Vorlesungen: 1. Semesterhälfte
(Ü) Projekt-Seminar: 2. Semesterhälfte
+ praktischer Teil → **Prüfungsstoff**

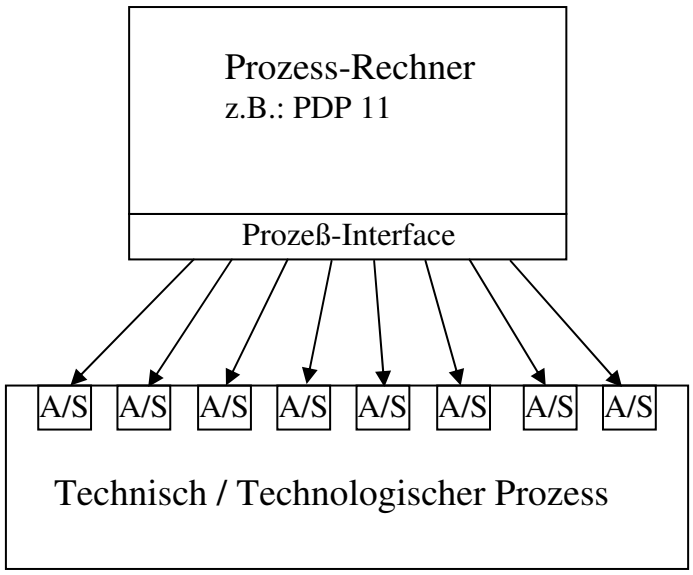
V1: 02.04.04
RN der PDV

Die. 06.04. entfällt
Fr. 16.04. nächster Termin

Literatur:
* Entschberger: "CAN-Bus"
* Schnell: "Bussysteme in der Automatisierungs-Technik", Vieweg
* Bender: "Profibus – Der Feldbus", Hanser

Inhalt:
1) Allgemeine Anforderungen an die Kommunikation in der PDV
2) Feldbus der unteren Leistungsebene (ASI)
3) Feldbus der mittleren Leistungsebene (Profibus)
4) Projekt zum Kommunikationssystem CAN (Automobil-Bereich)

Kapitel 1: Historisch



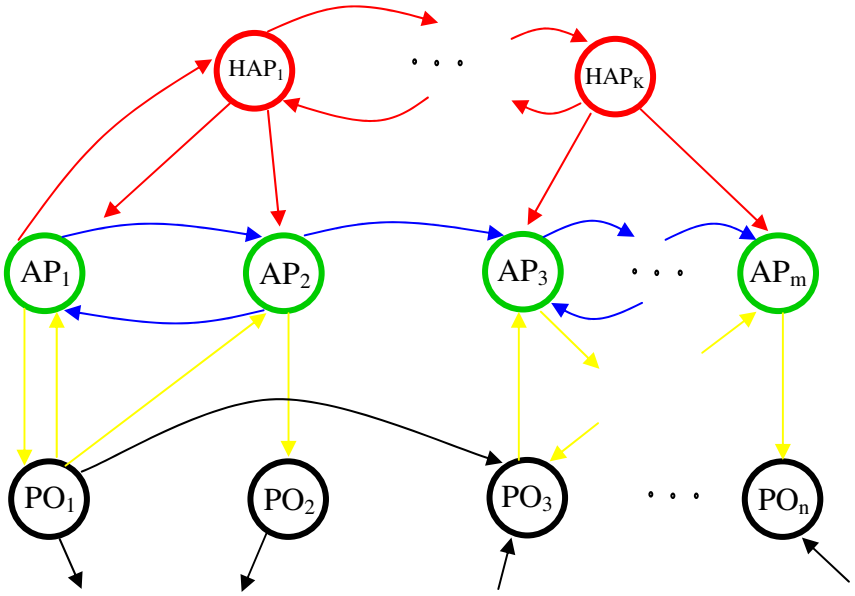
PDP: process-data-processor
(Firm. ***Digital***)

A/S: Aktor / Sensor

- sternförmige Ver-
kabelung (logisch)

- teure PDV-Einrichtung
→ teure Verkabelung möglich
→ Anlagengröße entsprechend

Heute:



* **PO_x** … **Prozess-Objekt** - Teil eines Technisch/Technolog. Systems
- typischerweise nicht isoliert – hat Beziehungen zu anderen **PO**

resultieren aus
technisch /technolog. Aufgabe

* **AP_x … Applikations-Prozesse:** - Informationsverarbeitung mit **direkten Beziehungen** zu anderen **PO** und zu **anderen AP**

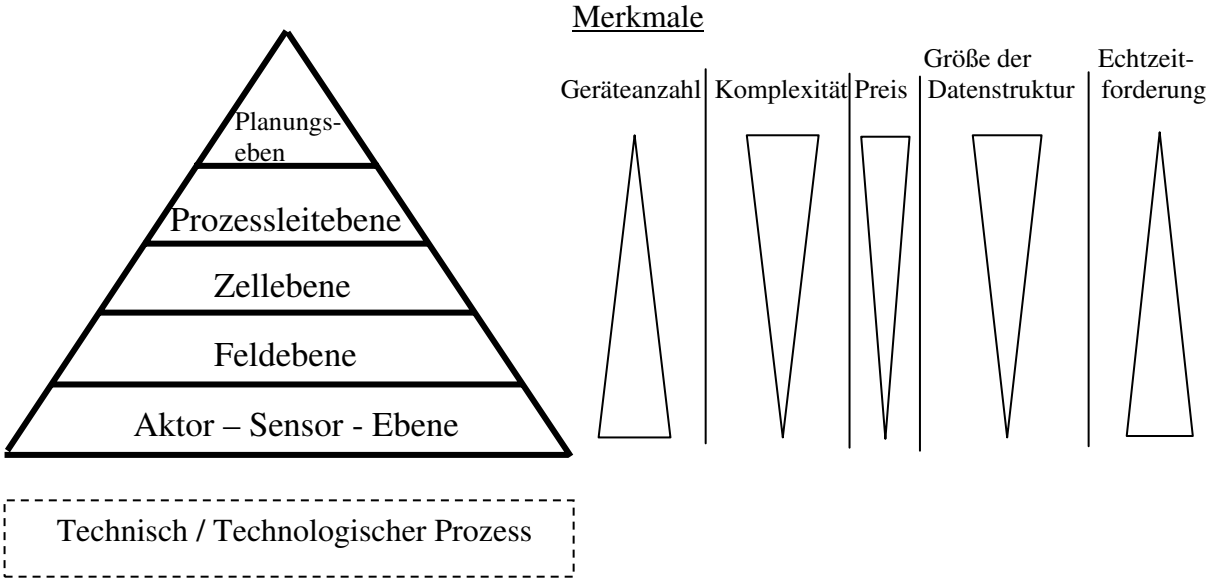
* **HAP_x … Hierarchisch übergeordnete AP**: - typischerweise nur **Beziehungen zu anderen AP / HAP**

➔ ***alle Beziehungen*** ➔ ***Kommunikation mit unterschiedl. Eigenschaften***

- Modelle mit nur 2 Ebenen eher untypisch

Struktur (Ebenenmodell) einer Automatisierungsanlage:

➔ *"Automatisierungspyramide"*



Aktor: Eingriffsmöglichkeit in den Prozess

Sensor: Messmöglichkeit im Prozess

Aktor-Sensor-Ebene: Ein-/Ausgabe von Prozessgrößen
- zeitzyklische Vorgänge (interessant für dig. Steuerungen, dig. Regelung und Signalverarbeitung)
- azyklische Vorgänge (Ereignisse, Alarme)

Feldebene:
- Steuerung, Regelung
- Signalvorverarbeitung
- Anzeige, Bedienen, Protokollieren
- Echtzeitforderungen gelten unmittelbar (Prozesse)

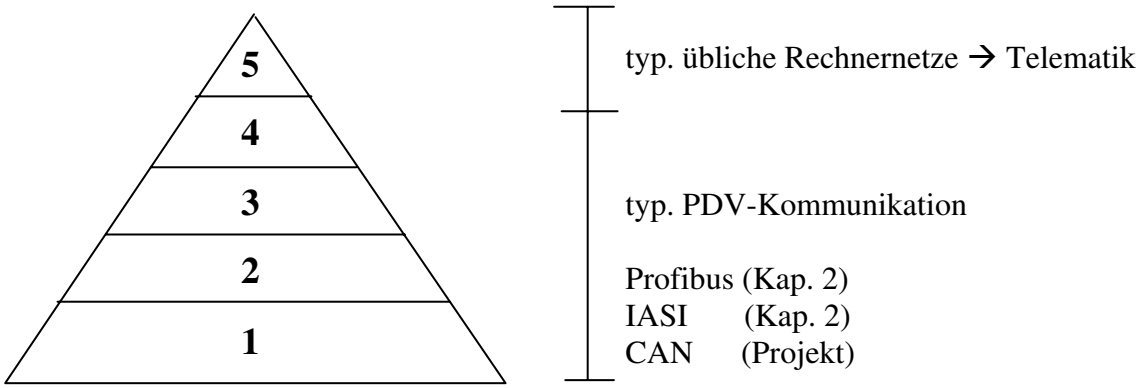
Zellebene:
- Zusammenfassen der Feldgeräte zu technologischer Einheit
  + Synchronisation + Koordination der Feldgeräte
  + Zielvorgaben für Feldgeräte
- zyklisch, azyklisch

Prozessleitebene:
- Zusammenfassen von Zellen zu techn. Anlage (relat. autonom)
- Synchron. / Koordination auf höherem Niveau
- evtl. Anlagenoptimierung / ~bedienung
- vorwiegend azyklisch

Planungsebene:
- Übergang kaufmännischer Bereich
- Produktionszielvorgabe
- azyklisch

V2: 16.04.04
RN der PDV

typische PDV-Kommunikationssysteme



Nachrichtenarten in den PDV-Ebenen:

Nachrichtenart	Alarme	Soll-/Ist-stellwerte	Synchron.-signale	Datenfiles	Steuer-programm	Prozeß-leitbilder
zulässige Wartezeit	0,1…10 ms	20..100 ms	1ms … 1s	1..100 s	1…100 s	1…100 s
Auftritts-häufigkeit	selten, azyklisch	häufig, zyklisch	mittel, zykl./azykl.	selten, azyklisch	selten, azyklisch	selten, azyklisch
Datenmenge	8..64 bit	256 bit	8..64 bit	1-10 KB	> 10 KB	> 10 KB
typ. Ebenen	1-3	1-3	2-4	(2),3,4	(2),3,4	(2),3,4

