CAN Bus over Ethernet

R. Löffler, C. Proske, M. Sharkhawy

28. Juni 2014

Inhaltsverzeichnis

1 CAN Bus

1.1 Was ist CAN?

Das Controller Area Network ist ein serielles Bus System und wurde 1983, zur Vernetzung von Steuergeräten in Automobilen, von der Firma Bosch GmbH entwickelt. Doch nicht nur in der Automobilindustrie, sondern auch in der Medizintechnick, Flug- und Raumfahrttechnik sowie in Aufzuganlagen und im Schiffsbau, kann man auf einen CAN Bus stoßen (nach Wik14a). Der Vorteil gegüber der herkömmlichen Verkablung lag darin, das mehrere Knoten über dieselbe Leitung Kommunizieren können. Alle Teilnehmer können Pakete priorisiert mit Kollisionserkennung auf den Bus legen, hierbei wird die Nachricht, nicht der Empfänger adressiert.
Die physikalischen Eigenschaften und der Formale Aufbau des CAN Busses wird in folgenden ISO Normen genau spezifiziert (nach Wik14a):

- ISO 11898-1:2003 Road vehicles — Controller area network — Part 1: Data link layer and physical signalling
- ISO 11898-2:2003 Road vehicles — Controller area network — Part 2: High-speed medium access unit
- ISO 11898-3:2006 Road vehicles — Controller area network — Part 3: Low-speed, fault-tolerant, medium dependent interface
- ISO 11898-4:2004 Road vehicles — Controller area network — Part 4: Time-triggered communication
- ISO 11898-5:2007 Road vehicles — Controller area network — Part 5: High-speed medium access unit with low-power mode
- ISO/NP 11898-6 Road vehicles — Controller area network — Part 6: High-speed medium access unit with selective wake-up functionality

Übertragungsgeschwindigkeiten

Aufgrund der Ausbreitungsgeschwindigkeit der Signale auf dem Bus, ist die Übetragungsgeschwindigkeit von der Leitungslänge abhängig. Die Tabelle 1.1 beinhaltet eine Zuordnung von Leitungslängen und deren maximaler Übertragunsgeschwindigkeit und andersrum.

1.2 Busankoplung/Vernetzung von CAN-Knoten

Der CAN Bus arbeitet nach dem Multi-Master Prinzip, wonach es jedem Knoten möglich ist, mit jedem anderen Knoten zu kommunizieren. Üblicherweise in Linientopologie verbunden, können laut ISO 119898 bis zu 32 Knoten damit verbunden werden. Die Signalübetragung erfolgt symmetrisch.

Auf dem physischen Medium werden die einzelnen Bits über Spannungsdifferenzen abgebildet. Man unterscheidet hier zwischen Highspeed-CAN und Lowspeed-CAN.

Bus Speed	Bus Length
1 Mbit/s	40m
500 kbit/s	100m
125 kbit/s	500m
50 kbit/s	1 km

Tabelle 1: Übertragungsgeschwindigkeiten

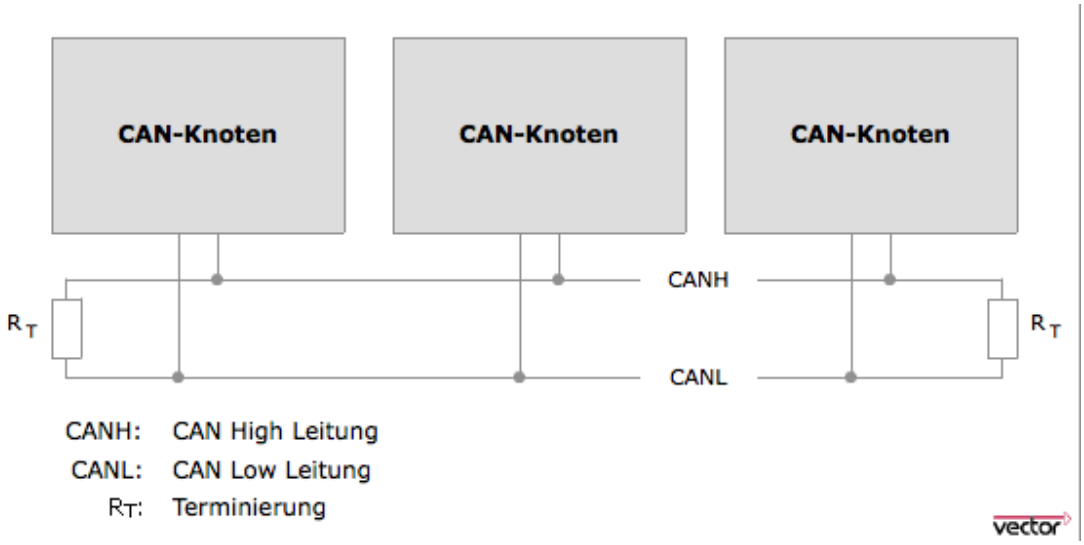

Abbildung 1: Vernetzung von CAN Knoten (Gmb14)

1.3 CAN-Bus Pegel

Beim CAN-Bus erfolgt die Übertragung der Bits über 2 Pegel:

- Dominant
- Rezessiv

Wenn beide Pegel gleichzeitig gesendet werden, so überschreibt immer der dominante Pegel den rezessiven Pegel.

