

CAN Bus over Ethernet

R. Löffler, C. Proske, M. Sharkhawy

26. Juni 2014

Inhaltsverzeichnis

1	CAN Bus	2
1.1	Was ist CAN?	2
1.2	Busankopplung/Vernetzung von CAN-Knoten	2
1.3	CAN-Bus Pegel	4
1.4	Bit Stuffing	5
1.5	CAN Frames	5
1.6	Zeichenkodierung	7
1.7	Synchronisation	8
1.8	Buszugriffsverfahren	8
1.9	Fehlermanagement	8
1.10	Sicherungsmechanismen und Fehlererkennung	8
2	CAN Bus over Ethernet	9
2.1	Probleme	9
2.2	CAN Schnittstelle	9
2.3	CAN Ethernet Bridge	9
2.4	Bestehende Lösungen	9
	Abbildungen	10
	Literatur	11

1 CAN Bus

1.1 Was ist CAN?

Das Controller Area Network ist ein seriellcs 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 Medizintechnik, Flug- und Raumfahrttechnik sowie in Aufzuganlagen und im Schiffsbau, kann man auf einen CAN Bus stoßen (nach Wik14a). Der Vorteil gegenüber 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 Übertragungsgeschwindigkeit von der Leitungslänge abhängig. Die Tabelle 1.1 beinhaltet eine Zuordnung von Leitungslängen und deren maximaler Übertragungsgeschwindigkeit und andersrum.

1.2 Busankopplung/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 11898 bis zu 32 Knoten damit verbunden werden. Die Signalübertragung 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

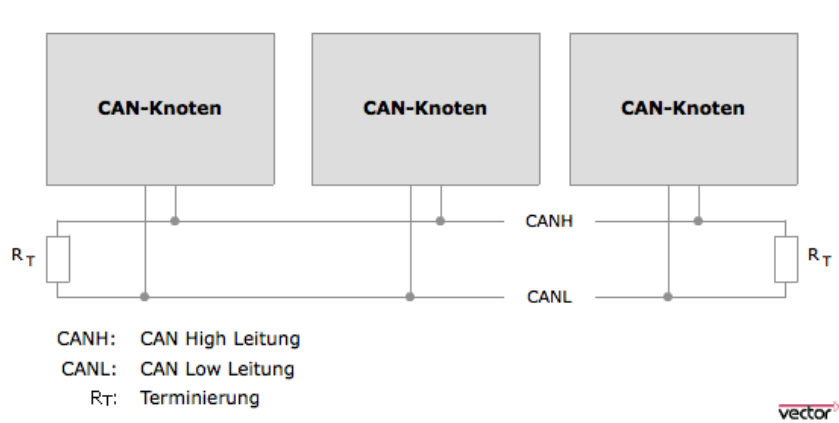


Abbildung 1: Vernetzung von CAN Knoten (Gmb14)

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.

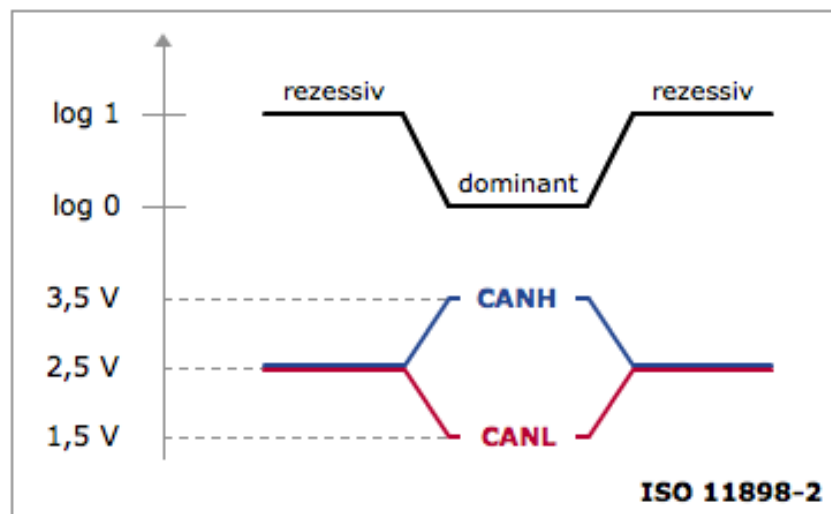


Abbildung 2: Highspeed-CAN Buspegel (Gmb14)

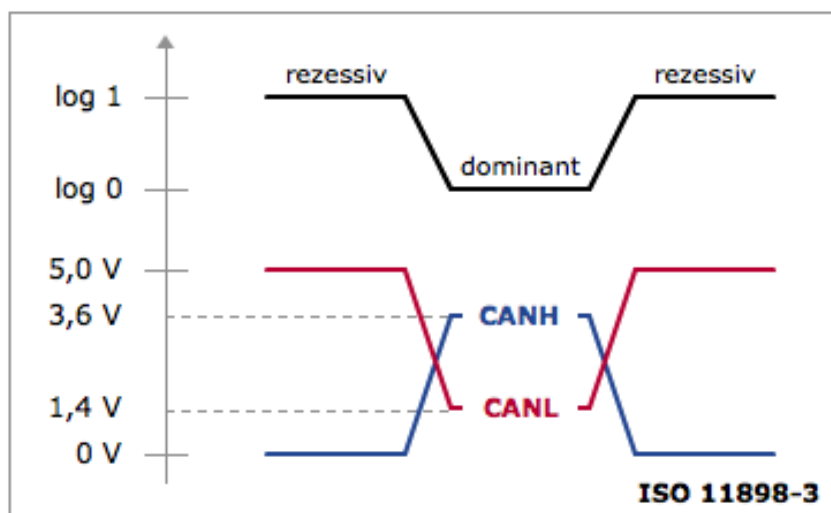


Abbildung 3: Loowspeed-CAN Buspegel (Gmb14)

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.

