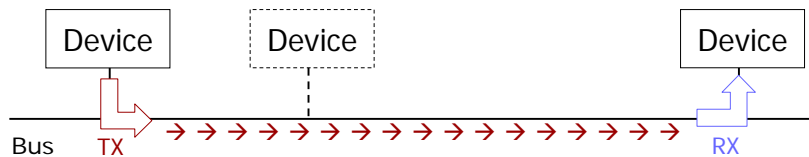


**Embedded Realtime Systeme (Teil 2)**

Prof. Dr. Ralf S. Mayer,  
Prof. Dr. Peter Altenbernd  
Fachbereich Informatik, h\_da

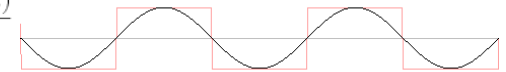
- Eigenschaften verteilter Systeme
  - Signallaufzeit
  - Buszuteilung
  - Bandbreite
  - Priorisierung
  - Zeitverhalten
- Bussysteme
  - Standards
  - Eigenschaften bezüglich OSI
  - Einsatz



- Bei der Verbindung von Geräten über ein Medium muss die Signallaufzeit berücksichtigt werden
- Maximale mögliche Ausbreitungsgeschwindigkeit:
  - Lichtgeschwindigkeit  $c_0 = 299.792.458 \text{ m/s} \approx 3 \times 10^8 \text{ m/s}$  im Vakuum
  - Geschwindigkeit im Medium:  $c_m < c_0$ 
    - In elektrischen Leitern (Kabeln) geht man von  $c_m = 17 - 20 \text{ cm/ns}$  aus.  $\rightarrow c_m \approx 2/3 c_0$
    - Geschwindigkeit im Medium in der Regel frequenzabhängig:  $c_m = c_m(f)$   $\rightarrow$  Dispersion
    - Dämpfung im Medium ist frequenzabhängig
    - Reflexion an Kabelenden ist möglich.  $\rightarrow$  Abschluss-Widerstände (Beispiel: Entspiegelung einer Brille)

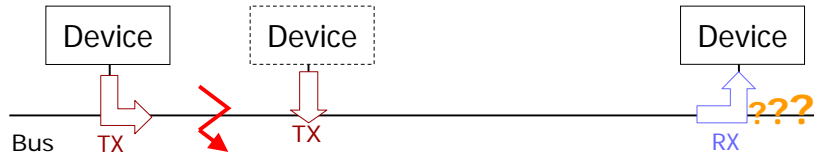
- Ein Rechtecksignal (binäre Übertragung) kann aus einer Überlagerung von Sinusfunktionen dargestellt werden, wobei
  - $f_0$  die Frequenz des Rechteck-Signals ist und (unendlich viele) ungerade Harmonische  $f_{n+1}$  dies Frequenz in geeigneter Amplitude  $a_{n+1}$  hinzuaddiert werden: harmonics: 1  $\rightarrow$  Fourier-Synthese

$$x_{\text{rechteck}}(t) = \frac{4}{\pi} \sum_{n=0}^{\infty} \frac{\sin((2n+1)t)}{2n+1}$$



- Bei einem realen Übertragungsmedium (Kabel, Lichtleiter etc) muss man aufgrund der Dispersion, Dämpfung und Reflexion davon ausgehen, dass ein ausgesandtes Rechtecksignal nicht mehr als Rechteck beim Empfänger ankommt.
- Auswirkung auf das System
  - sorgfältiges Design (elektrisch, physikalisch) notwendig
  - Reale Bandbreite kleiner als theoretische
  - Störeinflüsse von außen minimieren
  - Störung der Umgebung beachten (ElektroMagnetische Verträglichkeit)

## Bus-Zugriff (Arbitration)



Buszuteilung – Erkennung/ Vermeidung von Kollisionen:

- Master kontrollierte Netzwerke
  - Token-basierter Zugriff: Token-Passing in Ringnetzen. Token wird von Knoten zu Knoten weitergeleitet. Knoten der Frei-Token erhält, darf senden. Token-Ring, ARCNET, Profibus, ...
  - Feste / auf Anfrage zugeteilte Bandbreite
    - TDM(A): Zeitschlitz in Funknetzen (GSM), Kabelnetzen (ISDN)
    - FDM(A): Frequenzzuteilung (Rundfunk: Stereo), Kabel: ISDN/ADSL
  - Multimaster-Netzwerke: mehrere Teilnehmer können „beliebig“ senden. → Gefahr von Kollisionen
    - Ethernet: Kollisionserkennung
    - CAN: Kollisionsvermeidung
- ? Frage nach Determinismus der Verfahren.

## Verteilte Systeme - Busszugriff

- **CSMA/CA** - Carrier Sense Multiple Access with Collision Avoidance - Protokollvariante von CSMA/CD, bei der Kollisionen durch Vorausmassnahmen vermieden werden
- **CSMA/CD** - Carrier Sense Multiple Access with Collision Detection - Protokollstrategie, bei der nach Feststellen einer offenen Leitung jeder Teilnehmer senden kann und bei einer während des Sendens auftretenden Kollision fuer unterschiedl. (per Zufallsgenerator bestimmte) Zeit unterbricht (Konkurrenzbetrieb)
- **CSMA/CE** - Carrier Sense Multiple Access with Collision Elimination - Protokollvariante von CSMA/CD, bei der Kollisionen automatisch erkannt und behoben werden
- **CSMA/CP** - Carrier Sense Multiple Access with Collision Prevention - Protokollvariante von CSMA/CD, bei der Kollisionen durch Vorausmassnahmen verhindert werden.

Beispiele:

- CSMA/CD: Ethernet
- CSMA/CA: RS232 mit RTS/CTS, CAN
- CSMA/CE und /CP: bei LAN-Protokollen

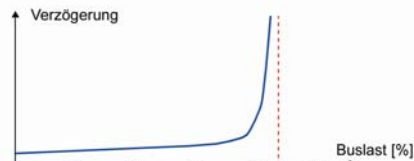
## Anforderungen an Bussysteme

Randbedingungen für echtzeitfähige Bussysteme:

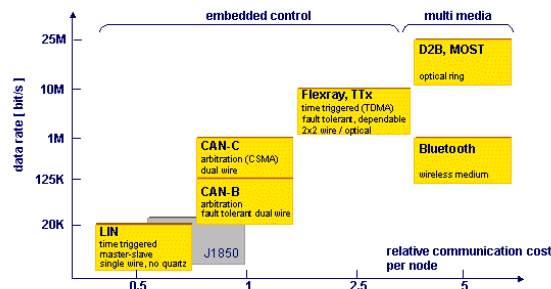
- Reaktionszeit
- deterministisch
- Bandbreite
- Sicherheit

Weiterhin:

- Kosten
- Störanfälligkeit
- Redundanz



CSMA/CD-Busszugriff von Ethernet: Übertragungszeiten nehmen mit steigender Buslast stark zu



