CAN

(Controller Area Network)

Ein Vortrag von: Marietta Stutz mstutz @cosy.sbg.ac.at
Eva Brunner ebrunner@cosy.sbg.ac.at
Susanne Stadler sstadler@cosy.sbg.ac.at
Georg Klima gklima@cosy.sbg.ac.at

Inhalt:

- Einführung in CAN.
- Entwicklungsgeschichte.
- Funktionalität.
- Skalierbarkeit.
- Implementierung.
- Quellen.

Einführung - CAN Definition:

- Verlässlicher Datenbus für Echtzeit - Applikationen.
- Event Triggered.
- Garantierte Transferrate von bis zu 1 Mbit/s.
- Ausgezeichnete Fehlererkennung und Eingrenzung.
- Garantierte Latenzzeiten.
- Vielfältigste Anwendungen in der Industrie - Automatisierung.
- ISO / OSI Konform.
- CAN Spezifikation definiert ISO / OSI Layer 1 und 2.
- Weiterentwicklung: OSI Session Layer TTCAN (Time Triggered CAN).
- ISO dokumentiert / zertifiziert (ISO 11898 / 11519).

Einführung - Kurze Information zum Entwickler (Bosch):

- 1886 von Robert Bosch in Stuttgart gegründet.
- Eines der größten Industrieunternehmen Deutschlands.
- 35 Mrd. € Umsatz im letzten Jahr (2002).
- 224.000 Mitarbeiter weltweit.
- Ca.: 2.000 neue Patente pro Jahr.
- Unternehmen auf die Finanzierung der Robert Bosch Stiftung ausgerichtet.

Einführung - Kurzgeschichte: Die Entwicklung von CAN:

- **1986** Präsentation des CAN Standards.
- **1987** Erster CAN Controller am Markt (Intel 82526).
- **1990** Entwicklung von DeviceNet und Smart Distributed System(SDS).
- **1992** Gründung von CAN in Automation (CiA).
- **1993** Entwicklung von CANopen.
- **1995** Weiterentwicklung von CANopen durch die CiA.
- **Inzwischen** Weiterentwicklung zu Time Triggered CAN (TTCAN).

Einführung - Einsatzgebiete von CAN - Bus - Systemen:

- **AUTO**: Verbinden von einzelnen Steuerkomponenten.
- **TEXTILINDUSTRIE**: Erste Anwendungen.
- **HAUS**: Steuerung von Heizsystemen, Industrie-Kaffeemaschinen.
- **BAHN**: Verbindung Führerstand – Antrieb.
- **MEDIZIN**: Komponenten basierte Systeme.
- **MARITIM**: Brücke – Maschinenraum.
- Fülle von Einsatzgebieten durch die Gründung von CiA und Entwicklung von CANopen.

Einführung - Argumentation zur Verwendung von CAN:

