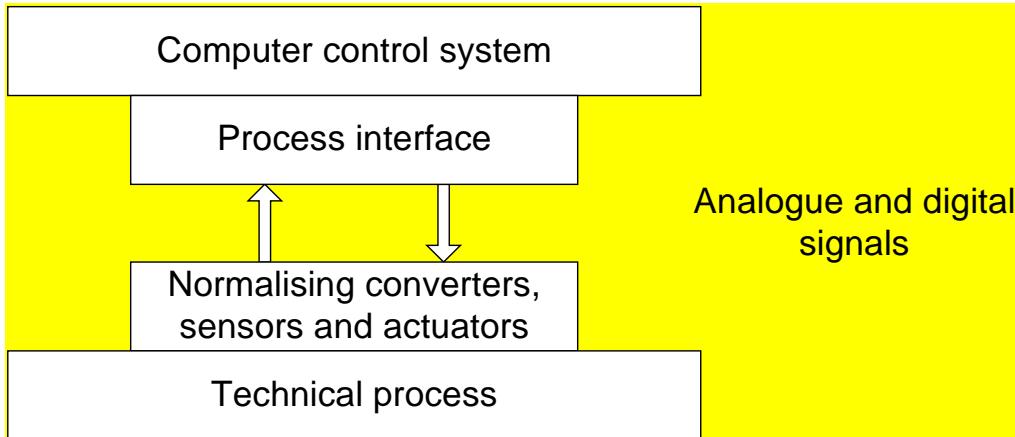


2.4 Prozess-Schnittstellen

Anbindung des Kontrollrechnersystems an den technischen Prozess.



Sensoren:

wandeln physikalische Größen (z. B. Druck, Temperatur, Geschwindigkeit) in elektrische Signale um.

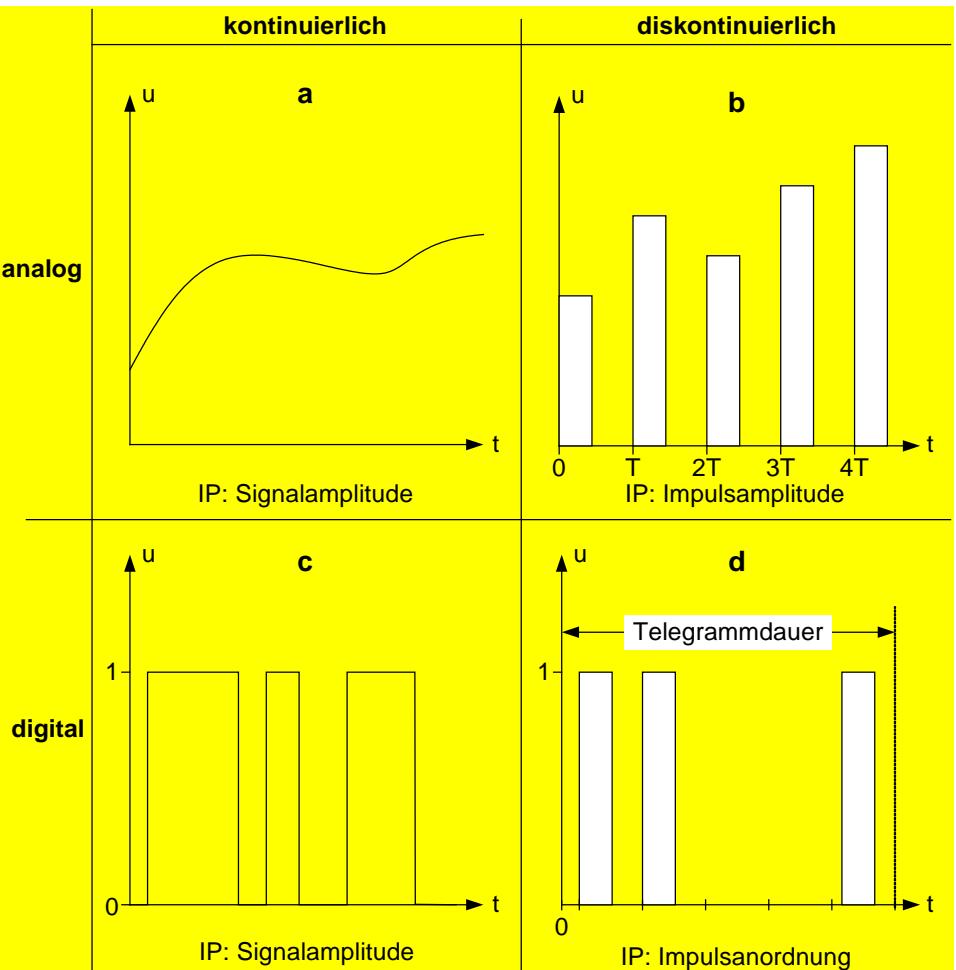
Aktoren:

wandeln elektrische Signale in physikalische Größen (Temperaturveränderung, mechanische Bewegung, Flussänderung über Ventile etc.).

Einzelne Sensoren und Aktoren arbeiten mit unterschiedlichen elektrischen Signalen (mV...kV, μ A...kA).

- Normalisierung auf gemeinsamen Spannungs-/Strombereich an der Prozessschnittstelle (z. B. 0...24V, 4...20mA)
- Galvanische Trennung des Rechners vom technischen Prozess (Schutzgründe)

Signalformen



- **analog-kontinuierliches Signal**
beliebige Zwischenwerte zu beliebigen Zeitpunkten (z. B. Thermoelement, Photozelle)
- **analog-diskontinuierliches Signal**
beliebige Zwischenwerte zu festen Zeitpunkten (z. B. CCD-Zeilen)
- **digital-kontinuierliches Signal**
nur bestimmte Werte zu beliebigen Zeitpunkten. Spezialfall binär-kontinuierliches Signal: Werte 0,1 (z. B. Taster)
- **digital-diskontinuierliches Signal**
nur bestimmte Werte zu festen Zeitpunkten (z. B. synchrone serielle Schnittstelle).

Interne Darstellung im Rechner

Umwandlung je nach Sensor erforderlich:

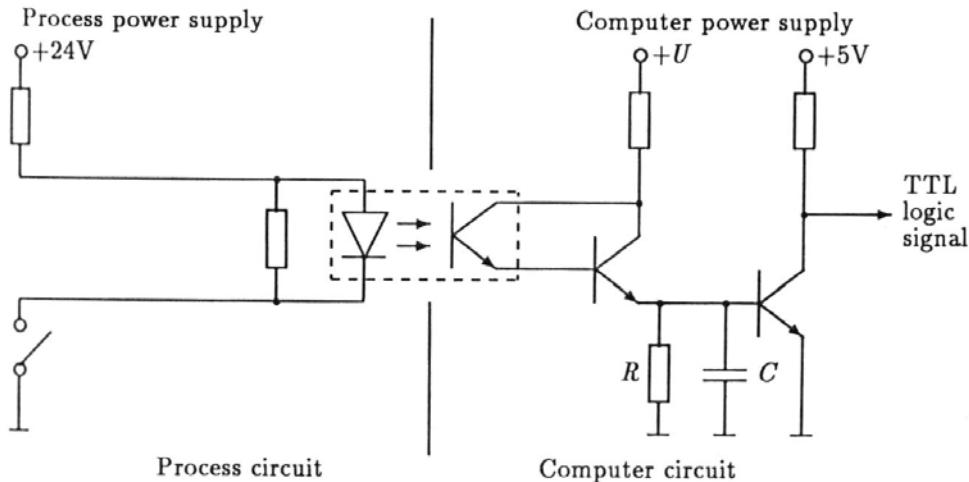
analog \Leftrightarrow digital (A/D- bzw. D/A-Wandler)

zeitkontinuierlich \Leftrightarrow zeitdiskontinuierlich (Abtastung bzw. Signalkodierung)

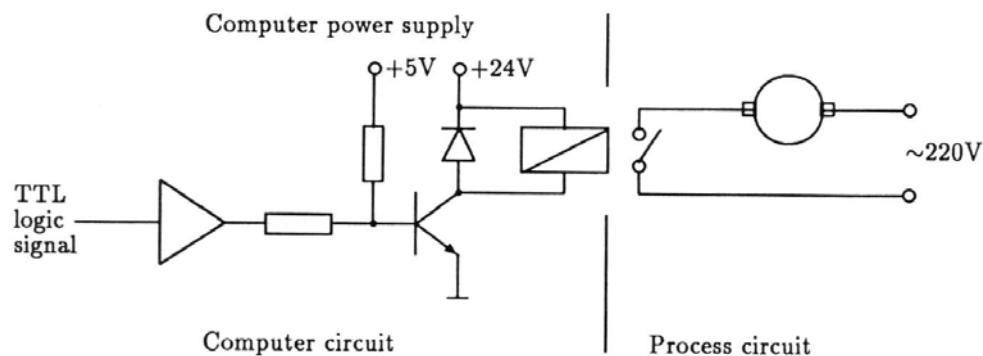
Auch umgekehrt für Aktor notwendig

Digitale Signale

Galvanische Trennung und Pegelanpassung

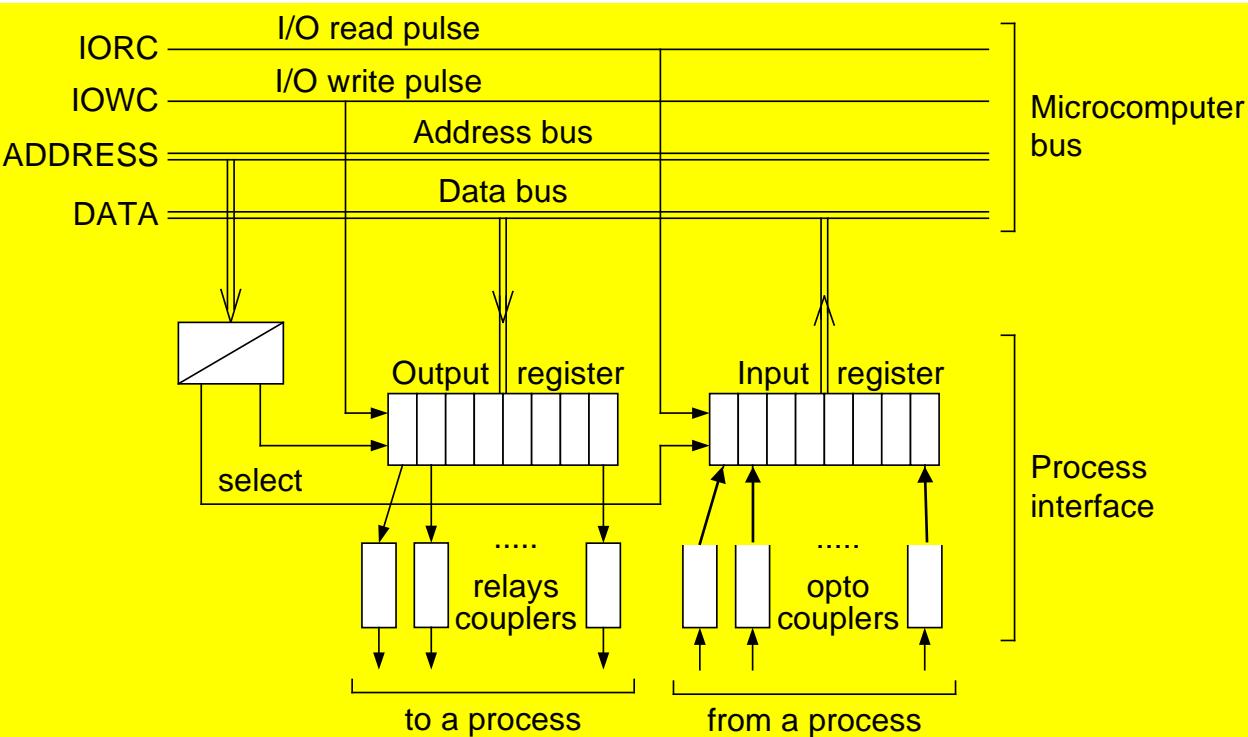


Optokoppler: Prozess steuert eine Leuchtdiode, deren Licht mittels eines Phototransistors empfangen und in ein TTL-Signal umgewandelt wird.



Relais: Rechner (TTL-Pegel) steuert eine Relaisspule an, deren Schaltkontakt ggf. hohe Ströme/Spannungen schaltet.

Abtastung



Prozesssignale (TTL-Pegel) werden in Eingabe- bzw. aus Ausgabe- Register unter Kontrolle des Rechners eingegeben bzw. ausgegeben (Parallele Ports).

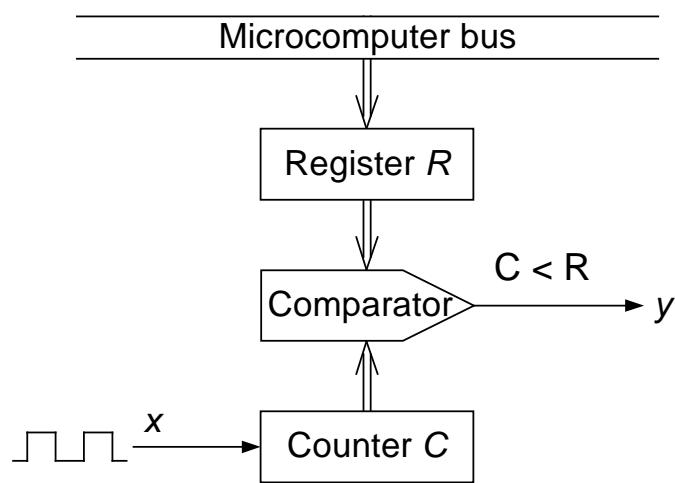
Weitere Varianten:

- Hw-Zähler für binäre impulsförmige Signale (Ausgabe)
- Hw-Timer zum Messen von Impulsabständen oder Impulslängen (Eingabe).
- Interrupt-Eingabe für dringende asynchrone oder sehr seltene Signale (z. B. Alarmmeldungen)
- etc.