- Allgemeine, i.a. qualitative Anforderungen

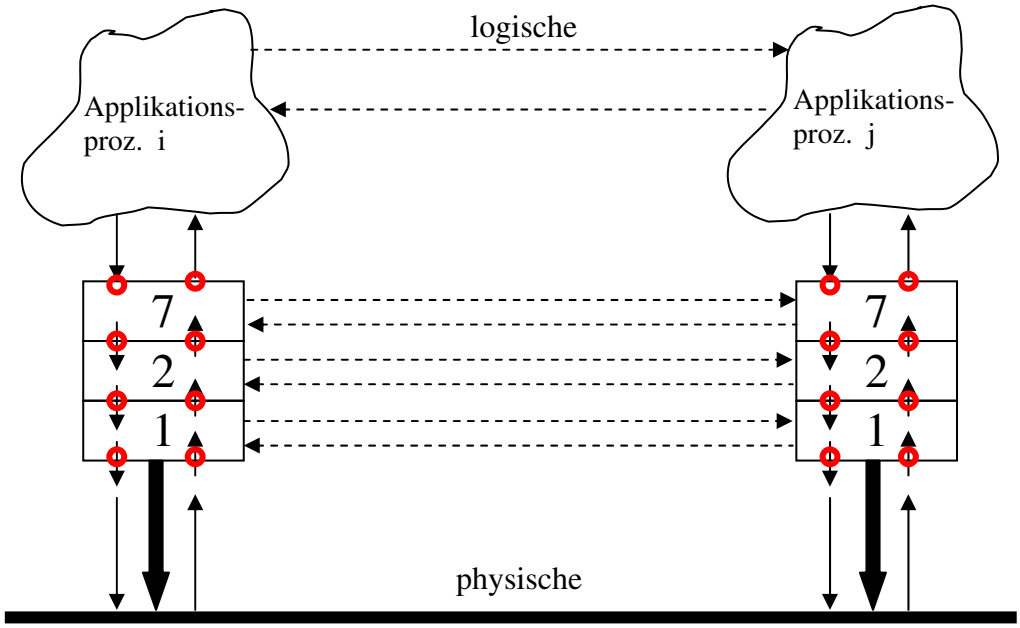
<u>These:</u> → nicht ein Kommunikationssystem kann alle Anforderungen erfüllen

max (Leistungen) **korreliert nicht** min(Preis)

➔ soviel Leistung wie nötig

Nichttechnische Gründe	Technische Anforderungen
* Marktgesichtspunkte * historische Entwicklung * angemessener Preis für Kommunikation in Abhängigkeit vom Gerätepreis * nationale und lokale Standards * Herstellerunabhängigkeit → Standardisierung	* Medium muß mit geringem technol. Aufwand verlegbar sein * Komm.-entfernungen: einige **m** bis **km** * Komm.-struktur muß der Anlagenstruktur anpassbar sein * je näher am Prozess, desto härter die Echt-zeitanforderung * Parametrisierbare Übertragungsraten (meist umgekehrt prop. zur Komm.-entfern.) * Störsicherheit bzgl. der umgebenden Prozesse * OSI-Schichten: 1,2,7 unterstützt

- ISO-OSI-Schichten → Standardmodell für ebenenorient. Kommunikation



7) Anwendungsschicht (application layer)

2) Verbindungsschicht (link layer)

1) Physische Schicht (physical layer)

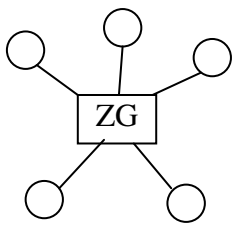
Vertikaler Datenaustausch: * Schicht 1 nutzt Dienste der Schicht *i-1* (Services)
* SAP… Service access point ○

PDV-Komm.-systeme hier: - homogen
- Nahverkehr
→ OSI 3..6 kann entfallen

Prinzipielle Realisierungsmöglichkeiten

Topologie:

Stern:

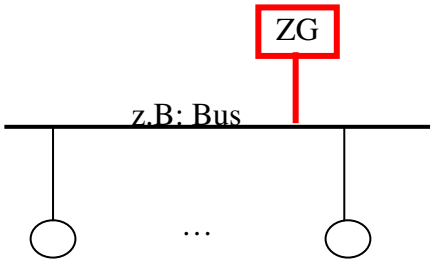


KT

Vorteile:
- relativ einfache Verwaltung
- mehrere Komm.-beziehungen gleichzeitig

Nachteile:
- hoher Verbindungsaufwand
- Ausfalltoleranz vom ZG=0
- ZG: **Flaschenhals** bzgl. Leistungsfähigkeit

Linie:



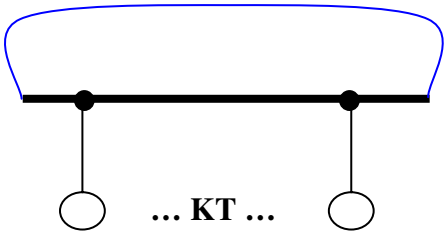
Vorteile:
- geringer Verbindungsaufwand
- Ausfalltoleranz einzelner KT

Nachteile:
- **Bus: Flaschenhals** bzgl. Leistungsfähigkeit
- evtl. höherer Verwaltungsaufwand

Bsp.: Profi-Bis

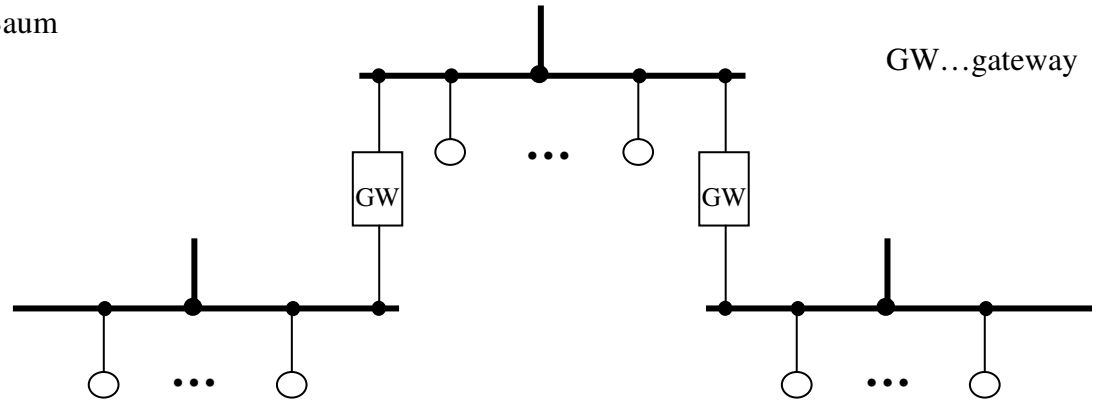
gemischt: Bsp.: ASI

Ring:



*** Zusammengesetzte Strukturen (Hierarchie)**:

→ Baum

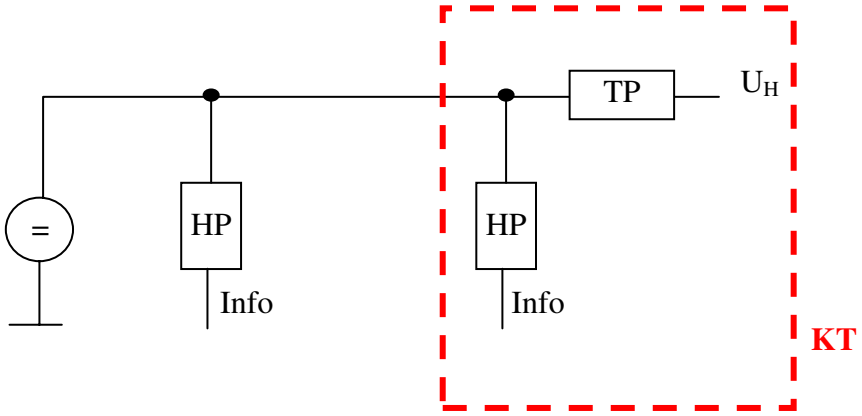


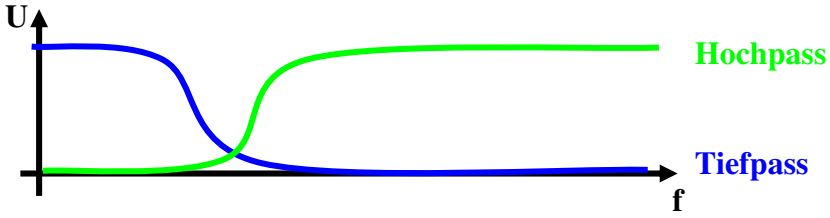
- auch unterschiedliche Strukturen in Hierarchisierung

Übertragungsmedium:

elektrisch	**optisch**
- einfacher verlegbar - kostengünstig - einfachere Hilfsenergieübertragung	- weniger störanfällig - höhere Übertragungsraten

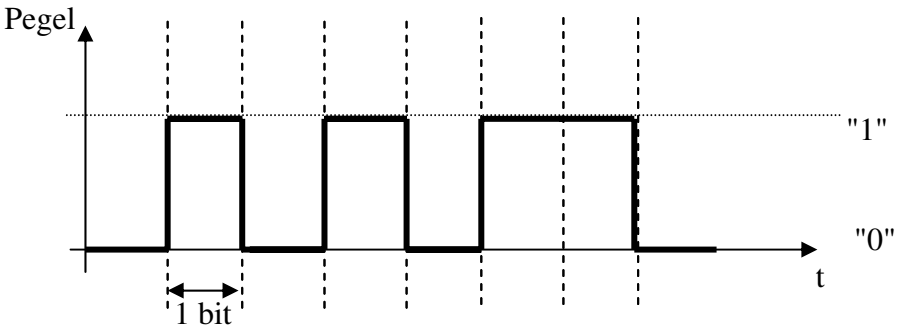
Übertragung der Hilfsenergie und Informationen auf einem Kabel:





*** Bitdarstellung**

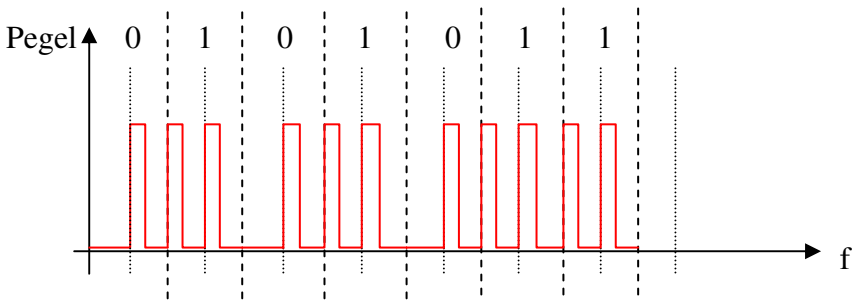
- physische Darstellung binärer Informationen (OSI-Schicht 1)

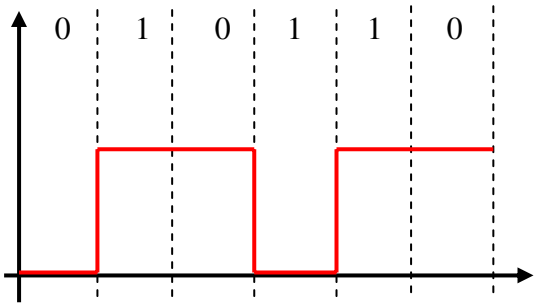


- Bitlänge wird über Zeit definiert ➔ Bitzeit

*** weitere Möglichkeit**

Pegel + Takt





0…kein Pegelwechsel

1…Pegelwechsel

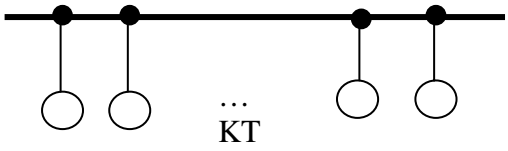
* Telegramm oder Datenrahmen → OSI-Schicht 2