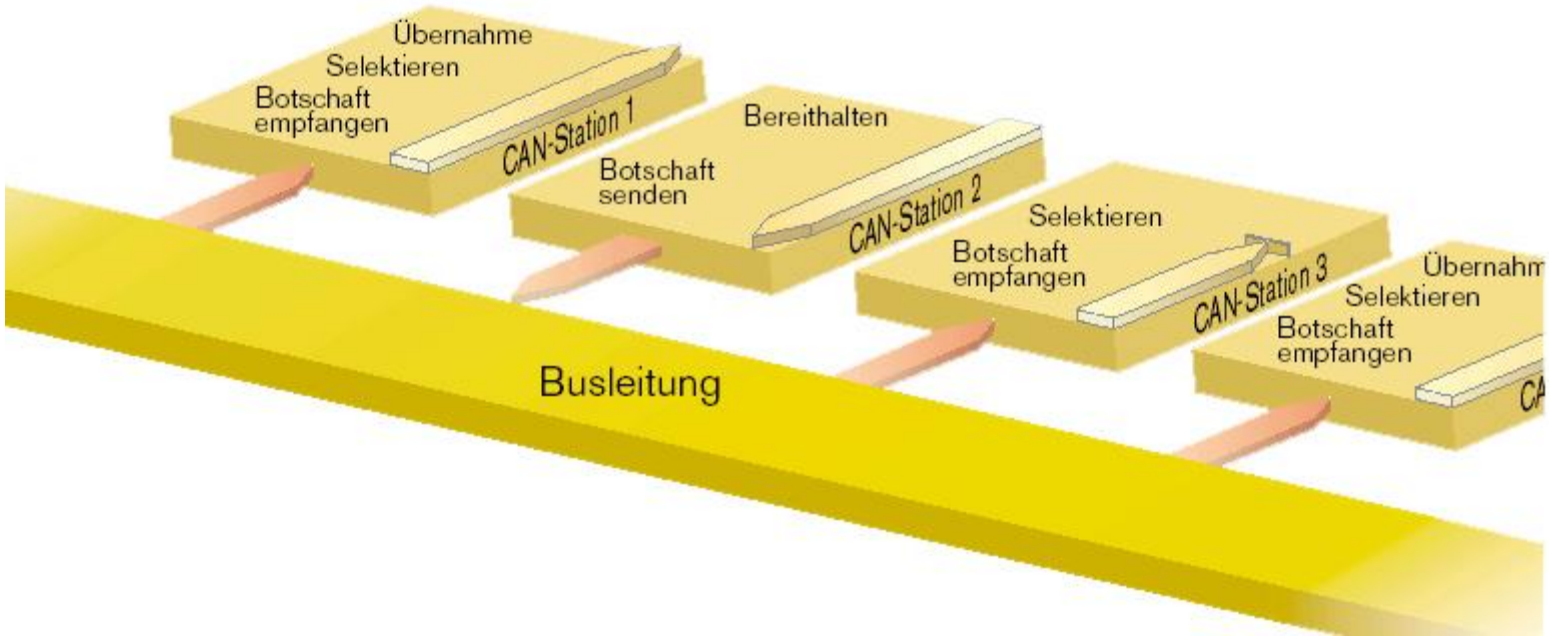
- Abstraktion der Kommunikation auf ein höheres Level.
- Hohe Modularisierung des Systems bei geringeren Entwicklungskosten.
- Im Automobilbereich: Leichtere Handhabung der Ausstattungsvarianten durch Modularisierung.
- Handhabbare Verkabelung durch Multiplexing.
- Testen von Einzelkomponenten als auch des kompletten Systems.

FUNKTION

Wie arbeitet CAN?