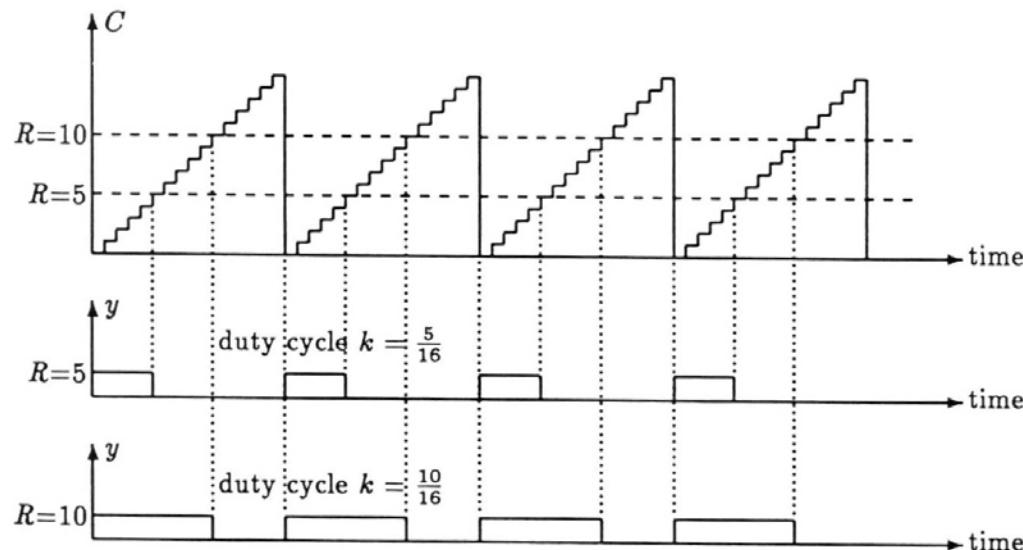
Vgl. Peripheriekomponenten typischer Mikrocontroller wie ATmega

Beispiel: Pulsbreiten-Modulator

a)



b)



Ausgangssignal

$$y =$$

$$\begin{cases} 0 & \text{für } C \geq R \\ 1 & \text{für } C < R \end{cases}$$

mit Frequenz

$$f_y = \frac{f_x}{2^n}$$

$$\text{und Impulsbreite} \quad k = \frac{R}{2^n}$$

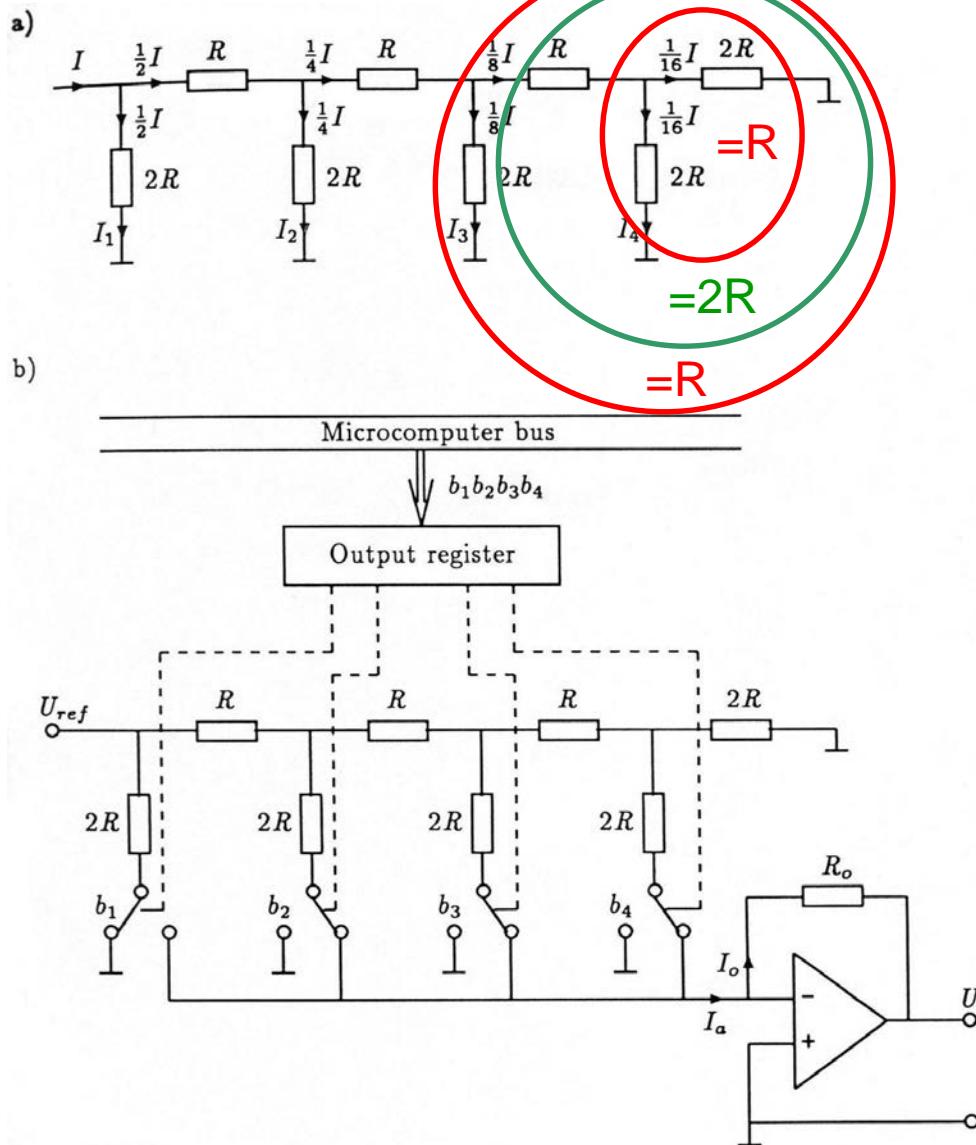
R: Register der Länge n Bits
x: Taktsignal des Zählers mit Frequenz f_x

Anwendung z. B. zur Motorensteuerung

Analoge Signale

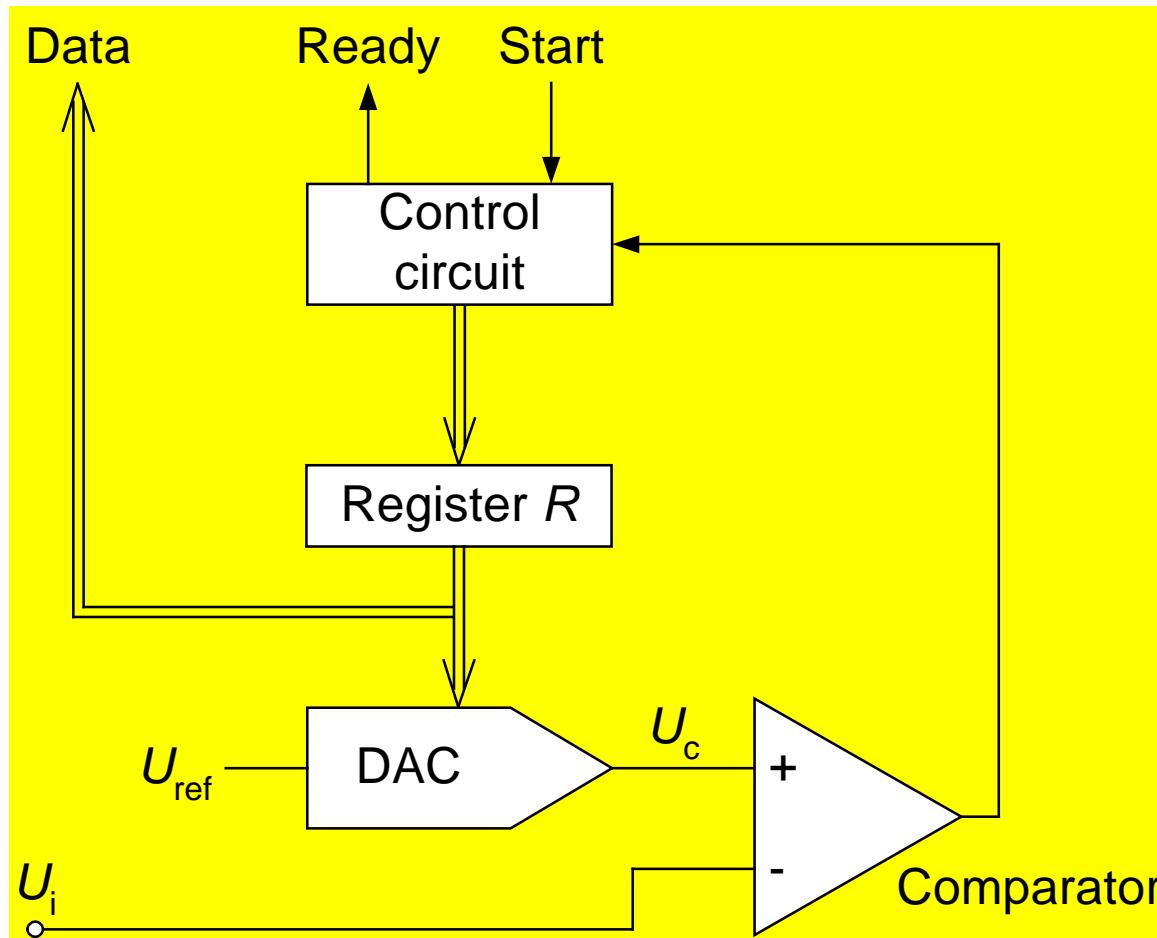
- Analog/Digital-Wandler für die Eingabe bzw. Digital/Analog-Wandler für die Ausgabe (verschiedene Varianten)
- Galvanische Trennung z. B. über Optokoppler
- Begrenzte Auflösung (z. B. $N = 8$ Bit, 12 Bit) und Abtastfrequenz (typ. kHz-Bereich)

