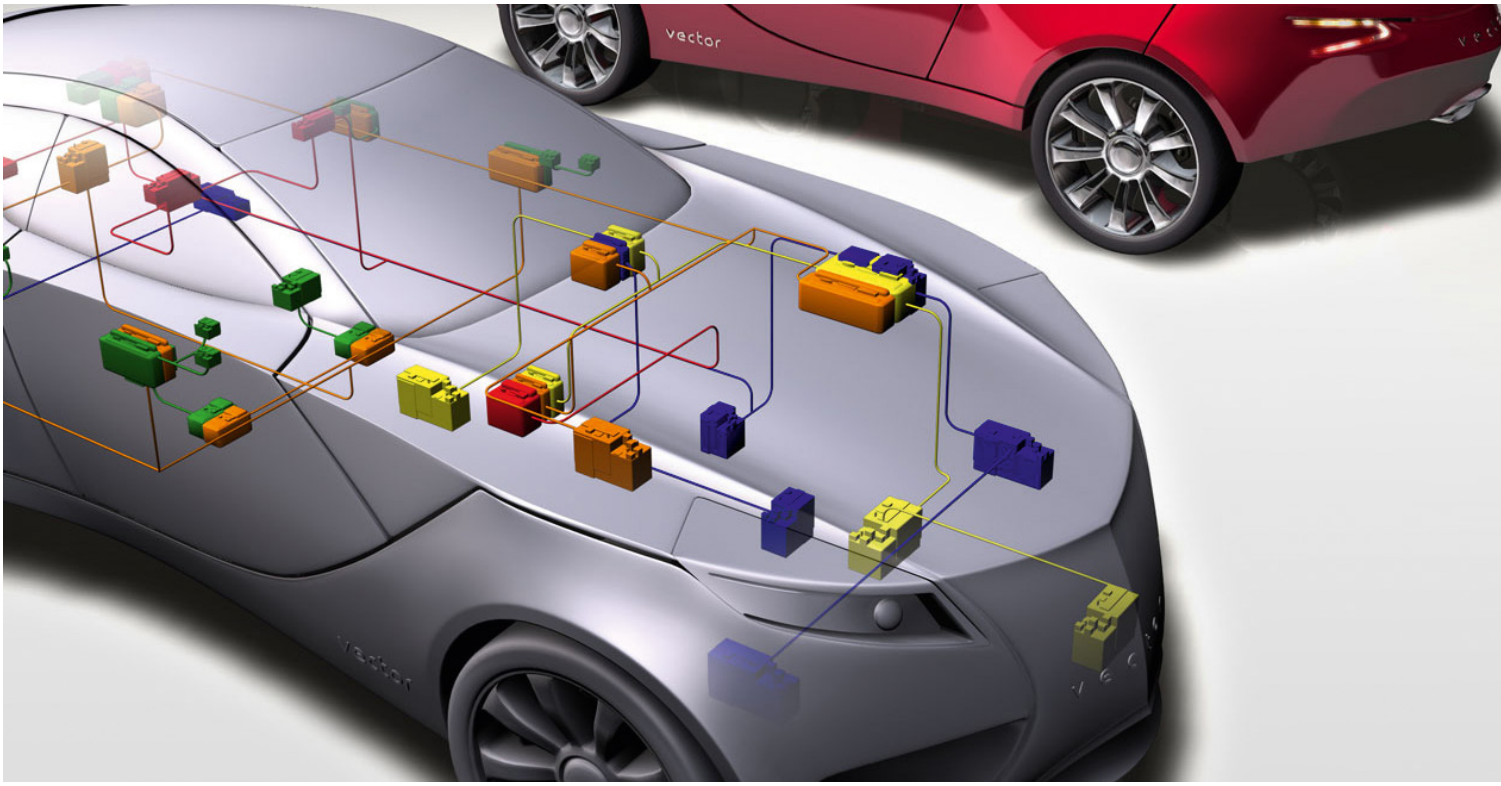


# Datenkommunikation im Automobil Teil 2:

## Sicherer Datenaustausch mit CAN

Wie im Teil 1 unserer Artikelreihe dargelegt, erfordern die komplexer werdenden elektronischen Systeme im Automobil ein höheres Maß an Datenaustausch zwischen den Steuergeräten. Damit dieser mit hinreichender Sicherheit und Schnelligkeit gewährleistet werden kann, wurde der CAN-Bus entwickelt. CAN steht für Controller Area Network, und wie der Name bereits vermuten lässt, kann der CAN-Bus ein größeres Gebiet vernetzen, er kann bis zu mehreren Kilometer lang sein.