## Eigenschaften von CAN

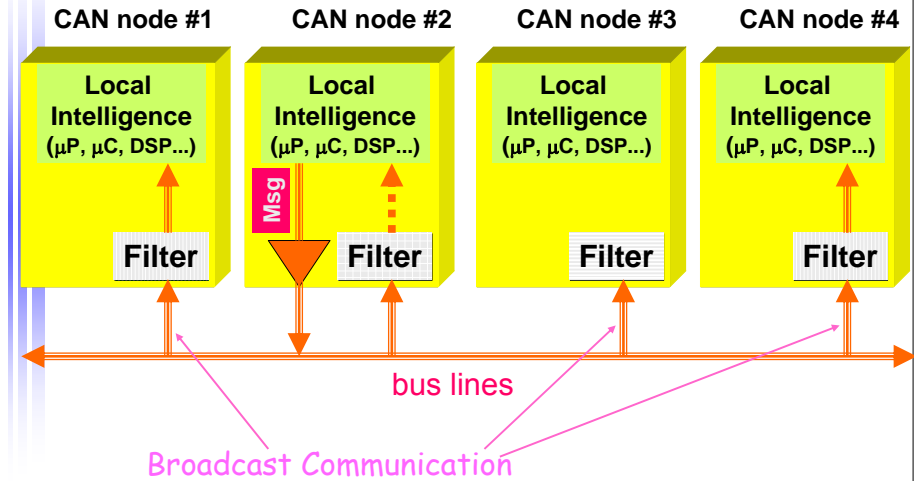
Control Area Network

- Ereignisgesteuert: Alle Knoten sind zu jedem Zeitpunkt gleichberechtigt auf das Kommunikationsmedium zuzugreifen.
  - **C**arrier **S**ense, **M**ultiple **A**ccess / **C**ollision **D**etection + Methode zum Erkennen und Auflösen von Kollisionen
    - Senden können, sobald Bus frei ist (Carrier Sense), mehrerer Knoten gleichzeitig (Multiple Access)
    - Dabei werden mögliche Kollisionen erkannt (CD) und behoben.
  - Bitweise Arbitrierung: Alle aktuellen Sender legen bitweise nacheinander die Bits ihrer Priorität auf den Bus
    - die höhere Priorität sendet weiter
    - die niedrigere(n) schalten sofort auf Empfang um
    - realisiert durch dominante („0“) / rezessive Bits („1“)
  - Beachte Gegensatz zu Ethernet (CSMA/CD)
    - bei entdeckter Kollision nehmen **alle** Sender ihre Signale vom Bus
    - die sendewilligen Knoten warten jeweils eine zufällig gewählte Zeit und greifen dann erneut auf den Bus zu
- Probleme bei hoher Netzlast:
- unbestimmte Reaktionszeit
  - Bandbreite nimmt ab

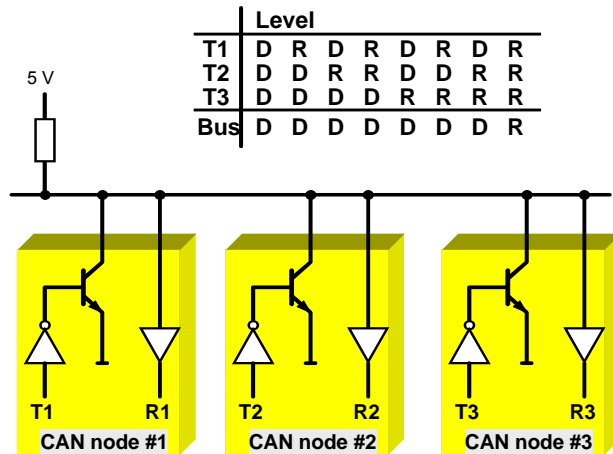
## Steckbrief CAN

- Broadcast-Bus bis 1 MBit/s
- Nachrichten (message) 0-8 Byte
  - Daten Message
  - Remote Transmission Request Message
  - Jeder Knoten versendet aus seiner Queue zuerst die Nachricht mit der höchsten Priorität
- Eindeutiger Identifier (unique identifier)
  - kein expliziter Empfänger
  - Empfänger implementieren Filter für Nachrichten
  - Collision Detection +
  - Collision Resolution: höhere Priorität erhält Vorrang
- CAN message format:
  - 47 Kontroll-Bits
  - 0-64 Daten-Bits
  - 0-19 Stopf-Bits (stuffing bits)