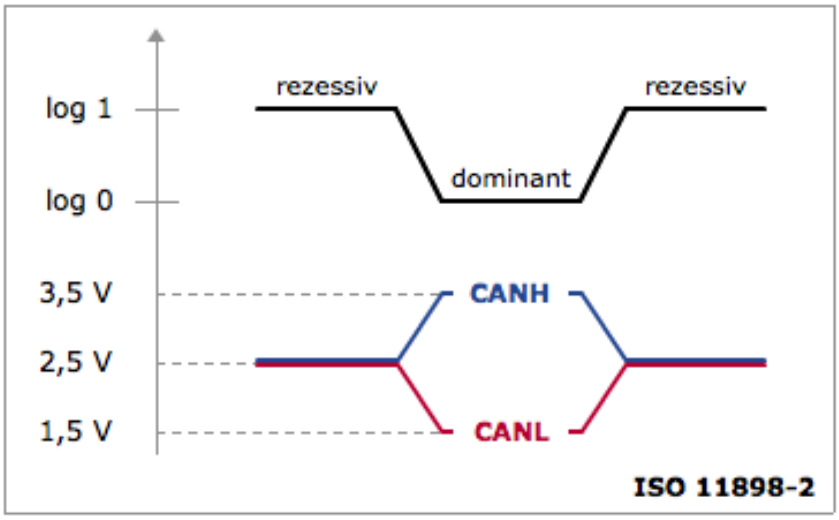

Abbildung 2: Highspeed-CAN Buspegel (Gmb14)

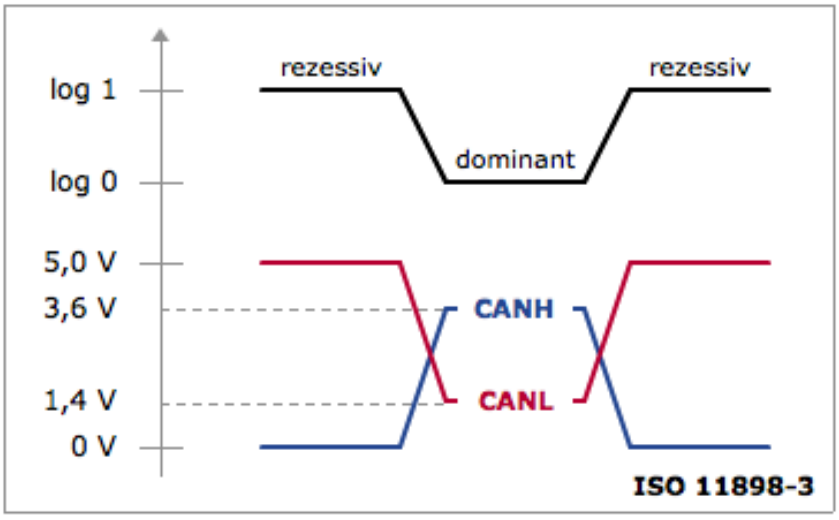

Abbildung 3: Loewspeed-CAN Buspegel (Gmb14)

1.4 Bit Stuffing

Bit-Stuffing beim CAN-Bus bedeutet, dass nach 5 Bits mit gleichem Pegel ein »Stuff Bit« mit inversem Pegel eingefügt wird. Die Empfänger filtern diese Stuff-Bits, dann nach dem gleichen Schema wieder heraus.
Bit stuffing wird einerseits eingesetzt, weil Bitfolgen mit mehr als 5 Bits mit gleichem Pegel für Steuerungszwecke eingesetzt werden, und andererseits auch wegen der NRZ-Kodierung. Das heißt dass der Pegel bei zB 4 rezessiven Bits gleich bleibt. Wenn dann beispielsweise 10 gleiche Bits übertragen werden, so kann der Empfänger möglicherweise nicht mehr unterscheiden, ob jetzt 10 oder 11 Bits übertragen wurden.

1.5 CAN Frames

Beim CAN-Bus gibt es 4 unterschiedliche Arten von Frames:

- Daten-Frame
- Remote-Frame
- Error-Frame
- Overload-Frame

Daten Frame

Die Daten-Frames dienen dem Transport von Daten und können bis zu 8 Byte an Nutzdaten enthalten. Bei den Daten-Frames kann zwischen Standard Format und Extended Format unterschieden werden, die sich hauptsächlich in der Länge des Identifiers unterscheiden. Der Aufbau eines Daten Frames kann aus Abbildung 4 entnommen werden.

