ulm university universität ulm

Grundlagen der Rechnerarchitektur

[CS3100.010]

Wintersemester 2014/15

Heiko Falk

Institut für Eingebettete Systeme/Echtzeitsysteme
Ingenieurwissenschaften und Informatik
Universität Ulm

ulm university universität ulm

Kapitel 8

Ein-/Ausgabe

Inhalte der Vorlesung

1. Einführung
2. Kombinatorische Logik
3. Sequentielle Logik
4. Technologische Grundlagen
5. Rechnerarithmetik
6. Grundlagen der Rechnerarchitektur
7. Speicher-Hardware
8. **Ein-/Ausgabe**

Inhalte des Kapitels (1)

8. Ein-/Ausgabe

- Einleitung
  - Datentransport von/zu Speicher oder E/A-Bausteinen
  - *Memory-mapped* I/O
  - Spezielle Ein-/Ausgabebefehle
  - *Interrupts*
- Prinzipen der Datenübergabe
  - Synchrone Busse / unidirektionales *Timing*
  - Asynchrone Busse / bidirektionales *Timing*
  - *Fully-Interlocked Handshaking*
  - Synchronisation Prozessor ↔ Geräten:
    Busy Waiting, *Polling*, *Interrupts*, DMA
- ...

Inhalte des Kapitels (2)

8. Ein-/Ausgabe

- ...
- *Point-to-Point* Verbindungen
  - Fallstudie: Serielle Schnittstelle (RS-232 / V.24)
  - Fallstudie: Paralleler Port (Centronics / IEEE 1284)
- Busse
  - Arbitrierungsverfahren:
    Daisy Chain, parallele und zentrale Arbitrierung, verteilte Arbitrierung
  - Interne Busse
    - Fallstudie: PCI-Bus, PCI-X, PCI-Express
  - ...

Inhalte des Kapitels (3)

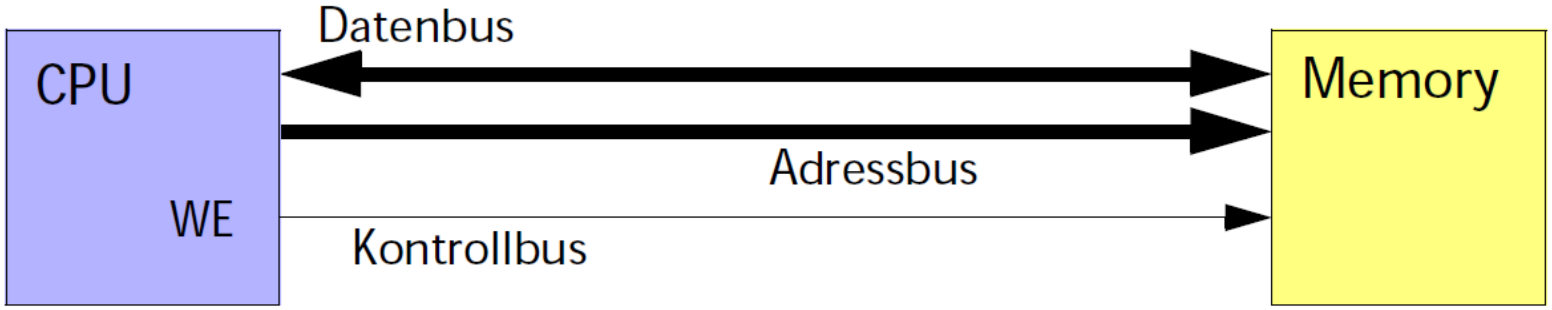
8. Ein-/Ausgabe

- ...
- Externe Busse
  - Fallstudie: IDE, ATAPI, SATA
  - Fallstudie: SCSI
  - Fallstudie: USB
- Fehlererkennung mit CRC-Zeichen
  - Polynom-Darstellung von Nachrichten
  - Berechnung der Prüfbits
  - Überprüfung von Nachrichten
  - Erkennbare Fehler
    - Einzelfehler
    - Ungerade Fehlerzahlen
    - Burstfehler