Digital-Analog-Wandler mit Widerstands-Leiter



$$\frac{U_o}{U_{ref}} = \frac{B}{2^N}$$

Analog-Digital-Wandler mit sukzessiver Approximation

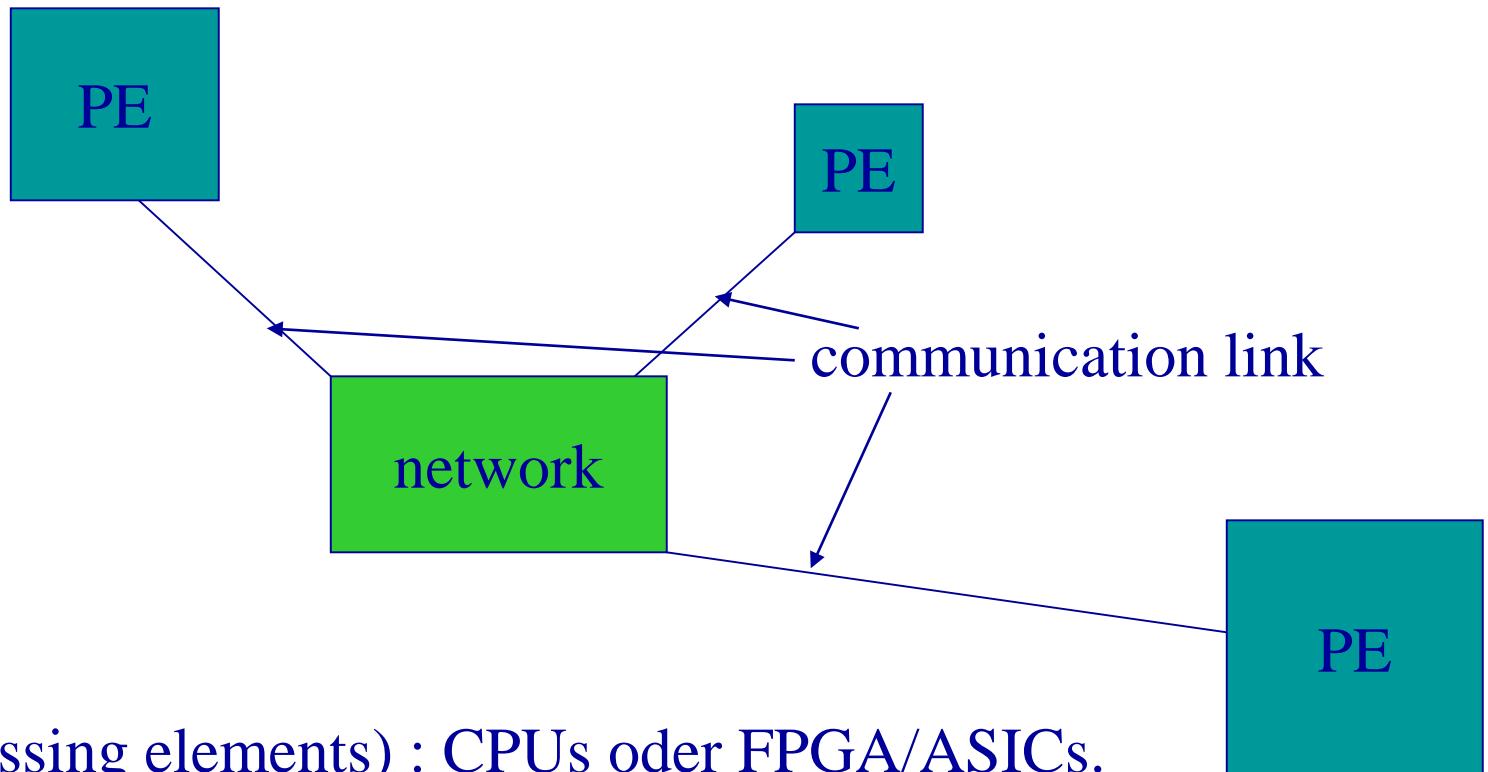


$$\frac{U_i}{U_{ref}} = \frac{R}{2^N}$$

2.5 Netzwerke für eingebettete Systeme

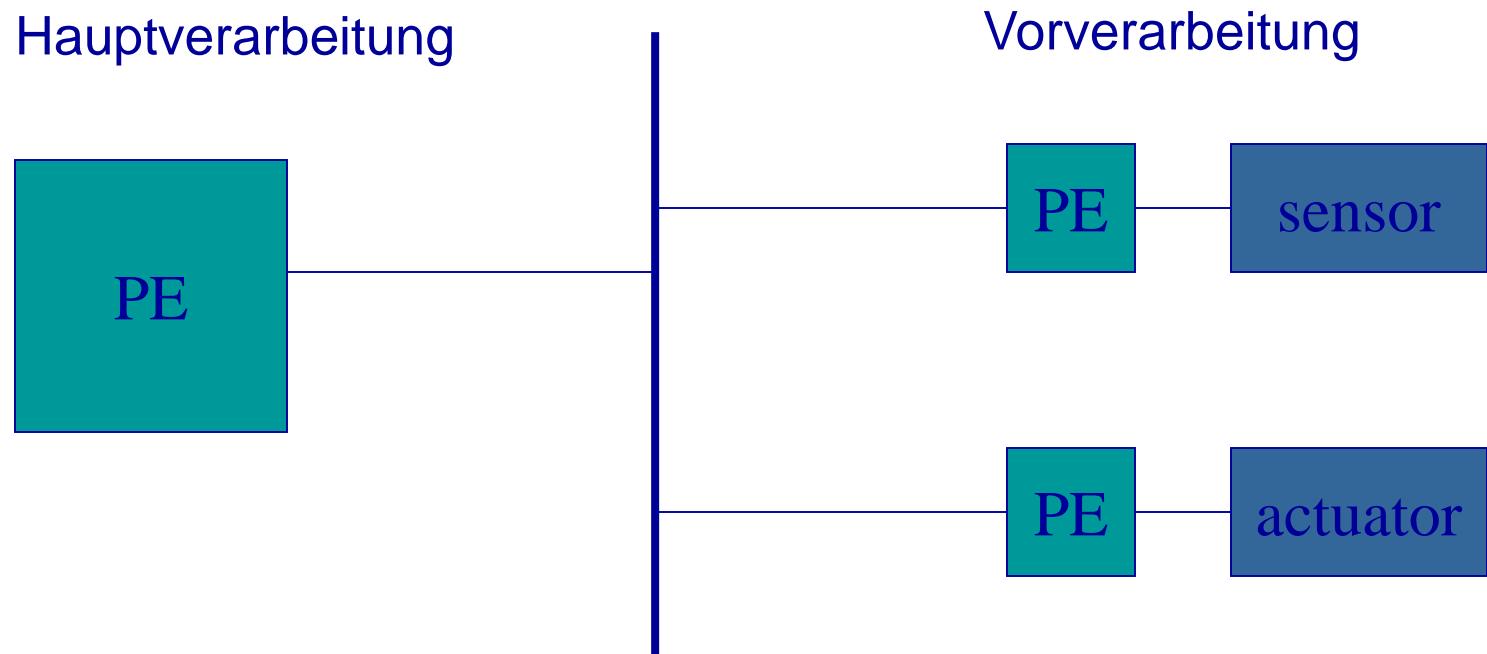
Netzwerkelemente

Verteiltes System



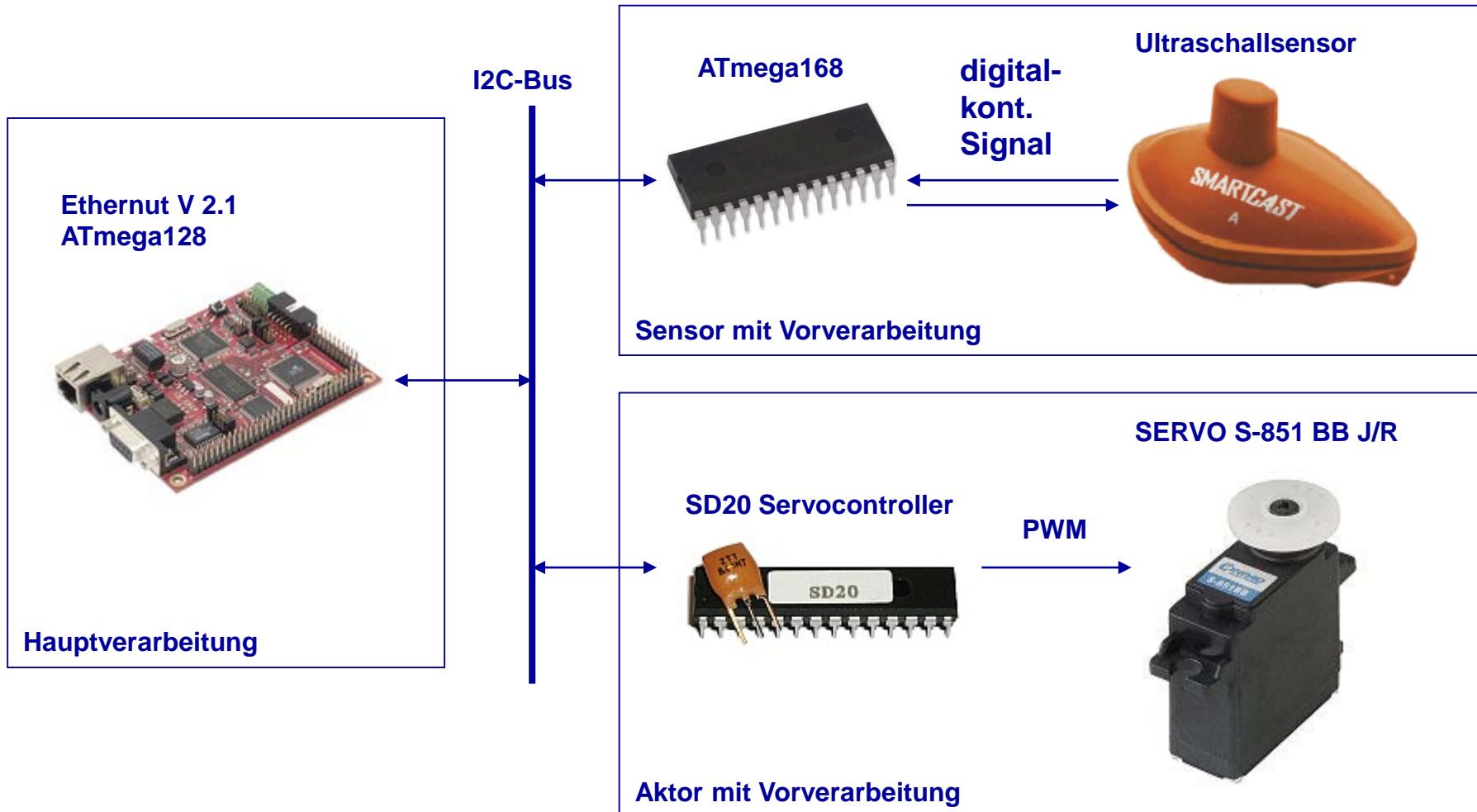
PEs (processing elements) : CPUs oder FPGAs/ASICs.

Netzwerke für eingebettete Systeme



Netzwerke für eingebettete Systeme

Beispiel: Unterwasserroboter MONSUN



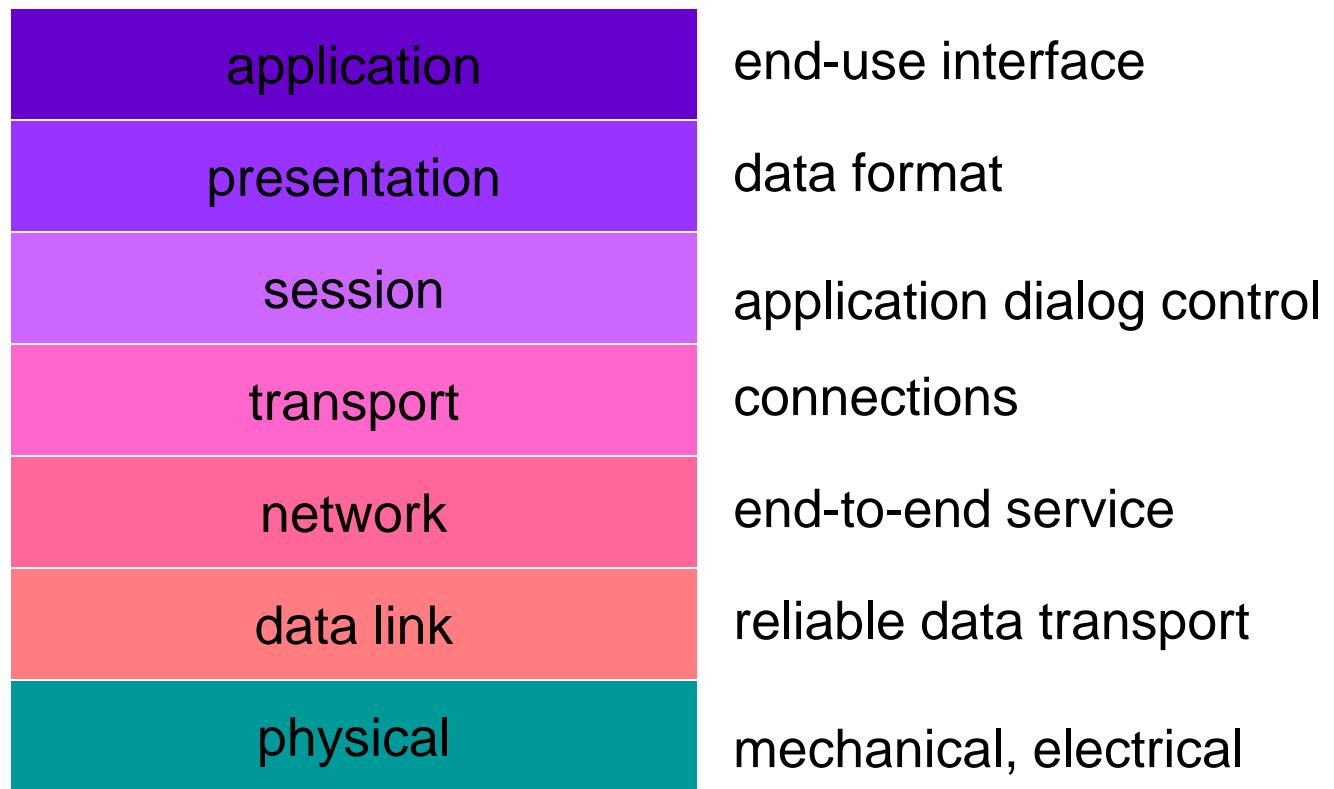
Warum verteilte Systeme?

- Höhere Leistung bei niedrigeren Kosten.
- Physikalisch verteilte Systeme: Echtzeitanforderungen können Übertragung an zentrale Stelle ausschließen.
- Verbesserte Debugging-Möglichkeiten: Eine CPU im Netzwerk kann verwendet werden, um andere zu debuggen.
- Einsatz fertiger Subsysteme mit integrierten Mikrocontrollern.
- Potential für Fehlertoleranz durch Redundanz
- Skalierbarkeit und modulare Erweiterbarkeit

Netzwerk-Abstraktionen

ISO/OSI Modell: 7 Schichten von Netzwerkprotokollen

Nicht immer alle
Schichten
bei
eingebetteten
Systemen
ausgeprägt!



ISO/OSI Protokollsichten

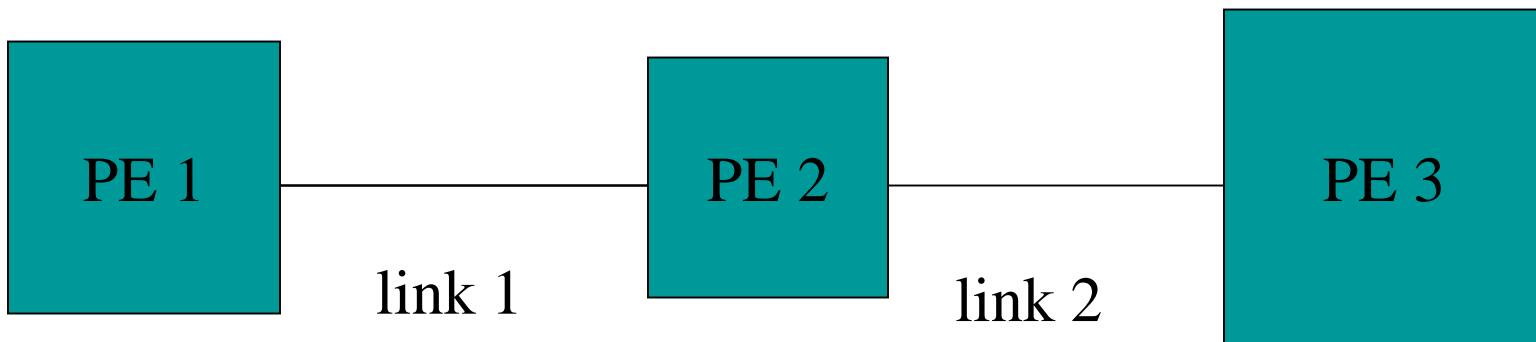
- Physical: connectors, bit formats, etc.
- Data link: error detection and control across a single link (single hop).
- Network: end-to-end multi-hop data communication.
- Transport: provides connections; may optimize network resources.
- Session: services for end-user applications: data grouping, checkpointing, etc.
- Presentation: data formats, transformation services.
- Application: interface between network and end-user programs.

Hardware-Architekturen

- Viele verschiedene Netzwerktypen, unterscheiden sich in:
 - Topologie
 - Kommunikations-Scheduling
 - Routing

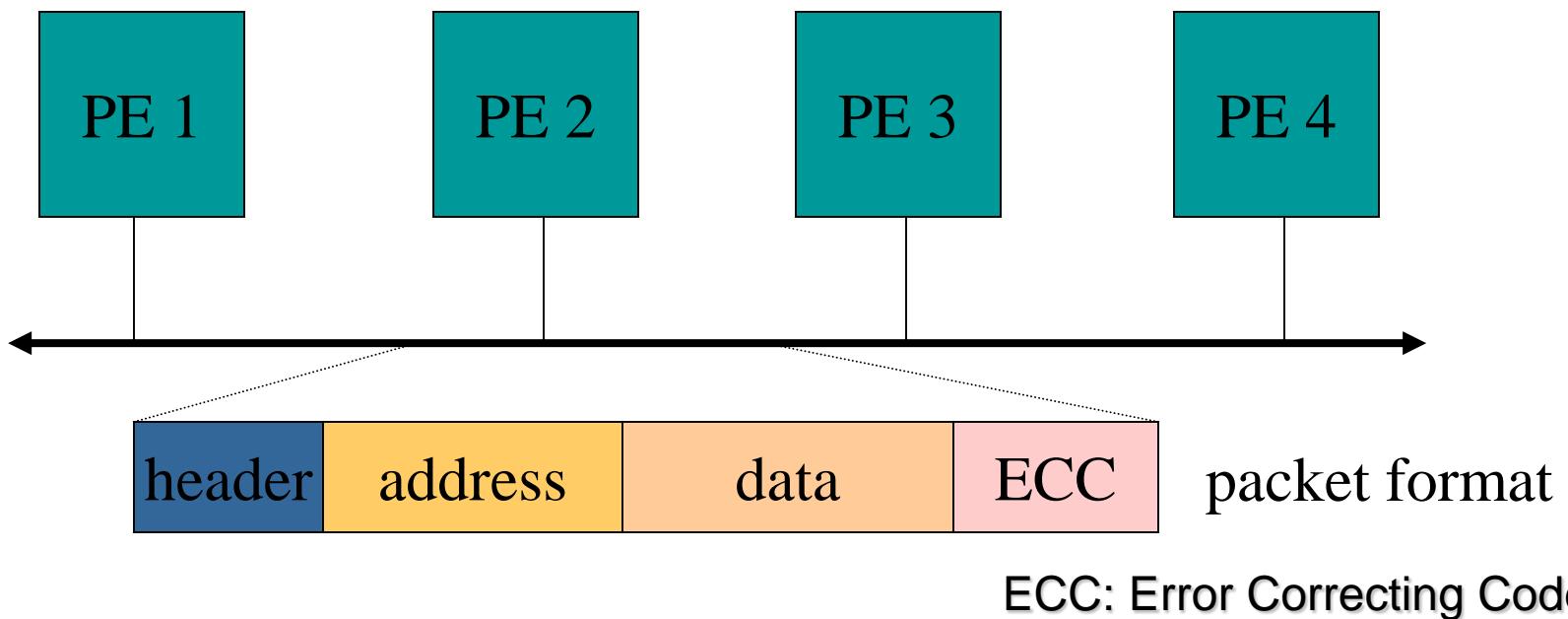
Punkt-zu-Punkt-Netzwerke

- Eine Quelle, eine Ziel (keine Switches)
- Dedizierte Leitungen (Links)
- Meist serielle Übertragung



