasenlagen eines Trägersignals zur Informationsübertragung eines Bits genutzt werden. In vielen Anwendungen wird z.B. 0V zur Übertragung einer logischen Null genutzt, für eine logisch eins z.B. 5V.
Ein Nachteil der NRZ-Kodierung ist, dass bei längerer Übertragung eines Zeichens keine Signaländerung statt findet. Dies kann bei asynchronen Übertragungen zum Verlust des Taktsignals führen, da der Takt nicht mehr zurück gewonnen werden kann.

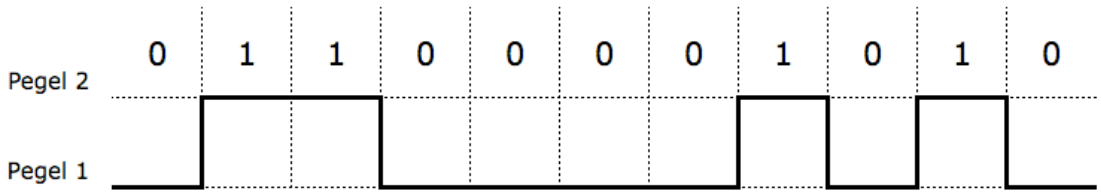


Abb. 19: NRZ-Kodierung

In der Abb. 19 ist der Zeitverlauf des NRZ-Codes dargestellt.

6.2 Manchester-Kodierung

Im Gegensatz zur NRZ-Kodierung werden bei der Manchester-Kodierung zwei unterschiedliche Pegel benötigt um ein Zeichen darzustellen. Dabei stellt z.B. der Übergang von einen high Pegel nach low eine logische null dar und der Übergang von low nach high eine logische eins. In der folgenden Abb. ist es genau andersherum (Übergang von low nach high eine logische null und Übergang von high nach low eine logische eins).

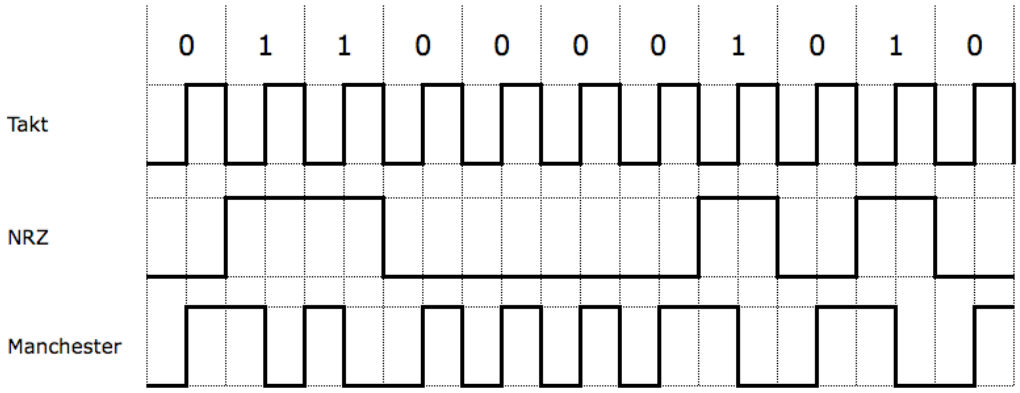


Abb. 20: Manchester-Kodierung

In der Abb. 20 ist der Zeitverlauf der NRZ- und Manchester-Kodierung dargestellt. Dort ist der Unterschied zwischen denn beiden Kodierungsarten gut zu erkennen. Ebenfalls ist in dieser Abbildung zu erkenn, das bei der Manchester-Kodierung die Taktrückgewinnung kein Problem darstellt, da bei jeder Zeichenübertragung ein Pegelwechsel stattfindet. Der Nachteil der Manchester-Kodierung liegt in der benötigten Bandbreite, die um das doppelte Größer als bei der NRZ-Kodierung ist.

7 Bit-Timing

Als erstes soll kurz erläutert werden, warum das Thema Bit-Timing im CAN-Bus so wichtig ist. Durch verschieden Oszillatoren (die unterschiedliche Toleranzen haben), Temperatur unterschiede und durch Zeitverzögerungen können sich die Bit-Zeiten in einem CAN-Netzwerk verändern. Deswegen muss sichergestellt werden, dass jedes Bit zum richtigen Zeitpunkt auf dem Bus abgetastet wird. Und dies wird mit dem Bit-Timing realisiert.
Die Realisierung des Bit-Timings findet direkt in jedem CAN-Kontroller statt. Da dies ein sehr komplexes Thema ist, wird hier nicht weiter darauf eingegangen. Es sein nur soviel gesagt, dass jedes Bit in vier Segmente (Synchronisation Segment, Propagation Segment, Phase 1 Segment und Phase 2 Segment) und 8 bis 25 Zeitquanten (t_{SCL}) eingeteilt ist. Das Synchronisations Segment ist immer ein Zeitquantum lang und befindet sich am Anfang des Bits. Die Abtastung findet immer zwischen den Segmenten Phase 1 Segment und dem Phase 2 Segment statt. In der Abb. 21 ist das Bit-Timing dargestellt.