- Zusammenfassung von mehreren binären Daten zu interpretierbarer Datenstruktur

Aufbau: + Nutzdaten (eigentliche zu übertragende Daten)
+ Steuerdaten (Art, Ziel, Quelle, u.ä.)
+ Sicherungsdaten (redundante Information zur Fehlererkennung bzw. Korrekturen)

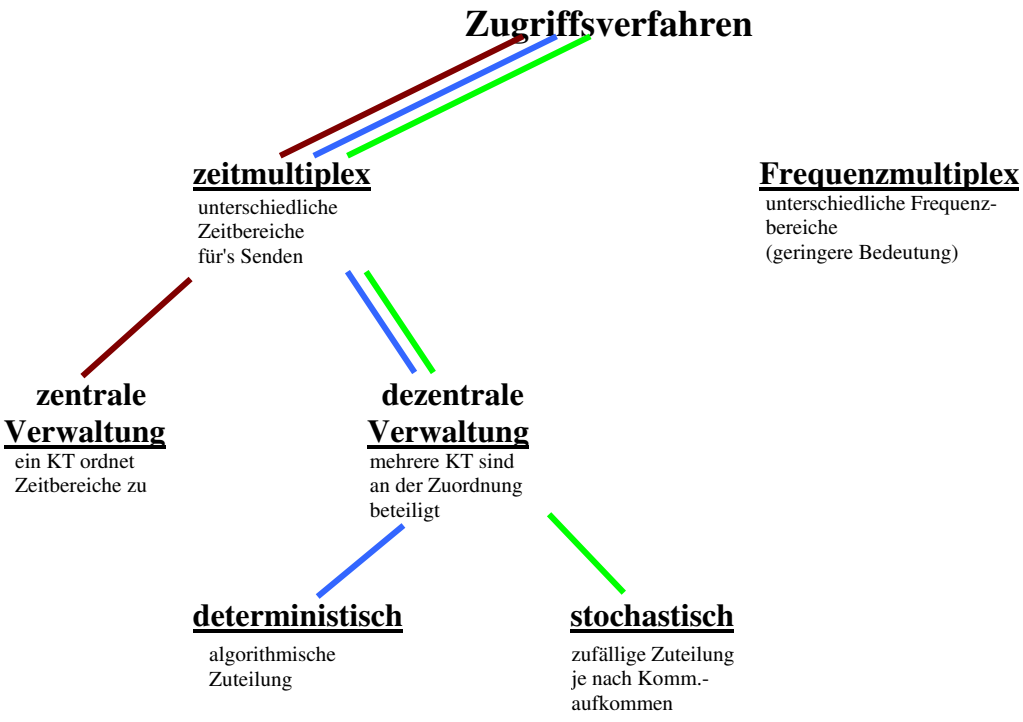
*** Zugriffsverfahren**:

1) mehrere sendefähige Teilnehmer (z.B. auf einer Linie)



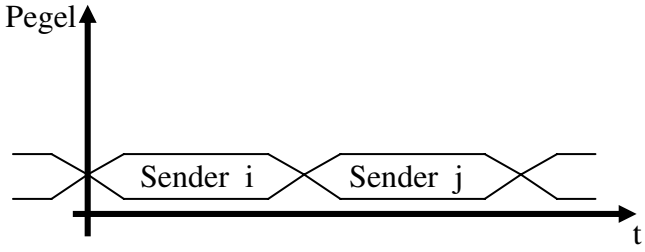
beim gleichzeitigen Senden → Problem

→ Lösung durch Zugriffsverfahren

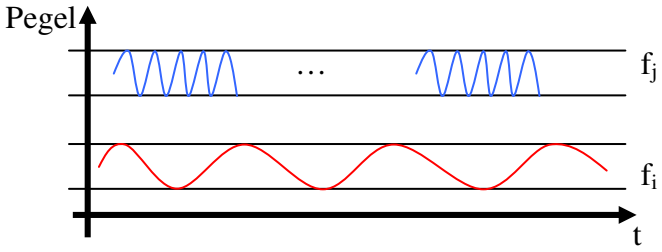


ASI-Bus (zeitmultiplex, zentral)
Profi-Bus (zeitmultiplex, dezentral, deterministisch)
CAN-Bus (zeitmultiplex, dezentral, stochastisch)

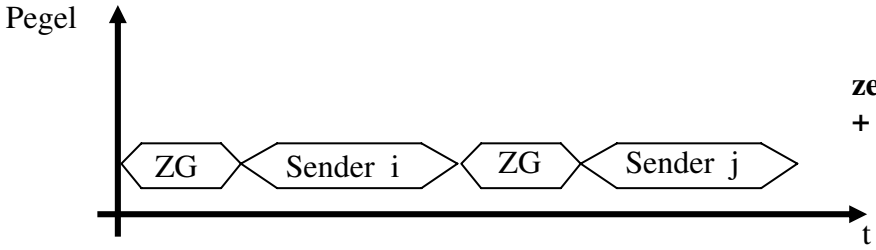
Beschreibung der einzelnen Klassen



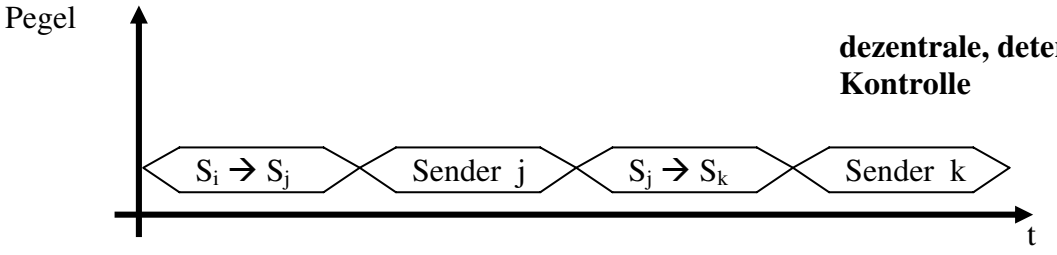
zeitmultiplex



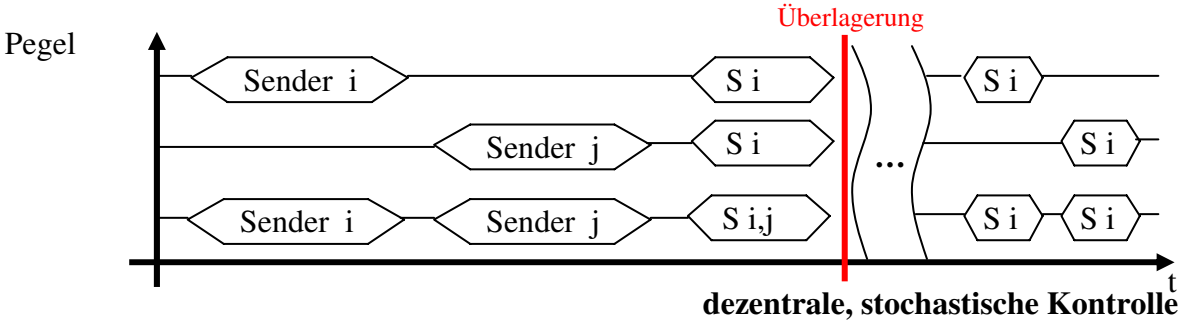
frequenzmultiplex



zeitmultiplex
+ zentrale Kontrolle (ZG)



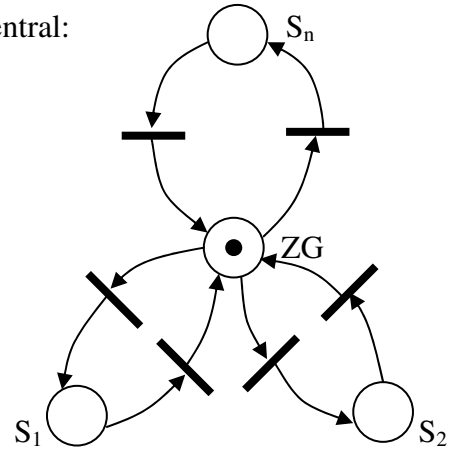
dezentrale, deterministische Kontrolle



dezentrale, stochastische Kontrolle

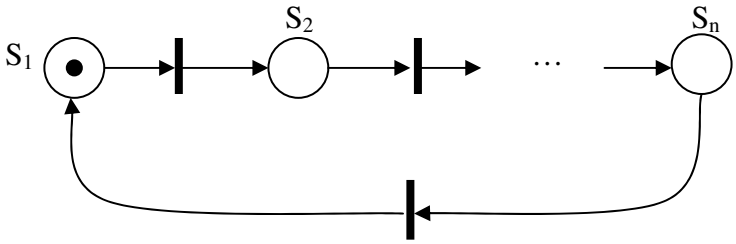
PN-Modell

* zentral:

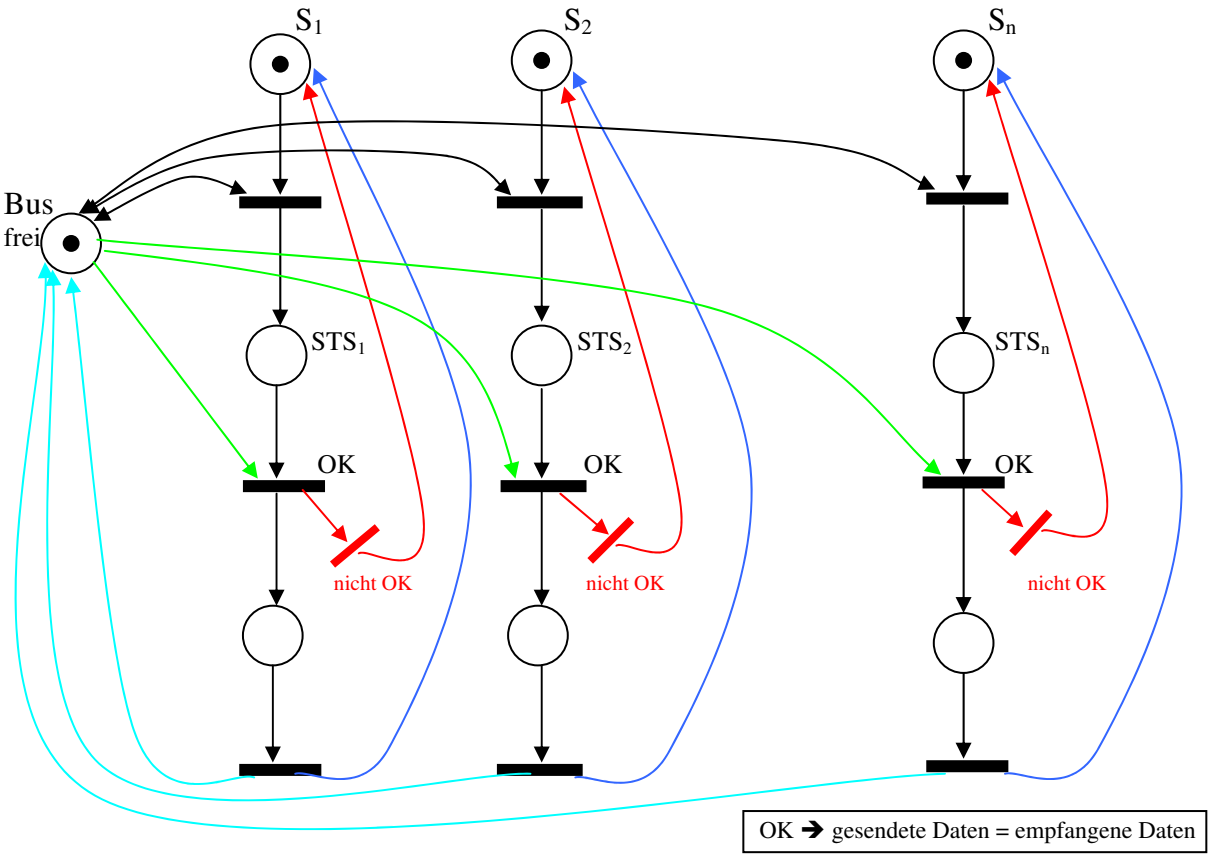


Marke: Kennzeichen "Senderecht"