Bus-Netzwerke

- Gemeinsames physikalisches Kommunikationsmedium
- Übertragung von Paketen
- Meist seriell



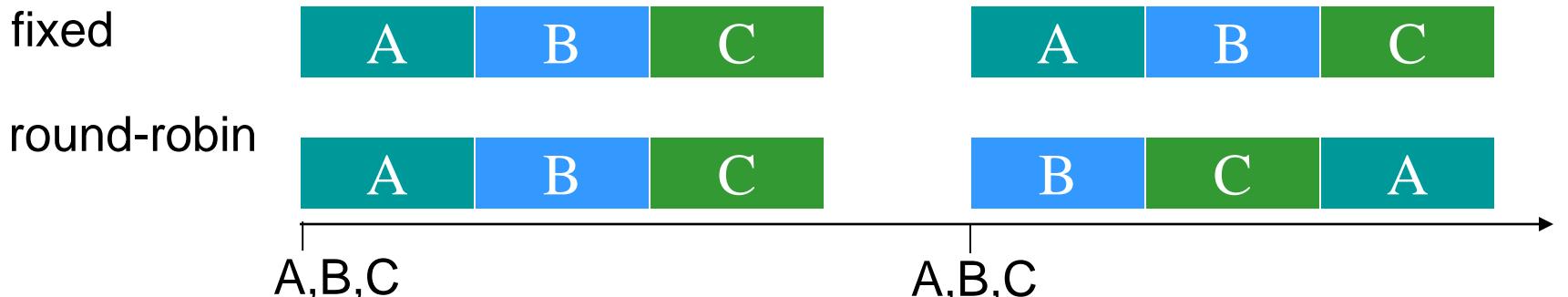
Bus-Arbitrierung

Bus kann immer nur für eine Kommunikation genutzt werden, d. h. alle Kommunikationen finden nacheinander statt (Flaschenhals!).

⇒ Es muss entschieden werden, wer den Bus wann benutzen darf (Arbitrierung)

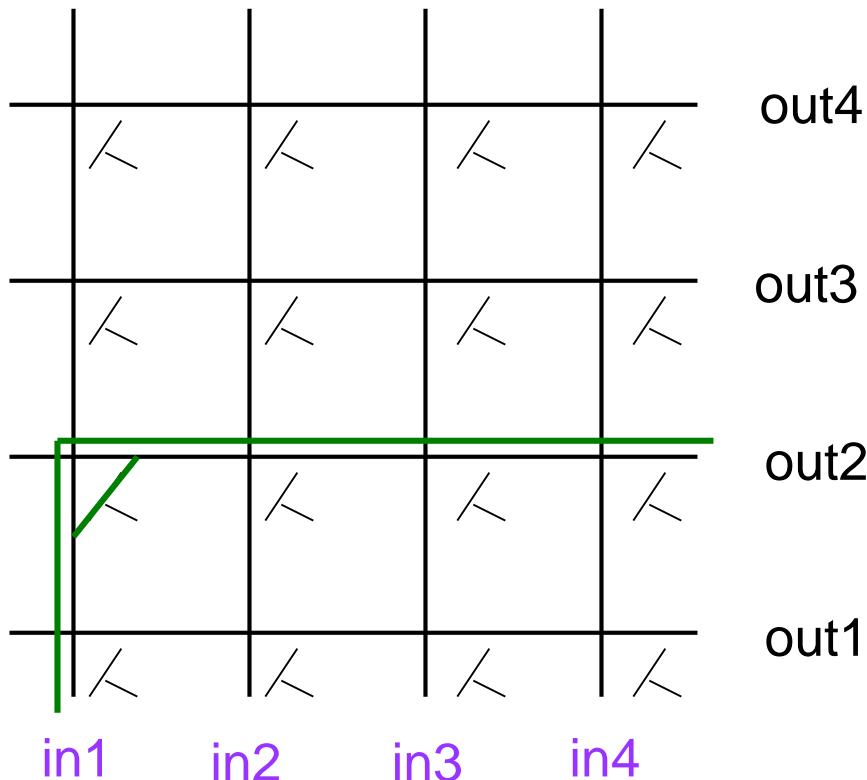
Beispiele

- Fixed: Jedesmal dieselbe feste Reihenfolge
- Fair: Jeder PE hat über längere Zeit die gleichen Zugriffschancen
 - Round-Robin: Rotation der Zugriffspriorität unter PEs



Kreuzschienenverteiler (Crossbar Switch)

N x N Schaltelemente, die Kommunikation von jedem Eingang mit jedem Ausgang erlauben

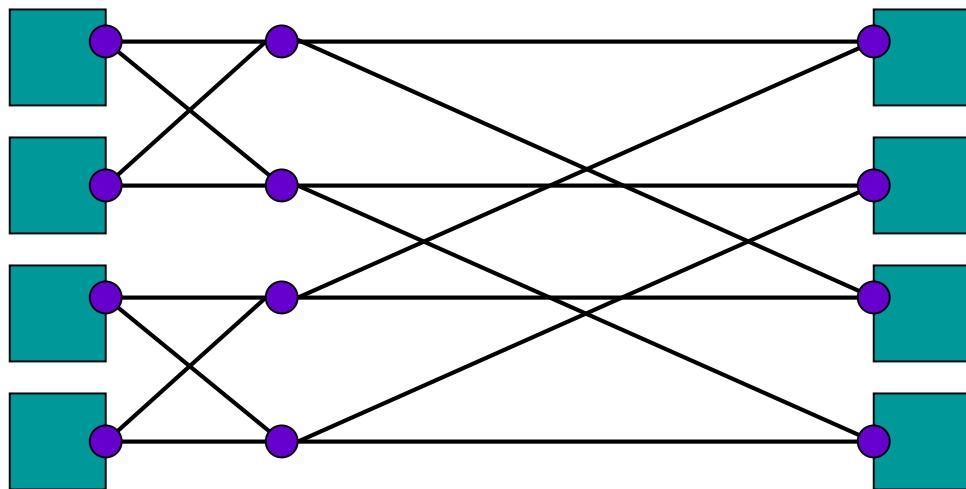


- Blockierungsfrei
- Multicast-fähig
- Komplexität $O(n^2)$

Hohe Bandbreite, aber wegen Komplexität nur für kleine n realisierbar!

Mehrstufige Netzwerke (Multi-stage Networks)

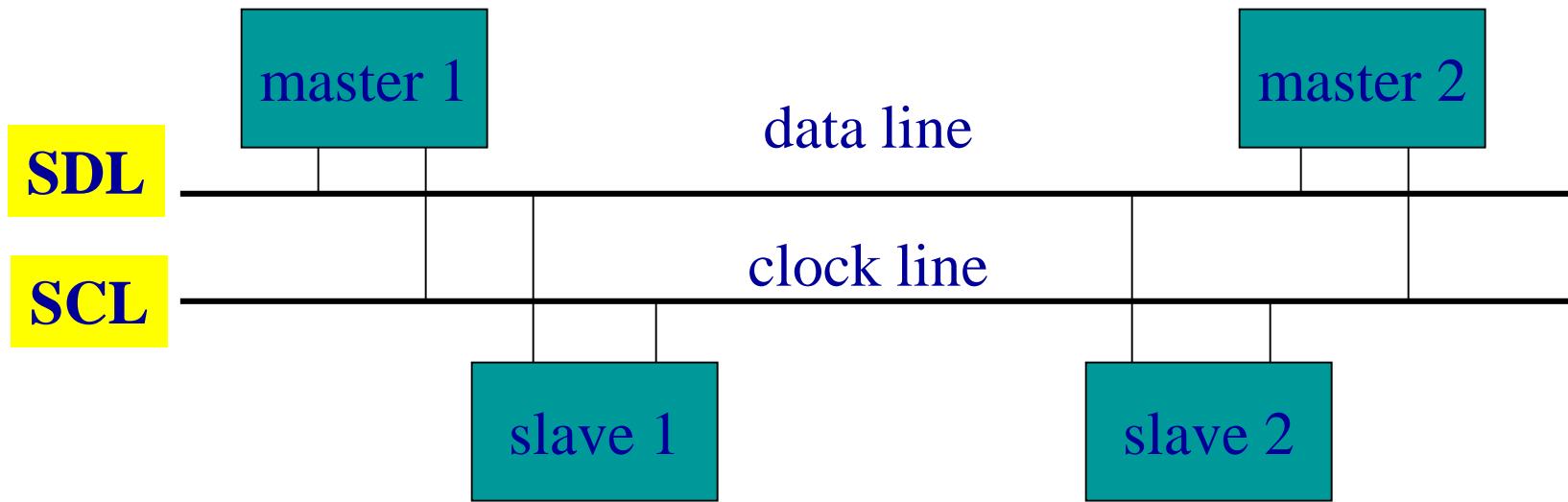
- Mehrere Stufen von Schaltelementen
- In der Regel blockierend
- Komplexität geringer als Crossbar, z. B. $O(n \log n)$
=> skalierbar auch für größere n



Beispiel: I²C-Bus

- Entworfen für preisgünstige Anwendungen mit mittleren Datenraten (bis zu 100 kbit/s Standard, bis zu 3.4 Mbit/s high-speed-mode).
- Charakteristika:
 - Seriell
 - Synchron
 - Multi-Master
 - Arbitrierung
- In vielen heutigen Mikrocontrollern integriert (u. a. ATmega -> TWI)
- Von zahlreichen Sensoren und Aktoren unterstützt