Der von Bosch [1] entwickelte CAN-Bus ist seit 1993 genormt und liegt als ISO-Norm 11898 vor **(Bild 1)**. Sie ist in mehrere Teile gegliedert. Der erste Teil umfasst das CAN-Protokoll und deckt vollständig den Data Link Layer (Framing, Adressierung, Buszugriff, Datensicherung) und die physikalische Signalcodierung als Teil des Physical Layer des ISO 7498 Referenzmodells der Datenkommunikation ab, dem sogenannten OSI-Schichtenmodell (OSI: Open Systems Interconnection). Zur Abwicklung des CAN-Protokolls wurde der sogenannte CAN-Controller entwickelt.

Die Teile 2 und 3 der ISO-Norm 11898 beschreiben zwei Ausprägungen des Physical Layer, nämlich CAN-High-Speed und CAN-Low-Speed. Letzterer wird oft auch Fault-Tolerant-CAN genannt, weil er beim Bruch einer seiner beiden Drähte weiter funktioniert, wenn auch mit geringerer Sicherheit. Die beiden Teile 2 und 3 decken den Physical Layer des ISO-Referenzmodels ab, darunter die physikalische Busankopplung, Datenraten und die Spannungspegel auf den beiden CAN-Leitungen.

CAN nutzt Differenzsignalübertragung, welche die Störempfindlichkeit verbessert und zwei Kommunikationsleitungen erfordert (CAN-High- und CAN-Low-Leitung), die zur Vermeidung von Reflexionen an beiden Enden mit dem Wellenwiderstand $R_T$ von 120 Ω abgeschlossen werden. Darüber haben wir in Teil 1 bereits geschrieben.

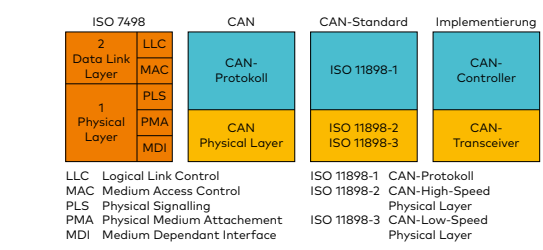


**Bild 1:** CAN-Standard

CAN-High-Speed kommt vor allem in Antriebs- und Fahrwerksapplikationen zum Einsatz. Man realisiert ihn im Wesentlichen durch den CAN-High-Speed-Transceiver, der eine maximale Datenrate von 1 Mbit/s unterstützt. Der CAN-Low-Speed Physical Layer wurde hauptsächlich im Komfortbereich eingesetzt. Dort wurde er in Türen, Sitze und Schiebedächer verlegt, wo er Biegungen unterliegt, wodurch Leitungen brechen können. Man nutzte hier den fehlertoleranten CAN-Low-Speed-Transceiver mit einer maximalen Datenrate von 125 kbit/s, der auch mit einem Draht auskommen kann. Inzwischen wird dieser CAN-Bus-Typ kaum noch verbaut.

Die CAN-Schnittstelle besteht also aus einem CAN-Controller und einem CAN-Transceiver **(Bild 2)**.
Der CAN-Controller wickelt das CAN-Protokoll ab, der CAN-Transceiver übernimmt die Aufgabe, den CAN-Controller physikalisch an die beiden CAN-Leitungen anzukoppeln und die Spannungspegel auf den beiden Drähten zu messen oder zu erzeugen.