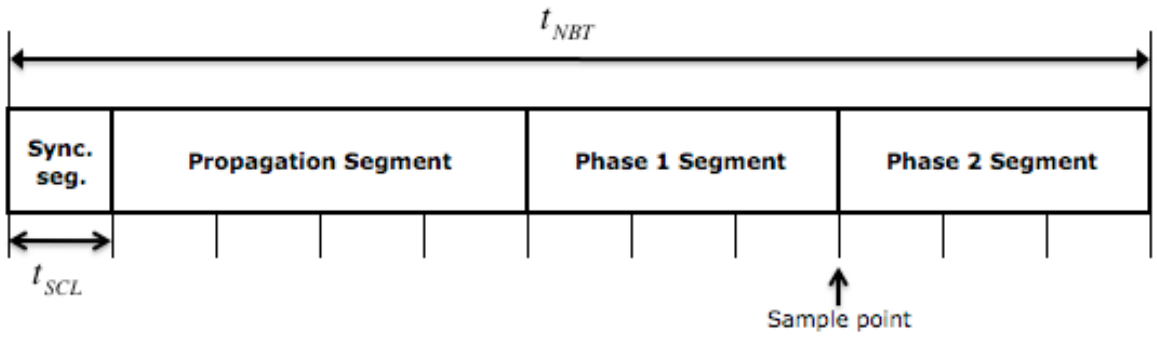


Abb. 21: Bit-Timing

8 Synchronisation

Im CAN-Bus sind zwei verschiedne Arten der Synchronisation vorgesehen,

- Hardware Synchronisation
- Resynchronisation

Bei der Hardware Synchronisation findet die Synchronisation am Anfang des Frames (SOF), also Synchronisation auf die erste Signalflanke.

Bei der Resynchronisation findet die Synchronisation während des Empfangs des Frames statt. Dabei wird die Synchronisation durch den Abgleich des internen Taktsignals auf die Flankenwechsel durchgeführt. Um regelmäßige Flankenwechsel zu Garantieren, gibt es im CAN-Bus die Stuff-Bits. Diese werden im Kapitel (12.2) genauer erläutert.

9 Akzeptanzfilterung

Über die Akzeptanzfilterung können an dem jeweiligen CAN-Knoten bestimmte Identifier herausgefiltert werden. Dies dient der Reduzierung der zu verarbeitenden Datenmenge. Der Full-CAN-Controller verwendet hierfür ein Akzeptanz-Code- und ein Akzeptanz-Masken-Register. In der folgenden Abbildung wird die Funktionsweise der Akzeptanzfilterung verdeutlicht.

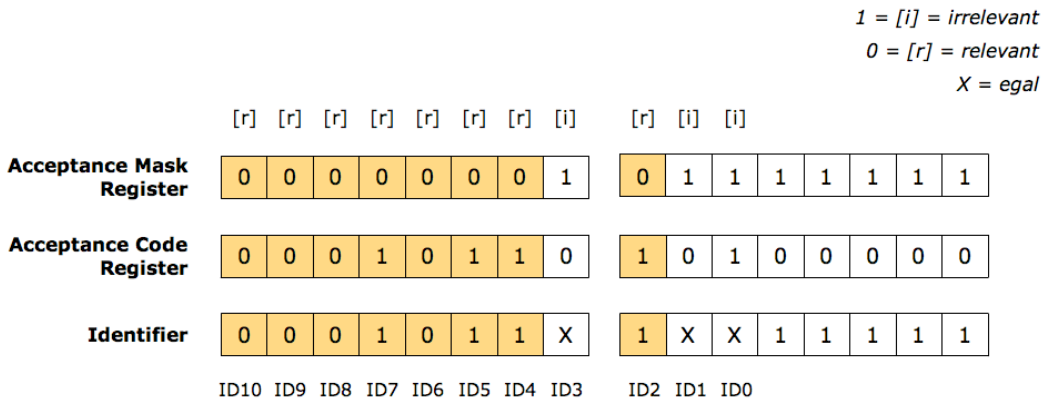


Abb. 22: Der Akzeptanzfilter anhand eines Beispiels [8]

9.1 Acceptance Mask Register

Im Acceptance Mask Register wird bestimmt welche Stellen im Acceptance Code Register für die Filterung relevant sind und welche nicht. Brauch eine Stelle im Acceptance Code Register nicht beachtet werden so wird an die dementsprechende Stelle im Acceptance Mask Register eine 1 geschrieben. Soll eine Stelle im Acceptance Code Register beachtet werden wird im Acceptance Mask Register eine 0 eingetragen.

9.2 Acceptance Code Register

Im Acceptance Code Register werden die Zustände der auszuwertenden Bits eingetragen.

9.3 Identifier

Für den Identifier werden alle relevanten Stellen des Acceptance Code Registers übernommen. An die Stellen die über das Acceptance Mask Register als irrelevant gekennzeichnet

wurden, wird ein „x“ (für egal: 0 oder 1) eingetragen. Ist im Identifier nur ein „x“ vorhanden kann nur dieses eine „x“ in 0 und 1 variieren. Es würden also nur 2 Identifier akzeptiert werden.
In unserem Beispiel in Abb. 22 kommt das „x“ dreimal vor. Es werden dementsprechend 2^3 = 8 Identifier akzeptiert. Die Binärwerte der Identifier werden dann zur Übersichtlichkeit in HEX-Werte umgerechnet.