## CAN: Broadcast-Prinzip, Filter

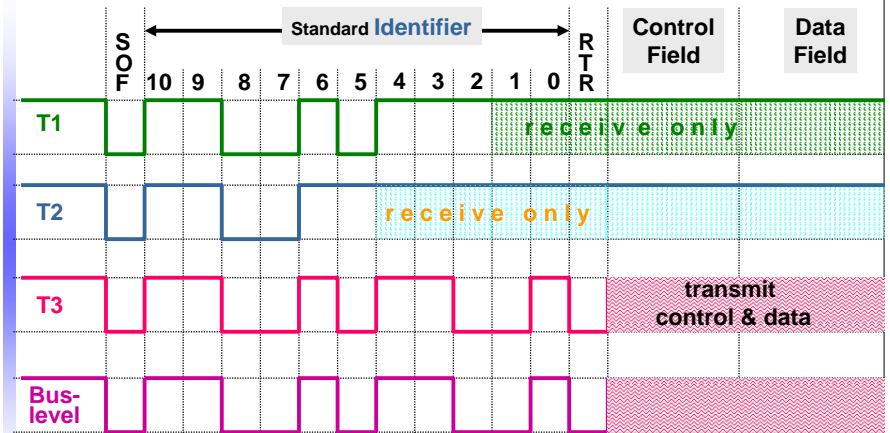


## CAN: Realisierung bitweise Arbitrierung



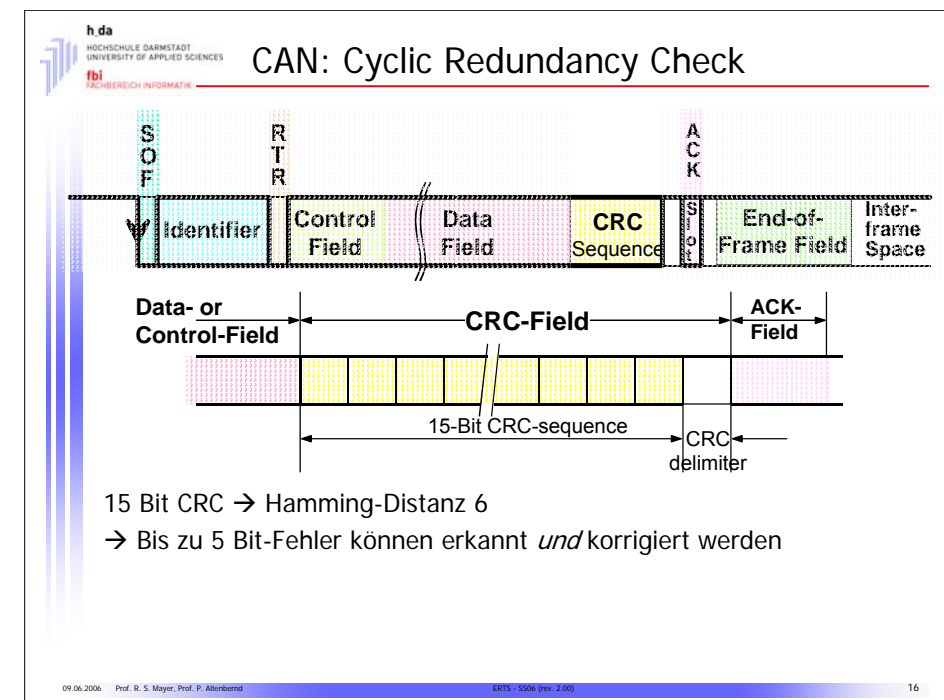
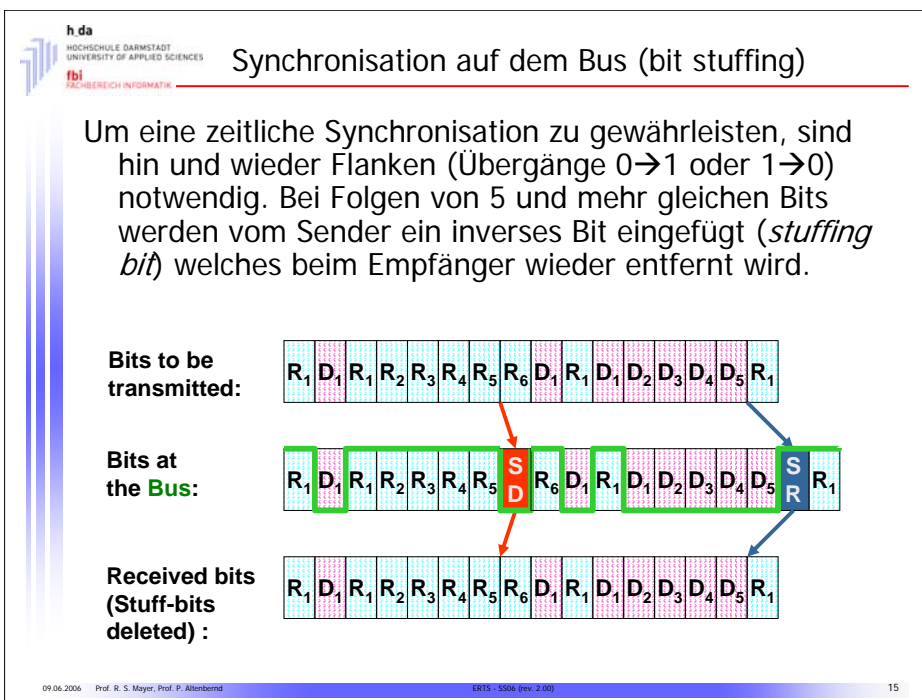
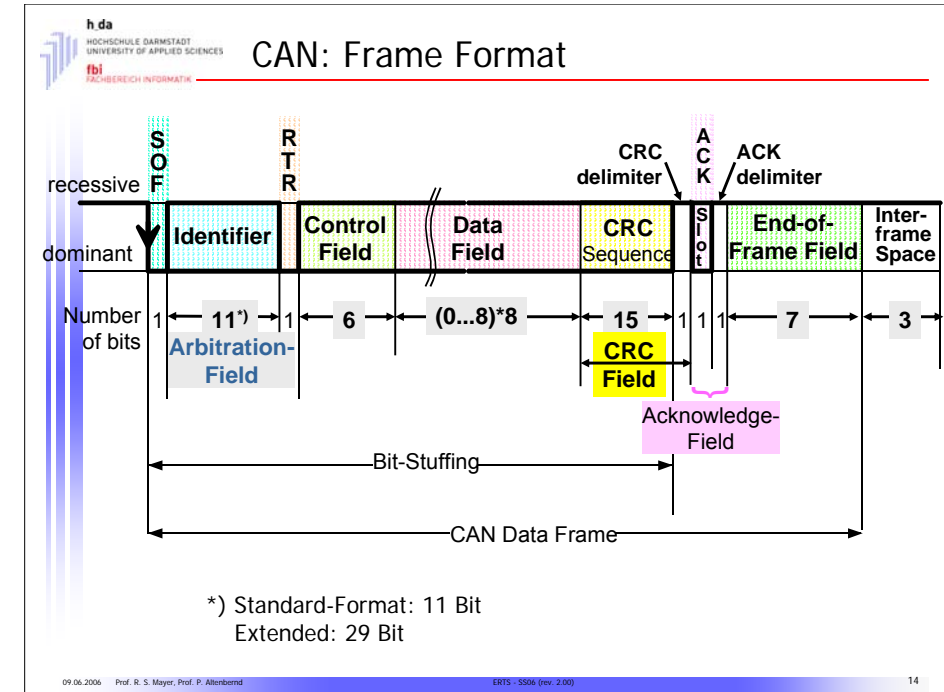
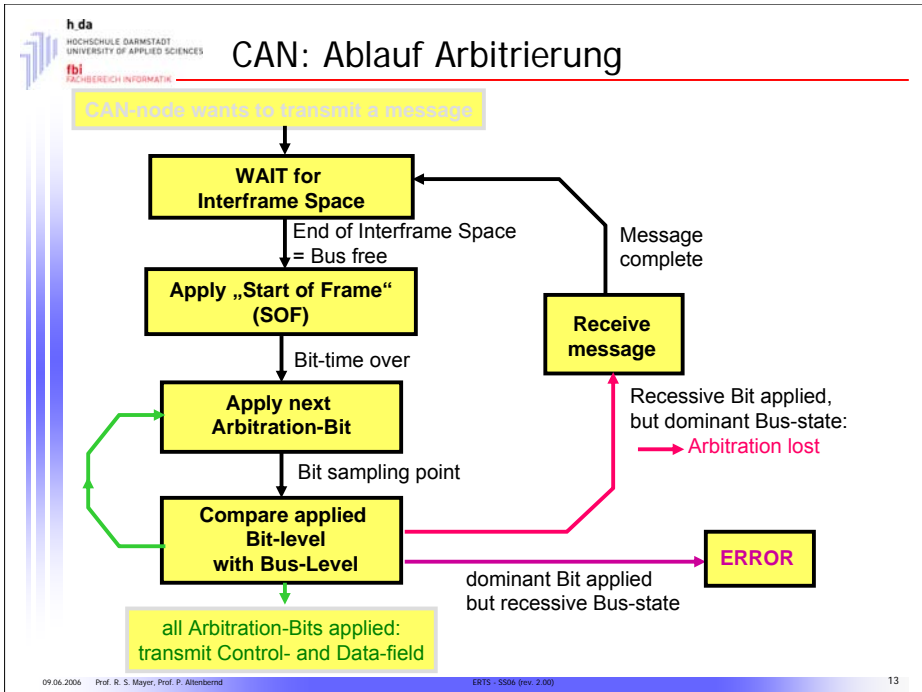
Verschaltung der Open-Collector-Ausgänge wirkt wie ein „wired“ AND.  
Dominantes Bit „0“ setzt sich gegenüber rezessivem Bit „1“ durch.  
→ CSMA/CD + Arbitration on Message Priority

## CAN: Realisierung bitweise Arbitrierung (2)

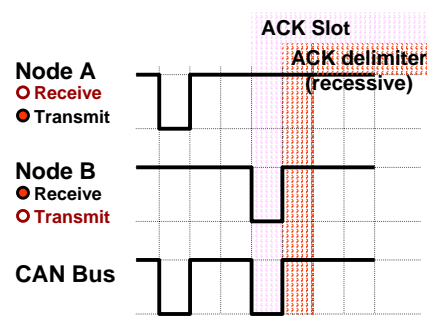
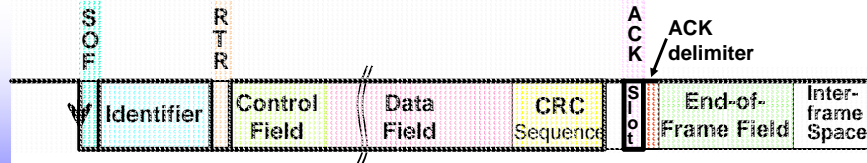


SOF: Start-of-Frame  
Identifier. 11 Bit (CAN 2.0A)  
29 Bit (CAN 2.0B)

RTR: Remote Transmission Request  
enthält Daten oder  
fordert zum Senden entspr. Daten auf



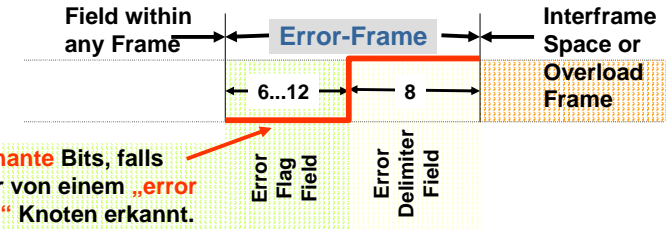
## CAN: Acknowledge



Alle Knoten checken CRC, Stuffing-Bits etc.  
Alle empfangenden Knoten quittieren – ungeachtet Korrektheit – mit einem dominanten Bit  
D.h. mindestens 1 Knoten hat korrekt empfangen  
Falls nicht, wird Sendung wiederholt  
Kein Error-Frame!