## Was muss von wo nach wo?

CAN nutzt eine empfängerselektive Adressierung. Die Identifier (ID) bezeichnen nicht das Ziel, sondern den Inhalt der übermittelten Daten. Eine Botschaft kann also von allen Steuergeräten am Bus empfangen und ausgewertet werden (Nachrichtenverteilung). Die Applikation eines empfangenden Steuergeräts entscheidet darüber, ob sie eine Botschaft auswertet. Sie kann sogar in ihrem CAN-Controller beim Start einen Akzeptanzfilter setzen, der nicht benötigte CAN-Botschaften anhand ihrer Identifier aus dem Nachrichtenstrom ausblendet. CAN bietet zwei Größen von Identifiern an: 11 und 29 Bit. Die kleinere Variante (Standardformat) wird in Pkw verwendet und bietet 2048 verschiedene Botschaften, während die größere Variante 536 870 912 bietet. Letztere wird hauptsächlich in Nutzfahrzeugen für auf CAN basierende Softwareprotokolle z. B. wie SAE J1939 benötigt, ist aber inzwischen im Pkw-Bereich auch anzutreffen.

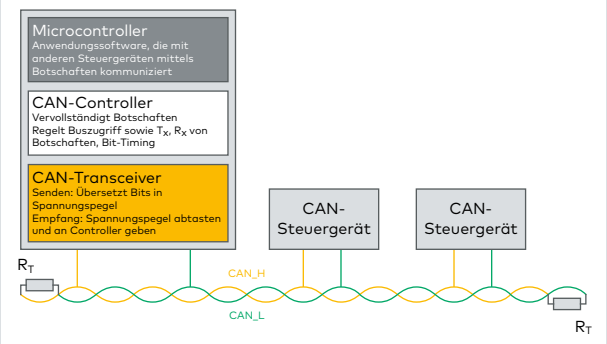


**Bild 2:** Aufbau eines CAN-Steuergeräts

Diese Form der Nachrichtenverteilung bietet folgende Vorteile:

- Kosteneinsparung durch gemeinsame Nutzung von Sensoren durch alle Steuergeräte am Bus
- Einfache Realisierung von verteilten Funktionen
- Die empfängerselektive Adressierung erlaubt unterschiedliche Konfigurationen ohne Anpassung von Hard- oder Software

## Ereignisse veranlassen die Übertragung von Nachrichten

Ereignet sich in unserem Alltag ein berichtenswertes Ereignis, so entstehen Nachrichten darüber. Auch in der Welt serieller Busse spricht man von einem Ereignis, wenn Informationen übertragen werden müssen. Für eine schnelle Übermittlung der Informationen ist es am besten, wenn das zugrundeliegende Ereignis die Übertragung der jeweiligen Daten unmittelbar auslöst. Man spricht von ereignisgesteuerter Übertragung.
Die Alternative wäre, Informationen immer nach einem vorgeschriebenen Zeitplan oder -raster zu übertragen. Wenn nun eine Information anfällt, das entsprechende Steuergerät jedoch nicht an der Reihe ist zu senden, muss gewartet werden.
Um dieses Warten zu vermeiden wurde CAN als ereignisgesteuertes Bussystem entwickelt. Jeder CAN-Knoten ist berechtigt, sofort nach Auftreten eines Ereignisses auf den CAN-Bus zuzugreifen, und die anfallenden Daten zu senden. Einzige Ausnahme dabei ist, wenn bereits ein anderes Steuergerät Daten überträgt. Die Höflichkeit gebietet, andere nicht zu unterbrechen.
Wichtig ist dabei, dass diese anderen nicht beliebig lange sprechen. Bei CAN hat man die Nachrichtenlänge auf maximal 130 Bit (bei 11-Bit-Identifiern) begrenzt. Bei der in Pkw üblichen Datenübertragungsrate von 500 kbit/s bedeutet dies eine Übertragungsdauer von ca. einer Viertel Millisekunde. Danach ist der Bus wieder verfügbar. Dies ist eine wichtige Voraussetzung für eine Datenübertragung, die für Anwendungen wie Antrieb und Fahrwerk hinreichend schnell sein muss.
Allerdings bleibt die Gefahr von Kollisionen, nämlich dann, wenn beim Freiwerden des Busses nach einer Übertragung mehrere Steuergeräte gleichzeitig mit dem Senden von Botschaften beginnen wollen. Diese Gefahr steigt mit zunehmender Buslast. Würden Botschaften bei einer Kollision zerstört, wäre die Folge ein Anstieg der Buslast, und somit ein Teufelskreis, der eine hinreichende Geschwindigkeit bei der Datenübertragung in Frage stellt. Um dies zu vermeiden setzt man im CAN-Netzwerk das CSMA/CA-Buszugriffsverfahren ein (Carrier Sense Multiple Access / Collision Avoidance).