*dezentral, deterministisch:



*dezentral, stochastisch:

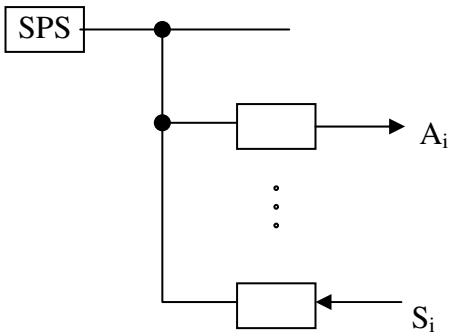


2. Feldbus der unteren Leistungsklasse (ASI)

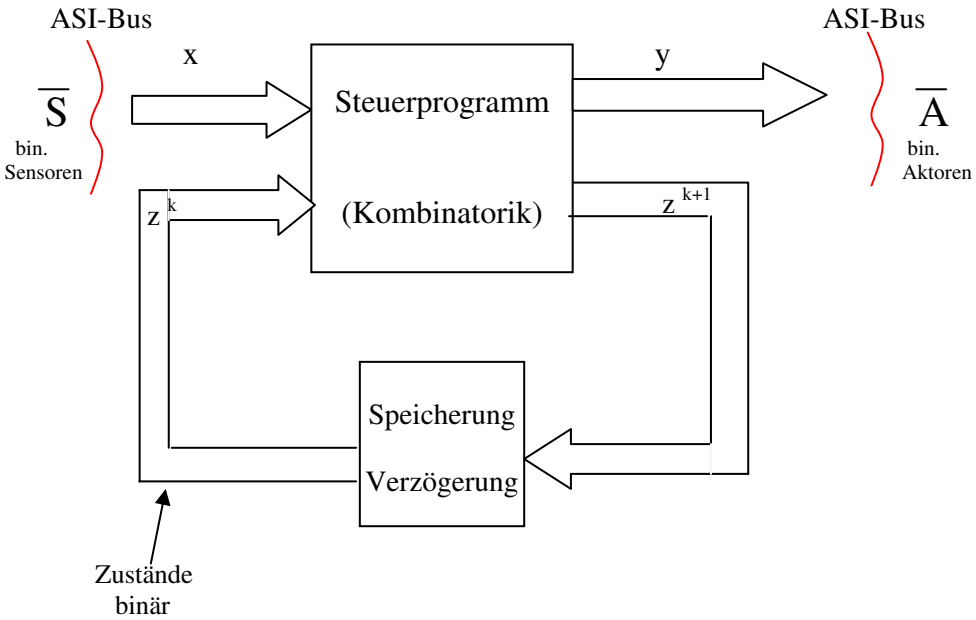
ASI…Aktor-Sensor-Interface

Einsatzbereich:

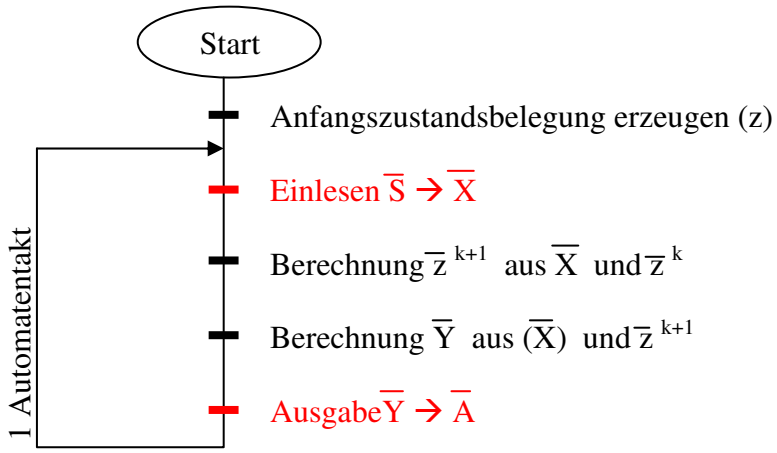
SPS: speicherprogrammierbare Steuerung



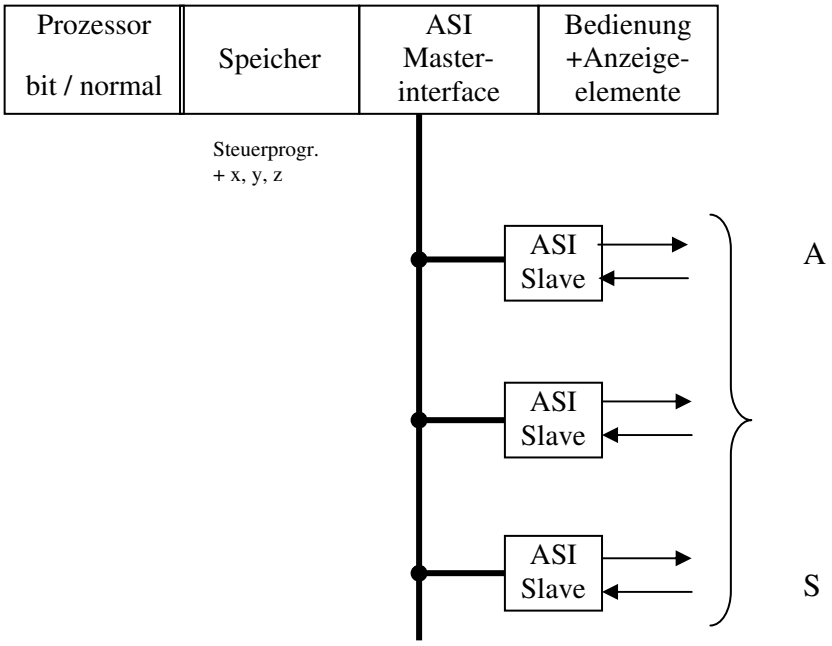
→ Einrichtung, die das klassische Automaten-modell abwickelt



***Grundprinzip der Abarbeitung:**



*** Aufbau ASI - SPS**

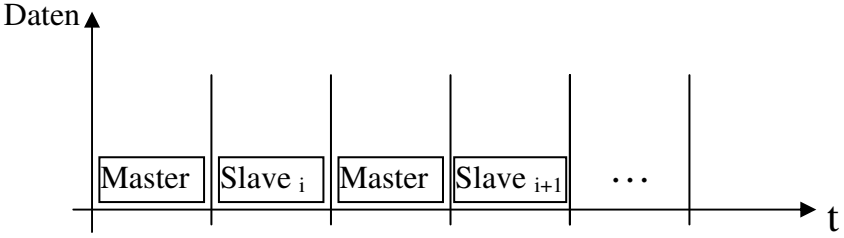


V4: 30.04.04
RN der PDV

ff.: ASI

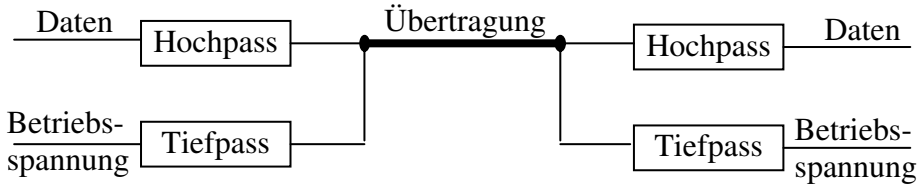
Allgemeine Eigenschaften:

* Charakter: serieller Bus
* Zugriffsverfahren notwendig
  → SPS verwaltet Zugriffsverfahren für den Bus
  → zentral, zeitmultiplex, deterministisch



Master: die SPS
Slave: i.a. E/A-Modul

- zur Unterstützung werden 2 Telegrammarten benötigt
- Daten- und Hilfsenergie werden über 1 gemeinsames Kabel übertragen (***frequenzmultiplex)***



➔ damit sind geringe Betriebs- und Übertragungskosten möglich

- Schichten (OSI): 1,2 (Schicht 7: Programmierwerkzeug-abhängig)

- Slave-Kosten sollen gering sein; Slave-Realisierung als digitale Hardware möglich

Verbindungen der Schicht 2:
- realisiert durch Telegramme (Master- und Slavetelegramm)

Mastertelegramm: - immer vom Master zum adressiertem Slave

ST	SB	A4	A3	A2	A1	A0	I4	I3	I2	I1	I0	PB	EB

ST… Kennzeichen für Beginn einer asynchronen, seriellen Kommunikation
SB… Steuer-bit; Unterscheidung der 2 Telegrammarten
A_i … Adress-bit; 5 Stk. → 2^5 Slaves unterscheidbar
I_j … Informations-bits; → 5 verschiedene Bitinformationen
PB…Paritäts-bit (nur 1 Bitfehler erkennbar; Korrektur unmöglich)
EB…Ende-bit; kennzeichnet das Telegrammende
- ist genau umgekehrt zum ST

- nach dem Mastertelegramm folgt prinzipiell die Masterpause
  → diese ermöglicht dem Slave die Antwort vorzubereiten
  (zw. 3 bis 10 bit-Zeiten)

➔ dann folgt die Slaveantwort über das

Slavetelegramm:

ST	I4	I3	I2	I1	I0	PB	EB

- Adresse entfällt, da Slaveantwort prinzipiell an den einen Master geht
- Slave-Telegramm hat nur einen Typ → SB entfällt auch

Behandlung von Übertragungsfehlern

- Erkennung über Paritätsbit (ist dann, bei einem Fehler, falsch)

→ **beim Slave:** - benutzt dann die Daten einfach nicht (macht also gar nichts !)
→ warten auf nächstes Master-Telegramm

→ **beim Master:** - benutzt die Daten ebenfalls nicht
→ warten auf nächstes Slave-Telegramm dieses Slaves

!! die Abfrage der Telegramme geschieht zyklisch !!