End-of-Frame: 7 rezessive Bits  
Entdeckt ein Knoten einen Fehler, sendet er einen Error-Frame aus

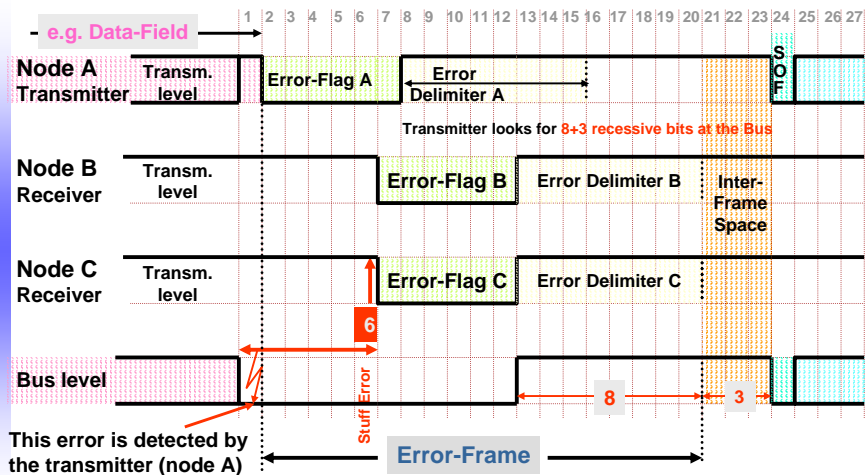
## CAN: Fehlerbehandlung



**Dominante Bits, falls Fehler von einem „error active“ Knoten erkannt.**

- „error active“ Knoten unterbricht bei Erkennung eines Fehler die Übertragung durch das Senden eines Error-Flags.
- „active error flag“ besteht aus 6 aufeinanderfolgenden dominanten Bits
- Alle Knoten erkennen eine Verletzung der Stopf-Bit (stuffing) Regel und senden ihrerseits einen Fehler-Rahmen
- → 6-12 dominante aufeinanderfolgende Bits auf dem Bus
- Nach Senden des „error delimitate“-Feldes → normaler Busbetrieb

## CAN: Fehlerbehandlung (2)



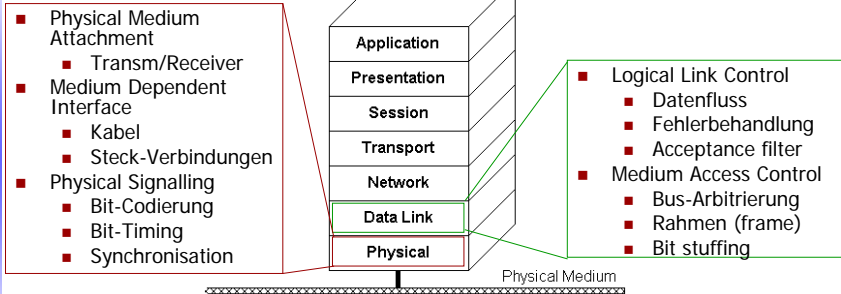
This error is detected by the transmitter (node A)

## CAN: Fehlererkennung

- Die Wahrscheinlichkeit  $\rho$ , dass ein fehlerhafter CAN-Standard-Frame nicht erkannt wird, beträgt:  
$$\rho = 4.7 \times 10^{-11} \times \text{Fehlerrate}$$
- Beispiel:  
1 Bit-Fehler pro 0.7 s,  
Übertragungsrate: 500 kBit/s,  
Betrieb: 8 h / d, 365 d / a (d:Tag, a:Jahr)  
→ statistisch: **1 unerkannter Fehler in 1000 Jahren**
- Extended CAN etwas schlechter als Standard CAN

## CAN: die physikalische Schicht

### The OSI Reference Model



## Bitraten/Buslänge-Beziehung (CAN)



### Bus

Bei CAN wird die Reaktion anderer Busteilnehmer auf das Aussenden einer Bitfolge erwartet. Ein von einem zweiten Sender ausgesandtes Bit muss in ca 2/3 der Bit-Zeit wieder beim ersten Sender eingetroffen sein, um dort eine Entscheidung zu ermöglichen. Bei verlorener Arbitrierung muss der erste Knoten innerhalb von 1/3 der Bitzeit auf Empfang umstellen.

Weiterhin ist die zu beachten:

- die Laufzeit auf der Busleitung
  - Die Laufzeit durch die Sende/Empfangs-Treiber
- Daraus ergibt sich die Abschätzung:

$$\frac{2}{3} \cdot t_{Bit} \geq 4 \cdot \tau_{TX/RX} + 2 \cdot \tau_{Line} \quad (Gl. 1)$$

## Bitraten/Buslänge-Beziehung (CAN) (2)

Buslänge [m]	max. Bitrate
40	1 MBit/s
100	500 kBit/s
500	125 kBit/s

Tab. 1: CAN Spezifikation

Geht man von  $\tau_{Line} = 17 \text{ cm/s}$  und  $\tau_{TXRX} = 25 \text{ ns}$  aus, erhält man folgende „Worst-Case-Designregel“:

$$Buslänge \times Bitrate < 40 \text{ m} \times 1 \text{ Mbit/s} \quad (Gl. 2)$$

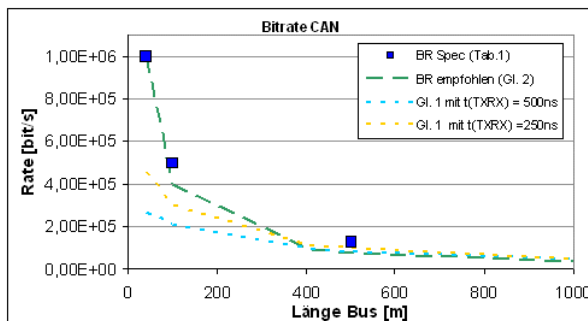


Diagramm: Abhängigkeit der maximal möglichen Bitrate in Abhängigkeit von der Kabellänge für CAN

Praktisches Limit für 1 MBit/s:  
Buslänge ca. 9 m

## CAN: Bit-Timing und Synchronisation

Prinzipiell müssen folgende Fälle bei der Kommunikation zwischen Knoten berücksichtigt werden:

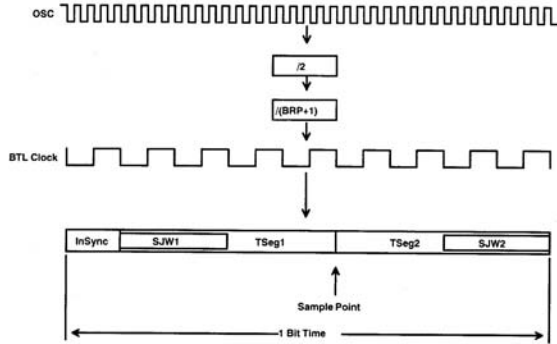
- Oszillator des Empfängers läuft etwas schneller/langsamer
- Laufzeit des Signals auf dem Bus

Man unterscheidet zwei Arten der **Synchronisation**:

- harte Synchronisation**: wird an der fallenden Flanke des Startbits durchgeführt (oder an weiteren fallenden Flanken). Legt den absoluten Beginn eines Bits fest.
- weiche Synchronisation**: wird innerhalb des CAN-Telegramms an Flanken durchgeführt, um die zeitliche Länge von Bits anzupassen.

Frage: Warum ist fallende Flanke dafür günstiger als ansteigende?