Die folgenden 8 CAN-Nachrichten werden also durchgelassen:

0xB4, 0xB5, 0xB6, 0xB7, 0xBC, 0xBD, 0xBE und 0xBF

10 Buszugriffsverfahren

10.1 Buszugriff über CSMA/CA

CSMA/CA bedeutet Carrier Sense Multiple Access/Collision Avoidance. Wie der Name schon sagt, handelt es sich hierbei um ein Buszugriffsverfahren, bei welchem Kollisionen vermieden werden. Kommt es also zu einem zeitgleichen Versuch mehrerer Teilnehmer, auf dem Bus etwas zu versenden, regelt das CSMA/CA-Verfahren den zerstörungsfreien (es gehen keine Daten verloren und es müssen keine Daten wiederholt gesendet werden) Buszugriff. Anhand des folgenden Beispiels wird die Funktionsweise von CSMA/CA dargestellt und erläutert.

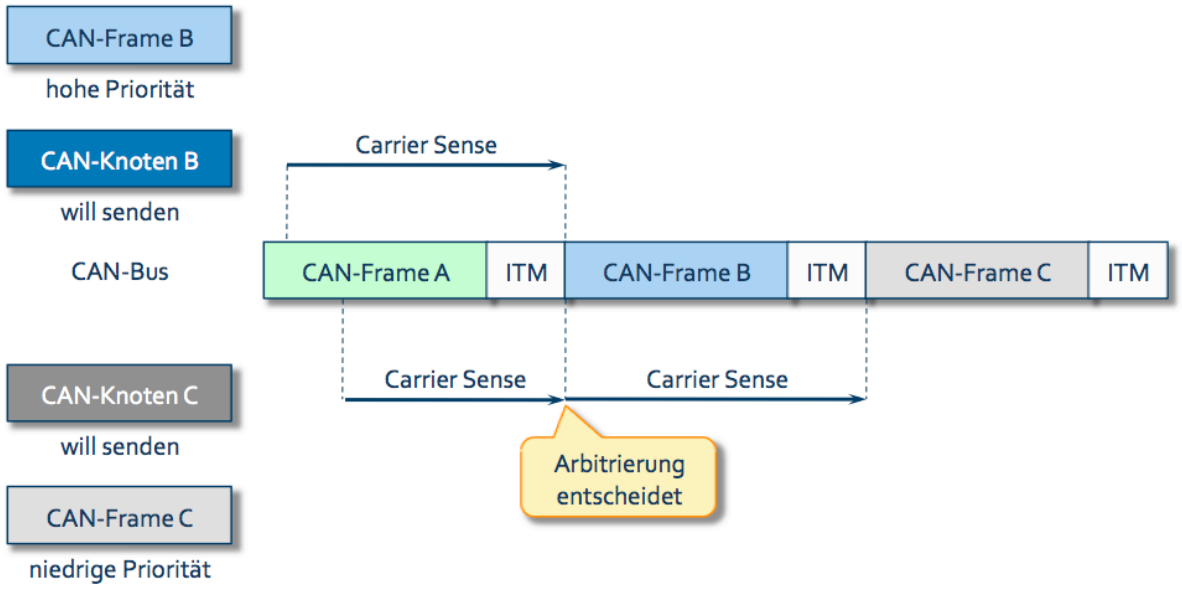


Abb. 23: CSMA/CA anhand eines Beispiels [8] (geändert)

In Abb. 23 wollen zwei Knoten (Knoten B und Knoten C) eines CAN-Netzwerkes gleichzeitig eine Nachricht auf dem Bus senden. Es kann immer nur ein Teilnehmer gleichzeitig eine Nachricht auf dem Bus senden (in diesem Fall CAN-Frame A). Alle anderen Teilnehmer des Buses (hier Knoten B und Knoten C) sind in dieser Zeit Empfänger und beobachten den Bus (Carrier Sense) solange bis er wieder frei ist (nach Abschluss des ITM- (Intermission-) Feldes).

Wie auf dem Bild zu erkennen ist, wollen Knoten B und Knoten C eine Nachricht auf den Bus senden während sich der CAN-Frame A noch auf dem Bus befindet. Knoten B will eine Nachricht mit hoher Priorität, d.h. mit niedrigem Identifier, und Knoten C eine Nachricht mit niedriger Priorität, d.h. mit einem hohen Identifier senden. Wird CAN-Frame A durch das ITM-Feld beendet ist der Bus wieder frei für andere Teilnehmer und es kommt ein Verfahren zum Einsatz, welches den zerstörungsfreien Buszugriff für Knoten B und Knoten C regelt - die bitweise Arbitrierung (siehe 10.2). Hierbei wird die Nachricht auf dem Bus gesendet welche die höhere Priorität hat. Da CAN-Frame B eine höhere Priorität als CAN-Frame C hat, gewinnt CAN-Frame B die Arbitrierung und darf gesendet werden. Knoten C beobachtet weiterhin den Bus bis dieser wieder frei ist um seine Nachricht zu senden. Hat Knoten B seine Nachricht erfolgreich gesendet, erhält automatisch Knoten C den Buszugriff, da kein anderer Knoten eine Nachricht senden will.

