## 6.2 CAN (Controller Area Network)

### 6.2.1 Wesentliche Eigenschaften

- Priorisierung von Nachrichten
- Garantierte Verzögerungszeiten
- Flexible Konfiguration von Systemen
- Multicast-Empfang durch mehrere Empfänger mit Zeitsynchronisation
- System-weite Datenkonsistenz
- Multimaster
- Fehlerdetektion und Fehlersignalisierung
- Automatische Neuübertragung, sobald der Bus wieder frei ist
- Unterscheidung temporärer Fehler und dauerhafter Fehlerbedingungen einzelner Knoten mit autonomer Abschaltung defekter Knoten
- Baudraten bis 1Mbaud – bei einer Auslastung von 50% und einem Protokolloverhead von 50% stehen in Multimastersystemen ca. 25Kbyte/s an Nutzdatenrate zur Verfügung

### 6.2.2 Herkunft, Anwendungen

Der CAN-Bus ist durch die Firma Bosch spezifiziert worden und wurde entwickelt für Anwendungen im Automobil. In Europa hat sich CAN auch durchgesetzt in Bereichen der Anlagenautomatisierung, z.B. der Kommunikation von Einheiten untereinander in größeren Geräten, wie z.B. Testgeräte in der Halbleiterindustrie, Laborautomatisierung, Vernetzung antriebstechnischer Einheiten.
Die CAN-Spezifikation umfasst dabei nur die physikalische Ebene, die Bittransfercodierung sowie die Übermittlung von Telegrammen und die Fehlerbehandlung. Weiterhin sind Dokumente zur angrenzenden Themen verfügbar. Strenge Auflagen zur Kompatibilität sichern einwandfreie Funktion, so sind z.B. CAN-Implementationen von Bosch in C und VHDL erhältlich.

### 6.2.3 Standardisierung von Funktionen: CAN-Open

Mit der sog. CAN-Open-Spezifikation steht ein Vorschlag zur Verfügung, wie Geräte aus unterschiedlichen Bereichen herstellerunabhängig mit definierten Telegrammen parametriert werden. Prinzipiell ist dadurch keine Anpassung der Systemsoftware mehr an die aktuell verwendete Hardware notwendig. Praktisch bedeutet CAN-Open jedoch einen Software-Overhead von ca. 16 Kbyte für ein typisches Gerät, sowie Anforderungen an Prozessorleistung und RAM. Da nicht alle herstellerspezifischen Features abgedeckt werden können, ist für jede Geräteklasse ein gewisser Funktionsumfang definiert, von dem meist nur ein Subset implementiert wird. Dadurch wird die Kompatibilität in vielen Fällen wieder erheblich eingeschränkt.

| Applikationsebene |
|---|
| Objektebene<br>Nachrichtenfilterung<br>Nachrichten- und Statusauswertung |
| Transferebene<br>Fehlerabschaltung<br>Fehlerdetektion und –Signalisierung<br>Validierung der Nachrichten<br>Bestätigung<br>Arbitration<br>Nachrichtenkapselung in Telegramme<br>Übertragungsrate und –Timing |
| Physikalische Ebene<br>Signalpegel und Bitrepräsentation<br>Übertragungsmedium |

**Abb. 6-10 : Umfang der CAN-Spezifikation**

### 6.2.4 CAN2.0A vs. 2.0B

In der ersten CAN-Spec 2.0A wurde ein Identifier von 11 Bit vorgesehen. Da jedoch jedes Peripheriegerät eine Reihe von Identifiern nutzen kann, und der Wunsch besteht, Identifier anhand der Nachrichtentypen unabhängig von der jeweiligen Kombination von Teilnehmern zu vergeben, wurde in der Spezifikation 2.0B der Identifier auf 21 Bit erweitert. Nahezu alle auf dem Markt verfügbaren Bausteine unterstützen 2.0B, bzw. können in Netzwerken mit 2.0B Nachrichten verwendet werden, wobei sie selbst nur 2.0A Telegramme verarbeiten (2.0B passive Bausteine).

### 6.2.5 Telegramtypen und deren Aufbau

CAN definiert DATA-FRAMES für die Übertragung von Daten, REMOTE-FRAMES zur Anforderung von Daten mit der selben ID, ERROR-FRAMES zur Signalisierung von Übertragungsfehlern, sowie OVERLOAD-FRAMES zur Einführung von Verzögerungen.

Die Abbildung zeigt den Aufbau von DATA-FRAMES. Das Datenfeld kann eine Länge von 0-8 Bytes haben. 0-Byte Telegramme dienen dabei beispielsweise dem auslösen Vorprogrammierter Ereignisse, oder REMOTE-FRAMES haben ebenfalls die gleiche Struktur, jedoch kein Datenfeld.
Das 15-Bit CRC erlaubt die sichere Erkennung von bis zu 5 Einzel-Bit-Fehlern sowie die Erkennung von Fehlerburst bis zu 14 Bit. Eine Fehlerkorrektur ist nicht vorgesehen.
Der / die Empfänger bestätigen den korrekten Empfang einer Nachricht direkt im Acknowledge-Feld des Telegramms durch einen dominanten Pegel.