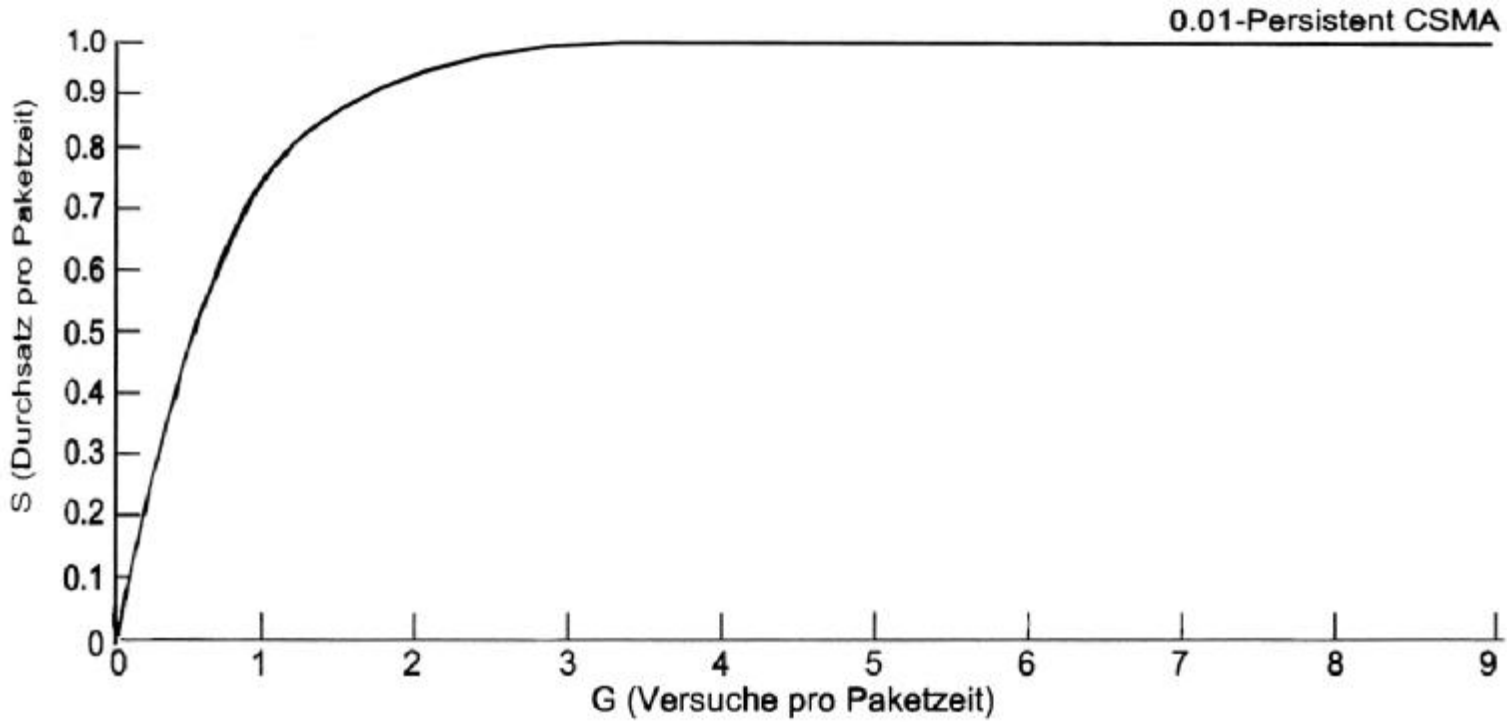
Funktion - Spezifikation:

- Vergleichbar mit Ethernet (10Base2) Topologie:
  - Serieller multimaster Datenbus.
  - Message- statt Station- IDs.
  - Kollisionsauflösung mittels Message IDs.
  - CSMA / CD + AMP (Carrier Sense Multiple Access / Collision Detection with Arbitration on Message Priority).



Funktion - Lastverhalten: CAN – Ethernet (1):