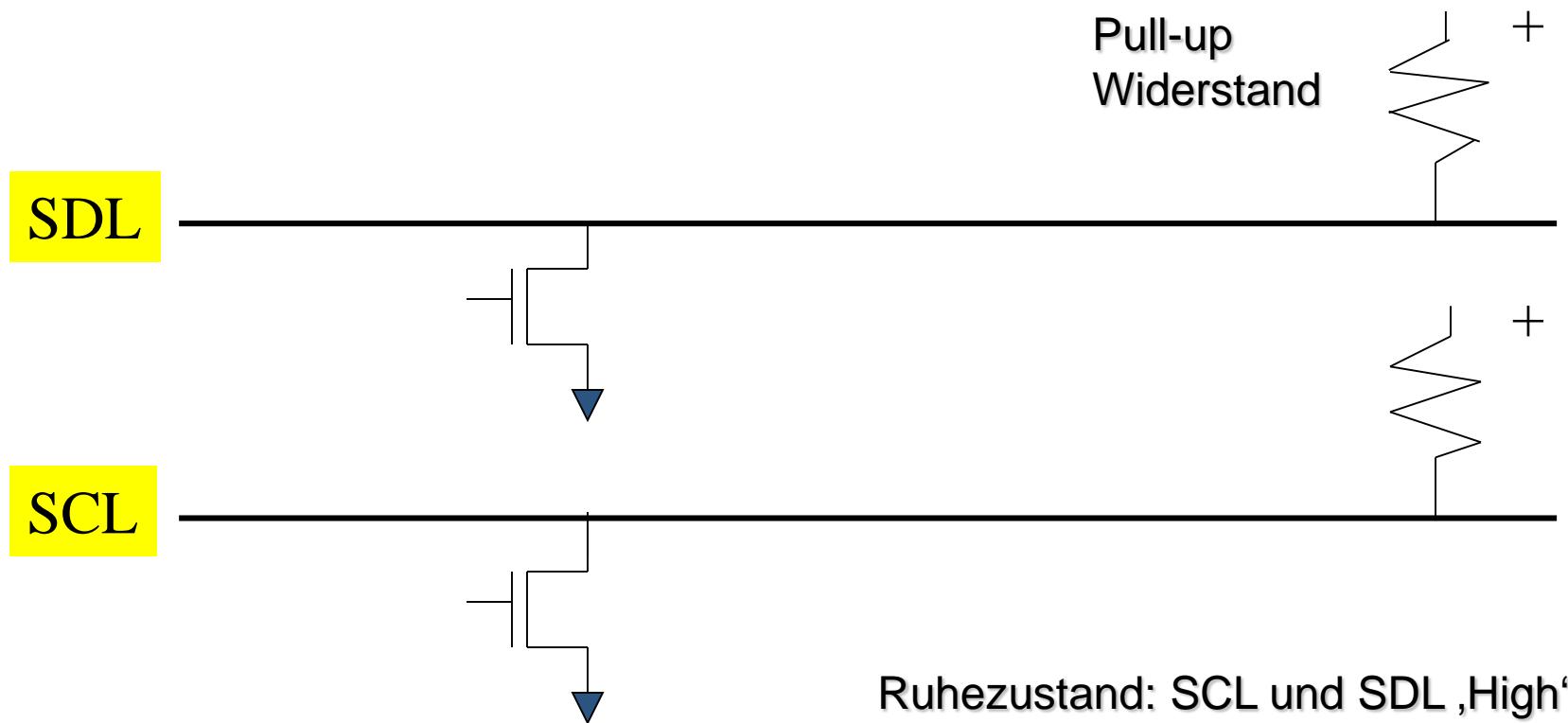
Physikalische Ebene



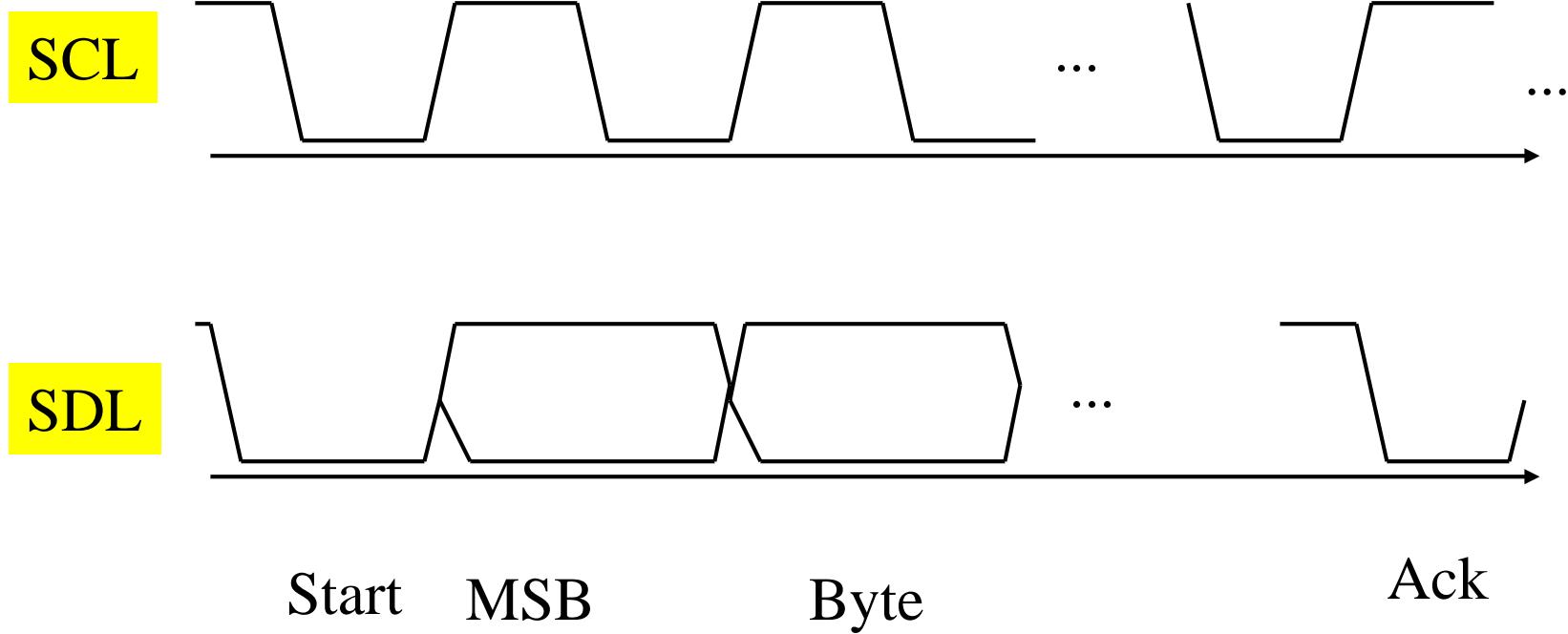
2 Busleitungen SDL (Serial Data Line)
SCL (Serial Clock Line)
(+ Masseleitung)

Elektrische Schnittstelle

- Open collector interface:



Timing bei der Übertragung eines Bytes



Start: SDL wird auf Low gezogen während SCL High bleibt (*Transmitter*).

Byte: Übertragung von einem Bit pro Taktzyklus (*Transmitter*).

Ack: *Receiver* setzt bei erfolgreicher Übertragung SDL auf Low, während SCL High ist.

Stop: SDL wird auf High gezogen während SCL High bleibt (*Transmitter*).

Adressierung

- Jeder Busteilnehmer (Device) hat eine Adresse (7 bit bei Standard, 10 bit bei „Extended“).
 - Bit 8 der Adresse signalisiert Read or Write.



- Jede Datenübertragung beginnt mit Zieladresse
- General Call Address (00...0) für Broadcast (Rundruf an alle).

Bus-Operationen

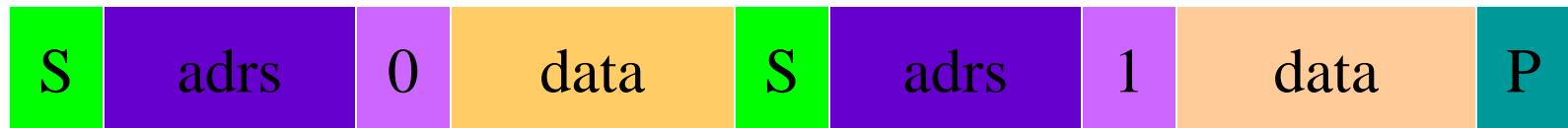
Multi-Byte Write



Read from Slave



Write, then Read



I2C - Das ACK-Bit und Empfang mehrerer Bytes

Jedes empfangene Byte wird vom Empfänger durch ein Acknowledge-Bit bestätigt.

Not-ACK bedeutet Fehler oder Ende der Kommunikation.

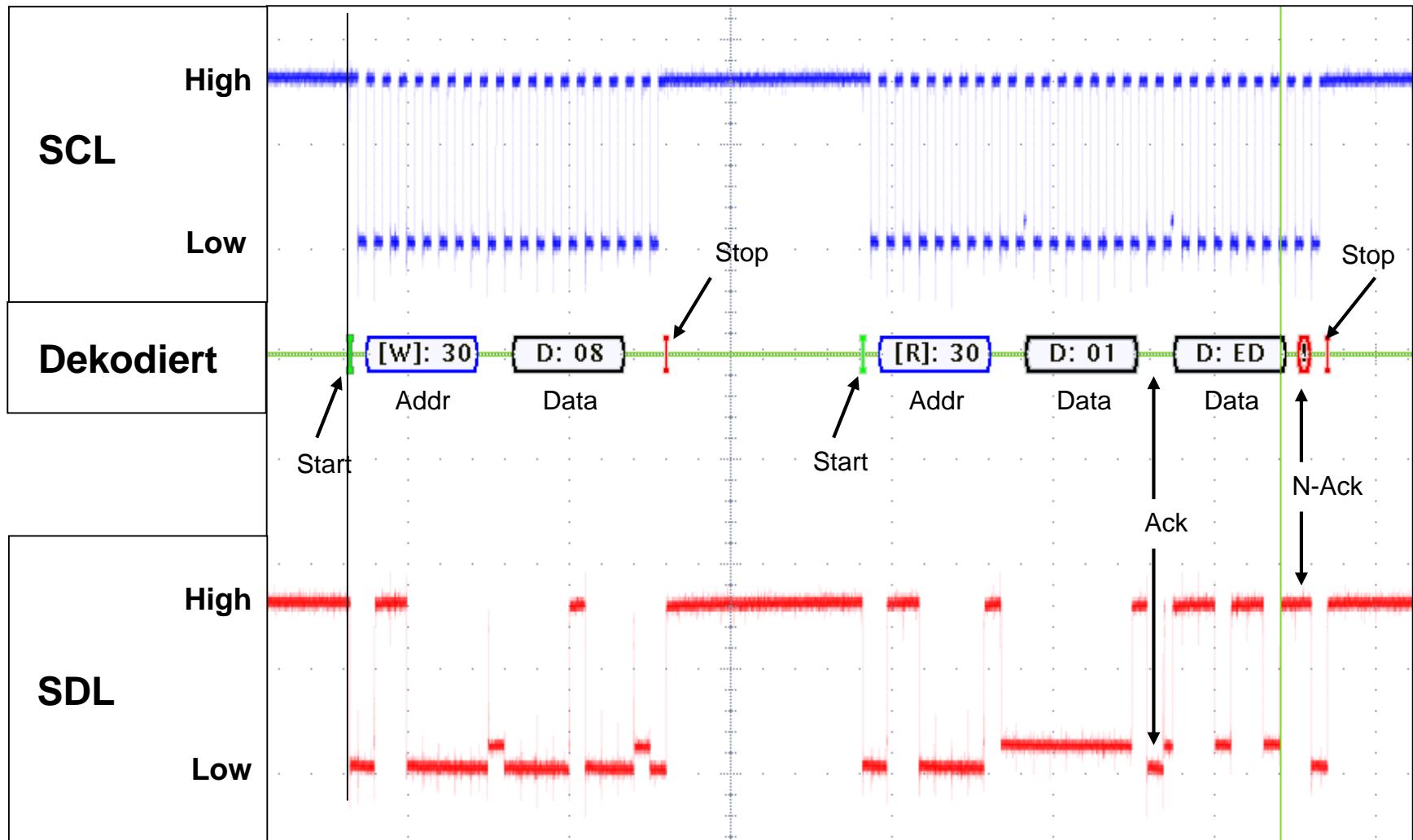
Read multiple bytes from Slave:

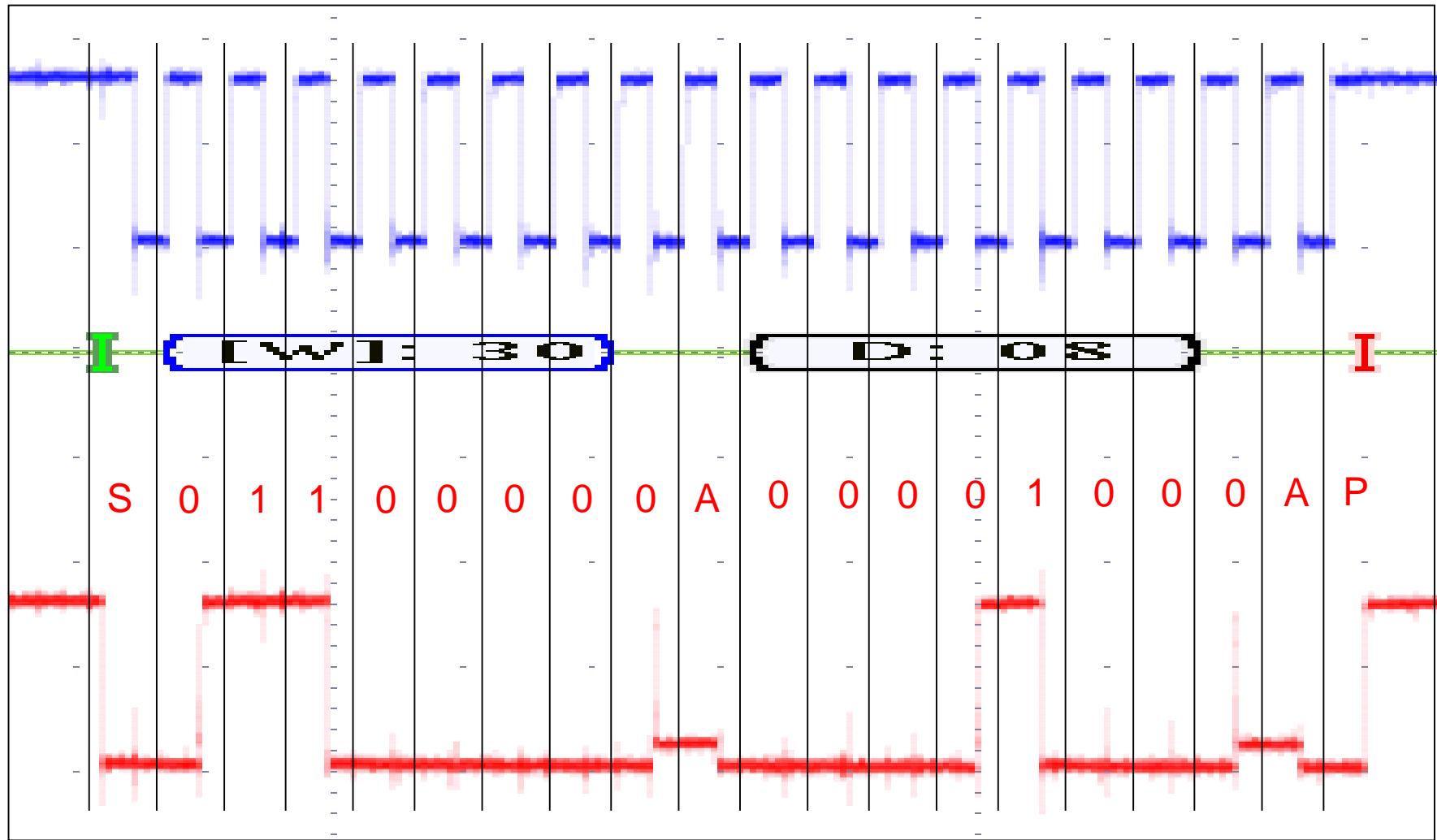


From Master to Slave

From Slave to Master

I2C – Ein Blick direkt auf den Bus





I²C-Bus Arbitrierung (mehrere Master)

- Master überwachen Bus (Start, Stop) und senden nur wenn keine andere Übertragung aktiv (CSMA: Carrier Sense Multiple Access)
 - Problem: gleichzeitiges Senden, “Überhören” von Start
- Sender hört Bus ab, während er sendet.
- Wenn der Sender einen **Konflikt feststellt** (sendet eine 1 aber hört eine 0), hört er **sofort auf zu senden** und gibt dem anderen Sender Priorität (d. h. die “kleinsten” Daten haben höchste Priorität).
 - Bei gleichzeitigem Senden gewinnt somit die “kleinste” Zieladresse, bei gleicher Zieladresse entscheiden die Daten.
- Die niederprioren Sender geben die Kontrolle schnell genug ab, so dass die Übertragung zuverlässig und ohne Verzögerung möglich ist.

Bem.: Senden zwei Sender ein identisches Byte gleichzeitig an den gleichen Empfänger, tritt kein Konflikt auf und beide Übertragungen sind erfolgreich!

Beispiel-Slave: Pyro-Sensor TPA81 mit Servocontroller

Einfache Wärmebild-Zeilen-Kamera:

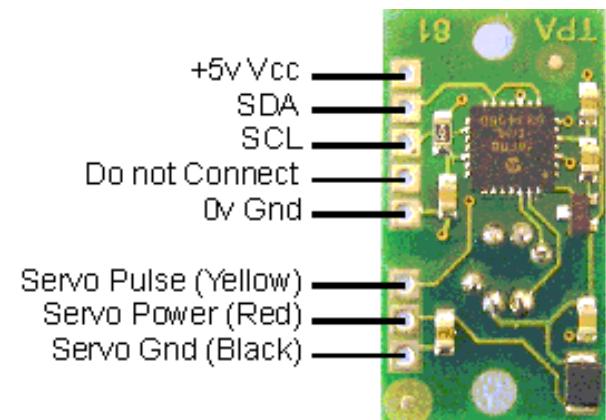
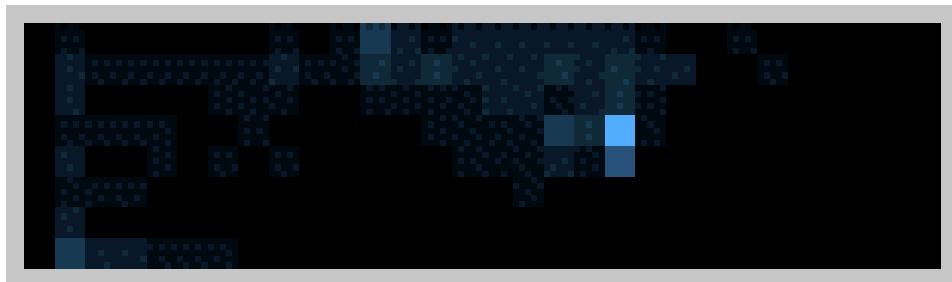
- Misst Wärmestrahlung (2-22 µm)
- Öffnungswinkel von $41^\circ \times 6^\circ$ (8 Pixel)
- Kann ein Servo steuern.