10.2 Bitweise Arbitrierung

Die bitweise Arbitrierung entscheidet bei einem Mehrfachzugriff auf den Bus über den Teilnehmer der seine Nachricht senden darf. Hierbei werden die Identifier der sendewilligen Nachrichten bitweise vom höchstwertigsten bis hin zum niederwertigsten Bit miteinander verglichen. Das funktioniert so, dass ein rezessiver Pegel (logisch „1“) immer von einem dominanten Pegel (logisch „0“) überschrieben wird (Wired-AND-Prinzip). Daten- und Remote-Frames werden über ihren Identifier priorisiert. Der Identifier stellt keine Adressierung eines Knotens dar, sondern ist eine eindeutige Zuordnung zu einer bestimmten Nachricht. Beruhend auf dem Prinzip von dominanten und rezessiven Pegeln hat bei Mehrfachzugriff stets die Nachricht mit dem niedrigsten Identifier Vorrang. Anhand der folgenden Abbildung wird dieses Verfahren verdeutlicht.

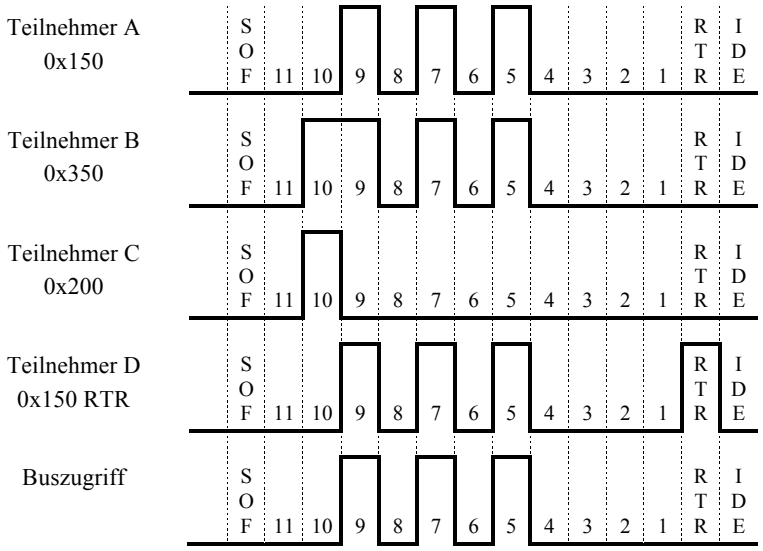


Abb. 24: Beispiel einer bitweisen Arbitrierung

Teilnehmer A (Identifier 0x150), B (Identifier 0x350) und C (0x200) wollen gleichzeitig eine Nachricht auf den Bus schicken. Hierbei werden die Bits der einzelnen Teilnehmer nacheinander (bitweise) auf den Bus geschaltet (von Bit 11 bis Bit 1). Dabei werden rezessive Pegel (logisch „1“) von dominanten Pegeln (logisch „0“) überschrieben ohne die Übertragung zu stören oder abzubrechen.

Ausscheiden von Teilnehmer B und C
Wird ein rezessiver Pegel, z.B. Bit 10 von Teilnehmer B und C von dem dominanten Pegel von Teilnehmer A und D überschrieben, so verlieren Teilnehmer B und C bereits die Arbitrierung und arbeiten von nun an als Empfänger und beobachten den Bus (Carrier Sense). Es kämpfen nur noch Teilnehmer A und Teilnehmer D um die Arbitrierung.

Ausscheiden von Teilnehmer D
Teilnehmer D scheidet aus, da er ein rezessives RTR-Bit sendet, welches von dem dominanten RTR-Bit von Teilnehmer A überschrieben wird. Teilnehmer D arbeitet nun auch als Empfänger.

→ Da alle Teilnehmer bis auf Teilnehmer A ausgeschieden sind, erhält Teilnehmer A den Buszugriff.

Somit hat eine Nachricht mit niedrigem Identifier automatisch immer Vorrang vor einer Nachricht mit hohem Identifier. So wird außerdem sichergestellt, dass ein Datenframe immer Vorrang vor einem Remoteframe hat. Der Teilnehmer der die Arbitrierung gewonnen hat darf seine Nachricht unverzögert auf dem Bus weitersenden.

In der folgenden Abbildung ist ein Mehrfachzugriff mit dementsprechender Busbelegung dargestellt. Außerdem ist im Empfangszweig die Auswirkung des für die jeweiligen Knoten eingestellten Akzeptanzfilters zu erkennen.