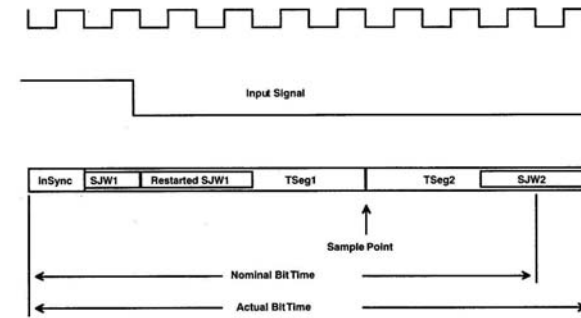
## Prinzip: Bit-Timing und Synchronisation



BTL: Bit Timing Logic  
BRP: Baud Rate Prescaler  
SJW: Synchronisation  
Jump Width

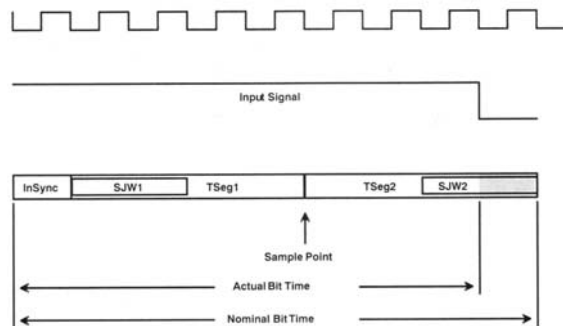
- Bitzeit ist in drei Segmente unterteilt: InSynch, TSEG1 und TSEG2
- Der Abtastzeitpunkt (Sample) ist immer genau zwischen TSEG1 und TSEG2
- Die programmierbaren Teile jedes Zeitsegments – das Synchronisationssegment SJW – werden zur Synchronisation benutzt.
- Die Flanke wird innerhalb des InSynch-Segments erwartet. In diesem Fall startet Empfänger TSEG1 und tastet an dessen Ende ab. Danach wird TSEG2 gestartet. Nach dessen Ablauf wird das nächste Bit erwartet.

## Synchronisation langsamerer Sender



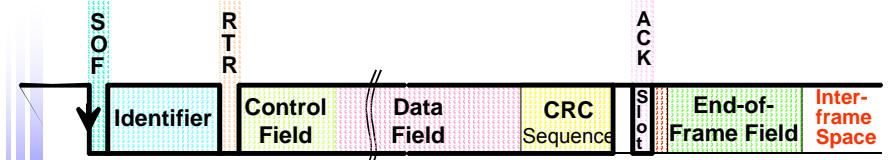
- Bitzeit des Senders länger als die des Empfängers (Sender hat niedrigere Übertragungsrate)
- Flanke wird erst im SJW1-Segment detektiert
- SJW1 wird erneut gestartet.
- Resynchronisation nur bei Flankenwechsel, sonst läuft normales Bit-Timing ab

## Synchronisation schnellerer Sender



- Bitzeit des Senders kürzer als die des Empfängers (Sender hat höhere Übertragungsrate)
- Flanke am Ende des Bits wird schon im SJW2-Segment detektiert
- SJW2 wird sofort gestoppt und
- Empfänger startet Bitzeit für das nächste Bit gestartet.
- Erneute Synchronisation erfolgt erst wieder mit einer Flanke

## CAN: Rahmen Format



- 1 Startbit
- + 11 Identifierbits
- + 1 RTR Bit
- + 6 Controlbits
- + 64 Datenbits
- + 15 CRC Bits
- + 1 CRC Delimiter
- + 19 (maximale Anzahl) Stuffbits
- + 1 ACK Slot
- + 1 ACK Delimiter
- + 7 EOF Bits
- + 3 IFS (Inter Frame Space) Bits
- = 130 Bit

CAN 2.0A Identifier = 11 Bit

→ Summe ≤ 130 Bit

CAN 2.0B Identifier = 29 Bit

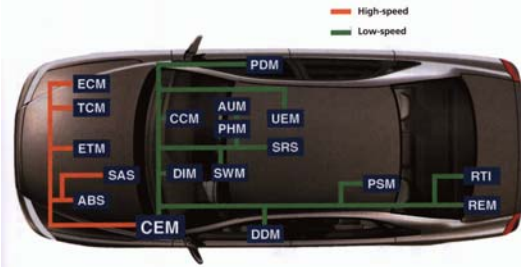
→ Summe ≤ 154 Bit

Bei 1MBit pro Sekunde (1 bit/μs):

Übertragungsdauer bei

- Standard-Frame: ≤ 130 μs
- Extended Frame: ≤ 154 μs

## Verteilte Systeme



Warum verteilte Systeme?

- Komponenten sind physikalisch verteilt
- Verteilte Systeme sind leistungsfähiger/preiswerter als eine einzige zentrale CPU
- Sicherheit: Ausfall betrifft einzelne Komponente nicht das ganze System.
- Redundanz einfacher möglich
- System skaliert: einzelne Komponenten können hinzugefügt werden.

## Response Time Analysis bei CAN

Modell:

- Die Prozesse  $P_j \in \mathcal{P}$  versenden Nachrichten
- Prozess  $P_j$  lässt sich beschreiben zu  $P_j = \langle T_j, C_j, J_j \rangle$ , dabei ist
  - $T_j$  die Periode (minimale Zeit zwischen zwei Sendungen)
  - $C_j$  minimale Übertragungsdauer für einen Rahmen (Frame)
  - $J_j$  der Jitter (Schwankung in der Periodizität  $T_j$ )
  - $R_j$  ist die Response Zeit des Prozesses  $P_j$

Für die Berechnung der Response-Zeit auf dem CAN-Bus kann gut das Modell des **Fixed Priority Scheduling** auf einem Prozessor herangezogen werden.

Sei Priorität  $\pi_i < \pi_j$ , ist die Anzahl  $n_i$  der Interferenzen, die  $P_i$  durch den Prozess  $P_j$  mit höherer Priorität erleiden kann

$$n_i = \left\lceil \frac{R_i}{T_j} \right\rceil \quad (\text{die Klammer bedeutet die } \mathbf{Ceiling}\text{-Funktion: kleinste Ganzzahl größer oder gleich der gebrochenen Zahl in der Klammer})$$

## Response Time Analysis bei CAN (2)

Die maximale Interferenz  $I_i$  beträgt also:

$$I_i = \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

In einem realen System kann aber nicht davon ausgegangen werden, dass alle Sendeprozesse zur gleichen Zeit und mit konstanter Periode gestartet werden  $\rightarrow$  **Jitter**.

Betrachten wir den Fall, dass  $P_j$  um  $J_j$  früher startet: Dann hat  $P_i$

- 2 Interferenzen, wenn  $R_i > T_j - J_j$ ,
- 3 Interferenzen, wenn  $R_i > 2T_j - J_j, \dots$ ,
- $n$  Interferenzen, wenn  $R_i > (n-1)T_j - J_j$

Daraus folgt:  $\frac{R_i + J_j}{T_j} > (n-1)$  oder  $(n-1) = \left\lceil \frac{R_i + J_j}{T_j} \right\rceil - 1$

Die maximale Interferenz  $I_i$  mit **Jitter**  $J_j$  beträgt also:

$$I_i = \left\lceil \frac{R_i + J_j}{T_j} \right\rceil C_j$$

## RTA bei CAN

Scheduling bei CAN ist Fixed Priority und nicht-preemptiv:

- Die Prozesse  $P_j \in \mathcal{P}$  versenden Nachrichten
- Prozess  $P_j$  lässt sich beschreiben zu  $P_j = \langle T_j, C_j, J_j \rangle$ , dabei ist
  - $T_j$  die Periode (minimale Zeit zwischen zwei Sendungen)
  - $C_j$  minimale Übertragungsdauer für einen Rahmen (Frame)
  - $J_j$  der Jitter in Queuing oder Release
  - $B_j$  die Blockierung durch alle Prozesse kleiner Priorität
  - $w_j$  die Interferenz durch alle Prozesse höherer Priorität
  - $R_i = J_i + w_i + C_i$ : Response Zeit des Prozesses  $P_i$
  - $\tau_{\text{bit}}, \tau_{\text{res}}$  die Zeit(unschärfe) für 1 Bit. Bei 1MBit/s  $\rightarrow 1 \mu\text{s}$

$$R_m = J_m + w_m + C_m \quad (\text{Gl. 3) worst case Release Time}$$

$$w_m = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{w_m + J_j + \tau_{\text{bit}}}{T_j} \right\rceil C_j \quad (\text{Gl. 4) wobei } hp(m) \text{ alle Nachrichten mit höherer Priorität als } m \text{ sind}$$

## RTA bei CAN (2)

In Gl. 4 kann die Blockierung im worst case Fall wie folgt angegeben werden:

$$B_m = \max_{\forall k \in lp(m)} (C_k) \quad (\text{Gl. 4}) \quad \text{wobei } lp(m) \text{ die Nachrichten mit kleinerer Priorität als } m \text{ sind}$$

(Gl. 2) Gl. 4 kann rekursiv beschrieben werden:

$$w_i^{n+1} = B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^n + J_j + \tau_{res}}{T_j} \right\rceil C_j$$

Dabei wird  $w_i^0 = 0$  gesetzt. Iteration ist beendet, wenn  $w_i^{n+1} = w_i^n$

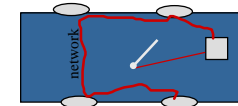
$C_m$  ist die maximale Zeit, die zur Übertragung der Nachricht  $m$  benötigt wird. CAN-Rahmen haben einen Overhead von 47 Bit, eine Stuffing-Breite von 5 Bit, jedoch werden nur maximal 34 von 47 Bits des Overhead gestopft.