* Fehler: **Slave wird nicht angesprochen**: → wie normale Nicht-Adressierung
(d.h.: tut also nichts)

angesprochener Slave antwortet nicht → wird wie Paritätsfehler behandelt
(d.h.: warten auf nächste Slaveantwort dieses Slaves)

<u>* Sonderfälle:</u>

*** Parameteraufruf:** - ein kompletter Zyklus
(Zyklus: Master kommuniziert einmal mit jedem Slave)
- Übertragung **keiner** E/A-Info's, sondern Parameter

*** Diagnoseaufruf:** - Suchen neuer Slaves durch den Master, durch einmaliges Ansprechen aller möglichen adressierbaren Slaves
→ die Antwortenden werden als vorhanden geführt
(diese Prozedur wird auch beim Zuschalten des des Masters ausgeführt)

3. Der Profi-Bus - Feldbus der oberen Leistungsebene

Profibus: process field bus

- standardisiert

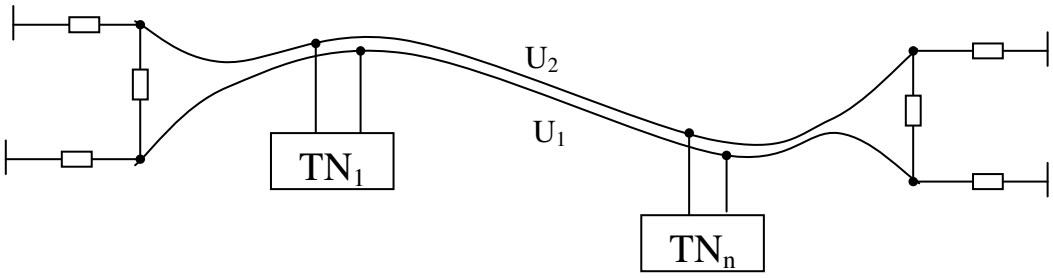
Zielgebiet: - Verbindung von Geräten / Komplexen der Feldebene und darüber
→ verschiedene Datenstrukturen: Realisierung über Schicht 7

- notwendig: Tolerierung von Teilnehmerausfällen

Funktionsweise:

* Übertragung: - elektrisch (über 2 Drahtleitungen) oder
- optisch (Lichtleiter)
- Hilfsenergie nicht Bestandteil des Busses

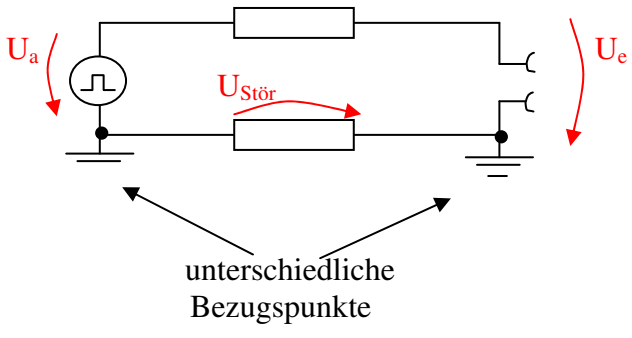
* Topologie: - wie bei ASI → serieller Bus



logische Werte: "1" : $U_1 > U_2$ } **Prinzip:** "Differenz-spannung"
"0" : $U_2 > U_1$

Vorteil: unempfindlich gegen Gleichtaktfehler

Warum ?

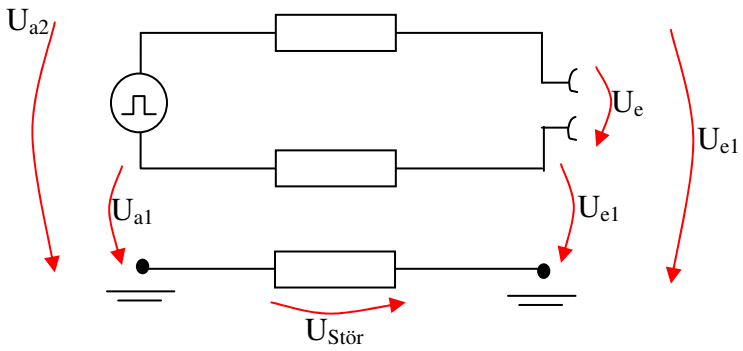


Übertragung (z.B.):

"1" = U } $U_e = f(U_a , U_{Stör})$
"0" = 0

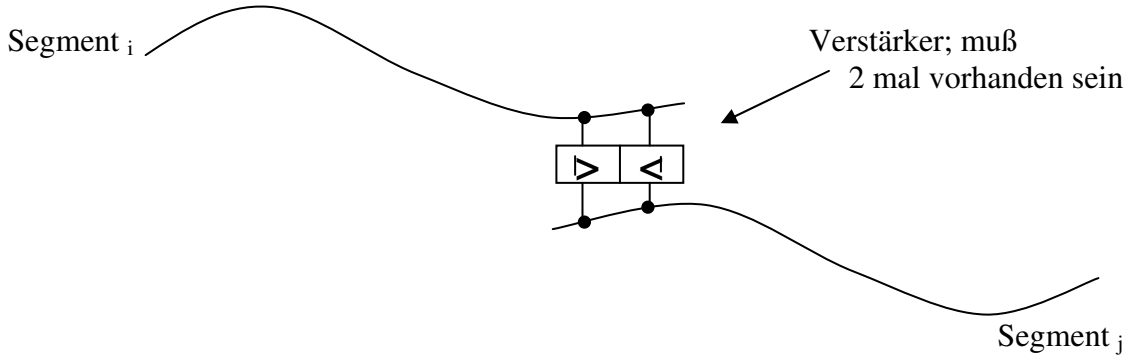
$U_e = f(U_{a1}, U_{a2})$

$\neq f(U_{Stör})$



Logikdarstellung: NRZ (non return to zero)
- ohne elektronische Verstärkung $< 2^5$ Teilnehmer
  (Grenze durch elektrische Parameter, nicht durch logische !)

* Verbindung von Segmenten:



Pr.VL 1
Rech.d.PDV
04.05.04

http://www.theoinf.tu-ilmenau.de/~nuetzel/bus.htm

Termine: - 2er Gruppen; 2 Termine á 180 min (H1501)
- im Juni
- 7-8 Gruppen pro Termin

- vorgeschlagene Tage: Mo, Die, Do, Fr (jeweils vor- und Nachmittag)

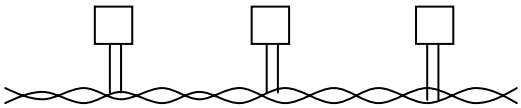
- letzte VL → "kleines Quiz zum Projekt"

Projekt zum CAN-Bus

CAN: controller area network
- in der Hierarchie unter dem Profi-Bus
- von Bosch (1983) für den Automobilbereich entwickelt
→ Protokoll; hier aber nur Schicht 1 und 2

- verdrillte 2-Drahtleitung mit Differenzspannung
- Schicht 2 ist voll in HW realisiert
- im Projekt: C164 CI (16-bit Micro-Controller)

Topologie: - Busstruktur (mit 3 Teilnehmern)



- verschiedene Typen:
  * Basic-CAN 2.0A (11bit Identifier)
  * Basic-CAN 2.0 B (11bit und 29bit Identifier)

besitzen Sende / Empfangspuffer (für Objekte und Telegramme)

* FULL-CAN 2.0 A/B
  - mehrere (typischerweise 15) Sende-/Empfangspuffer

-CAN-Bus ist Multi-Master-System (d.h.
- jede Station ist Master
- jeder kann Senden / Empfangen
- keiner braucht "fragen", ob er senden kann
→ "Schiedsrichter" erforderlich
➔ "***Bus-Arbitrierung***"

-CAN-Protokoll: - stochastisches Verfahren
- CSMA / MA (carrier sense multiple access with collision avoidence)

- definierte Vermeidung von Buskonflikten
(keine Vermischung der Sendesignale auf dem Bus)
- Telegramm:

11 bit	0-8 byte

Identifier | Daten (inkl. Größenangabe der Daten)

- 0-Pegel ist dominant !!! (1-Pegel ist unterordnend)

Regeln des Protokolls: 1) Alle sendebereiten Stationen senden gleichzeitig nachdem der Bus freigeworden ist

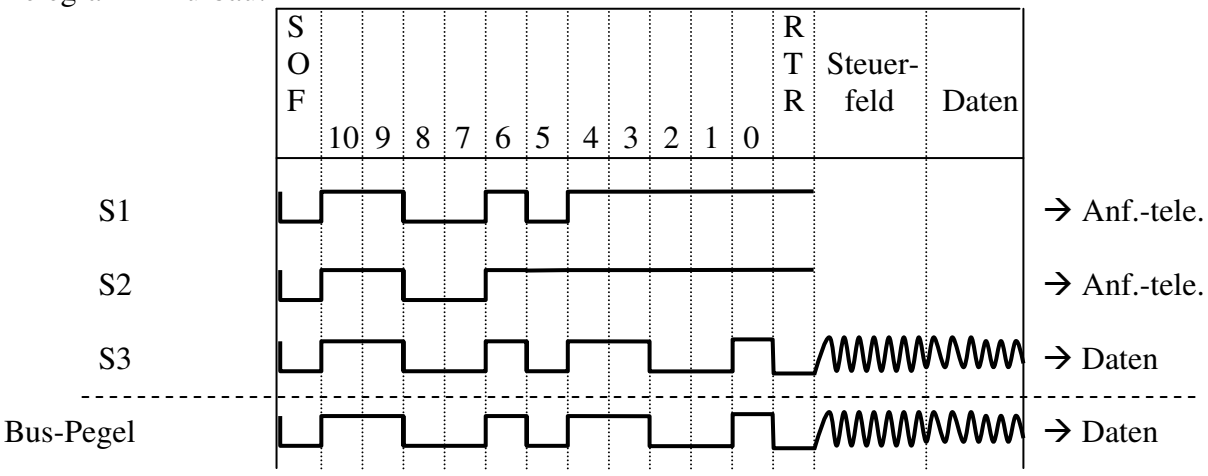
2) Identifier werden bit für bit gesendet und überwacht

3) Station, die eine "1" sendet, aber eine "0" auf dem Bus erkennt, bricht ihren Sendevorgang ab

- über den Identifier werden die **Nachrichten** priorisiert (nicht die Stationen selbst !!)