Detektiert eine Station einen Fehler in einer Übertragung, kann sie durch einen ERROR-FRAME dies allen Stationen übermitteln. Der Error-Frame besteht dabei in einer Verletzung des Bit-Stuffing-Protokolls, d.h. dass mehr als 5 dominante Bits aufeinander folgen. Diese Bedingung ist daher in jedem Fall feststellbar und sichert die Konsistenz bei Multicasts.

**Standard-Datenformat (2.0A)**

| FS | Identifier | | DLC | Data | CRC | | Ack | | EoF |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 11 | 3 | 4 | 0..8 Byte | 15 | 1 | 1 | 1 | 7 |

**Abb. 6-11 : Erweitertes Datenformat (2.0B)**

| FS | Identifier | | Identifier | | DLC | Data | CRC | | Ack | | EoF |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 11 | 2 | 18 | 3 | 4 | 0..8 Byte | 15 | 1 | 1 | 1 | 7 |

**Abb. 6-12 : Aufbau der Telegramme bei CAN2.0A und CAN2.0B**

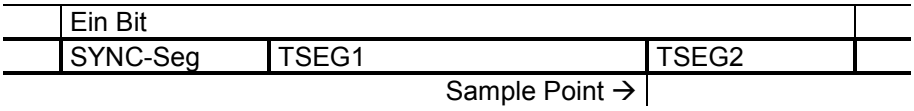


**Abb. 6-13 : Aufbau der einzelnen Datenzellen**

## 6.2.6 Arbitration mit Priorisierung

Prinzipiell kann ein Sender nur senden, wenn der Bus frei ist (CSMA/CD-Prinzip). Der CAN-Bus nutzt sowohl für die Auflösung von Kollisionen bei der Arbitration ein logisches Verordern der Informationen auf dem Bus, als auch für die Herstellung von Datenkonsistenz bei der Bestätigung von Multicast-Nachrichten. Dieses Konzept ist auch z.B. vom Bussystem I²C her bekannt. Es bedingt aber gleichzeitig, dass die Datenlaufzeit nicht höher sein darf als ein Bruchteil der Länge einer Bit-Zelle. Dadurch sind Größe des Bussystems und maximale Datenrate eng miteinander verbunden. Jeder Baustein kann beginnen zu senden, sobald der Bus frei ist. Dabei folgt nach einem Startbit, dass auch der Synchronisation der Sender untereinander dient, die Übertragung des Nachrichtenidentifiers, der u.A. den/die Zielknoten identifiziert. Sollten zwei Sender zugleich eine Übertragung starten, z.B. weil der Bus gerade frei geworden ist, so verliert derjenige den Masterzugriff, der als erstes einen „rezessiven“ (=passiven, d.h. durch die Abschlusswiderstände hergestellten) Pegel in seinem Telegramm enthält, der durch einen „dominanten“ Pegel des anderen Senders überdeckt wird. Der Sender, der die Arbitrationsphase verloren hat bricht dann seine Übertragung ab und startet erneut, sobald der Bus wieder frei ist. Auf den ersten Blick scheint die statische Kopplung der Prioritäten an die Nachrichten-IDs nachteilig zu sein. Sie kann jedoch aufgrund des großen Adressraums aufgelöst werden, indem die obersten Adressbits alleine zur Codierung der Priorität genutzt werden, während die niederwertigen Adressbits den Empfänger codieren.