Gestarted wird ein Frame durch das **SOF** (start-of-frame) bestehend aus einem dominanten Bit, das der Synchronisation aller CAN-Knoten dient. Darauf folgt gleich das Arbitrationsfeld und dem Kontrollfeld. Abhängig vom Format bestehen diese aus **Basis Identifier** (11 Bits), **Extended Identifier** (18 Bit), einem **RTR**-Bit (remote transmission request; bei Remote-Frames rezessiv) sowie einem **SRR**-Bit (substitude-remote-request; rezessiv). Ist das **IDE**-Bit (identifier extension) rezessiv, so handelt es sich um das Extended Format, das somit immer Nachrang über dem Standard Format hat.
Im Kontrollfeld befindet sich neben 1-2 reservierten (aktuell nicht verwendeten) Bits der aus 4 Bits bestehende **DLC** (data length code), der die Länge des nachfolgenden **Datenfeldes** angibt.
Wie bereits erwähnt können mit den Daten-Frames bis zu 8 Byte, also 64 Bit, übertragen werden. dies kann jedoch nur in einheiten von 8 Bits geschehen. Anschließend an das Datenfeld ist das CRC-Feld (cyclic redundancy check; Prüfsummenfeld) bestehend aus 16 Bit (15 Bit + 1 rezessives Delimiter-Bit). Des weiteren enthält ein Daten-Frame ein **ACK**-Feld (Acknowledge). Dieses wird verwendet, um den Empfang eines korrekten

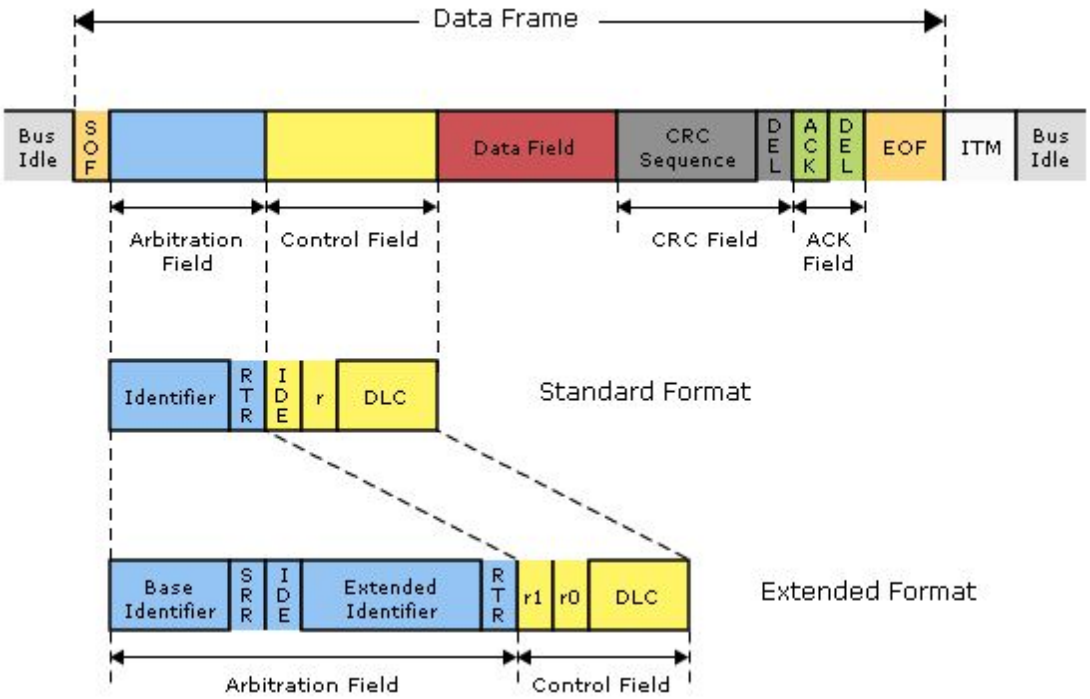

Abbildung 4: Aufbau eines Daten-Frames (Gua10)

Frames zu bestätigen. Der Sender sendet dafür ein rezessives Bit. Jeder Empfänger der keinen Fehler feststellen konnte, setzt einen dominanten Pegel und überschreibt somit den rezessiven des Senders. Im Falle einer negativen Quittierung (rezessiver Pegel) muss der Knoten, der den Fehler erkannt hat, nach dem ACK-Delimiter einen Error-Frame aussenden.
Das Ende des Frames wird mit 7 rezessiven Bits dem **EOF** (end of frame) angezeigt. Anschließend an den Frame muss ein ITM-Package (intermission oder inter-frame-space) bestehend aus mindestens 3 rezessiven Bits gesendet werden.

Remote Frame

Mit Hilfe der Remote-Frames können Daten-Frames von anderen Teilnehmern angefordert werden. Der Frame unterscheidet sich vom Daten-Frame durch ein rezessives Bit im »RTR«-Slot, wodurch Remote-Frames im Falle einer Kollision immer Nachrang gegenüber den Daten-Frames haben, und durch ein Fehlen des Datenfeldes.