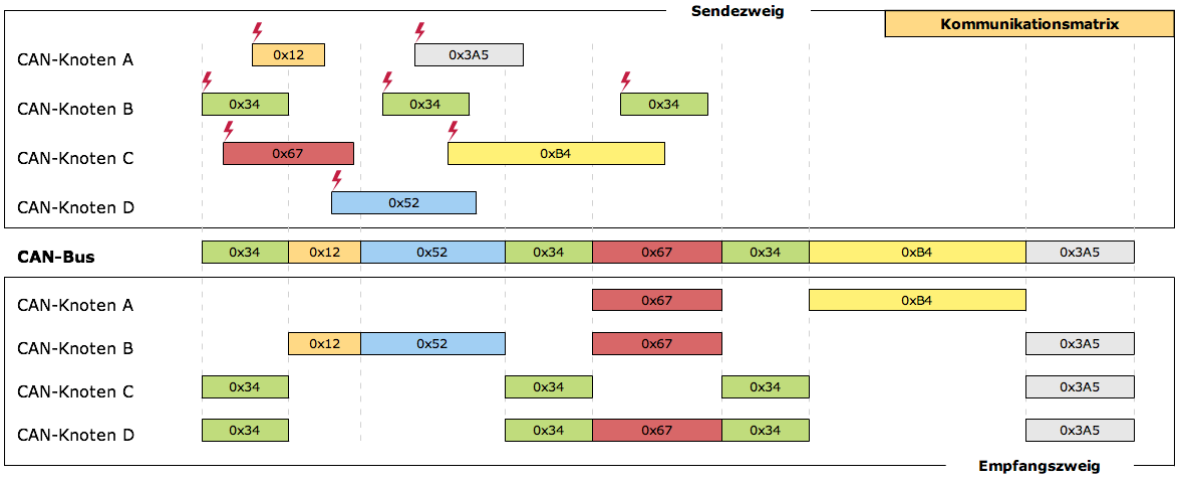


Abb. 25: Busbelegung bei Mehrfachzugriff, Akzeptanzfilterung von Nachrichten [8]

Sendezweig
CAN-Knoten A, B, C und D wollen alle zu gewissen Zeitpunkten Nachrichten verschicken. Teilnehmer A bis D erhalten der Reihe nach, nach dem Prinzip der bitweisen Arbitrierung den Zugriff auf den Bus. Bei niederprioren Nachrichten (z.B. 0x3A5 von Knoten A) kann es zu längeren Wartezeiten kommen bis sie ihre Nachricht auf dem Bus senden dürfen, wodurch die Echtzeitfähigkeit eingeschränkt ist.

→ bei hoher Buslast ist nur bei hochprioren Nachrichten Echtzeitfähigkeit gewährleistet

Empfangszweig
Teilnehmer (Knoten A bis D) können nach dem Prinzip der Akzeptanzfilterung für nur bestimmte Nachrichten empfänglich gemacht werden. Der µC eines Knotens verarbeitet dann nur Nachrichten welche im Akzeptanzfilter des CAN-Controllers zugelassen sind.

Die Akzeptanzfilter der verschiedenen Knoten sind wie folgt eingestellt:

- Knoten A empfängt nur Nachrichten mit den Identifiern: 0x67, 0xB4
- Knoten B empfängt nur Nachrichten mit den Identifiern: 0x12, 0x52, 0x67, 0x3A5
- Knoten C empfängt nur Nachrichten mit den Identifiern: 0x34, 0x3A5
- Knoten D empfängt nur Nachrichten mit den Identifiern: 0x34, 0x67, 0x3A5

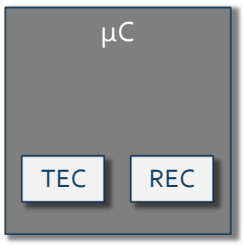
11 Fehlermanagement

Um anhaltende Störungen an einem CAN-Knoten zu vermeiden, gibt es drei verschiedene Fehlerzustände in denen sich ein solcher Knoten befinden kann. Zur zusätzlichen Differenzierung ob es sich um einen Sende- oder Empfangsfehler handelt, enthält jeder µC eines CAN-Knotens einen Sendefehlerzähler (TEC) und einen Empfangsfehlerzähler (REC).

11.1 Fehlerzähler TEC und REC

Die Fehlerzähler werden bei einer fehlerfrei gesendeten oder empfangenen Nachricht um 1 dekrementiert.

Enthält eine Nachricht allerdings Fehler werden die dementsprechenden Fehlerzähler je nach Schwere des Fehlers um 1 bis 8 inkrementiert.



11.2 Die verschiedenen Fehlerzustände eines CAN-Knotens

Active error:
CAN-Knoten ist ein vollwertiger Kommunikationsteilnehmer [10]. Er kann senden, empfangen und Fehler melden.

Passive error:
CAN-Knoten hat nur noch passives Fehlermeldungsrecht. Fehlerhafter Empfang kann nur über das ACK-Feld signalisiert werden [10].

Bus OFF:
CAN-Knoten wird elektrisch vom Bus getrennt. Um aus diesem Zustand herauszukommen muss betroffene Teilnehmer zurückgesetzt werden.

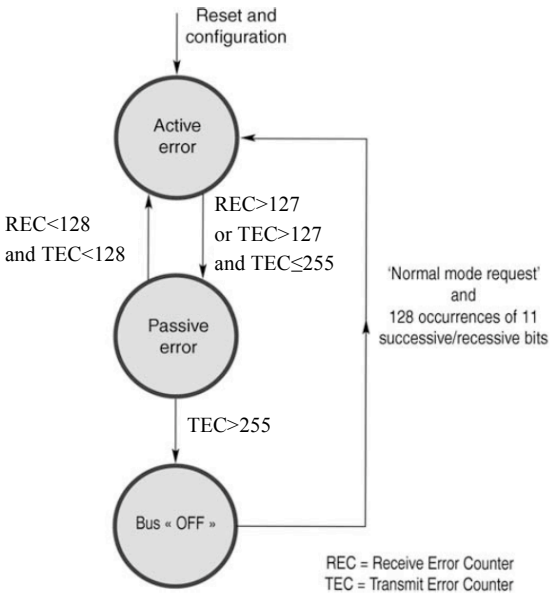


Abb. 26: Fehlerzustände eines CAN-Knotens [5] (geändert)

12 Sicherungsmechanismen/Fehlererkennung

Im CAN-Protokoll und in der Hardware sind verschiedene Sicherungsmechanismen zur sicheren Datenübertragung und Fehlererkennung integriert. In den folgenden Abschnitten werden diese vorgestellt und beschrieben.