## Prioritäten anstatt Kollisionen

Will ein Steuergerät senden, muss es prüfen, ob der Bus frei ist (Carrier Sense — CS). Ist er belegt, muss es warten. Wird der Bus wieder verfügbar, kann es sein, dass noch weitere Steuergeräte darauf gewartet haben. In diesem Fall dürfen alle Steuergeräte mit dem Senden beginnen (Multiple Access — MA). Um die sich nun anbahnenden Schäden bei dieser Kollision zu vermeiden (Collision Avoidance — CA oder Collision Resolution — CR) wird nun etwas getan, was man als bitweises Arbitrieren bezeichnet. Das Wort stammt vom englischen „arbiter", auf Deutsch „Schlichter".

Alle Steuergeräte mit Sendeauftrag senden gleichzeitig den Identifier ihrer jeweils zu übertragenden CAN-Nachricht bitweise vom höchst- zum niederwertigsten Bit.

Ein Bit mit Wertigkeit 0 ist am CAN-Bus dominant. Das bedeutet, wenn zwei Steuergeräte gleichzeitig unterschiedliche Bitwerte übertragen, setzt sich auf den Busleitungen die 0 gegenüber der 1 durch. Man nennt dies „Wired-AND-Buslogik" (0 = dominant, 1 = rezessiv). Mit dem Senden eines jeden ID-Bits vergleicht jedes Steuergerät den Buspegel mit dem von ihm gesendeten Pegel. Man nennt dies Bitmonitoring. Die Regeln der sogenannten Arbitrierungslogik entscheiden, ob ein Steuergerät weiter senden darf oder das Senden einstellen muss **(Bild 3)**.

Ein Bit nach der Übertragung der Identifier endet die Arbitrierung. Alle Steuergeräte, die beim Senden ihres Identifiers eine logische 1 übertrugen, jedoch eine 0 am Bus vorfanden, mussten das Senden einstellen. Nur ein Steuergerät musste nicht stoppen und kann nun seine gesamte Botschaft unangefochten zu Ende übertragen.

Verlierer einer Arbitrierung wechseln zunächst in den Empfangsmodus und warten, bis der Gewinner seine Botschaft fertig übertragen hat. Sobald der Bus wieder frei ist, greifen sie für einen erneuten Sendeversuch darauf zu. Die Bus- und die Arbitrierungslogik verhindern so nicht nur Kollisionen (Collision Avoidance), sondern sorgen auch für einen prioritätengesteuerten Buszugriff; denn je kleiner der numerische Wert eines Identifiers, desto höher ist die Priorität der betreffenden CAN-Nachricht. Die Botschaft mit dem kleinsten Identifier (ID = 0) wird deshalb ohne Verzögerung übertragen. Botschaften mit numerisch höheren ID-Werten haben bei Kollisionen ein Risiko der Verzögerung durch das Verlieren der Arbitrierung.

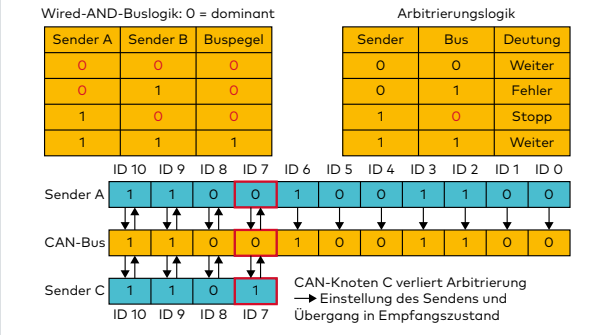


**Bild 3:** Dominanz des Null-Bits und die Arbitrierung

Bei niedriger Buslast ist die Wahrscheinlichkeit von Kollisionen klein, und diese Art des zufälligen, zerstörungsfreien und prioritätengesteuerten Buszugriffs sorgt für einen gerechten und sehr schnellen Buszugriff. Mit zunehmender Buslast häufen sich Kollisionen und die Verzögerungen vor allem von niederprioren Botschaften wachsen an. Im schlimmsten Fall kommen dann Informationen zu spät oder gar nicht mehr bei ihren Empfängern an. Deswegen ist es notwendig, bereits bei der Planung und Entwicklung eines CAN-Busses eine nicht zu hohe Buslast sicherzustellen.