Error Frame

Erkennt ein CAN-Knoten einen Fehler, so sendet er einen Error-Frame an alle anderen CAN-Knoten im Netzwerk. Bei diesen Frames wird das Bit-Stuffing bewusst ignoriert. Der Error-Frame besteht aus 2 Feldern, den Error-Flags und dem Error-Delimiter (8 rezessive Bits). Die Error-Flags sind abhängig vom Modus in dem sich ein CAN-knoten befindet.

Ist der Knoten im Fehler-Status *»error active«* so setzt er die Error-Flags auf 6 dominante Bits. Befindet er sich hingegen im Fehler-Status *»error passive«* so sendet er 6 rezessive Bits.

Overload Frame

Overload-Frames werden als Zwangspause zwischen Daten- und Remote-Frames genutzt. Dabei hat ein Overload-Frame das gleiche Format wie ein Active-Error-Frame. Wie in Abbildung 5 ersichtlich ist, besteht der Overload-Frame ebenfalls aus 2 Feldern: dem Overload-Flag (6 dominante Bits) und dem Overload-Delimiter (8 rezessive Bits). Ein Overload-Frame kann jedoch nur während eines Interframespaces gesendet werden. Dies ermöglicht die Unterscheidung von den Error-Frames, die während der Übertragung einer Nachricht gesendet werden, sobald eben ein Fehler erkannt wurde.

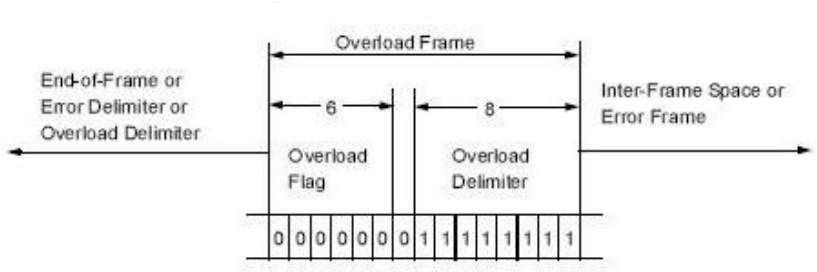

Abbildung 5: Aufbau eines Overload-Frames (CBM09)

1.6 Zeichenkodierung

Auf der Busleitung werden die einzelnen Bits der Pakete über den Non-Return-to-Zero Code (NRZ) codiert. Hierbei hat jeder Wert eines Bit einen konstanten Zusatand auf der Leitung (siehe Abb. 6).

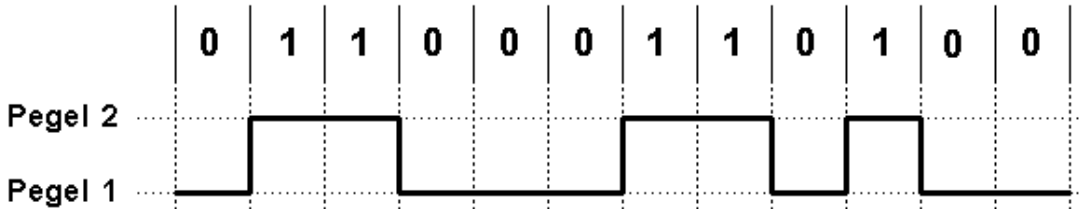

Abbildung 6: NRZ-Code (Wik14b)

Probleme bei dieser Codierung können auftreten, wenn mehrere gleiche Symbole in Folge gesendet werden. Abhilfe schafft hier Bit-Struffing, näheres dazu in Kapitel 1.9.

Als Vergleich zur NRZ Codierung sollte man noch den Manchester-Code erwähnen, bei dem jeweils eine steigende, bzw. fallende Taktflanke einen Bit-Wert repräsentiert (Abb. 7). Ein Bitstuffing ist hier nicht nötig, da auch gleiche aufeinanderfolgende Bits durch eine Taktflanke reproduzierbar sind.

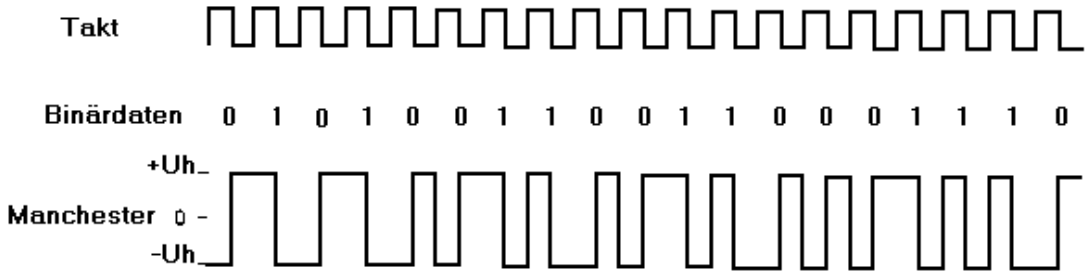

Abbildung 7: Manchester-Code (wis09)

1.7 Synchronisatzion

Da die CAN Knoten nur eine gemeinsame Baud-Rate, jedoch keinen gemeinsamen Takt besitzen, wird der Daten-Frame selbst zur Synchronization verwendet. Es gibt eine Hard- und eine Soft-Synchronization, wobei die Hard-Synchronization die fallende Flanke des Startbits zur Synchronizierung verwendet. Alle nachfolgenden abfallenden Flanken werden zur Soft-Synchronizierung verwendet. Durch die Soft-Synchronizierung kann sich der Takt nur noch in einem spezifizierten Rahmen verschieben (nach DM).