$s_m$ : Anzahl Daten-Bytes (0-8)

$$C_m = \left( \left\lceil \frac{34 + 8s_m}{5} \right\rceil + 47 + 8s_m \right) \tau_{bit} \quad (\text{Gl. 6})$$

## Problem bei eventgesteuertem Bus

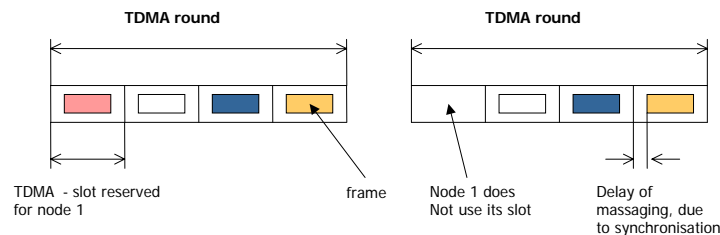
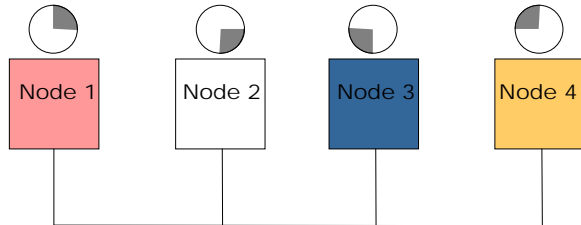
- nur Nachricht mit höchster Priorität ist determiniert.
- niedrige Prioritäten können verdrängt werden
- höhere Prioritäten können durch niedrigere zeitweilig blockiert werden.
- Reihenfolge von Nachrichten nicht determiniert
- Einfluss von Jitter
  - selbst kleine Schwankungen der Periodizität der einzelnen Nachrichten haben ungünstigen Einfluss auf Response-Zeit
  - nicht periodisch auftretende Nachrichten haben negativen Einfluss



→ Übergang zu zeitgesteuerten Protokollen bei kritischen Anwendungen (z.B. drive-by-wire, X-by-wire)

## Zeitgesteuerte Protokolle

Zeitsteuerung (Time Triggered Protocol) kann beispielsweise durch TDMA (Time Division Multiplex Access) erreicht werden



## Zeitgesteuerte Protokolle

- Übergang von Ereignissteuerung (event triggered) → Zeitsteuerung (Time Triggered Protocol)
- Sicherheitskritische Anwendungen besser beherrschbar
- Zeitgesteuert := alle Aktivitäten werden durch das Fortschreiten der Zeit ausgelöst
- Abhängigkeit von einer Systemzeit.
- Sendezeitpunkt aller Nachrichten von vornherein bekannt „Bus-Fahrplan“
- Entlastung der Anwendungssoftware
- Bei CAN Übergang zu TTCAN Erweiterung des Protokolls um einen **Session Layer** (OSI) Datentransport und Physikalische Schicht bleiben gleich.

## TTCAN: Erweiterung von CAN zum TTP

### Beispiel: TTCAN

- TTCAN ist vollständig rückwärtskompatibel zu CAN, da CAN physical + link layer verwendet wird
- Somit auch CAN-Knoten direkt als Empfänger/Sender verwendbar
- Standardisierung erfolgt unter ISO 11898-4
- Damit aber auch Beschränkung der Datenrate auf 1 Mbps (Brutto!)

### Grundprinzip der Erweiterung erfolgt in 2 Schritten

- Schritt 1:
  - Einführung einer Referenznachricht einer Zeitbasis
  - Ausfallsicherheit wird durch Redundanz dieser Zeitbasen erreicht
- Schritt 2:
  - Eine globale Zeitbasis wird eingeführt
  - Korrektur der zeitlichen Drift zwischen den CAN Controllern wird eingeführt

## TTCAN Mechanismen

### TTCAN Mechanismen

- Verfügungenstellen von Zeitbasen
  - Local\_Time, Cycle\_Time, and Global\_Time
- Drei verschiedene Arten von Zeitfenstern:
  - Exklusive, arbitrierend, frei
- Zeitgesteuerte und periodische Kommunikation werden durch eine zeitgesteuerte Referenznachricht gesteuert
- Die System Matrix beschreibt die zeitliche Abfolge der Nachrichten in jedem Grundzyklus

### Einführung von Zeitfenstern ("TDMA")

- Exklusive Zeitfenster
- Exklusiv reserviert für eine Nachricht, keine konkurrierenden Netzwerkzugriffe möglich
- Nachrichten im exklusiven Zeitfenster werden nicht verzögert (=deterministisch)

### Arbitrierungsfenster

- Während dieses Zeitfensters erfolgt die klassische bitweise Arbitrierung

### Freie Fenster

- Reserviert für mögliche zukünftige Erweiterungen

## TTCAN Referenznachricht

### Referenznachricht

- Die Referenznachricht wird durch ihren Identifier gekennzeichnet

### Im Schritt 1

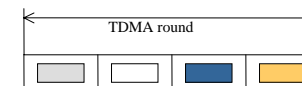
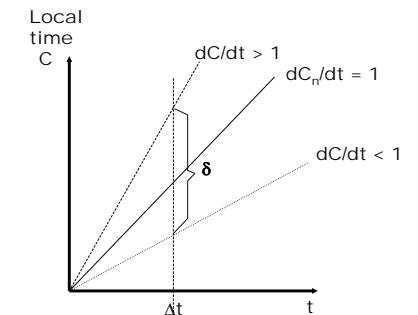
- Die Referenznachricht enthält 1 Byte Kontrollinformation. Der Rest der CAN Nachricht kann für Datentransfer genutzt werden

### Im Schritt 2

- Die Referenznachricht enthält zusätzliche Kontrollinformation, wie bspw. die globale Zeitbasis des TTCAN Zeitmasters
- Die Referenznachricht enthält 4 Bytes an Kontrollinformation. Die verbleibenden 4 Bytes können für Datentransfer genutzt werden.