Das bitweise Arbitrieren hat noch einen weniger offensichtlichen Nachteil: Ein Bit muss zeitlich lang genug sein, damit die Verzögerungen beim Senden keine Probleme bereiten. Ein elektrisches Signal breitet sich in Kupfer mit ca. 200 000 km/s oder 0,2 m/ns aus. Dazu kommen noch Verzögerungen durch die CAN-Controller und -Transceiver. Ein Bitpegel muss daher lange genug auf dem Bus anliegen, damit auch die vom Sender am weitesten entfernten Steuergeräte ihn beim Arbitrieren noch rechtzeitig feststellen und auswerten können. Daraus folgt ein reziproker Zusammenhang zwischen Buslänge und Übertragungsrate.

## Daten bestellen und liefern

Es gibt im Straßenverkehr Linienbusse, die regelmäßig nach Fahrplan fahren, und Charterbusse, die nur auf Bestellung fahren. Beide Modelle finden wir auch bei CAN. Botschaften können in regelmäßigen Abständen zyklisch gesendet werden, oder aber nur auf Anforderung.

CAN-Busse in Fahrzeugen leben hauptsächlich von zyklischen Übertragungen. CAN-Busse sind im Pkw kürzer als 40m und im Lkw kürzer als 200m. Somit sind hohe Übertragungsraten möglich, die ein zyklisches Senden erlauben, ohne dass die Buslast zu hoch wird.

Lange CAN-Busse wie in Gebäudekomplexen oder Industrieanlagen jedoch ermöglichen aufgrund ihrer Länge von Kilometern nur kleine Übertragungsraten. Dort macht ein zyklisches Senden von Botschaften nur mit entsprechend langen Zyklen Sinn. Jedoch können Daten auf Bestellung gesendet werden.

Zum Übertragen von Daten gibt es den Data Frame **(Bild 4)**, zum Anfordern von Daten den Remote Frame. Weil der Identifier den Inhalt der Daten bezeichnet, wird in einem Remote Frame derselbe Identifier zum Anfordern von Daten benutzt wie im Data Frame für die Übertragung der angeforderten Daten. Damit es beim Arbitrieren nicht zu Problemen kommt, wenn ein Remote Frame und der dazugehörige Data Frame kollidieren, gibt es am Ende des Identifiers das Remote Transmission Request Bit (RTR), welches

auch noch zur Arbitrierung herangezogen wird. Ist RTR = 1 liegt ein Remote Frame vor, anderenfalls ein Data Frame.

## Bitte ein Bit zum Uhrenvergleich

Grundvoraussetzung für die Übertragung von Data und Remote Frames ist ein Gleichlauf zwischen Sender und Empfänger. Aus Kosten- und Aufwandsgründen verzichtet man bei CAN auf eine Taktleitung. Man realisiert den Gleichlauf mit Signalflanken und einem definierten Mechanismus zur Synchronisation. Wird bei CAN nichts gesendet (Bus-Idle) ist der Bus-Pegel rezessiv, also wie wenn nur Bits mit dem Wert 1 gesendet würden. Eine Botschaft beginnt daher mit dem Übertragen eines dominanten Synchronisationsbits (Start of Frame — SOF) auf dessen fallende Signalflanke alle Empfänger ihre Uhren justieren.

Jeder Empfänger stellt während der gesamten Übertragung durch Auswertung jeder fallenden Signalflanke die Synchronisation sicher und passt gegebenenfalls sein eigenes Zeitraster (Bit Timing) an. Lange Folgen gleichwertiger Bits weisen keine Signalflanken auf. Deswegen unterbricht CAN diese und fügt nach fünf gleichen Bits ein komplementäres Bit ein und erzeugt somit eine Signalflanke. Man nennt dies Bit Stuffing. So erzwingt man eine fallende Flanke nach spätestens 10 Bits. Nach dem Synchronisieren verwerfen die Empfänger diese Stuffbits.