1.8 Buszugriffsverfahren

Durch die Multi-Master-Architektur des CAN-Bus kann es passieren, dass mehrere Knoten gleichzeitig veruschen Daten über den Bus zu senden denn jeder Knoten hat das Recht zu jederzeit und ohne Absprache Daten zu senden. Um dieses Szenario zu vermeiden, bzw. aufzulösen verwendet man CMSA/CA (Carrier Sense Multiple Access/Collision Avoidance). Bei diesem Verfahren “lauscht” jeder Teilnehmer vor dem Senden ob der Bus frei ist.

Sollte es durch simultanen Zugriff dennoch zu einer Collision kommen, wird durch eine bitweise Arbitrierung. Über den Basis Identifier im Arbitrationsfeld des Pakets wird dann entschieden, wer den Zugriff auf den Bus bekommt. Entscheidend ist hier, welche Nachricht die höhere Priorität besitzt, bzw. als erstes ein rezessives Bit sendet.

Abbildung 8 zeigt ein Beispiel der bitweisen Arbitrieung. An Punkt (1) beginnen beide Knoten gleichzeitig mit dem senden Ihres Identifieres. Node 2 sendet an der 5. Bistelle des Identifiers ein dominantes Bit, wohingegen Node 1 ein rezessives Bit sendet. An diesem Punkt (2) beendet Node 1 seine Übetragung und der Bus gehört Node 2.

Der Identifier addressiert die Nachricht und dient gleichzeitig als Priorisierung. Jeder Identifier kann nur einen Sender, jedoch mehrere Empfänger haben.

1.9 Sicherungsmechanismen und Fehlererkennung

Für die Fehlererkennung stehen gleich mehrere Verfahren zur Verfügung:

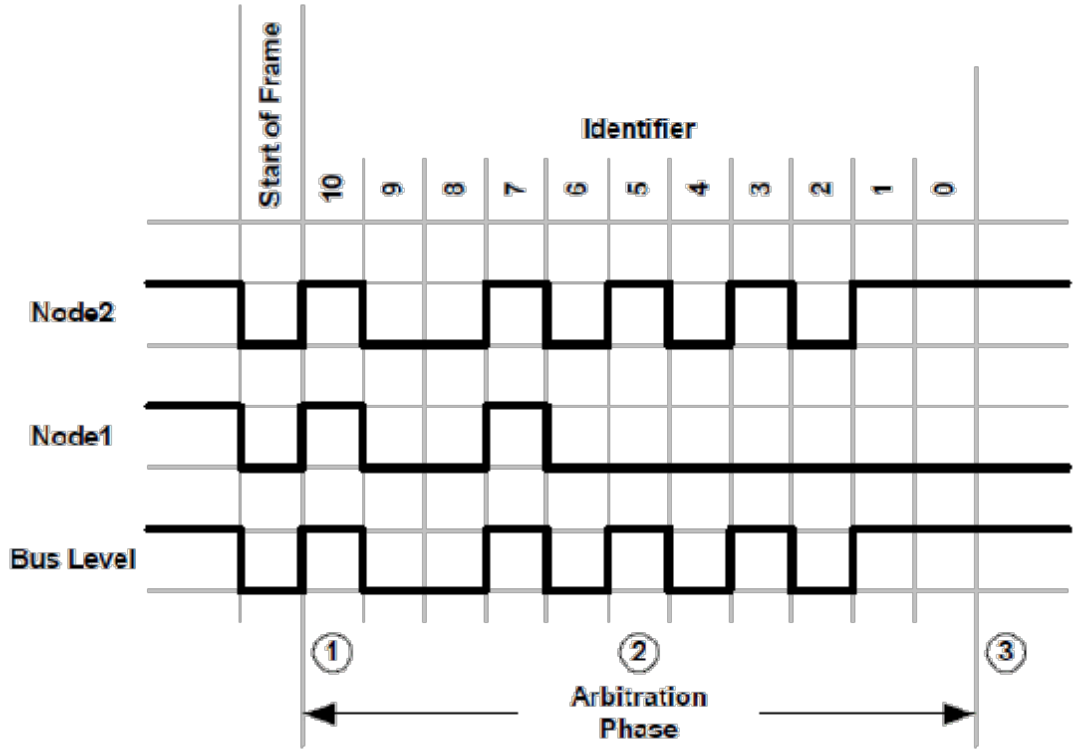

Abbildung 8: Beispiel: Bitweise Arbitrierung (Rat09)

Bitmonitoring

Dies ist Aufgabe des Senders. Er vergleicht ob das gesendete Bit mit dem Buspegel übereinstimmt. Sollten diese nicht übereinstimmen wird mit die Fehlerbehandung eingeleitet. Auf das Arbitration Field und den ACK-Slot findet dieses Verfahren keine Anwendung.