## TTP Uhren-Synchronisation

- Uhren synchronisiert in den Grenzen  $\delta$
- Justierung der Uhren der Knoten hoch oder runter (Uhren sind durch Zähler realisiert)
- MSGs für Synchr.
  - Expected arrival (EA)
  - Real arrival (RA)
  - Alle Uhren auf Mittelwert



EA	0	25	50	75
RA	0	23	52	79
Diff	0	2	-2	-4

## TTCAN Beschreibung

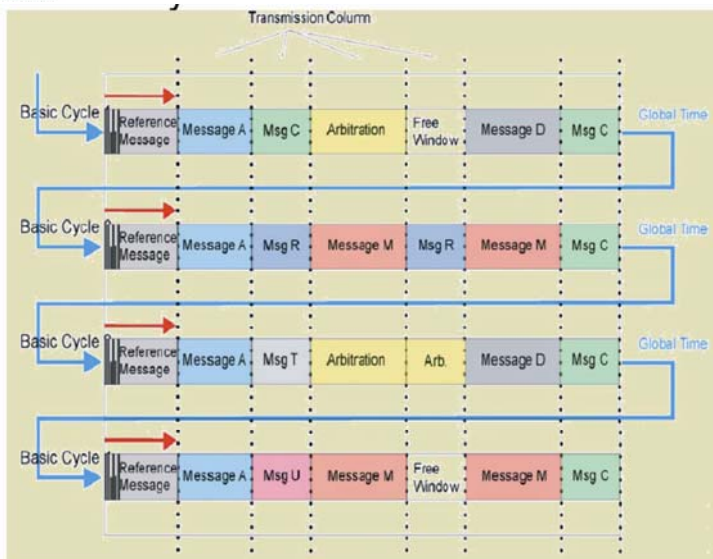
- Systemweit synchronisierte Zeitbasis
- Referenznachricht mit bestimmten Identifier
- Alle Knoten synchronisieren damit ihre eigene Uhr
- Sobald jeder Knoten seine Zeitmarke erreicht hat, versendet er seine Nachricht
- Zeit zwischen zwei Referenznachrichten bildet Basiszyklus
- Mehrere Basiszyklen sind erlaubt.
- Abfolge aller Basiszyklen bilden Gesamtzyklus
- → unterschiedliche Wiederholzyklen sind damit realisierbar
- Sogar nichtperiodische Referenznachrichten sind möglich z.B. Drehzahl der Kurbelwelle → Berechnung Einspritzzeitpunkt
- TTCAN ist Hybrid: neben Zeitsteuerung auch weiterhin ereignisgesteuerte Nachrichten möglich
- Uhren dennoch immer leicht unterschiedlich
- TTCAN bietet Möglichkeit Uhren auch zwischen zwei Referenznachrichten zu resynchronisieren.  
→ Genauigkeit bis 1 µs möglich

## TTCAN Systemmatrix

### TTCAN System Matrix

- Bei TTCAN müssen nicht alle Grundzyklen gleich sein
- Unterschiedliche Grundzyklen werden durch den Zyklenzähler unterschieden
- Der Zyklenzähler wird in jedem Takt erhöht bis zu einem Maximalwert. Danach erfolgt ein Zurücksetzen
- Die System Matrix ergibt sich aus der Kombination aller unterschiedlicher Zyklen, und
- gibt einen Überblick über die Konfiguration des TTCAN Netzwerkes

## TTCAN Systemmatrix (2)



## TTCAN Systemmatrix (3)

- TTCAN Erweiterung erfolgt durch 2 zusätzliche funktionale Blöcke:
  - Trigger Memory
  - Frame Synchronization Entity (FSE)
- Trigger Memory speichert die Zeitpunkte der Systemmatrix, die zu den Nachrichten im RAM gehören
  - Diese Daten werden der Frame Synchronization Entity zur Verfügung gestellt
- Die FSE ist eine Zustandsmaschine, welche die TT Kommunikation kontrolliert. Sie synchronisiert sich mit den Referenznachrichten und kontrolliert die Zyklen. Die FSE besteht aus 6 Blöcken:

**TBB:** Time Base Builder

**TSO:** Time Schedule Organizer

**AOM:** Application Operation Monitor

**CTC:** Cycle Time Controller

**MSA:** Master State Administr.

**GTU:** Global Time Unit

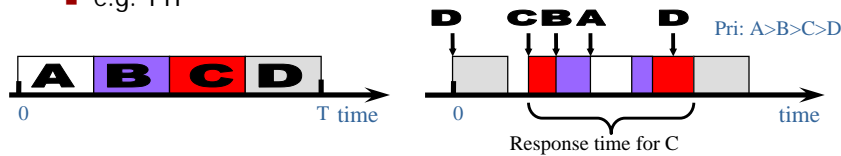
## Vergleich event-/zeitgesteuert

### Time triggered

- TDMA
- Pre-Scheduled
  - Predictable
  - Testable
- Static
- e.g. TTP

### Event driven

- CSMA/CR
- Priority driven
  - Online Scheduling
  - Flexible
  - e.g. CAN



## TTP - CAN: A comparison

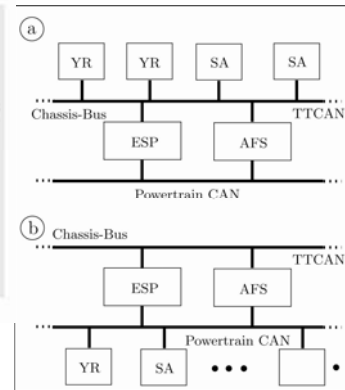
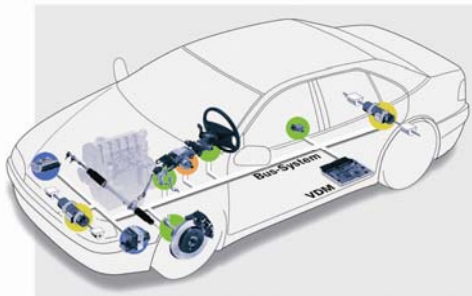
### TTP

- time triggered: Static TDMA
- Complete package (costly)
- easy to analyse
- high overhead for simple systems
- overhead not as bad for complex systems
- Ultra-reliable systems
- Includes DS-functionality
  - Clock synchronisation
  - Fault tolerance

### CAN

- event driven: CSMA/CR
- Priority-based
- just a protocol (cheap)
- difficult to analyse
- efficient for simple systems
- efficiency lower with complex systems
- > FlexRay: combination

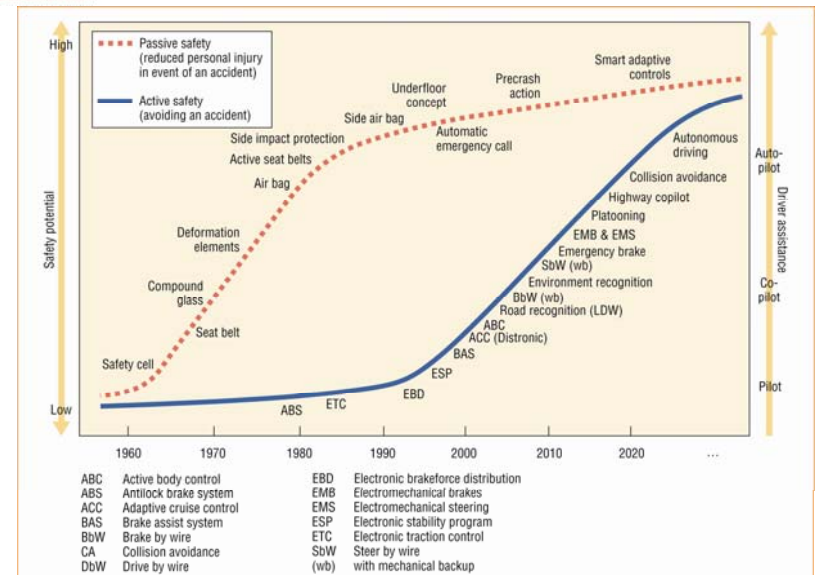
## TTCAN: Anwendungen und Redundanz



VDM Vehicle Dynamics Management  
 ESP Electronic Stability Program  
 YR Yaw Rate Sensor  
 SA Steering Angle Sensor  
 AFS Active Front Steering  
 EAR Electronic Active Roll Stabilizer

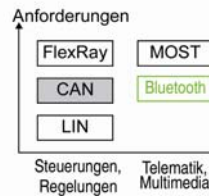
*Kritische Systeme können redundant miteinander vernetzt werden  
 → Ausfallsicherheit*