Nach dem SOF folgt der entweder 11 Bit oder 29 Bit lange ID. Seine Länge ist am IDE-Bit (Identifier Extension) ersichtlich. IDE = 0 bedeutet kurze IDs, IDE = 1 die langen.

Zur Übertragung von Nutzinformationen stellt CAN das maximal 8 Byte breite Data Field zur Verfügung. Mittels DLC (Data Length Code) wird die genaue Anzahl der Nutzbytes angegeben. Nach dem Data Field folgt das 15 Bit große Check Field oft auch Prüfsumme oder Cyclic Redundancy Check — CRC genannt.

Aus allen zu übertragenden Bits errechnet der Sender diese Prüfsumme. Mathematisch formuliert tut er dies mittels einer Division der zu übertragenden Bitfolge durch das Polynom 15-ten Grades 0xC599 oder $1x^{15} + 1x^{14} + 1x^{10} + 1x^{8} + 1x^{7} + 1x^{4} + 1x^{3} + 1x^{0}$ auf dem binären Körper, also $x \in \{0,1\}$.

Jeder Empfänger tut dasselbe mit den ankommenden Bits. Danach vergleicht er beide Sequenzen und bewertet die empfangene Botschaft nach dem rezessiven CRC-Delimiter im Acknowledge Slot (ACK-Slot). 0 steht für gut, 1 für Fehler. Eine CAN-Botschaft wird nach dem rezessiven ACK-Delimiter mit dem 7 Bit langen, rezessiven EOF (End of Frame) abgeschlossen. Weil nach dem CRC keine Stuffbits mehr eingefügt werden, ist das EOF eine eineindeutige Kennung für das Ende der Botschaft. Nach dem EOF folgen drei rezessive Bits, die allerdings nicht mehr zur Botschaft gehören. Man nennt sie Intermission (ITM) oder Inter Frame Space (IFS). Erst danach darf eine weitere Botschaft gesendet werden.

## Fehlererkennung und der Elefant im Porzellanladen

Die Wahrscheinlichkeit, dass Fehler in CAN-Nachrichten unerkannt bleiben, ist mit $4{,}7 * 10^{-11}$ oder kleiner sehr gering [2]. Das CAN-Protokoll verfügt über definierte Fehlererkennungsmechanismen. Dazu gehören auf der Empfängerseite neben dem CRC die Überprüfung des Formats (Form Check) und die Prüfung gemäß der Bitstuffing-Regel (Stuff Check). Der Sender prüft per Bitmonitoring, ob ein Bitpegel am Bus dem Sendeauftrag entspricht, und er wertet den ACK-Slot aus.

Weist zum Beispiel an einem CAN-Bus statistisch jede tausendste Botschaft einen Fehler auf, und ist er jährlich 1000 Stunden im Betrieb bei einer Datenrate von 500 kbit/s, einer mittleren Buslast von 25 % und einer durchschnittlichen Nachrichtenlänge von 80 Bit, so würde nach den Regeln der Statistik nur alle 4000 Jahre eine fehlerhafte CAN-Botschaft auftreten, deren Fehler unerkannt bleibt.

Sobald ein Steuergerät einen Fehler bemerkt, bricht es die Nachrichtenübertragung ab, indem es sechs dominante Bits sendet. Dieses sogenannte Error Flag ist der sprichwörtliche Elefant im Porzellanladen des CAN-Protokolls. Denn wo immer es auftaucht, verletzt es bewusst geltende Regeln, zumeist die Bitstuffing-Regel. Dadurch wird die betroffene CAN-Übertragung so deutlich zerstört, dass alle Steuergeräte am CAN-Bus selbst nur lokal feststellbare Fehler wahrnehmen. Es folgt ein zweites Error Flag, weil ja nun alle übrigen Steuergeräte den durch das erste Error Flag erzeugten Fehler sehen. Danach wird für 8 rezessive Bits gewartet. Man nennt diese Error Delimiter. Die beiden Error Flags und den Error Delimiter nennt man Error Frame **(Bild 5)**. Er ist unverzichtbar für die netzweite Datenkonsistenz.