- Lastverhalten von Ethernet mit nur CSMA ohne CD

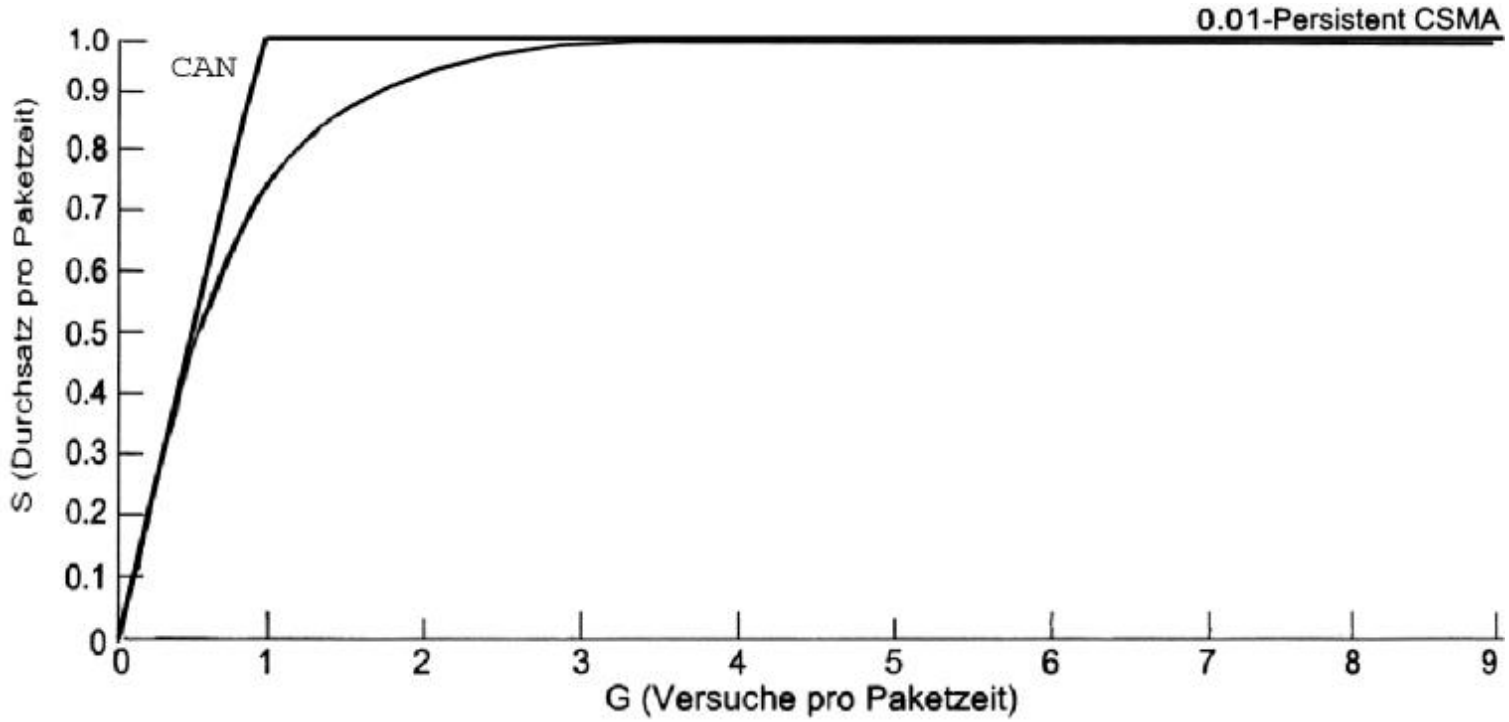


S = Rahmen pro Zeiteinheit **à** 0 <= S <= 1 (S > 1 geht nicht!)

G = Übertragungsversuche pro Paketzeit **à** G >= S

Funktion - Lastverhalten: CAN – Ethernet (2):

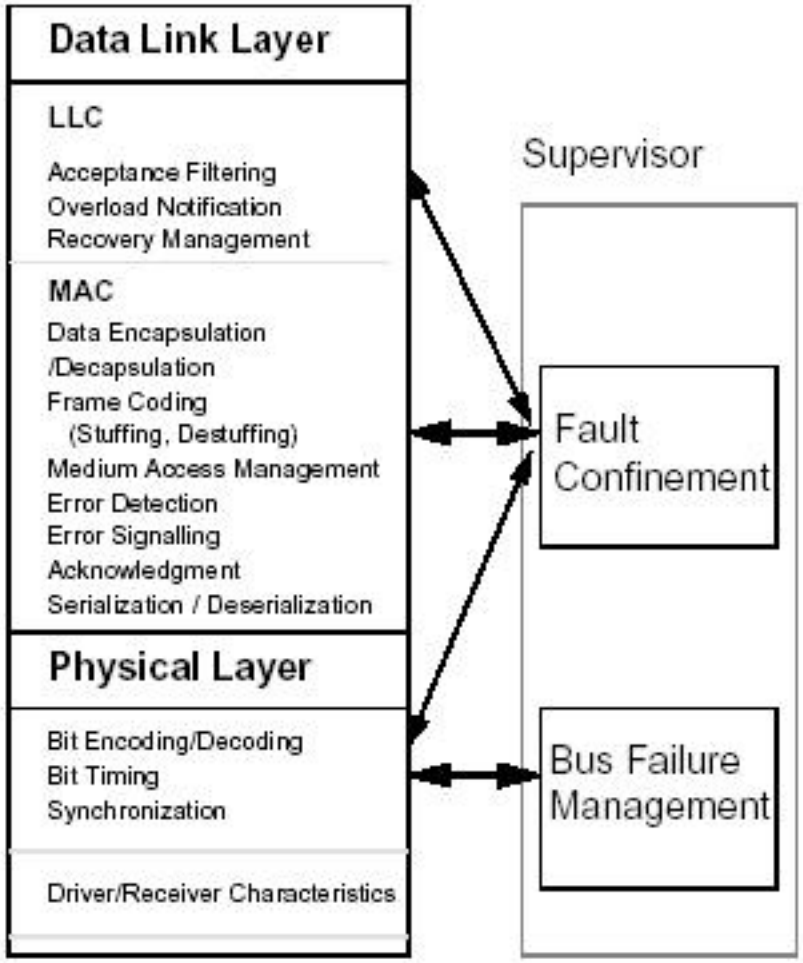
- Lastverhalten von CAN mit CSMA / CD + AMP