Beispiel (wieder mit 3 Stationen):

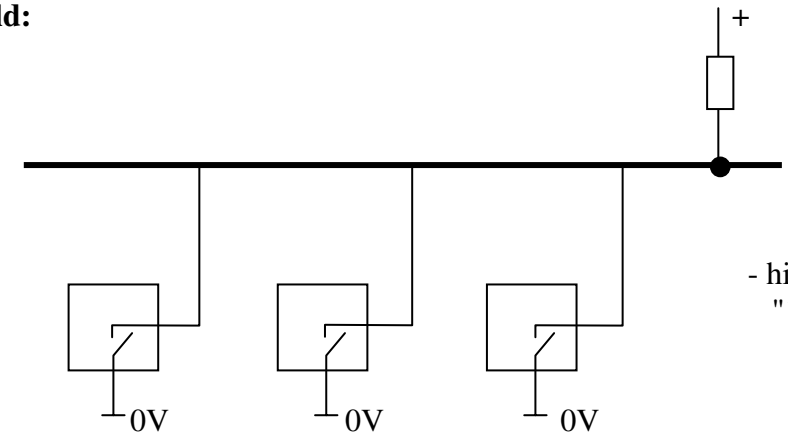
Telegramm-Aufbau:



(Anf.-tele.: Anforderungstelegramm
Daten: hier werden Daten gesendet)

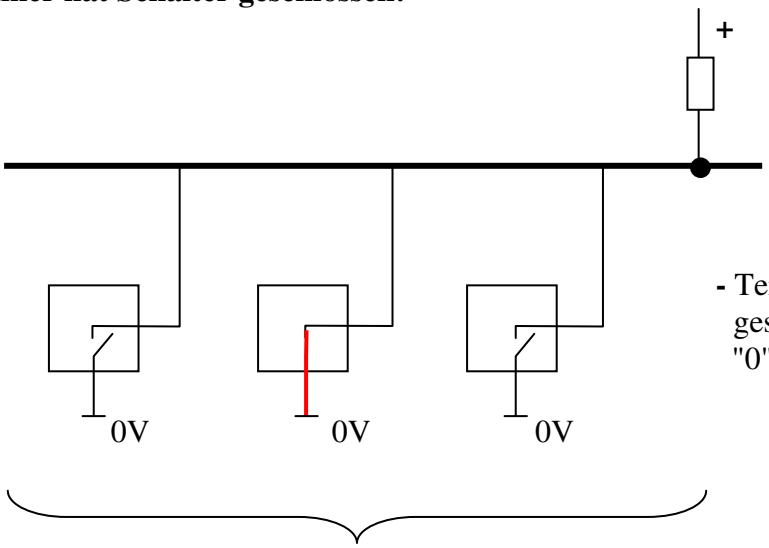
SOF: Start of Frame
RTR: remote transmission request (Anforderung)
bei "0"-Pegel → **Daten**
bei "1"-Pegel → **Anforderung** (~von Daten)

Schaltbild:



- hier senden noch alle "1"

2. Teilnehmer hat Schalter geschlossen:



- Teilnehmer 2 hat Schalter geschlossen und sendet "0" → priorisiert

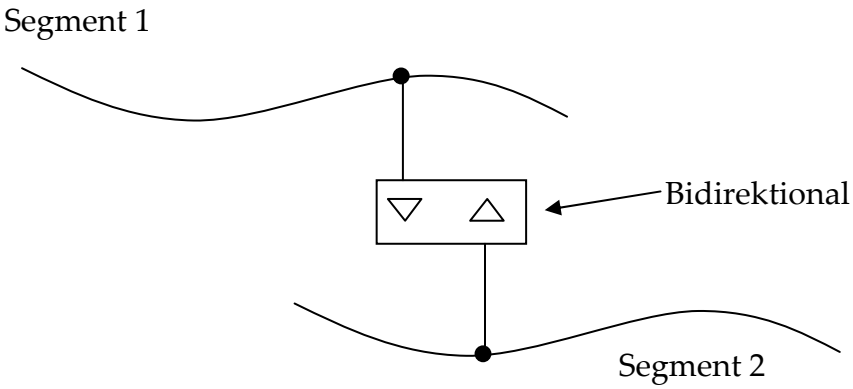
entspricht Schreibfunktion

3.2. Profibus (Physical Layer) – Schicht 1

V: 28.05.04
RN d. PDV

- physisch begrenzt auf 31 Teilnehmer
  (auf Grund elektrischer Bedingungen)

➔ Vergrößerung durch "Repeater"

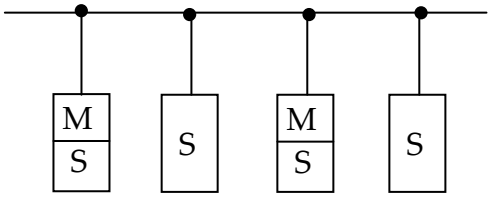


3.3. Link Layer – Schicht 2

→ Busverwaltung (zeitmultiplex, deterministisch, dezentral)

→ Protokolle des elementaren Datenaustausches
(Schicht 7 benutzt diese Protokolle)

* Prinzip der Busverwaltung:
Tokenbus (elektrisch)



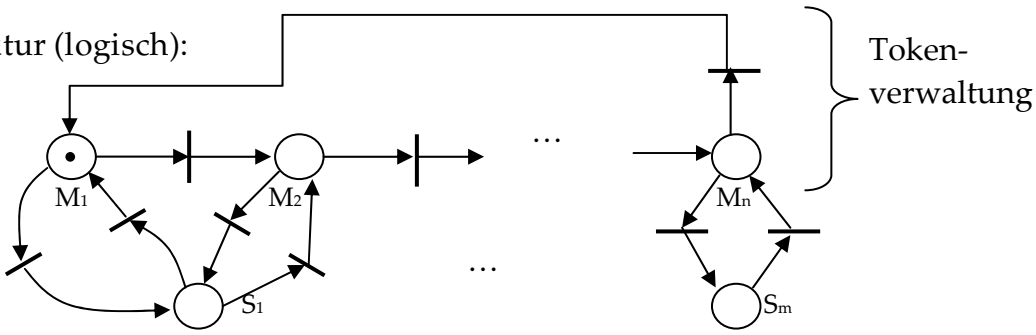
M…Master (Busverwaltung)
S… Slave (Datenaustausch)

Token…Kennzeichen:
Senderecht

Grenzwerte: * aller Teilnehmer (M/S)

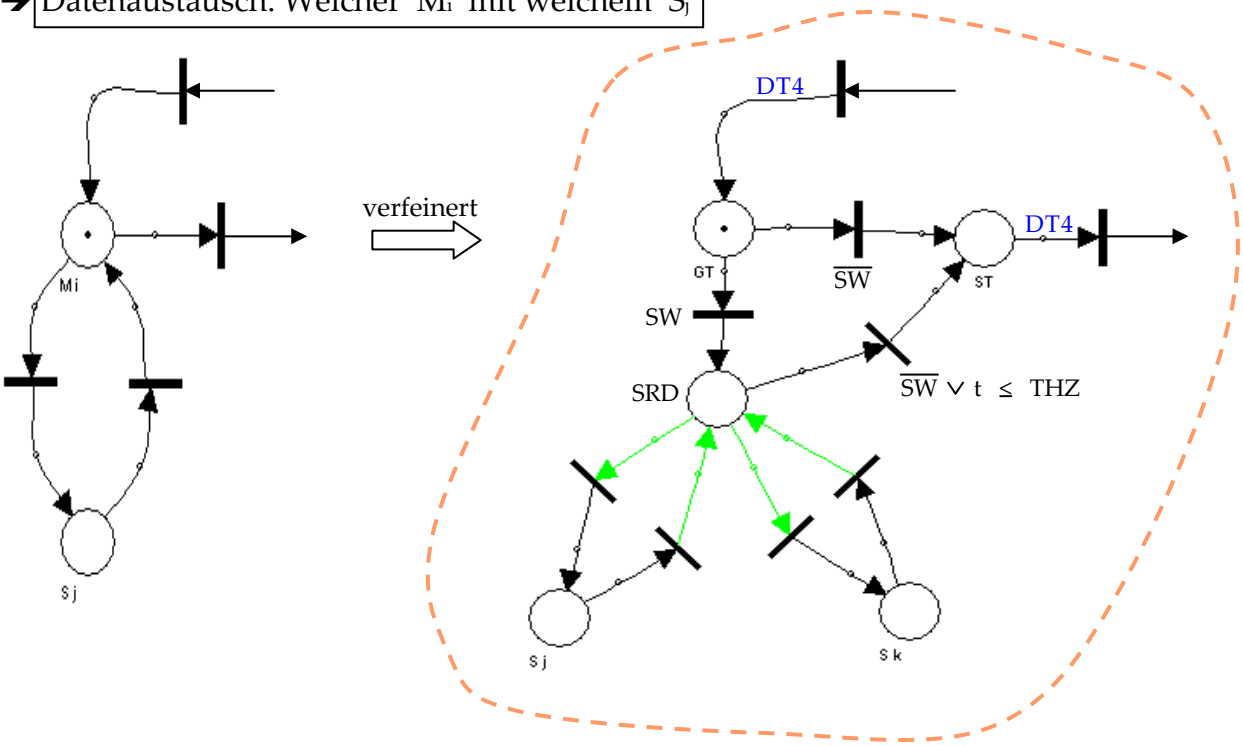
* mind. 1 (M/S), Rest (S) (sonst würde es nicht fkt.en)

Struktur (logisch):



- Master-Slave-Beziehung hängt von jeweiliger Situation ab
  (ist ausschließlich Applikationsabhängig)

➔ Datenaustausch: Welcher M_i mit welchem S_j



Sollverhalten ohne Fehler und Fehler-behandlung

→ $DT_{1..3}$

DT4… Datentelegramm 4 (Tokenweitergabe)
GT … Get Token
ST … Send Token
$\overline{SW}$… kein Sendewunsch; d.h. Applikations-Schicht benötigt keinen Datentransfer
SW… Sendewunsch
SRD... Send & Receive Data
THZ...maximale Tokenhaltezeit

Telegramme $DT_{1..4}$

DT_4 * Tokenweitergabe

SD_4	DA	SA

SD_i … Sende Limiter i (vom jeweiligen Token)
DA … Destination Address
SA … Source Address
- Prüfzeichen entfällt hier auf Grund der "Source Address"

Mögliche Fälle:

Keine Fehler: - Tokennachfolger (DA) erkennt DT_4 korrekt;
- übernimmt den Token
- Vorgänger erkennt Übernahme durch Datenverkehr

Fehler: → korrigierbar durch 2-3 malige Widerholung der Token-sendung

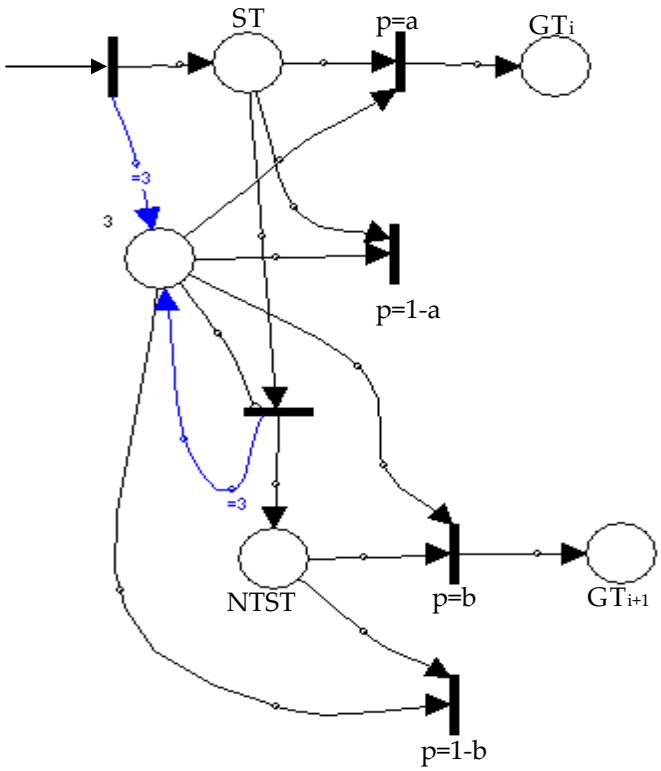
- *nach Tokentelegramm:* Ruhe auf dem Bus
  → dann Wiederholung (2-3 mal)
  - positiv i.a. wenn Störung des DT
  - wenn negativ wird Masterausfall vermutet u. nächster aktiver Master wird gesucht (worst case: er findet sich selbst wieder → totaler Busausfall)