Stuff Check

Dies ist Aufgabe des Empfängers. Sollte diese sechs homogene Bits lesen, liegt ein Bitstuffing-Fehler vor (siehe Kapitel 1.4). Auch diese zieht eine Fehlerbehandlung nach sich.

Form Check

Auch dies ist Aufgabe des Empfängers. Er vergleicht den ankommenden Bitstrom mit dem Datenformat. Ein dominantes Delimiter-Bit (CRC oder ACK) oder ein dominantes Bit im EOF signalisiert einen Formatfehler und leitet eine Fehlerbehandlung ein.

Cyclic Redundancy Check

Der Empfänger überprüft die ankommenden Bits mit der CRC Sequence. Ein Fehler leitet eine Fehlerbehandlung ein.

ACK Check

Sollte das im ACK Slot gesendete rezessive Bit nicht von mindestens einem Empfänger überschrieben werden, liegt ein Bestätigungsfehler vor, der eine Fehlerbehandlung nach sich zieht.

Abbildung 9 verdeutlicht das Auftreten der jeweiligen Fehler in den einzelnen Bereichen einer Nachricht.

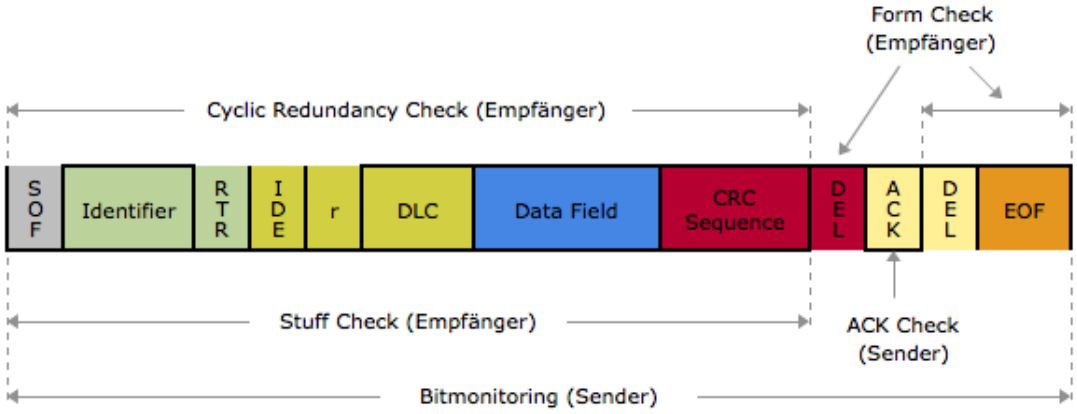

Abbildung 9: Fehlererkennung (Gmb14)

Fehlerbehandlung

Sollte ein Fehler erkannt werden, sendet der jeweilige CAN-Knoten ein Fehlersignal (Error-Flag). Dieser besteht aus sechs dominanten Bits, welcher absichtlich die Bitstuffingregel verletzt. Durch den Error-Flag werden alle Teilnehmer des CAN über einen Fehler informiert (Bitstuffingfehler).

1.10 Fehlerverfolgung

Um die Netzweite Datenkonsitenz sicherzustellen, können, wie oben schon erwähnt, alle Knoten die einen Fehler erkennen die Nachricht auf dem Bus abbrechen. Hierbei kann es auch vorkommen das dies irrtümlicherweise geschieht.

Jeder CAN-Controller besitzt eine Sendefehlerzähler (TEC, Transmit Error Counter) und einen Empfängerfehlerzähler (REC, Receive Error Counter). Nach jeder erfolgreichen Übertragung eines Data oder Remote Frames veringert sich der jeweilige Fehlerzähler (TEC=TEC-1; REC=REC-1). Enteckt ein Knoten einen Fehler und sendet einen Error Flag wird der entsprechende Fehlerzähler erhöht. Hierbei gilt für den Sender: TEC=TEC+8 für den fehlererkennenden Empfänger: REC=REC+1. Für den Fehler verursachenden Empfänger gilt zudem REC=REC+8.

Je nach höhe des Fehlerzählers gibt es verschidene Zustände des CAN-Knotens. Zu Beginn befinden sich alle Knoten im “Active Error”-State. In diesem Modus werden vom CAN-Knoten Error Flags beim erkennen eines Fehlers gesendet. Sollte einer der Fehlerzähler (TEC oder REC) über 127 steigen, fällt der CAN-Knoten in den “Error Passive”-State. In deisem Zustand können entdeckte Fehler lediglich mit sechs homogenen rezessiven Bits signalisiert werden. Fehlererkennenden Empfängern wird so die Möglichkeit genommen, entdeckte Fehler zu globalisieren. Knoten, die sich im „Error Passive“-State befinden, müssen beim Senden von zwei aufeinanderfolgenden Data oder Remote Frames zusätzlich die „Suspend Transmission Time“ (8 Bits) abwarten.