12.1 Sicherungsmechanismen in Sender und Empfänger

Die verschiedenen Kontrollmechanismen sollen für eine hohe Datenkonsistenz im CAN-Netzwerk sorgen. Welche Kontrollmechanismen die jeweiligen Sender und Empfänger nutzen ist in der folgenden Tabelle ersichtlich.

<table>
<tr><th></th><th>Sender</th><th>Empfänger</th></tr>
<tr><td>Bitebene</td><td>Bus Monitoring
• ist der gesendete Pegel ungleich dem Buspegel liegt ein Fehler vor</td><td>Stuff Check
• Verletzung der Bit Stuffing Regel innerhalb SOF und CRC gilt als Fehler</td></tr>
<tr><td rowspan="2">Botschaftsebene</td><td>Acknowledgement Check
• wird kein dominantes ACK vom Sender erkannt, wird EOF zerstört und ein Error Frame gesendet</td><td>Frame Check (message frame check)
• Überprüfung der Struktur des Rahmens
• Bitfelder und Rahmenlänge werden mit dem vorgegebenen festen Format vergli chen</td></tr>
<tr><td></td><td>CRC Check
• sichert die Information des Rahmens
• berechnet 15 Bit CRC für den Bereich vom SOF bis zum letzten Datenbit</td></tr>
<tr><td></td><td colspan="2">→ Alle Fehler ziehen eine Fehlerbehandlung (error signalling) nach sich und senden somit einen Error Flag, fehlerhafte Botschaft wird abgebrochen
→ Im Anschluss beginnt der Sender seine Botschaft erneut zu senden</td></tr>
</table>

In der folgenden Abbildung ist zu erkennen in welchen Abschnitten eines Daten- oder Remoteframes die jeweiligen Checks bestimmte Fehler erkennen können.

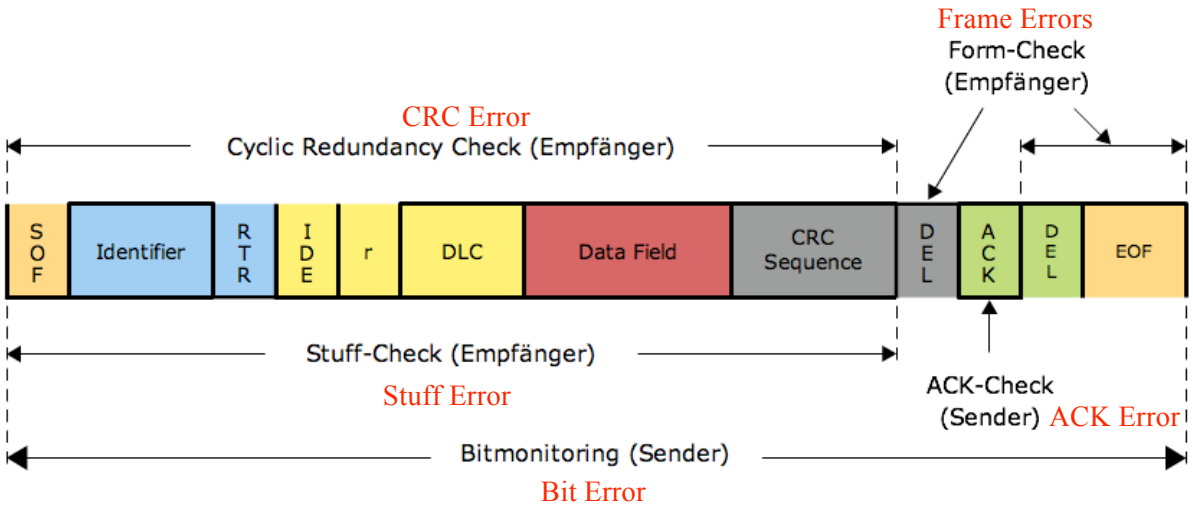


Abb. 27: Bereiche der Fehlererkennung in Daten-/Remoteframe [8]

12.2 Bit-Stuffing

Das sogenannte Bit-Stuffing wird einerseits zur Synchronisierung eingesetzt, und andererseits, um z.B. Leitungsstörungen zu erkennen. Nach der Bit-Stuffing-Regel wird nach fünf aufeinander folgenden gleichwertigen Bits ein invertiertes Bit eingefügt (s. Abb. 28). Die Anzahl der zu übertragenden Bits pro Nachricht wird dadurch erhöht.