Einsatzbeispiel:

Sensor wird senkrecht angebracht, und waagerecht von einem Servo geschwenkt.

Ergebnis: Wärmebild 32x8 Pixel.

Bsp: Kerze in ca. 2 m Entfernung:



Quelle: www.roboter-teile.de/datasheets/tpa81.pdf

Beispiel-Slave: I2C-Register des TPA81

Um gezielt auf die einzelnen Daten zugreifen zu können, stellen viele I2C-Geräte Register zur Verfügung, die direkt über I2C angesprochen werden.

Register	Lesen	Schreiben
0	Software Revision	Befehls-Register
1	Umgebungs-Temperatur °C	Servo Bereich (nur V6 oder höher)
2	Pixel 1 – Temperatur °C	-
3	Pixel 2	-
4	Pixel 3	-
5	Pixel 4	-
6	Pixel 5	-
7	Pixel 6	-
8	Pixel 7	-
9	Pixel 8	-

Beispiel-Slave: I2C-Befehle des TPA81

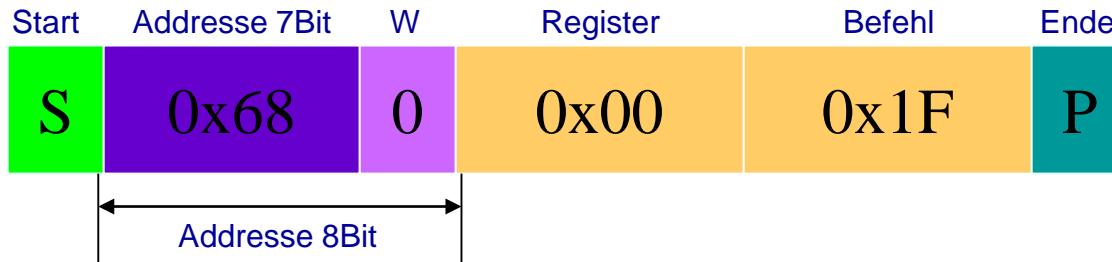
In das Register 0 können folgende Befehle an den TPA81 geschrieben werden:

Befehl		Aktion
Dezimal	Hex	
0	0x00	Setzt Servo Position auf Minimum
nn	nn	Setzt Servo Position
31	0x1F	Setzt Servo Position auf Maximum
		-
160	0xA0	Erstes Byte in Sequenz zur Änderung der Moduladresse
165	0xA5	Drittes Byte in Sequenz zur Änderung der Moduladresse
170	0xAA	Zweites Byte in Sequenz zur Änderung der Moduladresse

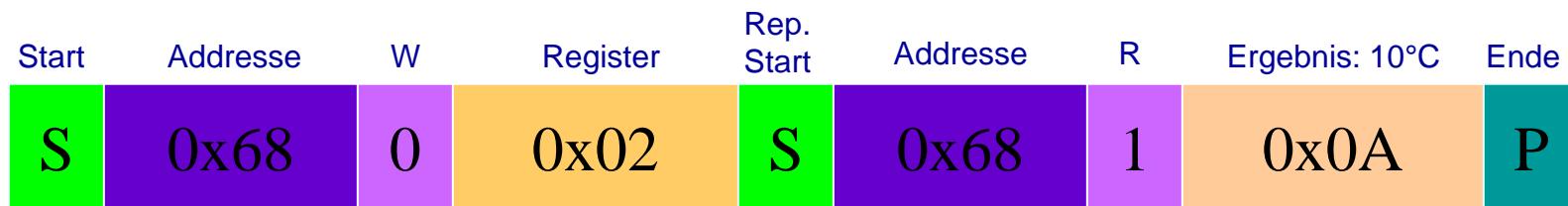
Beispiel-Slave: Lesen eines Registers & Senden eines Befehls

I2C-Adresse des Gerätes: 0xD0 (8Bit) == 0x68 (7Bit)

Befehl: Setze das Servo auf die Maximale Position (Schreibe 0x1F in Register 0x00):

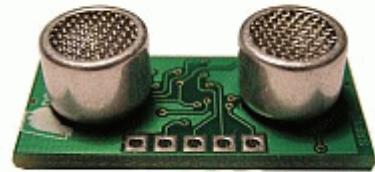


Lese die Temperatur des Pixels 1 (Register 0x02) aus:



Beispiele weiterer Sensoren und Aktoren, die über I2C angesteuert werden können.

Ultraschall-Entfernungsmesser SRF10



Kompass CMPS03



Motorcontroller MD22

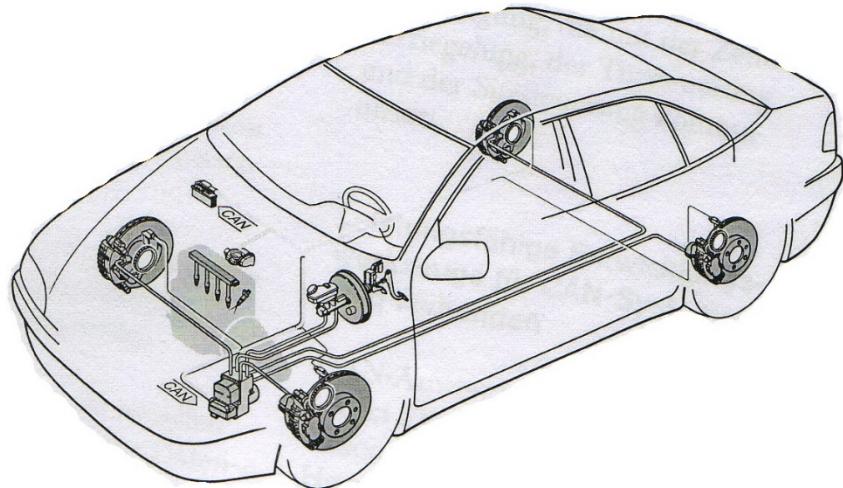


Servocontroller SD21



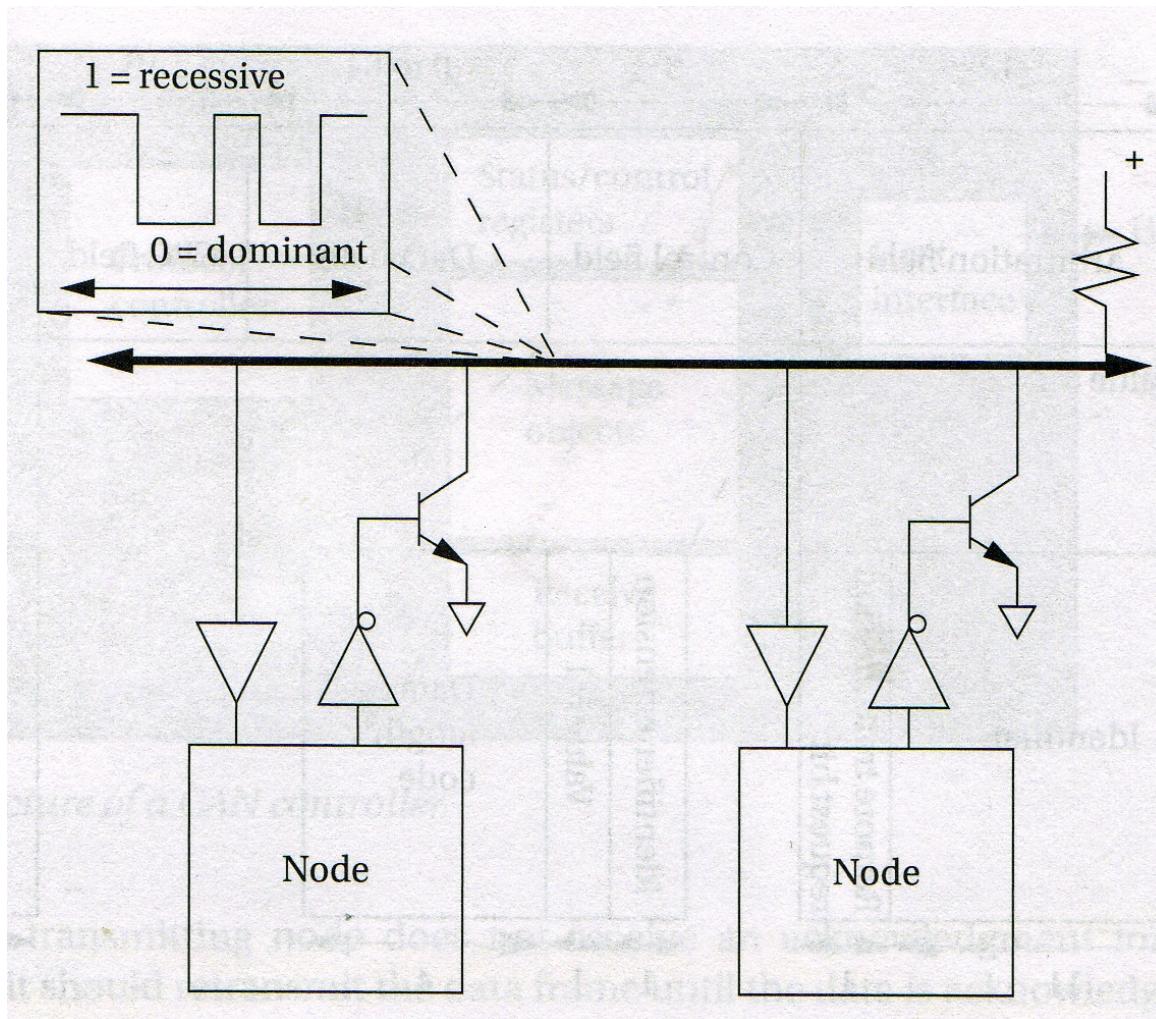
Beispiel: CAN-Bus

Entwickelt von Bosch für die Vernetzung in Automobilen (standardisiert ISO 11898). Mittlerweile auch in vielen anderen Anwendungen (z. B. Medizintechnik, Landwirtschaftstechnik, Robotik, Gebäudeleittechnik, Textilmaschinen, Aufzüge) eingesetzt.



- Bitserielle Übertragung
- bis zu 1Mb/s, typisch 500 kbit/s
- max. 40 m
- Twisted-Pair-Kabel oder optisch
- Multi-Master
- *bedingt echtzeitfähig*

Physikalische Ebene des CAN-Bus



Wired-AND

Jeder Busteilnehmer kann Bus auf 0 herunterziehen d. h. 0 ist dominant.

Wenn alle Teilnehmer eine 1 senden, ist der Bus im rezessiven Zustand.

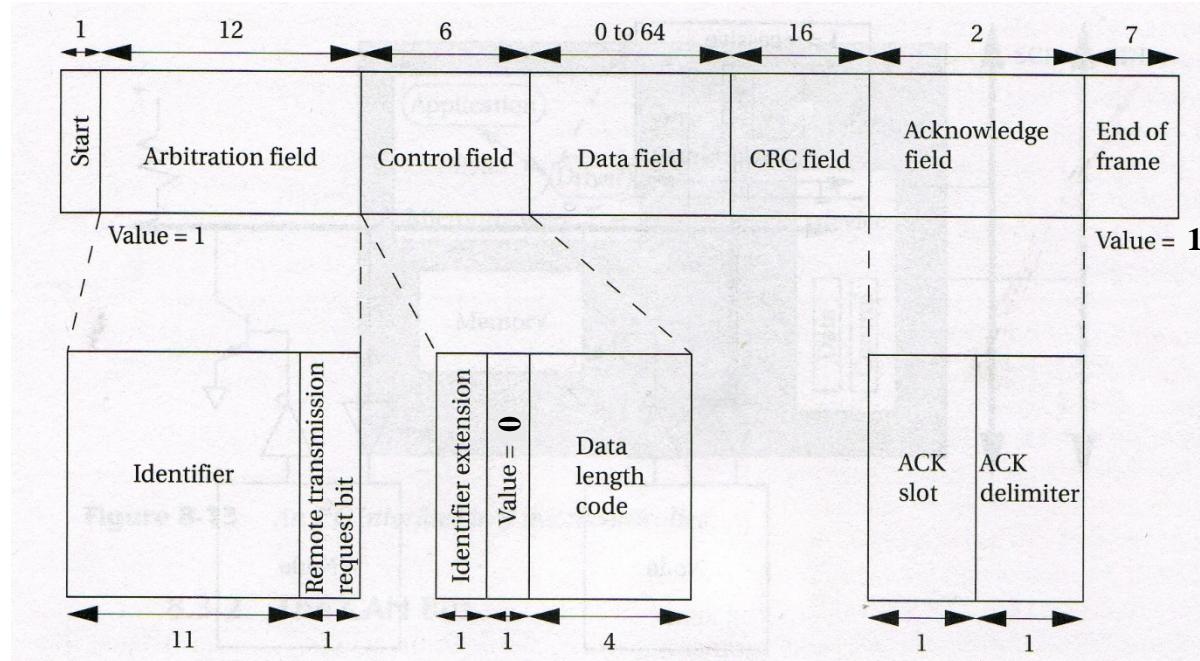
Data-Link-Ebene

Data Frame: Datenübertragung

Remote Frames: Mit Frames mit RTR=1 können Daten angefordert werden. Der Empfänger schickt dann die Daten zurück. Keine Parameterübergabe möglich, d. h. jede Anforderung muss eigenen Identifier haben.