Sollte der TEC über 255 steigen geht der CAN-Knoten in den “Bus Off”-State und trennt sich damit vom Bus ab. Der Zustand Bus Off kann nur durch Eingriff des Host (mit 128 x 11 Bit Zwangswartezeit) oder per Hardware-Reset verlassen werden.

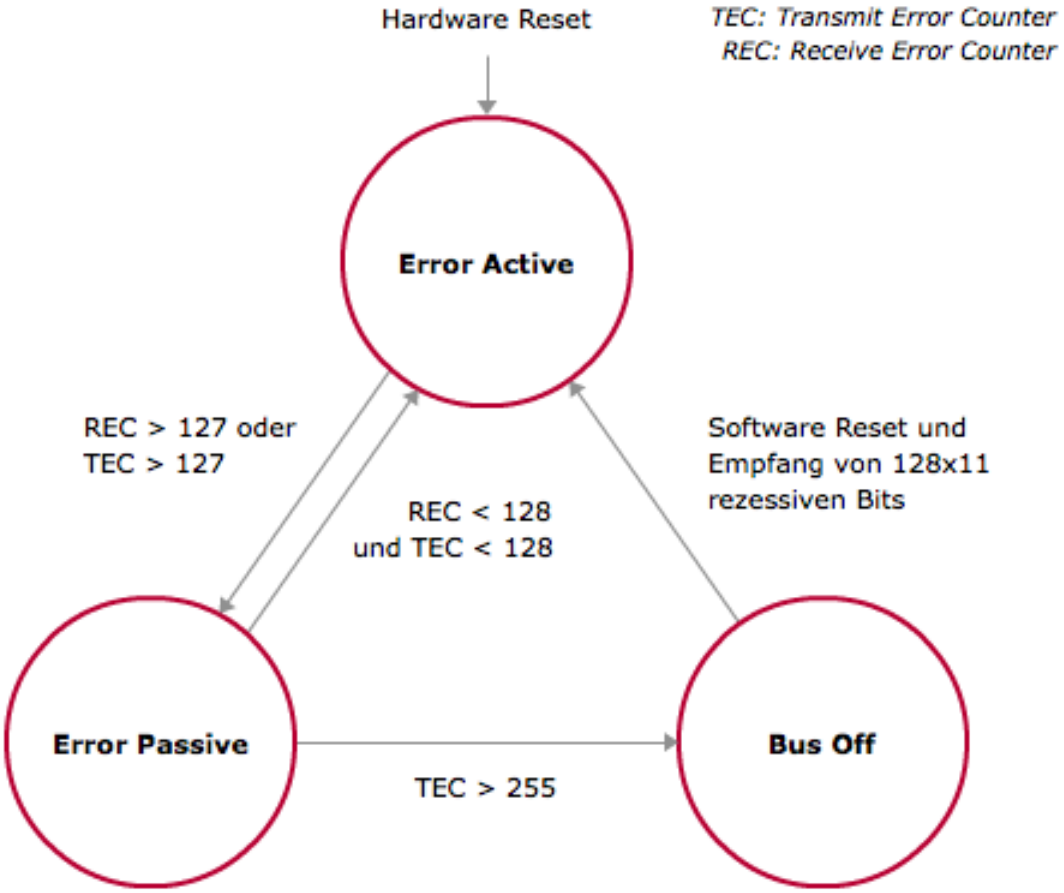

Abbildung 10: Fehlerverfolgung (Gmb14)

2 CAN Bus over Ethernet

2.1 Probleme

2.2 CAN Schnittstelle

CAN Schnittstellen erlauben es, eine CAN-Bus Struktur an Ethernet Topologien anzubinden. Dies ermöglicht anderen Netzwerkgeräten mit den Teilnehmern im CAN-Bus und vice versa zu kommunizieren. Typischerweise wird eine CAN Schnittstelle verwendet, um den CAN Bus über Ethernet mit einem PC zu verbinden. Dies ermöglicht einen flexiblen Zugriff auf die CAN-Systeme über LAN oder sogar Internet. Die CAN Schnittstelle verhält sich dabei wie ein normaler CAN Knoten. (ele14)

Die CAN Nachrichten werden vom CAN Gateway in Ethernet Packete eingepackt und dann über die Ethernet Verbindung gesendet. Dabei gehen jedoch wichtige Eigenschaften des CAN-Busses verloren. Zum Beispiel wird das ACK-Feld im CAN-Frame nicht mehr vom eigentlichen Empfänger, dem PC, sondern direkt von der CAN Schnittstelle gesetzt. Dadurch kann nicht sichergestellt werden, ob die Nachricht auch tatsächlich fehlerfrei beim PC angekommen ist. Abbildung 11 zeigt die Topologie des Netzwerkes bestehend aus CAN-Netzwerk, CAN-Schnittstelle und Ethernet.