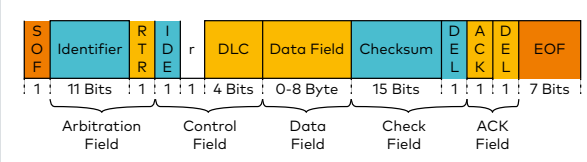


**Bild 4:** CAN Data Frame

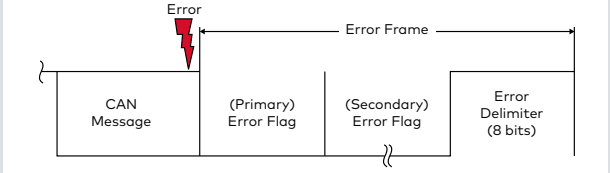


**Bild 5:** CAN Error Frame

Fehler müssen korrigiert werden! Der Sender der fehlerhaften Botschaft tut dies durch Wiederholung, sobald der CAN-Bus wieder frei ist, also nach dem Error Frame und ITM. Allerdings gibt es keine Garantie für sofortiges Wiederholen, denn nach jedem ITM darf jedes Steuergerät beginnen zu senden und die Arbitrierung entscheidet, welche Botschaft sich durchsetzt. Die Fehlererholzeit hängt also von der Nachrichtenpriorität und der Buslast ab.
Was würde passieren, wenn ein Steuergerät per Error Flag Fehler signalisiert, wo keine sind? Laufende Nachrichtenübertragungen würden grundlos abgebrochen und die Kommunikation würde erliegen. Man vermeidet dies durch Selbstüberwachung der CAN-Controller mittels Fehlerzähler. Erkennt ein Steuergerät einen Fehler als erstes, d.h. es sendet das erste der beiden Error Flags, dann muss es seinen Zähler um 8 Punkte erhöhen. Als Sender des zweiten Error Flags muss es nur um 1 hochzählen. Wird eine Botschaft ohne Fehler empfangen, darf der Zähler um 1 reduziert werden. Erkennt das Steuergerät den Fehler beim Senden, erhöht es den Transmission Error Counter (TEC), anderenfalls den Receive Error Counter (REC). Übersteigt einer dieser Zähler den Wert 127, schaltet der CAN-Controller in den Modus Error Passive um. Nun kann er kein Error Flag bestehend aus dominanten Bits mehr senden, also kein anderes Steuergerät mehr unterbrechen oder stören. Er kann jetzt nur noch Error Flags mit rezessivem Pegel erzeugen und dadurch lediglich sich selbst beim Senden stören. Der TEC wird im Fehlerfall weiter inkrementiert, bis er beim Wert 255 das Abschalten des Steuergeräts erzwingt **(Bild 6)**.

## CAN-Botschaften werden benotet

Jede Nachrichtenübertragung wird von allen Empfängern gleichzeitig benotet. Sie alle senden in die Übertragung des Senders, exakt im ACK-Bit eine Bewertung zurück. Man nennt so etwas eine In Transmission Response oder auch In Frame Acknowledgement. Ein dominanter Pegel entspricht der Note Gut, ein rezessiver Pegel der Note Mangelhaft. Da der Sender den ACK-Slot rezessiv sendet, reicht nur eine Note Gut aus, um die Korrektheit der Nachrichtenübertragung zu bestätigen. Schlechte Noten anderer Steuergeräte würden dadurch überschrieben und blieben zunächst ungehört. Jedoch senden diese nach dem ACK-Delimiter ein Error Flag.
Liegt überhaupt keine gute Bewertung vor und bleibt der ACK-Slot rezessiv, stellt der Sender einen ACK-Fehler fest und bricht seine Nachrichtenübertragung mit einem Error Flag ab.