- *durch Datenstörung:* DA wird von mehreren (mind. 2) Mastern als eigene interpretiert
  → Mithören zeigt allen sendenden Mastern, dass eigene Sendung gestört
  → Abbruch der Sendung → Ruhe auf Bus
  Tokenabsender verhält sich wie oben ( )
  + Tokenvernichtung

- *Integration neuer Teilnehmer i. d. Bus:* (nach Zuschalten oder Störungsbeseitigung)
  → längere Störungssendung
  → alle anderen brechen ab + Tokenvernicht.
  → Neuinitialisierung (alle versuchen neues Token zu generieren)
  - es beginnt der Master, mit der niedrigsten Adresse
  → keine Ruhe auf Bus → alle anderen brechen ab
  → Suche des nächsten akt. Teilnehmers durch Adressinkrementieren; Tokentelegramm und bis zu 3maliger Wiederholung

  erstmalige Neuinitialisierung erfolgt ebenso

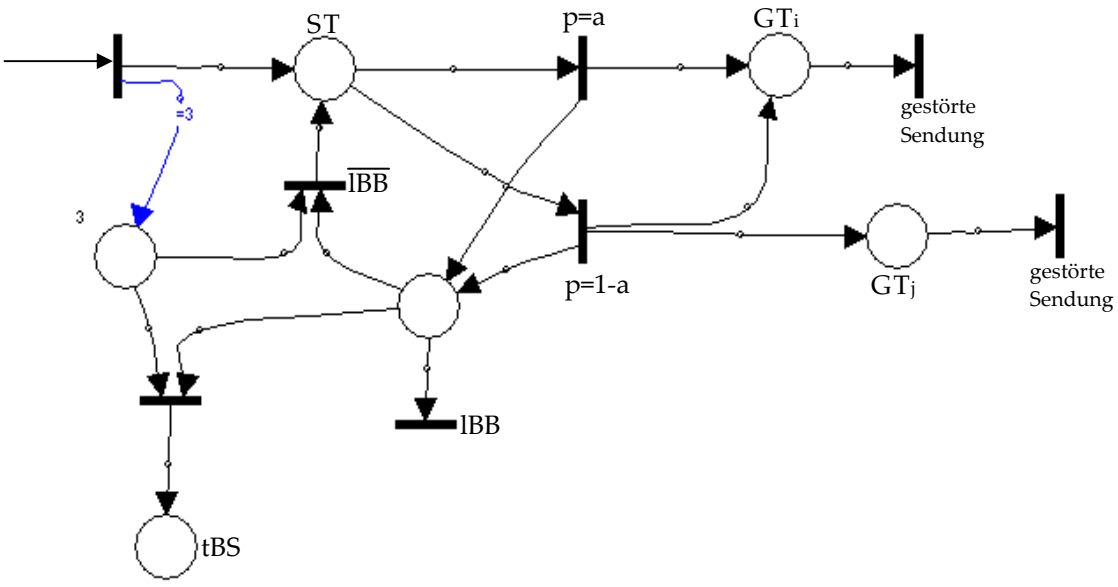
Tokenverlust



NTST … Nächsten Token und Send Token

- also neue Tokenadresse und Send Token

Tokenverdopplung



lBB … längerer BusBetrieb (mehr als 1 Zeichen)
tBS … totale BusStörung

Telegramme zum Datenaustausch:

DT_1 - Steuercode

SD_1	DA	SA	FC	FCS	ED

SD_i … Sende Limiter
DA … Destination Address
SA … Source Address
FC … Frame Control (Steuercode)
FCS… Frame check sum (Prüfzeichen)
ED … End Delimiter

DT_3 - für Daten mit fester Länge

SD_3	DA	SA	Daten (8 Byte)	FC	FCS	ED

DT_2 - für Daten mit variabler Länge

SD_2	LE	LER	SD_2	DA	SA	Daten (n-Byte)	FC	FCS	ED

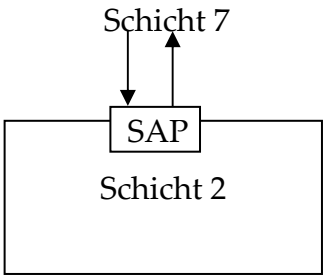
LE … Length
LER … Length repeat

- mehr Telegramme existieren nicht beim Profibus

ff.: "Dienste" zum Profibus

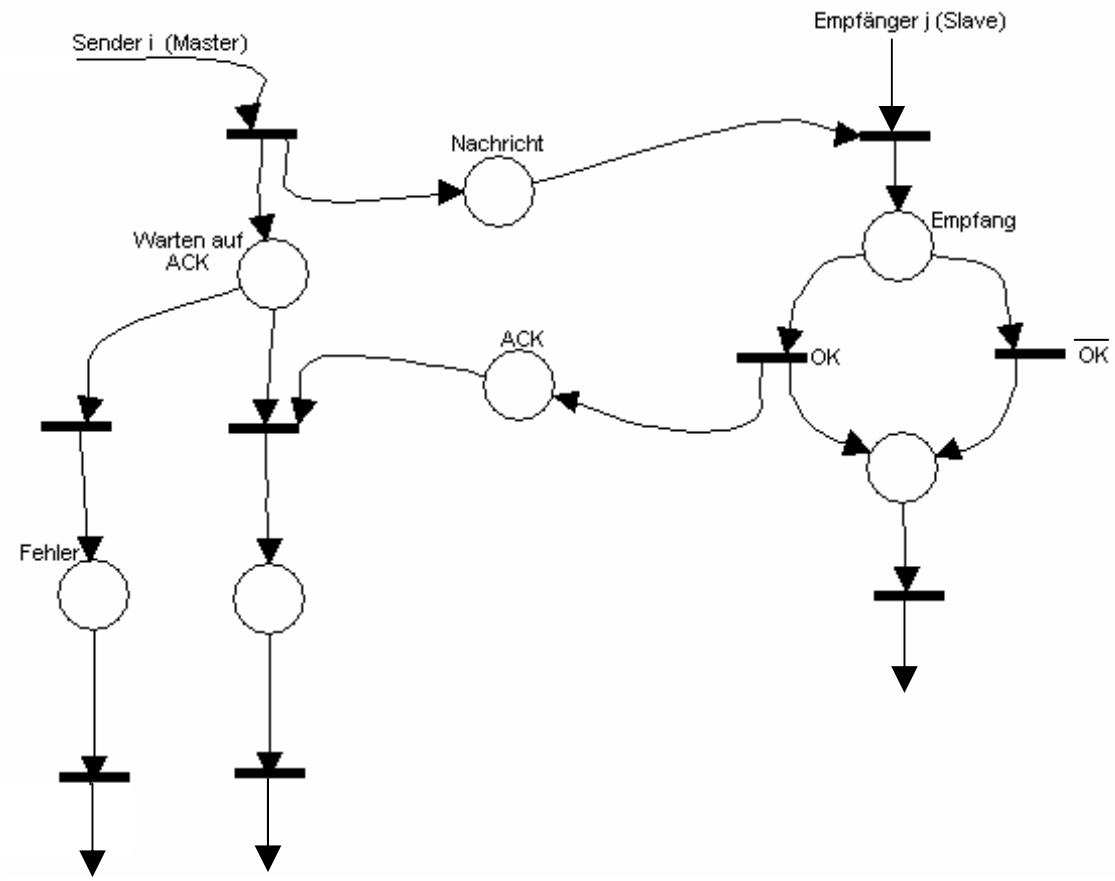
V10: 11.06.04
RN d PDV

* Dienste (Services)



1. SDA (Send Data with Acknowledge)

- Daten senden mit Bestätigung

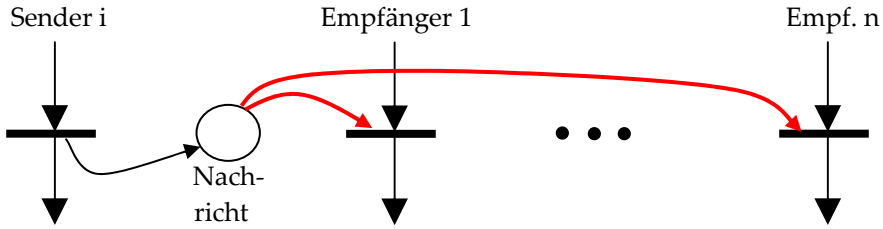


- nur bei 1:1 Datenverbindung möglich
- bei Kommunikation M ⟷ M muß der 2. Master auch Slave-Funktionalität haben

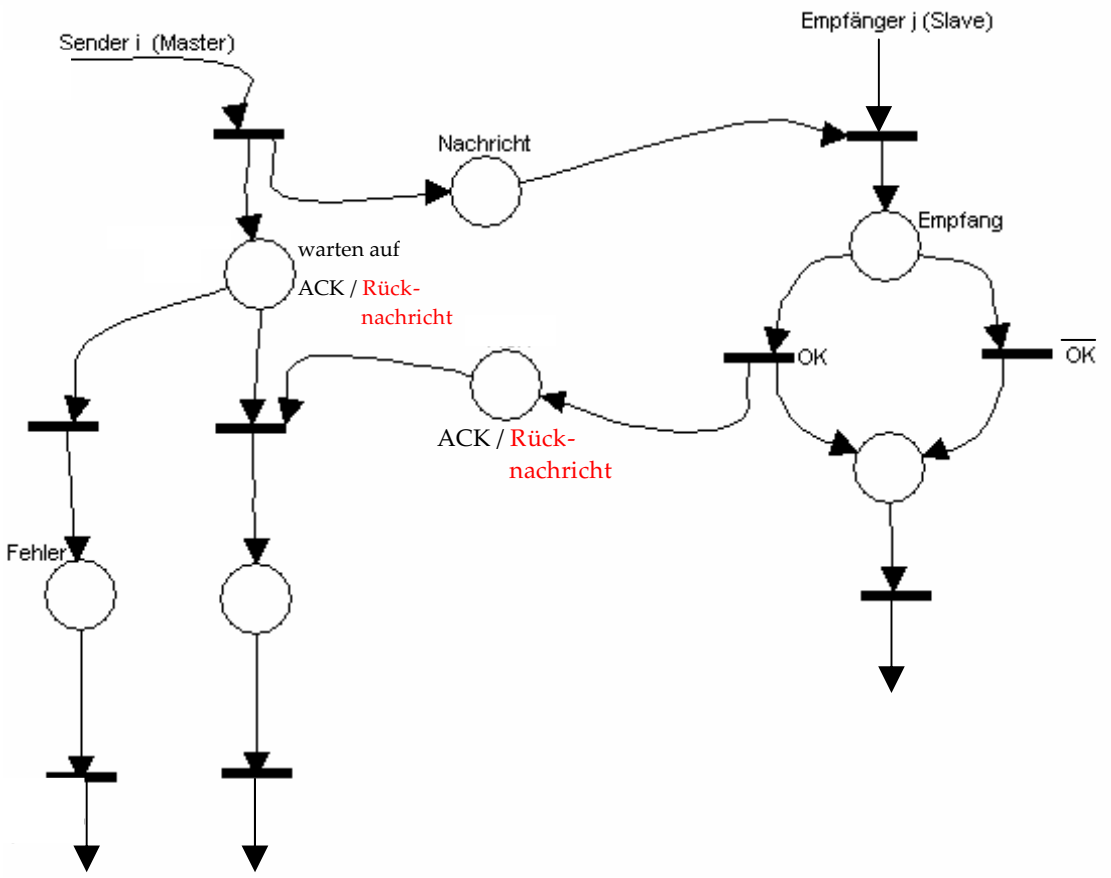
2. SDN (Send Data with no Acknowledge)