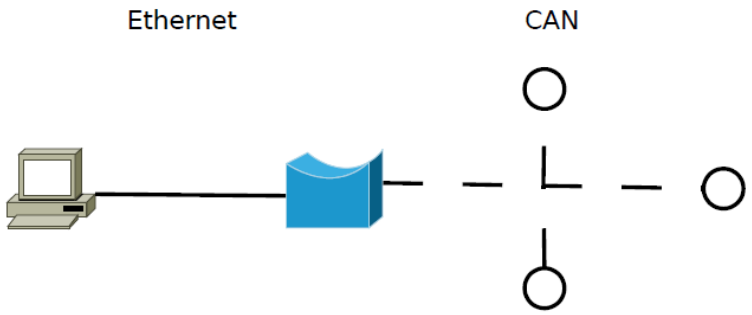

Abbildung 11: CAN-Schnittstelle

2.3 CAN Ethernet Bridge

Die CAN Ethernet Bridge ermöglicht es, mehrere CAN Netzwerke miteinander über Ethernet (LAN oder Internet) zu verbinden. Abbildung 12 zeigt zwei CAN Busse, die über Ethernet miteinander verbunden sind. Die CAN Bridges verhalten sich hierbei wieder wie CAN Teilnehmer. Auch hier gehen wichtige Eigenschaften des CAN-Busses, wie beispielsweise das Setzen des ACK-Feldes durch den Empfänger oder das eigentliche Zeitverhalten im jeweiligen CAN-Bus, verloren. Denn das ACK-Feld wird direkt von der Bridge und nicht von den eigentlichen Empfängern gesetzt.

Vorteil einer solchen Entkopplung der CAN Netzwerke ist, dass eine Filterung der gesendeten Nachrichten möglich ist, und somit der Datenfluss zwischen den CAN Netzwerken beeinflussbar ist. So können beispielsweise Fehler abgefangen und begrenzt werden. Ein weiterer Vorteil einer Entkopplung ist, dass das Zeitverhalten der Busse getrennt betrachtet werden kann und für Analysen anstatt eines riesigen Busses nur einen Teil des Busses betrachtet werden muss.

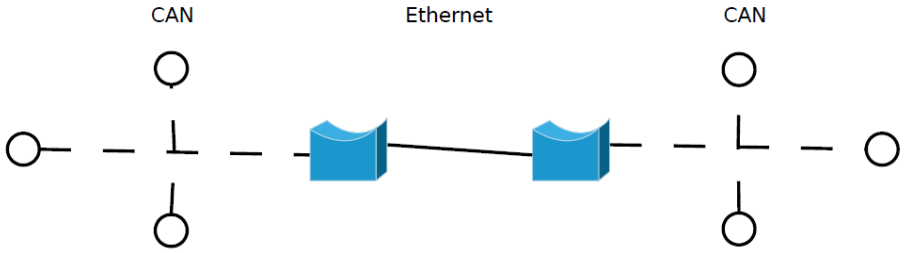

Abbildung 12: CAN Ethernet Bridge

2.4 Transparente Ethernet Bridge

2.5 Bestehende Lösungen

Abbildungsverzeichnis

Literatur

[CBM09] CAN BUS MESSAGE FRAMES - Overload Frame,Interframe Space. http://rs232-rs485.blogspot.co.at/2009/11/can-bus-message-frames-overload.html. Version: 11 2009, Abruf: 20.06.2014

[DM] DSR Management, Inc.: Bit Synchronization on Controller Area Network (CAN) Bus. http://www.dsrminc.com/whitepaper/bit_synchronization_on_controller_area_network__rev_1_.pdf, Abruf: 26.06.2014

[ele14] electronics, SYS T.: CAN-Ethernet Gateway. http://www.systec-electronic.com/de/produkte/industrielle-kommunikation/schnittstellen-und-gateways/can-ethernet-gateway-v2. Version: 6 2014, Abruf: 27.06.2014

[Gmb14] GmbH, Vector I.: Einführung in CAN. https://elearning.vector.com/index.php?seite=vl_can_introduction_de&root=376493&wbt_ls_kapitel_id=504193&wbt_ls_seite_id=504213&d=yes. Version: 2014, Abruf: 22.06.2014

[Gua10] Guardigli, Marco: Hacking Your Car. http://marco.guardigli.it/2010/10/hacking-your-car.html. Version: 10 2010, Abruf: 20.06.2014

[Rat09] Ratheesh: IMPLEMANTATION OF CAN PROTOCOL. http://dc387.4shared.com/doc/4g-s6_G0/preview.html. Version: 12 2009, Abruf: 28.06.2014

[Wik14a] Wikipedia: Controller Area Network. http://de.wikipedia.org/wiki/Controller_Area_Network. Version: 6 2014, Abruf: 21.06.2014

[Wik14b] Wikipedia: Non Return to Zero. http://de.wikipedia.org/wiki/Non_Return_to_Zero. Version: 1 2014, Abruf: 26.06.2014

[wis09] wissen.de rn: Manchester Codierung. http://www.rn-wissen.de/index.php/Manchester-_Codierung. Version: 10 2009, Abruf: 26.06.2014