Bei G > S **à** Überlast am Bus

Wie erreicht man diese Skalierbarkeit?

Funktion - Netzwerkebene: (ISO / OSI Model)



LLC = Logical Link Control
MAC = Medium Access Control

Funktion - Logische Buszustände:

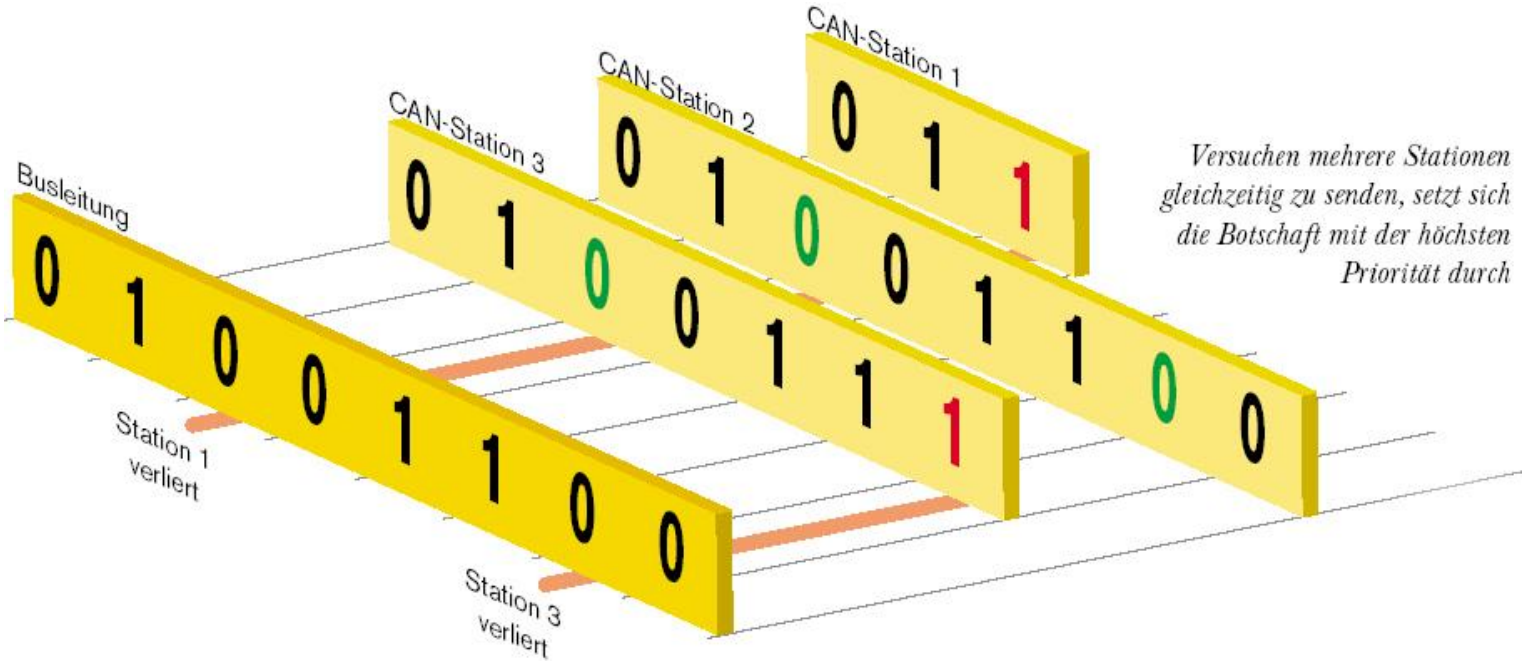
- Die Buszustände (abstrakter gesehen):
  - Dominant, Rezessive, Idle.
- Die logischen Buszustände:
  - Logisch 0 entspricht dominant.
  - Logisch 1 entspricht rezessiv.
  - Logisch 1 entspricht ebenfalls Idle (also auch rezessiv).
- Der Schluss:

  Rezessive Buszustände werden durch dominante überschrieben.

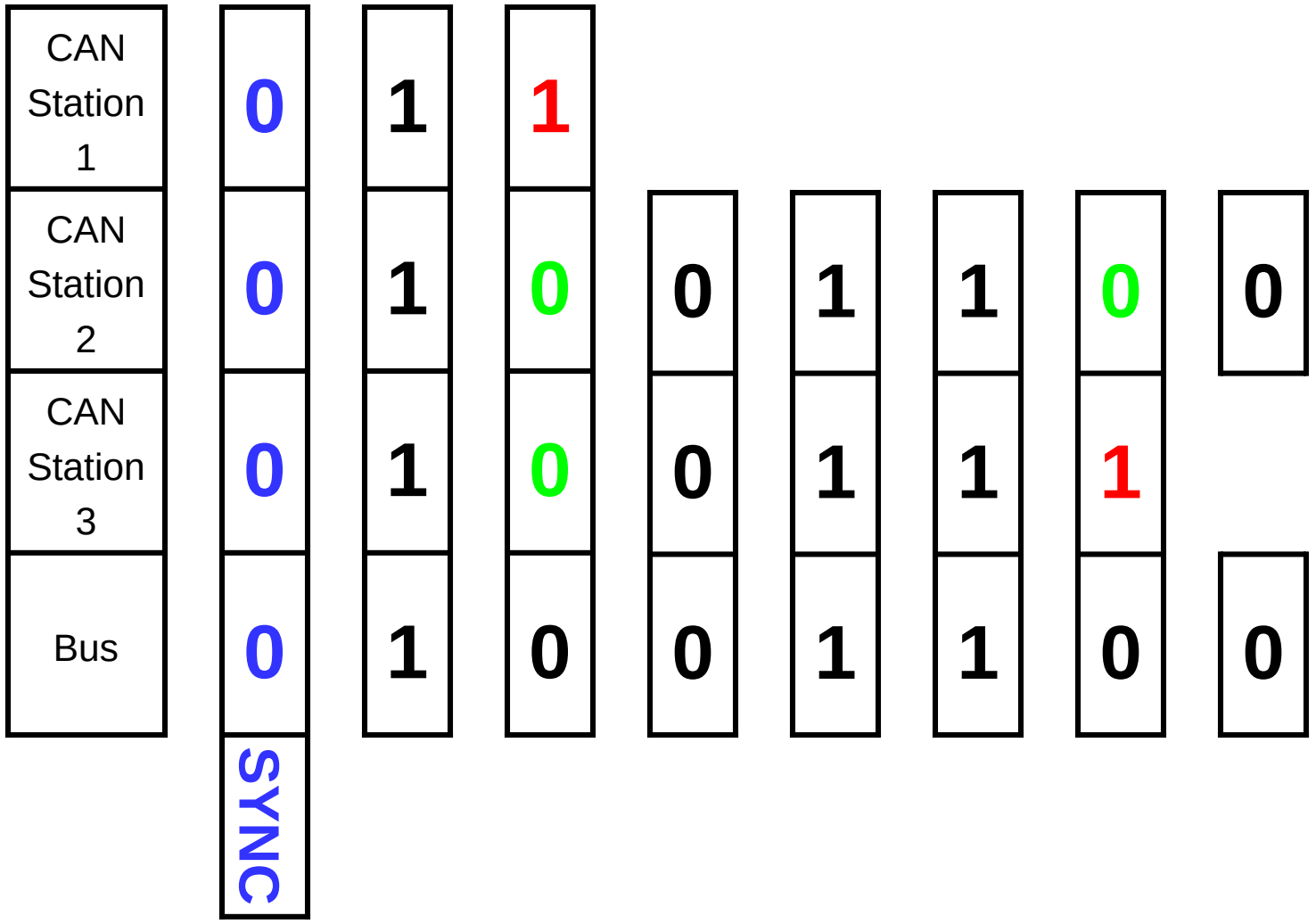
  Also eine logische 0 überlagert eine logische 1.