Error Frames: Werden von beliebigen Knoten im Fehlerfall generiert

Overload Frames: Zeigen an, dass ein Knoten überlastet ist



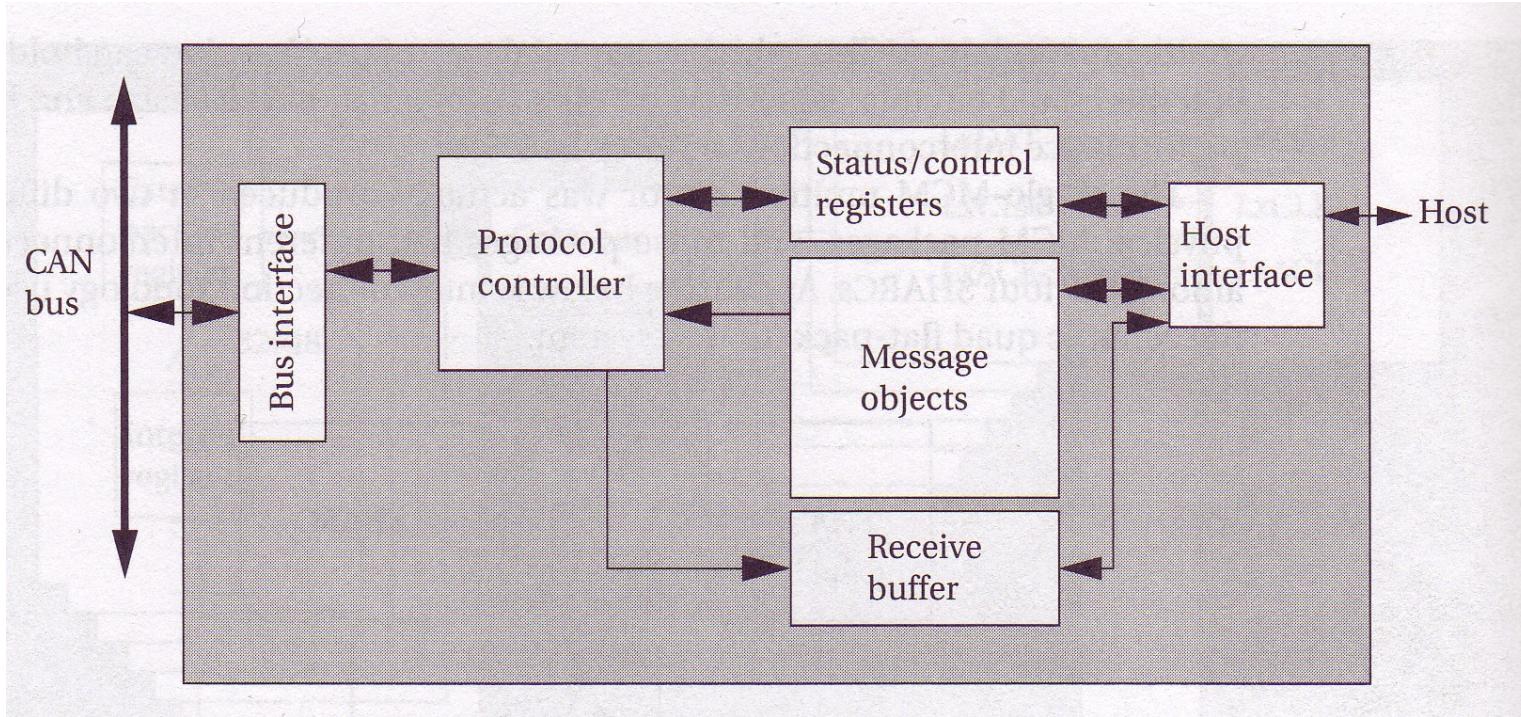
- Start: eine 1, End of frame: siebenmal 1
- Arbitration field: Nachrichten-Identifier (11Bit) und RTR
Letztes Bit des Arbitration Field RTR gibt an, ob geschrieben (RTR=0) oder gelesen werden soll (RTR=1).
- Control Field: Identifier extension und 4 Bit Data Length Code mit Länge der Nutzdaten, getrennt durch Feld mit einer 0.
- Data field: 0 bis 64 Bit Nutzdaten (in 8-Bit Einheiten),
- CRC field: Cyclic Redundancy Check zur Fehlererkennung
- Acknowledge field: Sender setzt ACK slot auf 1, Empfänger zieht es bei korrektem Empfang auf Null (dominant). Automatische Neuübertragung im Fehlerfall. ACK-Delimiter grenzt zum End-of-Frame ab.

Arbitrierung:

Alle Master senden gleichzeitig Identifier. Wenn ein Knoten ein dominantes Bit hört aber rezessiv sendet, zieht er sich zurück, d.h. Nachricht mit kleinstem Identifier hat höchste Priorität (00...0) (vgl. I2C-Bus).

CSMA/AMP: Carrier Sense Multiple Access with Arbitration on Message Priority

CAN Controller

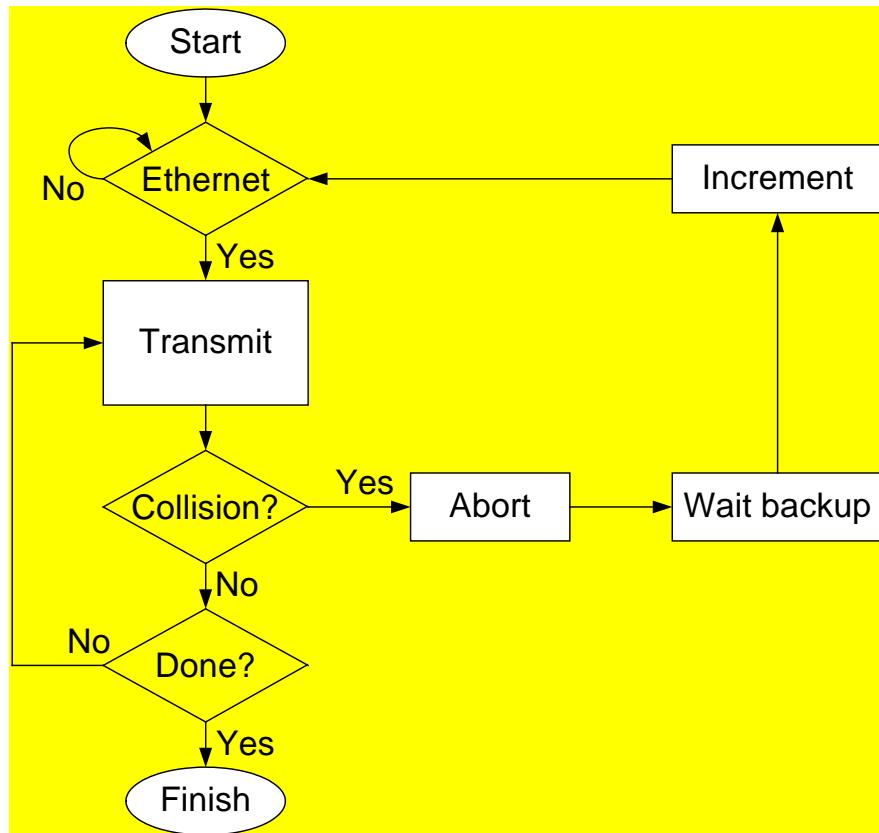


- Implementation des CAN-Protokolls in einem Chip zur Ankopplung von Mikrocontrollern an CAN-Bus
- Auch Mikrocontroller mit ein oder mehreren integrierten CAN Controllern

Beispiel: Ethernet

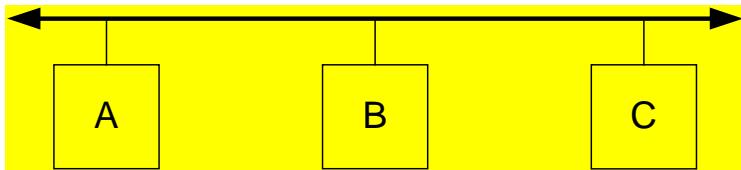
Ethernet in der Office-Welt weit verbreitet, aber aufgrund des CSMA/CD-Verfahrens und des TCP/IP-Protokolls nicht echtzeitfähig, daher bestenfalls für *weiche* Echtzeit brauchbar.

CSMA/CD (Carrier Sense Multiple Access/Collision Detection) bei Ethernet



Bei Kollisionen erneute Versuche
nach Exponential Backoff
Algorithmus mit Zufallskomponente

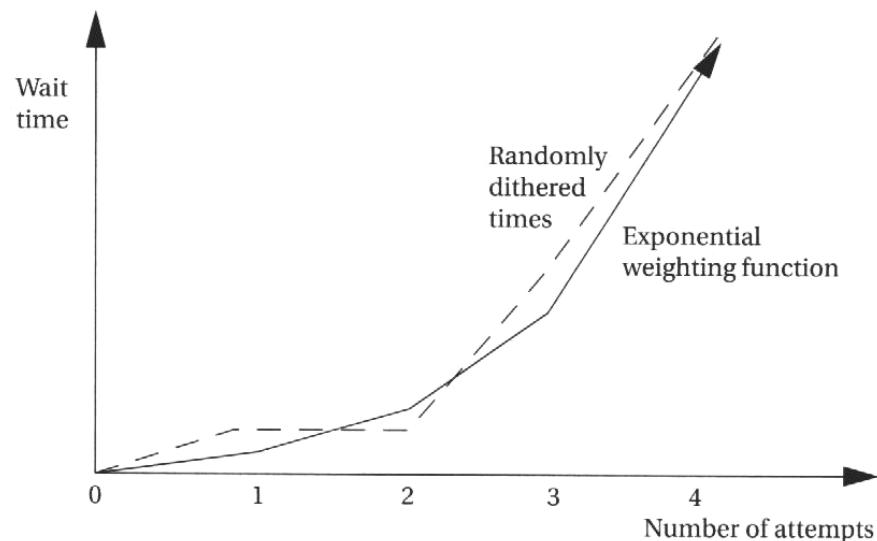
Klassischer Ethernet-Bus



Kollisionswahrscheinlichkeit steigt mit Abstand der Stationen (bis zu einigen 100 m)

Max. Kollisionsfenster: 2x Laufzeit des Signals

Wartezeit als Funktion der Versuche



Starker Anstieg mit Zahl der Versuche
(Exp. Backoff)

Zufallskomponente, um erneute Kollision zu vermeiden.

⇒ *Ethernet nicht für harte Echtzeit geeignet, bestenfalls weiche Echtzeit bei niedriger Last und damit geringer Kollisionswahrscheinlichkeit*

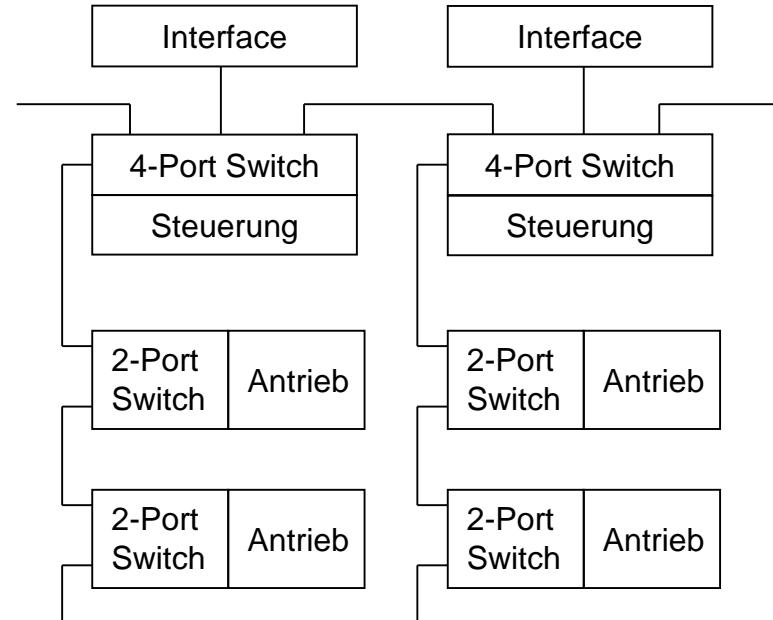
TCP/IP wegen automatischem Timeout mit Retransmission im Falle von Fehlern auch nur bedingt echtzeitfähig. Außerdem hoher Protokolloverhead (offenes System).

PROFINet

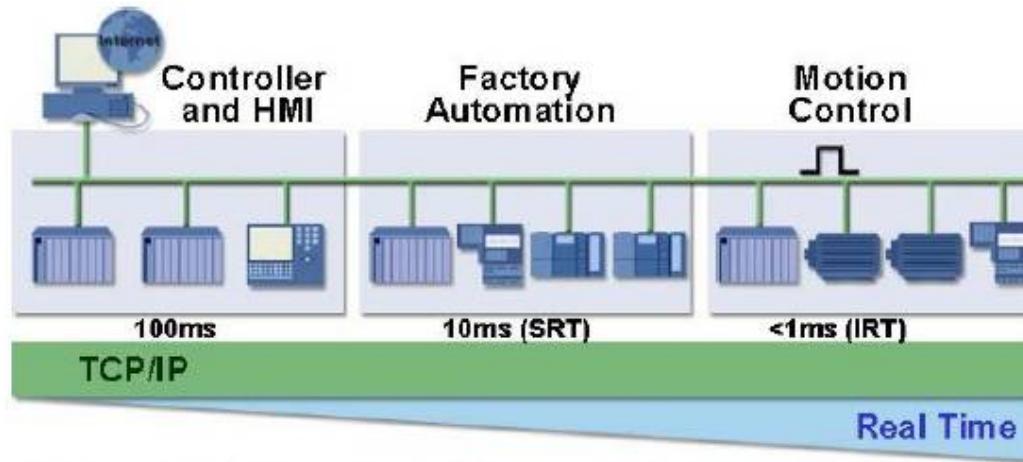
Erweiterung von Ethernet um hart echtzeitfähige Protokolle, so dass durchgängige Vernetzung von der LAN- bis in die Feldbusebene möglich. Standardisiert (IEC1158) seit 2003, PROFINet Nutzerorganisation (PROFIBus angegliedert) mit führenden Firmen als Mitgliedern.