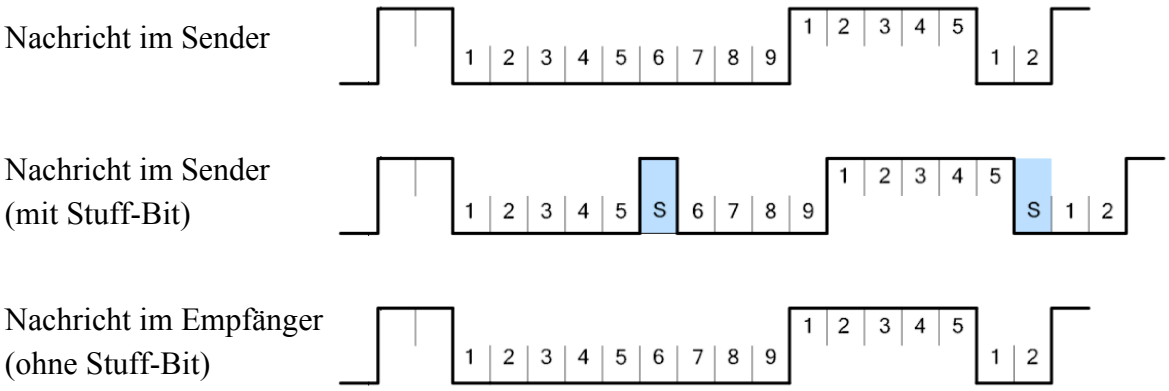


Abb. 28: Beispiel Bit-Stuffing [10]

Kommt die Nachricht mit Stuff-Bits beim Empfänger an, filtert dieser die Stuff-Bits wieder heraus. Die Nachricht bleibt somit unverändert und kann vom Empfänger weiterverarbeitet werden.

Durch das Einfügen von Stuff-Bits kommt es außerdem zur Synchronisierung zwischen den einzelnen Teilnehmern. Bei einer sehr langen Folge von Bits gleichen Pegels könnte es zu einer zu großen Verschiebung der einzelnen Taktsignale der jeweiligen Knoten kommen, sodass eine Nachricht nicht mehr fehlerfrei übertragen werden kann. Aus diesem Grund hat man sich durch einen guten Kompromiss aus Übertragungsrate und Synchronisierung auf das Einfügen eines Stuff-Bits nach dem Aufeinanderfolgen fünf gleichwertiger Bits geeinigt.

12.3 CRC-Check

Mit dem CRC-Check wird eine Hamming Distanz von d = 6 erreicht, d.h. es werden 5 Fehler sicher erkannt. Der CRC-Check wird über die Nutzdaten gebildet (s. Abb. 29).

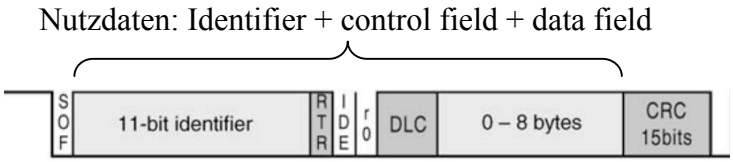
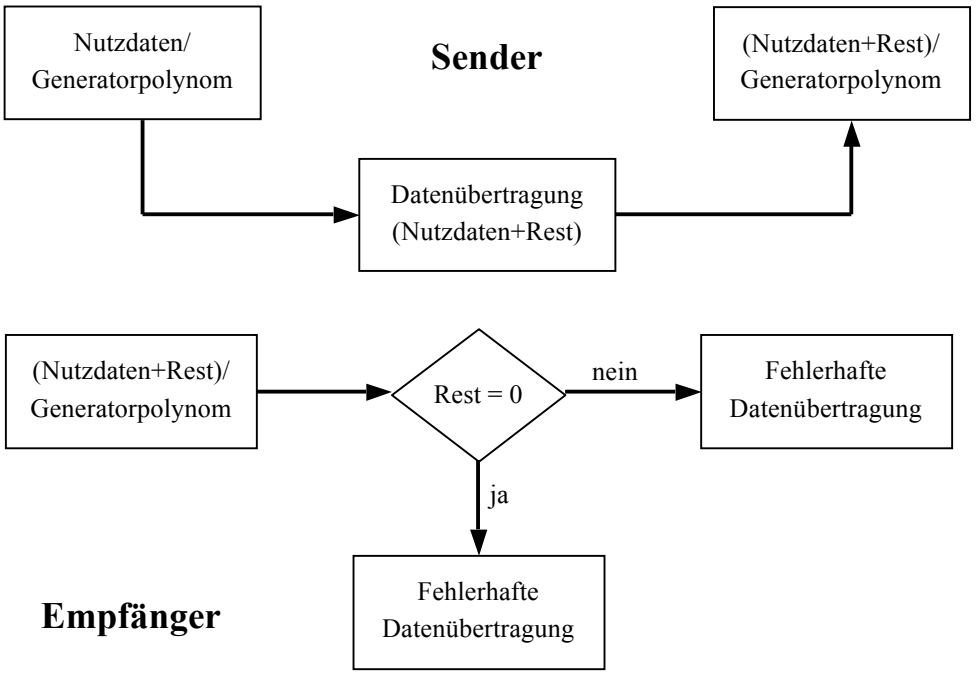


Abb. 29: Relevante Nutzdaten für den CRC-Check [5]

In den beiden folgenden Diagrammen ist der Ablauf des CRC-Checks in Sender und Empfänger zu sehen.