Funktion - Arbitration / Priorisierung (1): (MAC sublayer)

- Ethernet: CD + Binary Exponential Backoff **à** Bei Kollision Abbruch (Zeitverlust).
- CAN: CD + AMP-Verfahren **à** Bei Kollision verlustlose Auflösung.



Funktion - Arbitration / Priorisierung (2): (MAC sublayer)



Funktion - Arbitration / Priorisierung (3): (MAC sublayer)

- Vorgang der Arbitration:
  - Der Bus ist Idle (Leerlauf).
  - Senden eines dominanten Bits zur Synchronisation der Stationen.
  - Gleichzeitiges Schreiben und Lesen des Busses (Monitoring).
  - Unterscheidet sich der gelesene vom geschriebenen Wert **à** wird die Station automatisch zum Empfänger.
  - Solange gelesenes und geschriebenes Bit übereinstimmen bleibt die Station Sender.
  - Am Ende bleibt eine Station als Sender übrig, der Rest empfängt.
  - Der Sender überträgt den Rest seines Frames ohne Unterbrechung.

Funktion - Messageframeformat:

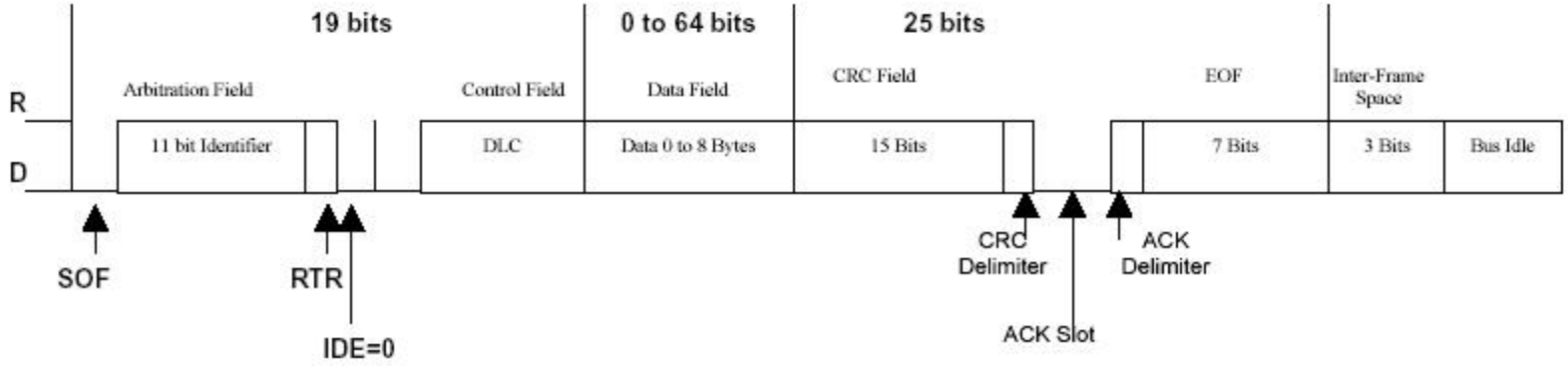
- DATA FRAME:
  Beinhaltet Daten entsprechend dem Identifier.

- REMOTE FRAME:
  Leerer Data-Frame mit Remote-Bit gesetzt. Fordert Daten entsprechend dem Identifier an.