Basis:

- Fast Ethernet (100 Mbit/s) geschaltet
⇒ Kollisionen prinzipiell vermeidbar.
- Spezielle industrietaugliche Kabel und Stecker
- Spezielle Ausführung der Switches und NICs (Network Interface Cards)
- Voll kompatibel mit Standard Fast Ethernet und TCP/IP bzw. UDP/IP für zeitunkritische Kommunikation

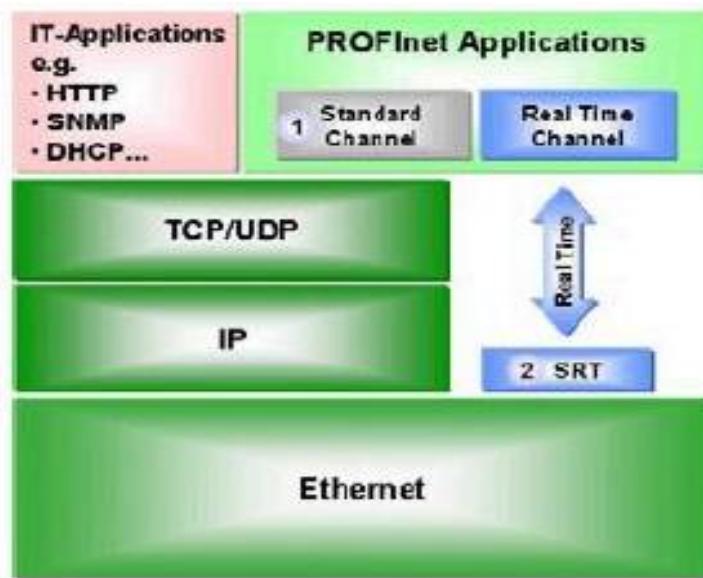


Spezielle Echtzeitprotokolle für zeitkritische Kommunikation



SRT: Soft Real-Time
(z. B. für Prozessautomatisierung)
IRT: Isochronous Real-Time
(z. B. für Motion Control)

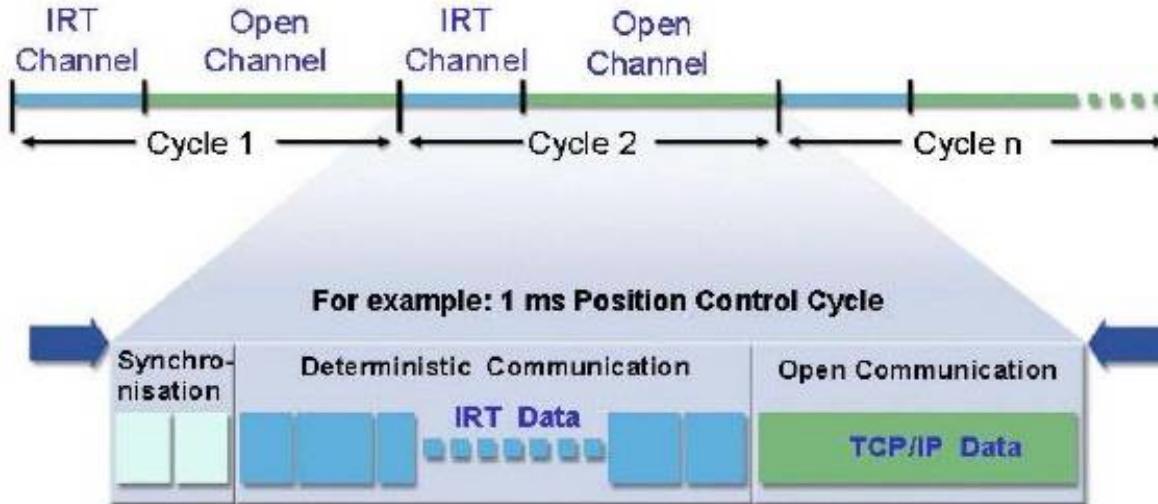
SRT: Priorisierter Kommunikationskanal



Setzt direkt auf Ethernet-Schnittstelle auf (kein TCP/IP)
Pakete mit Prioritätsklasse Prio 6 nach IEEE 802.1Q (vgl. Internet-Telefonie Prio 5)

Reine Softwarelösung

IRT: Zeitschlitzgesteuerter synchronisierter Kommunikationskanal



Leistungsparameter von PROFINet IRT

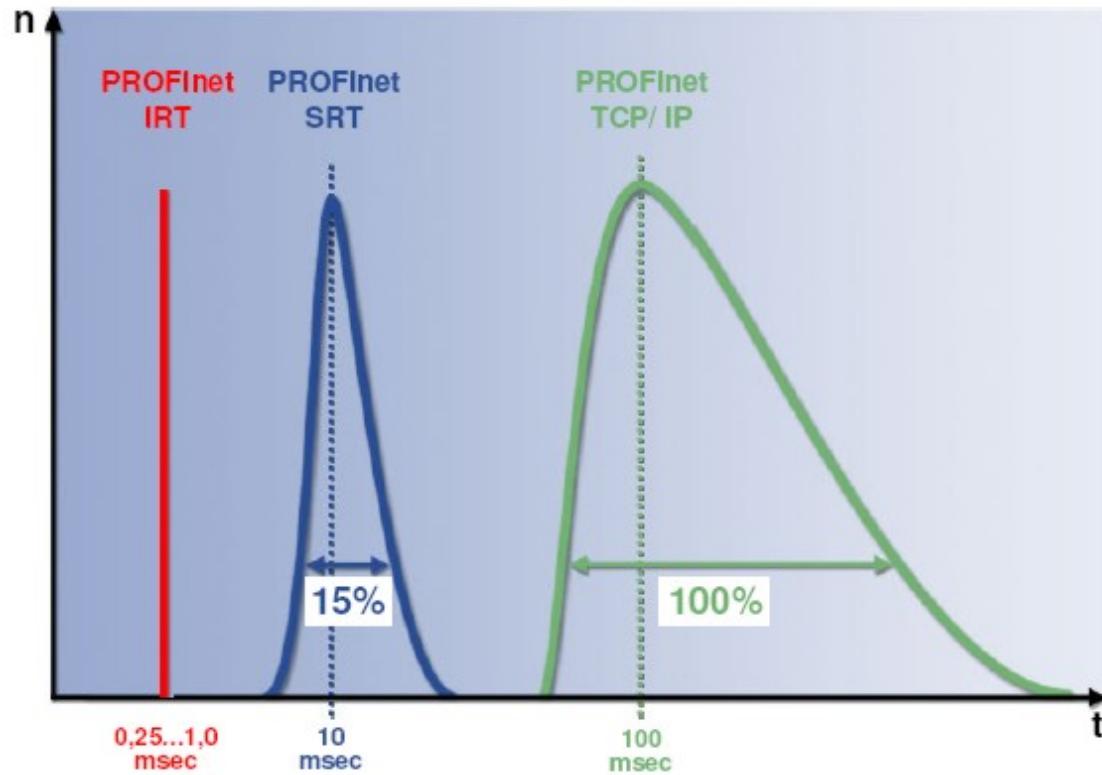
Zykluszeit	1 msec	250 µsec
Jitter	<1µsec	<1µsec
Anzahl Teilnehmer	70	150
Gleichzeitig übertragbare TCP/IP-Daten	9 MByte/sec	6 MByte/sec

Hardwareimplementierung (ASIC) zur Zyklussynchronisation und Zeitschlitzsteuerung.

Zykluszeiten von 1 ms bei 1 µs Jitter für bis zu 100 Teilnehmer.

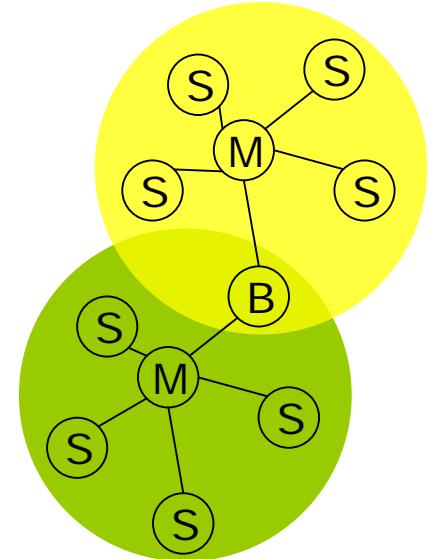
Einsatz auch für sehr harte Echtzeitanforderungen wie Motion Control möglich.

Verteilung der Aktualisierungszeiten bei PROFINet



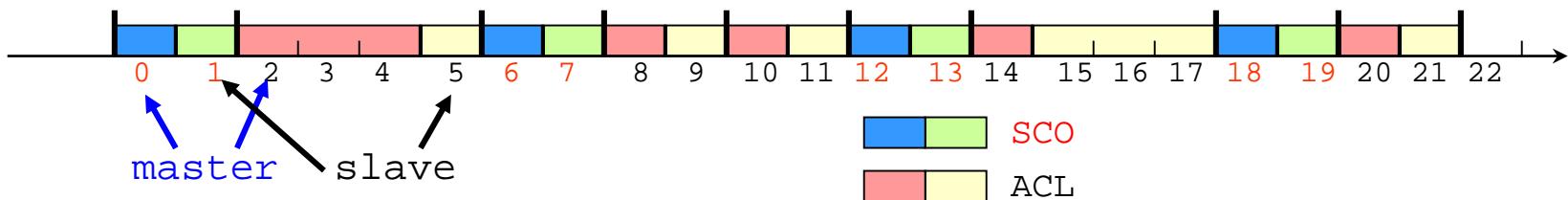
IEEE 802.15.1 („Bluetooth“)

- Bitübertragung
 - 79 Kanäle im 2.4 GHz ISM Band
 - Pseudo-random Frequency-Hopping
 - Wie finden sich Knoten?
 - Bandbreite ca. 1 mbps
 - Energieeffizient bei hoher Datenrate
- Arbitrierung basiert auf fester Zuteilung
 - Piconet: 1 Master + max. 7 Slaves
 - Master synchronisiert Slaves und gibt Hopping-Sequenz vor
 - Jeder Slave hat einen Zeitslot, in dem er mit dem Master kommunizieren kann
 - Scatternets: Überlappende Piconets
 - Bridge: gemeinsamer Knoten
 - Zeitlicher Wechsel zwischen Piconets



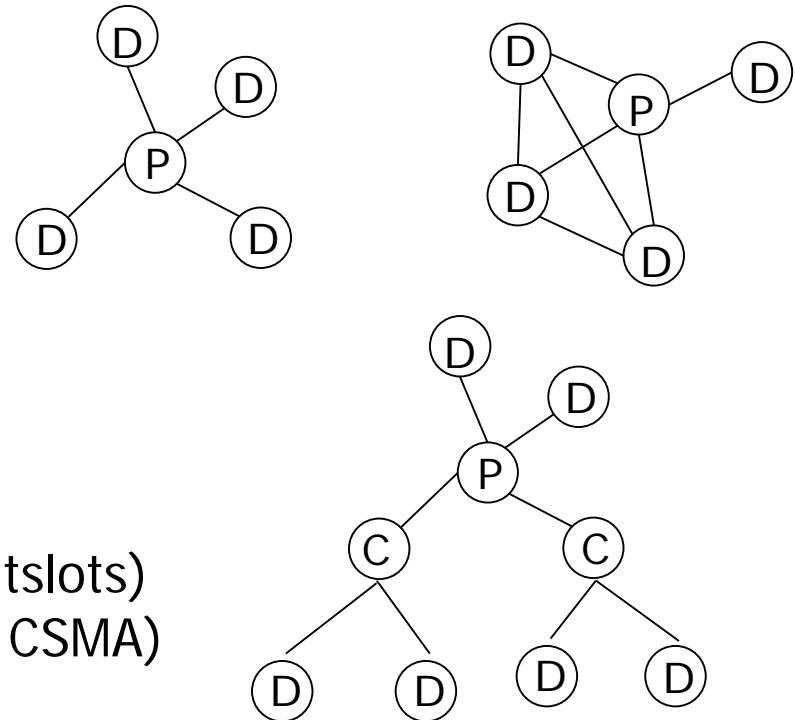
Bluetooth: Echtzeitkommunikation

- Synchrone verbindungsorientierte Komm. (SCO)
 - Periodische Zeitschlitte
 - Typischerweise fuer Sprachübertragung
 - Symmetrisch 64 kbit/s bidirektional mit Fehlerkodierung
- Asynchrone verbindungslose Kommunikation (ACO)
 - Pakete mit variabler Länge (1-5 Zeitschlitte)
 - Asymmetrisch max. 732.2 kbit/s bzw. 57.6 kbit/s
 - Symmetrisch 2 x 432 kbit/s



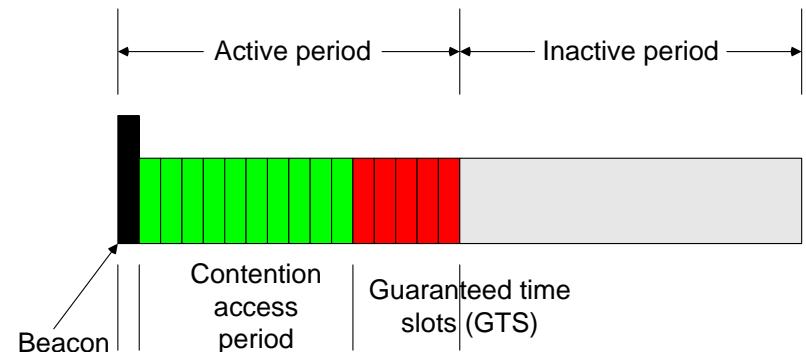
IEEE 802.15.4 („Zigbee“)

- Bitübertragung
 - 20 – 250 kbps in verschiedenen Frequenzbereichen (868 MHz, 926 MHz, 2.4 GHz)
 - Zum Teil mehrere Kanäle mit dynamischer Kanalwahl
- Knotenrollen
 - PAN Coordinator
 - Coordinator
 - Device
- Knotentypen
 - Full Function Device
 - Reduced Function Device
 - Nur „Device“ Rolle
- Netztopologien
 - Stern (Beacon-Mode = CSMA + Zeitslots)
 - Peer-to-Peer (Non-Beacon-Mode = CSMA)
 - Cluster Tree



Zigbee: Echtzeitkommunikation

- Coordinator und Devices bilden sternförmiges Netz
 - Coordinator sendet regelmässig Beacon
 - Synchronisation
 - Zeiteinteilung im Frame
 - Zuteilung von Guaranteed Time Slots (GTS)
 - Active Period
 - CAP: Nachrichtenübertragung mittels CSMA
 - GTS: Nachrichtenübertragung mittels TDMA
 - Inactive Period
 - Schlafen



Aktuelle Trends bei verteilten eingebetteten Echtzeitsystemen

- Integration von Netzwerkschnittstellen schon für einfache Mikrocontroller (z. B. I2C-Bus, SPI-Bus, CAN-Bus, USB-Bus) neben seriellen Standard-Schnittstellen (z. B. RS-232, RS-485)
- (Standard-)Ethernet weit verbreitet, trotz Problemen mit harter Echtzeit
- Web-Anbindung (Sicherheitsproblematik!)
- Funkvernetzung (z. B. Bluetooth, ZigBee, WLAN, GSM, UMTS)
- Drahtlose Sensornetze heute wichtiger aktueller Forschungsgegenstand