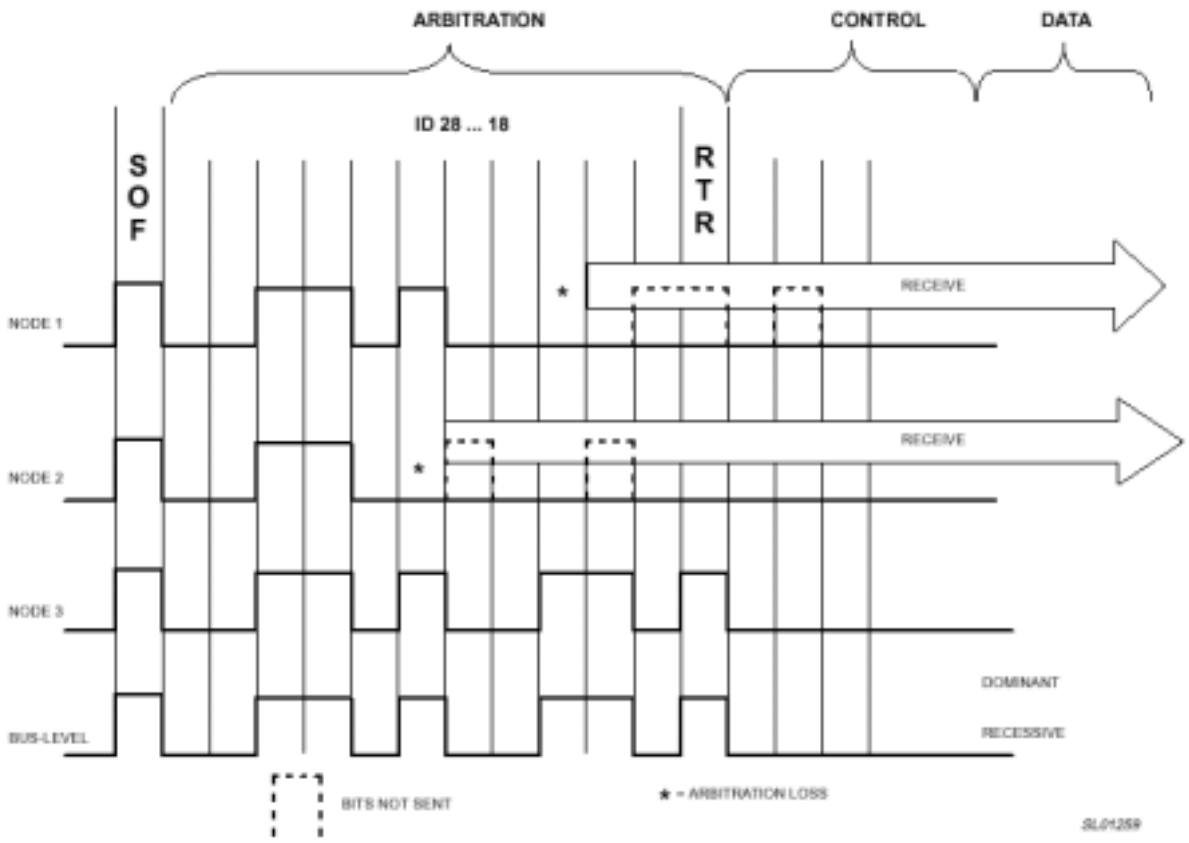


**Abb. 6-14 : Arbitration durch logisches Verodern der Nachrichten-IDs auf dem Bus aus [1]**

## 6.2.7 Codierung

CAN verwendet eine direkte Codierung der Bits (NRZ), d.h. stabiler Logikpegel während der gesamten Bitübertragung. Dabei wird zu Synchronisationszwecken nach je 5 gleichen Bits ein Pegelwechsel erzwungen, indem ein Bit der gegenüberliegenden Priorität eingeschoben wird. Spezielle Frames, wie der Error-Frame können jederzeit erkannt werden, indem sie gegen diese Konvention verstoßen.

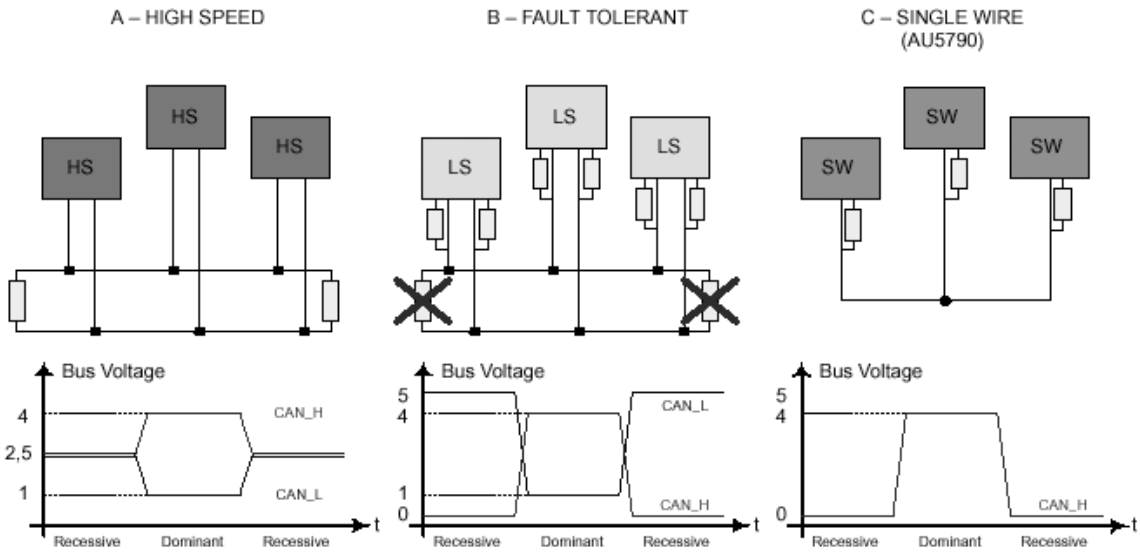


**Abb. 6-15 : Physikalische Medien für A: Standard-CAN, B: Low-Speed, Fault-tolerant, C: Single-Wire [2]**