- ERROR FRAME:
  Abschließender Frame nach einem „erkannten“ Error Flag.

- OVERLOAD FRAME:
  Durch den anfordernden Receiver nach einem Remote-Frame
  (Zum Verhindern von einem Overflow oder einer Race Condition).

Funktion - Frameformat:



SOF 1 Bit	**Message ID 11 Bits**	**Steuer Bits**	**Daten 0 - 64 Bits**	**CRC + Delimiter 15 Bits**	**ACK Slot + Delimiter**	**EOF 7 Bits**	**Inter Frame Space 3 Bits**

Funktion - Safety (1): (Fehlererkennung)

- Fehlererkennung:
  - Monitoring (Vergleich gesendetes / empfangenes Bit).
  - CRC 15 Bit.
  - Bit Stuffing / Destuffing.
  - Message Frame Check (fixes Vorkommen bestimmter Bits).
- Möglichkeiten der Fehlererkennung:
  - Alle globalen Fehler werden erkannt.
  - Alle lokalen Fehler beim Sender werden erkannt.
  - Bis zu 5 zufällig in einer Message verteilte Fehler werden erkannt.
  - Burst Errors mit weniger als 15 Fehlern werden erkannt.
  - Jede ungerade Anzahl von Fehlern in einer Message wird erkannt.

Funktion - Safety (2): (Fehlerwahrscheinlichkeit)

- Fehlerwahrscheinlichkeit bei einer bestimmten Fehlerrate:

  Speziell: message error rate * 4.7 * 10^{-11}

  (message error rate wird im Systemtestlauf gemessen).

  Allgemein: 10^{-11}.

Funktion - Safety (3): (Fehlerzustände)

- 3 Fehlerzustände der Stationen:
  - **Error Active:** Jeder (lokal) erkannte Fehler wird global bekannt gemacht (Verletzung der Bit Stuffing Regeln **à** 6 *dominante* Bits).
  - **Error Passive:** Jeder (lokal) erkannte Fehler wird möglicherweise global bekannt gemacht (Verletzung der Bit Stuffing Regeln **à** 6 *rezessive* Bits).
  - **Bus Off:** Die Station wurde vom Bus wegen „Unzuverlässigkeit“ getrennt. Kann erst nach einem Reset und einer speziellen Bussequenz wieder **Bus On** werden.
- Die Fehlerzustände werden durch eine interne unabhängige Logik ermittelt (mittels spezieller Zähler und einem ausgeklügeltem Regelwerk).

Implementierung

Implementierung - Arten(1): (Basic CAN)

- Basic CAN:
  - Billigere CAN Variante.
  - Mitverwendung des Host Mikrocontrollers für CAN.
  - Message Behandlung durch Host.
  - Remote Frame Behandlung durch Host.
  - 2 Receive Buffer.
  - Überlauf Philosophie: halte älteste Nachricht, neue gehen verloren.
  - Kein Message Filtering.
  - Message Aufbau durch Host.

Implementierung - Arten(2): (Full CAN)

- Full CAN:
  - Teurere CAN Variante.
  - Hat eigene Mikrocontroller.
  - Unterstützt Message Filtering.
  - Hat Identifier bezogene Mailboxen für Senden und Empfangen.
  - Überlauf Philosophie: Halte die neuesten Nachrichten.
  - Zum Teil dynamisch rekonfigurierbare Mailboxen.
  - Kommunikation mit dem Host über Shared Memory.

Implementierung - Application Layer:

- Teilweise Frameworks für CAN System Integration:
  - CAL (CAN Application Layer): Lizenz-frei, Applikationsunabhängig.
  - CANopen: Implementierung von CAL (eigentlich von Philips med. sys.).
  - DeviceNet: freier Fieldbus für ein Can Sensoren Netzwerk.
  - SDS (Smart Distributed System): Honeywell interner CAL.

Quellen:

- http://www.can.bosch.com/
- http://www.canopen.us/
- http://can-cia.de/
- http://www.mjschofield.com/
- http://www.algonet.se/~staffann/developer/frames.htm

Danke!

Noch Fragen?