- 1 Sender → n Empfänger

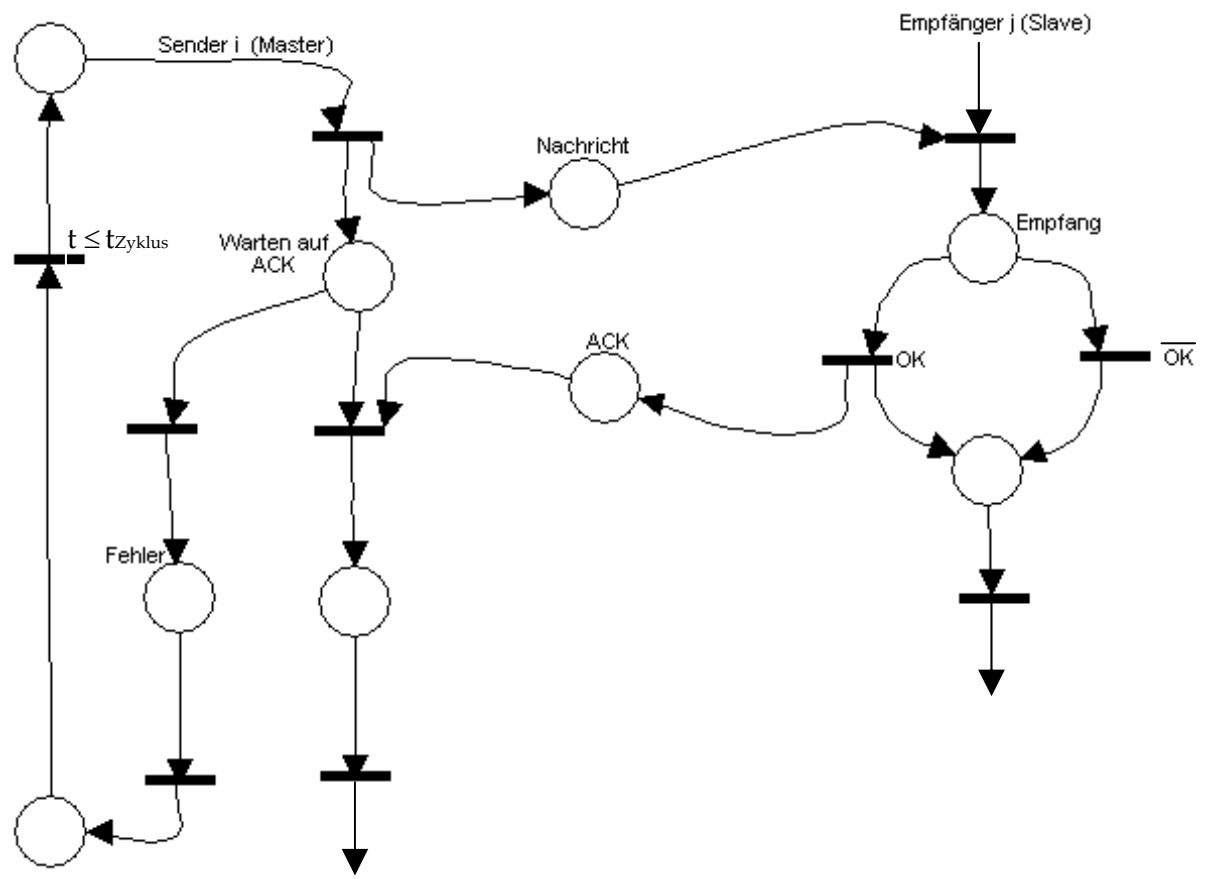
Schema (vereinfacht!):



3. SRD (Send requested Data with Reply)



4. CSRD (cyclic SRD)



3.3. Anwendungsschicht (Schicht 7)

- bildet Schnittstelle zur Anwendung;
- nutzt Schicht-2-Dienste

Datenstrukturen: - ähnlich wie bei höheren Programmiersprachen;
spezielle Erweiterungen für Steuerung und Echtzeitbetrieb

➔ * Boolean; Integer (8, 16, 32, 64 bit);
* String
* unsigned (8, … , 64 bit); Bitstring (Zusammenfassung von Binär-variablen)
* Float Point
* octet string (Kette von Oktalzahlen)

Zeitformate: * Date; time of Date; Delay; Time Difference;

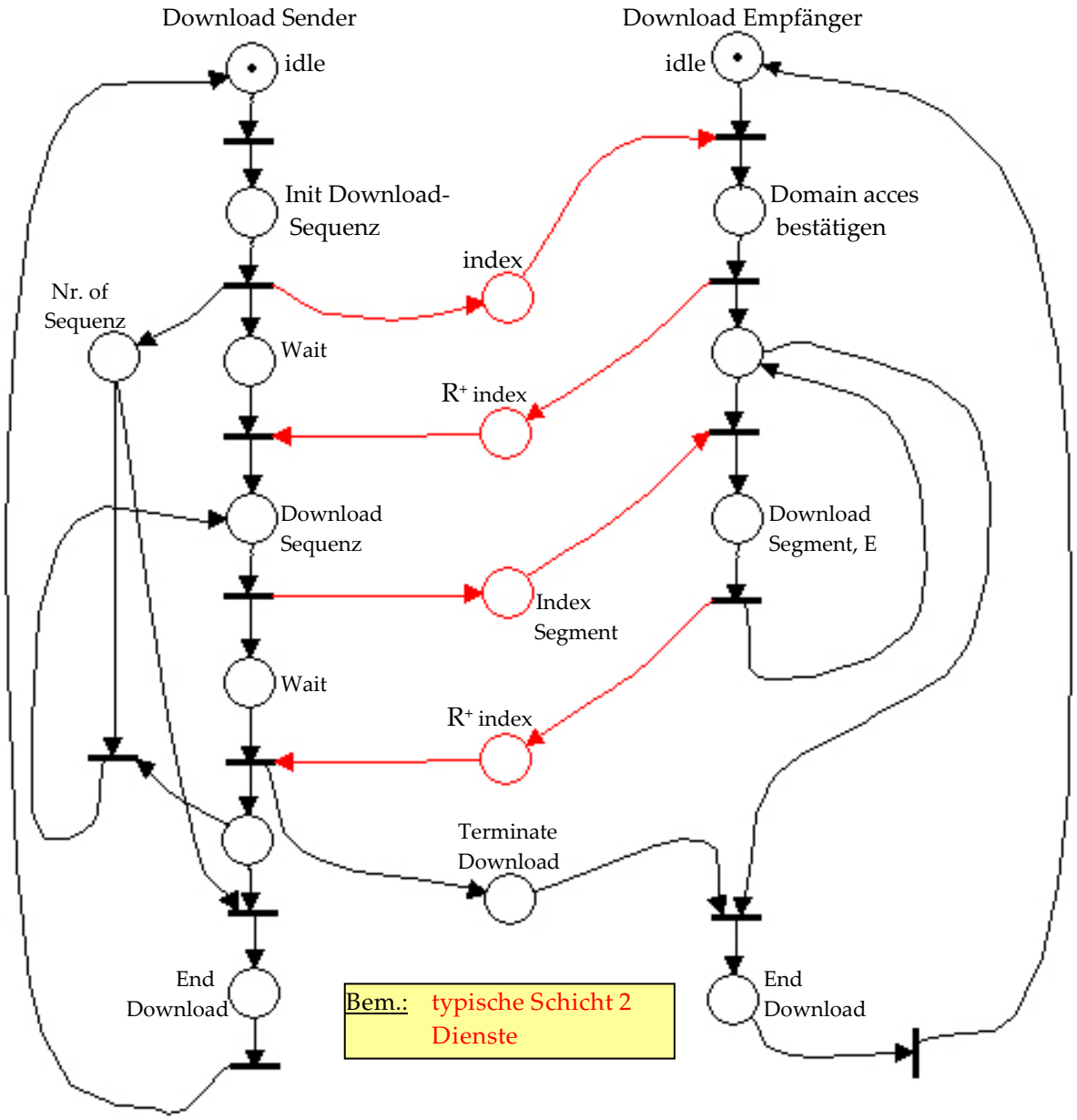
Objekte der Schicht 7 (keine Objekte im OO-Sinn !!!!)

* Variable
* array-Variable (Feld gleicher Variablen)
* Record-Variable (Feld ungleicher Variablen)
* Domainobjekte (logisch zusammenhängender Speicherbereich; enthält typ.weise Daten/Programme)
* Alarme
* Variable List (zur Laufzeit zusammengefasste statische Variable)
* Programm Invocation (Zusammenfassung mehrerer Domains; allg. lauffähiges Programm)

Dienste: i) Anwendungsdienste

- Datenaustausch zwischen Prozessen
- read/write: lesen/schreiben einer Variablen
- read/write with type: lesen/schreiben einer Variablen mit Typüberwachung
- physical read/write: physisches lesen/schreiben einer Speicher(zelle/adresse)
- Information report: lesen/schreiben mit unbestätigtem Dienst)
- Define/Delete Variable List: Erzeugen/Löschen einer Var.-Liste
- Domain access: lesen/schreiben einer Domain

Bsp.: Download Domain (ohne Fehlerbehandlung)
Umsetzung Schicht 7-Dienst in Schicht 2-Dienste



- Programm Invocation
  + Create/Delete
  + Start / Stop
  + Reset (in Anfangszustand zurück)
  + Resume (Fortsetzen eines gestoppten Prgr.)
  + Kill

- Event Management:
  - E-Notifikation (Ereignis anmelden)
  - Acknowledge E-Notifikation (Bestätigung)
  - Alter Event Condition-Monitoring

ii) Verwaltungsdienste
- Arbeit im Objektverzeichnis
  (Objektverzeichnis enthält alle Objekte einer Profibuskonfiguration)
- Abbrechen u. Einrichten von Komm.-verbindungen
  (ungestörte Kommunikation)

iii) Netzwerkmanagementdienste
- Behandlung von Fehlern u. Ausnahmesituationen (gestörte Komm.)