Das verwendetet Generatorpolynom ist 16 Bit lang und in der CAN-Norm (siehe [1]) festgelegt und lautet:

$$X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1 = 1100010110011001$$

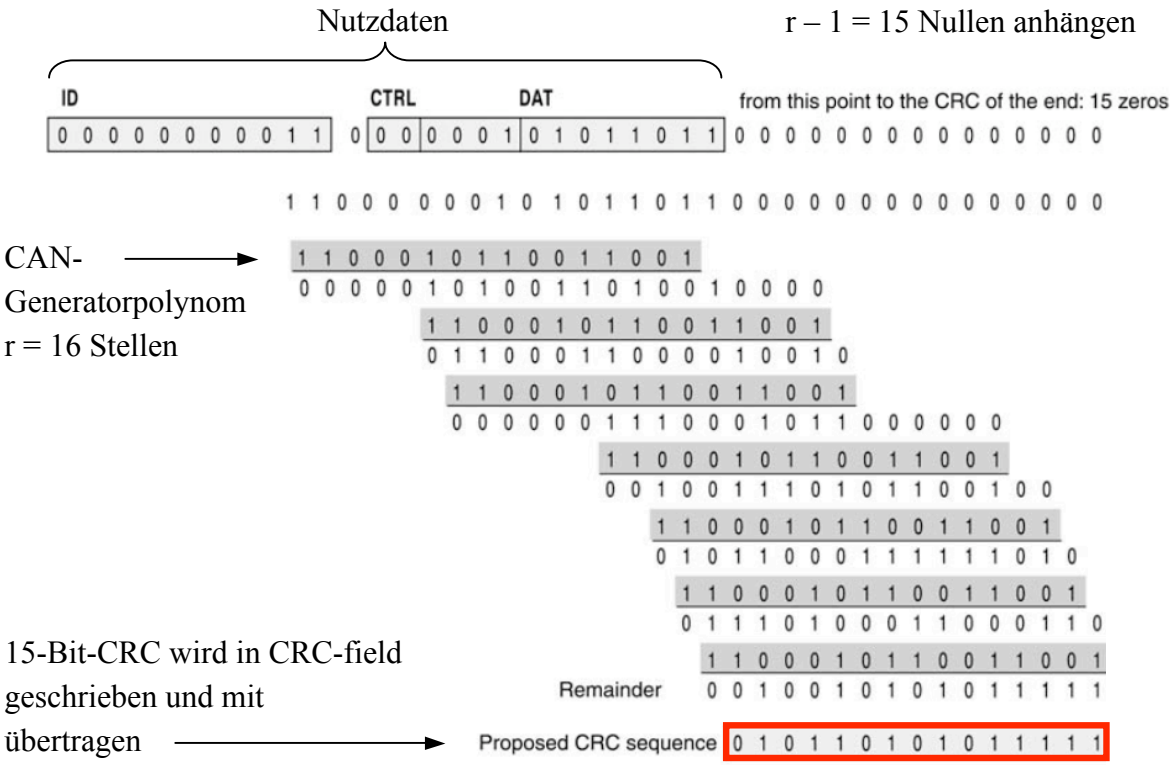


Abb. 30: CRC-Check anhand eines Beispiels [5]

Die 15-Bit CRC-Sequenz wird folgendermaßen gebildet:

1. An die Nutzdaten werden r-1 = 15 Nullen angehängt
2. Das Generatorpolynom wird mit der ersten Stelle an die erste „1“ in den Nutzdaten angelegt und darunter geschrieben
3. Jetzt wird jede Stelle miteinander XOR-verknüpft und das Ergebnis darunter geschrieben
4. Anschließend wird wieder das Generatorpolynom, abschließend mit der ersten „1“ vom vorigen Ergebnis, darunter geschrieben
5. Die Ergebniszeile oberhalb des Generatorpolynoms wird mit Zahlen der Nutzdaten bis zur letzten Stelle des Generatorpolynoms aufgefüllt
6. Jetzt werden die Schritte 3. bis 5. solange wiederholt bis der Rest (Remainder) die gleiche Anzahl an Stellen hat wie das Generatorpolynom
7. Nun wird die erste Stelle des Restes gestrichen und man erhält die gewünschte CRC-Sequenz, welche in das CRC-Feld geschrieben und mit übertragen wird

Abbildungen

Quellen

[1] ISO 11898-1:2003 Road vehicles – Controller area network (Data link layer and physical signalling)

[2] ISO 11898-2:2003 Road vehicles – Controller area network (High-speed medium access unit)

[3] ISO 11898-3:2006 Road vehicles – Controller area network (Low-speed, fault-tolerant, medium-dependent interface)

[4] Vernetzung im Kraftfahrzeug (Robert Bosch GmbH)

[5] Paret, D.: Multiplexed Networks for Embedded Systems CAN, LIN, FlexRay, Safe-by-Wire
ISBN-13: 978-0470034163
Wiley & Sons, 1. Auflage (Juli 2007)

[6] Schnell, G.; Wiedemann, B.: Bussysteme in der Automatisierungs- und Prozesstechnik
ISBN-13: 978-3528465698
Vieweg Verlag; 5. Auflage (15. Januar 2003)

[7] Controller Area Network (CAN) Schedulability Analysis

[8] http://www.vector-elearning.com/vl_index_de,,376493.html (23.10.09)

[9] http://www.automite.de/produkteundtechnologien/canbusanalyseservice/02c4279c770bc5805/index.html (23.10.09)

[10] Bernsdorf, I.: Entwicklung einer Controller Area Network Schnittstelle für Abgassensoren (Diplomarbeit 08/09)

[11] Lawrenz, W.: Controller Area Network Grundlagen und Praxis
ISBN 3-7785-2780-0
Hüthig Verlag, 4. Auflage, Heidelberg 2000

[12] http://www.softing.com/home/en/industrial-automation/products/can-bus/more-can-bus/ (5.12.09)