## Ausblick auf zukünftige Automotive Syst.

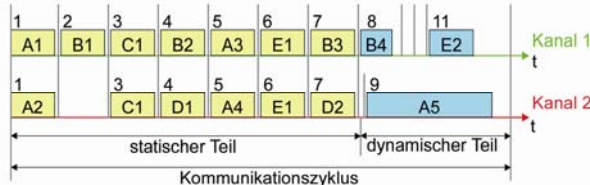
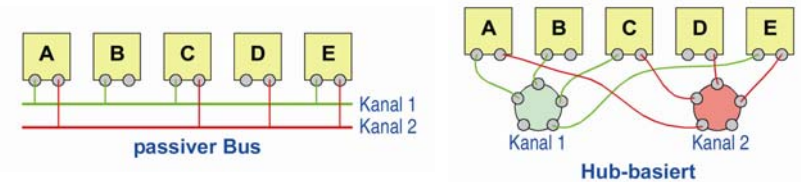


## Weitere Busse im Automobilbau

- CAN ist derzeit dominierend
- MOST (Media Oriented System Transport) für Infotainment und (Verkehrs-)Telematik-Bereiche
- LIN (Local Interconnect Network)
  - Einsatz wo geringe Kosten und dafür auch geringere Leistung nötig,
  - lose Ankopplung von Sensoren und Aktuatoren
- FlexRay: Konsortium vieler Automobil-Firmen zum Einsatz eines neuen Busses
  - 10 MBit/s Übertragungsgeschwindigkeit
  - echtzeitfähig
  - fehlertolerante Konfiguration möglich



- von BMW und DaimlerChrysler für X-by-Wire-Systeme entwickelt
- ähnlich zu TTCAN von Bosch (ebenfalls für X-by-Wire-Systeme)
  - → hat sich durchgesetzt (Bosch unterstützt auch FlexRay)
- High-End-Bus im KFZ-Steuerungsbereich:
- synchrone und asynchrone Übertragung von bis zu 10 MBit/s
- Kupfer- und Lichtwellenleiter möglich
- Topologie: - passiver Bus (bis 3 MBit/s)
  - - sternförmig über Hubs (bis 10 MBit/s)
  - - optionaler zweiter Kanal für - Redundanz (Ausfallsicherheit)
  - - Erhöhung der Bandbreite



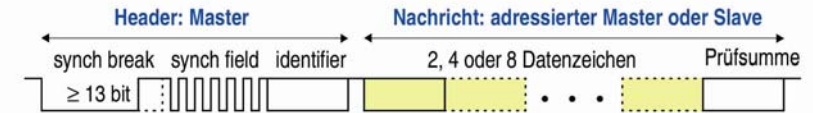
- in mehrere Zeitfenster unterteilt (vgl. TDMA)
- Zyklus beginnt mit einem SYNC-Signal einer zentralen Station zur Synchronisation aller Stationen (exakte globale Zeitbasis)
- statischer Teil mit fester Zuordnung der Zeitschlitze zu Nachrichten
- dynamischer Teil für prioritätsbasierte Übertragung asynchroner Nachrichten
- → zur Busauslastung in diesem Teil beginnen die Stationen nach kurzer Timeout-Zeit mit neuem Zeitschlitz, wenn kein Übertragungswunsch vorliegt
- deterministisch für kritische X-by-Wire-Daten (feste Zeitschlitze im statischen Teil), als auch der maximalen Ausnutzung der zur Verfügung stehenden Bandbreite

## Systembusebene: MOST (Media Oriented Systems Transport)

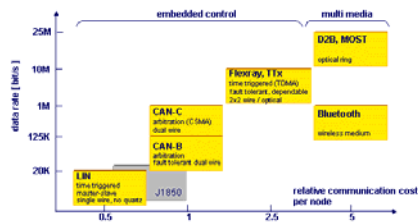
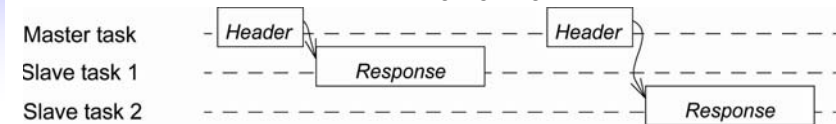
- seit 1998 für Multimedia und Telematik (Infotainment-Systeme)
  - Festlegung aller Protokollschichten (inkl. Anwendungsschicht)
- Medium: preisgünstigen Kunststoff-Lichtwellenleiter (polymer optical fiber, POF)
- Topologie: - Ring-, Stern- und Kettenstrukturen
  - bis 64 Knoten (Plug&Play)
  - eindeutige Send- und Empfangsadressen für asynchrone Daten
- Unterstützung von synchroner und asynchroner Übertragung
  - synchrone Echtzeitdaten: Reservierung fester Kanäle (vgl. TDMA bei FlexRay)
    - (Daten werden ohne vorgegebenes Format und Zieladresse verschickt, damit alle interessierten Empfänger dieselben Daten annehmen können [z. B. zwei Fond-Displays das selbe Bild])
  - asynchronen Daten- und Kontrollnachrichten: CSMA/CA über Adressen
- Datenrate: - 24 MBit/s für synchr. Audio- und Videodaten (150 MBit/s bereits als Prototyp)
  - 14 MBit/s für asynchrone Daten
  - asynchroner Steuerkanal mit 700 kBit/s zur Gerätesteuerung und MMI

## Sensor-Aktuator-Ebene: LIN (Local Interconnect Network)

- „Billigbus“ unterhalb von CAN für Vernetzung von einfachen Sensoren (z. B. Temperatur- und Feuchtigkeitssensoren) und Aktuatoren (z. B. Beleuchtungselemente) einzelner Subsysteme
- Einsatz: wo Bandbreite und Flexibilität von CAN nicht benötigt wird  
→ im Allg. werden die mit LIN vernetzten Sub-Systeme (z. B. Türen, Sitze, Lenkrad) an übergeordnete, CAN-basierte Netzwerke (Body, Cassis) angeschlossen
- Master-Slave-Bus (eine Master-Station)  
→ Echtzeitfähigkeit durch Master-Slave-Kommunikation und Rahmenformat
- Medium: 12 V-Eindrahtbus
- Topologie: Linienbus mit typ. bis zu 16 Stationen  
(keine feste Obergrenze)
- Bandbreite: max. 20 kBit/s
- Kodierung: UART-Zeichen
- Nachrichten: keine Zieladressen, sondern nachrichtenorientiert (wie CAN)

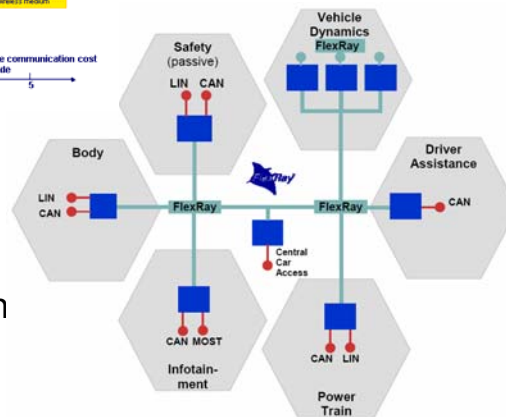


- 1. Master:
  - Rahmenbeginn: 13 Bit langer dominanter Pegel (*sync break*)
  - Synchronisation: wechselnde 1-0 Folgen (*sync field*)
  - 6 Bit lange Nachrichten-ID
- 2. Master oder ein Slave, der über die Nachrichten-ID angesprochen wird:
  - 1 bis 8 Datenbytes
  - Prüfsumme.
- Kommunikationsmöglichkeiten (*immer vom Master initiiert*)
  - vom Master zu einem oder mehreren Slaves
  - von einem Slave zum Master und/oder anderen Slaves
  - → direkte Slave-to-Slave-Übertragung möglich



Kosten

Visionen



- [1] Real-Time Systems and Programming Languages; Alan Burns, Andy Wellings; Pearson-Addison Wesley; 3rd Ed 2001
- [2] Hard Real-Time Computing Systems; Giorgio C. Buttazzi; Springer; 2005
- [3] Analysing Real-Time Communications: Controller Area Network (CAN); K. W. Tindell, H. Hansson A. J. Wellings; 1994
- [4] Informations- und Kommunikationstechnik im Kraftfahrzeug, Michael Bolle
- [5] Echtzeitkommunikation, Bernd Schürmann