Gestartet 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 Arbitrationfeld 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 (substitute-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

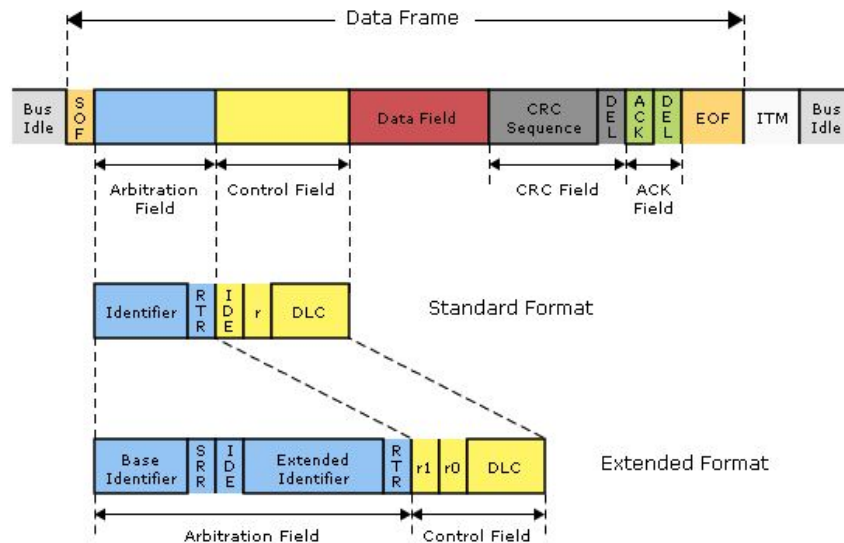


Abbildung 4: Aufbau eines Daten-Frames (Gua10)

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.

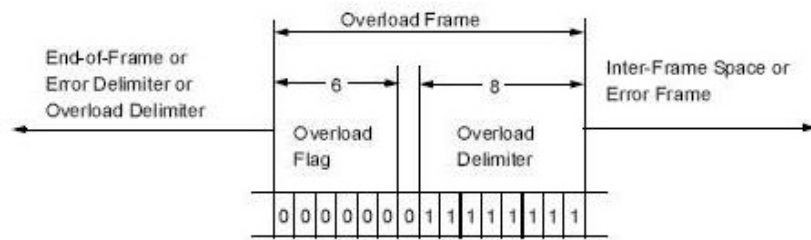


Abbildung 5: Aufbau eines Overload-Frames (CBM09)

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 Zustand auf der Leitung (siehe Abb. 6).

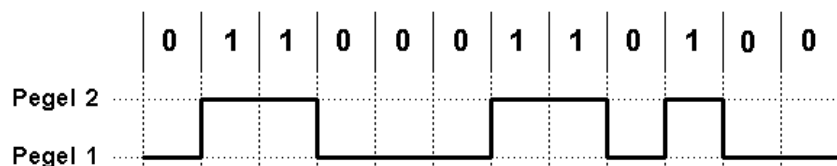


Abbildung 6: NRZ-Code (Wik14b)

Probleme bei dieser Codierung können auftreten, wenn mehrere gleiche Symbole in Folge gesendet werden. Abhilfe schafft hier Bit-Struffing, näheres dazu in Kapitel 1.10.

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.

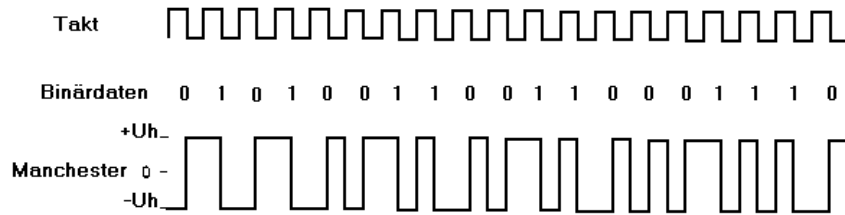


Abbildung 7: Manchester-Code (wis09)

1.7 Synchronisation

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 Synchronisierung verwendet. Alle nachfolgenden abfallenden Flanken werden zur Soft-Synchronisierung verwendet. Durch die Soft-Synchronisierung kann sich der Takt nur noch in einem spezifizierten Rahmen verschieben (nach (nach DM)).

1.8 Buszugriffsverfahren

1.9 Fehlermanagement

1.10 Sicherungsmechanismen und Fehlererkennung

2 CAN Bus over Ethernet

2.1 Probleme

2.2 CAN Schnittstelle

2.3 CAN Ethernet Bridge

2.4 Bestehende Lösungen

Abbildungsverzeichnis

1	Vernetzung von CAN Knoten (Gmb14)	3
2	Highspeed-CAN Buspegel (Gmb14)	4
3	Loowspeed-CAN Buspegel (Gmb14)	4
4	Aufbau eines Daten-Frames (Gua10)	6
5	Aufbau eines Overload-Frames (CBM09)	7
6	NRZ-Code (Wik14b)	7
7	Manchester-Code (wis09)	8

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
- [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
- [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