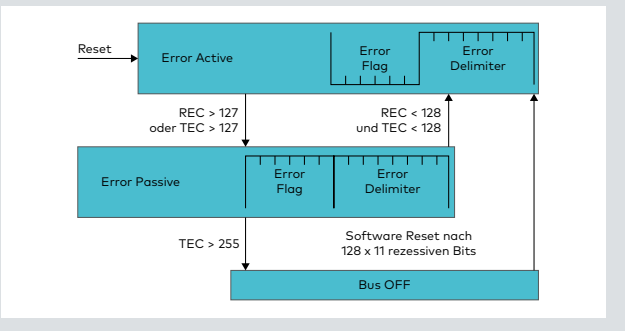


**Bild 6:** Zustände eines CAN Controllers

## Die Grenzen von CAN

Noch immer ist CAN die am häufigsten verbaute Bustechnik im Automobil. Jedoch hat CAN systemische Grenzen. Durch das Prinzip Arbitrierung ist CAN selbst bei zyklischem Senden von Botschaften nicht deterministisch. Das bedeutet, dass sehr zeitkritische Anwendungen mit CAN nicht hinreichend sicher funktionieren. Zudem können mit CAN maximal nur eine Million Bits pro Sekunde übertragen werden. Zum anderen ist CAN durch das hohe Maß an Sicherheit zu teuer für einfache Anwendungen, die darauf verzichten können.
Fahrerassistenz oder autonomes Fahren aber auch Komfortanwendungen im Bereich Audio und vor allem Video erfordern deutlich höhere Datenübertragungsraten und erheben dadurch höhere Ansprüche bzgl. rechtzeitiger Verfügbarkeit von Daten. In diesen Bereichen haben sich Bussysteme wie FlexRay und Netzwerke wie MOST und Ethernet etabliert. Aber auch CAN wurde weiterentwickelt, und mit CAN FD ist eine Erweiterung von CAN geschaffen worden, welche deutlich höhere Übertragungsraten ermöglicht. Seit Dezember 2015 liegt der überarbeitete CAN Standard ISO 11898-1:2015 vor, in dem CAN FD enthalten ist.
Auf der anderen Seite füllt LIN (Local Interconnect Network) die Lücke bei der kostengünstigen Vernetzung von Sensoren und Aktoren im Komfortbereich, wie beispielsweise der Steuerung von Fensterhebern, Sitzen, Außenspiegeln, Schiebedächern oder der Klimaanlage.

## Zuverlässige Steuergerätevernetzung

Unsere Spezialisten von Vector [3] unterstützen Hersteller und Zulieferer der Fahrzeug- und anderen Branchen nicht nur bei der CAN-Vernetzung, sondern auch bei den Kommunikationssystemen wie LIN, FlexRay, CAN FD und Ethernet.
Wir bieten durchgängige Ketten aus Werkzeugen für Planung, Design, Entwicklung und Wartung, aber auch Softwarekomponenten und Basissoftware für AUTOSAR-Steuergeräte. Für den Einstieg in die Steuergerätevernetzung vermitteln wir Grundlagen zu CAN, LIN, FlexRay und Ethernet in Seminaren. Für die notwendigen Fertigkeiten im Umgang mit oben genannten Werkzeugen lernen Ingenieure und Techniker in Workshops mit den vielfältigen Entwicklungsaufgaben rund um die Elektronik im Automobil vertraut zu werden [4].

Im ersten Teil dieser Reihe [5] befassten wir uns allgemein mit seriellen Bussystemen. In weiteren Folgen werden die seriellen Bussysteme LIN, FlexRay und MOST behandelt. Der interessierte Leser findet zu den bereits veröffentlichten Themen auf der Vector E-Learning Plattform [6] ergänzende und vertiefende Informationen.

**Literatur und Links:**

[1] de.wikipedia.org/wiki/Controller_Area_Network
[2] Unruh, J.; Mathony, H. J.; Kaiser, K.H: Error Detection Analysis of Automotive Communication Protocols. SAE International Congress 1990.
[3] www.vector.com
[4] www.vector-academy.de
[5] Christmann, E.: Datenkommunikation im Automobil Teil 1: Architektur, Aufgaben und Vorteile serieller Bussysteme.
[6] elearning.vector.com

**Ernst Christmann, Physiker, Mathematiker**
ist Senior Technical Trainer im Bereich Software Tools für Steuergerätetests und Steuergeräteentwicklung und seit 2004 bei der Vector Informatik GmbH in Stuttgart tätig.