Ein-/Ausgabe aus Sicht der CPU (1)

Datentransport aus der CPU heraus oder in die CPU hinein

- Adress- und Datenbus der CPU plus Kontrollleitungen
- In der Regel Datentransport zum oder vom Speicher

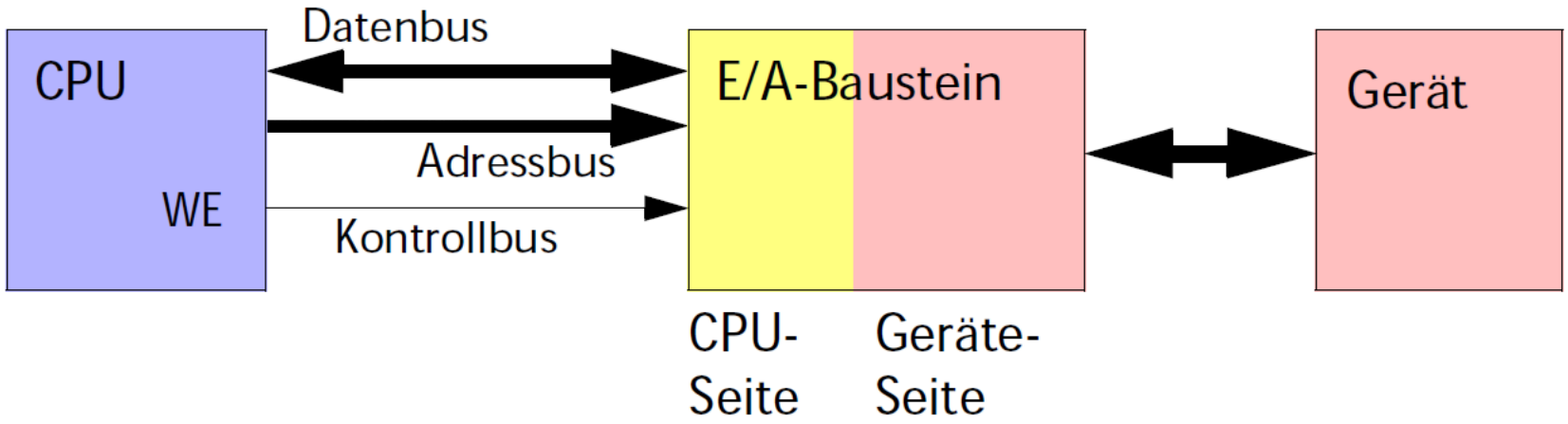


  - Vereinfachte Darstellung
- In der Praxis vor dem Speicher ein Chipsatz zur Speicheransteuerung
  - Generierung der RAS- und CAS-Signale
  - Multiplexen der Adressleitungen zum Speicherchip
  - Chipsatz sorgt für überlappende Speicherzugriffe
- In der Praxis viele Kontrollleitungen

Ein-/Ausgabe aus Sicht der CPU (2)

Datentransporte außerhalb des Hauptspeichers

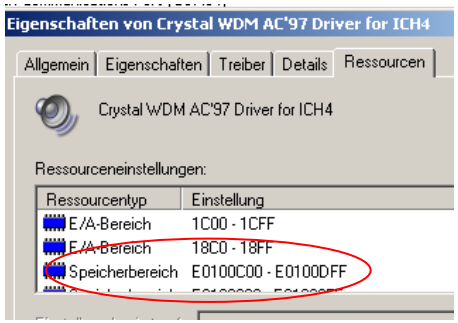
- Coprozessor-Interaktion
  - Datentransport vom oder zum Co-Prozessor
- Datentransport zu Ein-/Ausgabebausteinen
  - Externe Bausteine am CPU-Bus
  - Bieten spezielle Speicherstellen an (Register zur Interaktion)

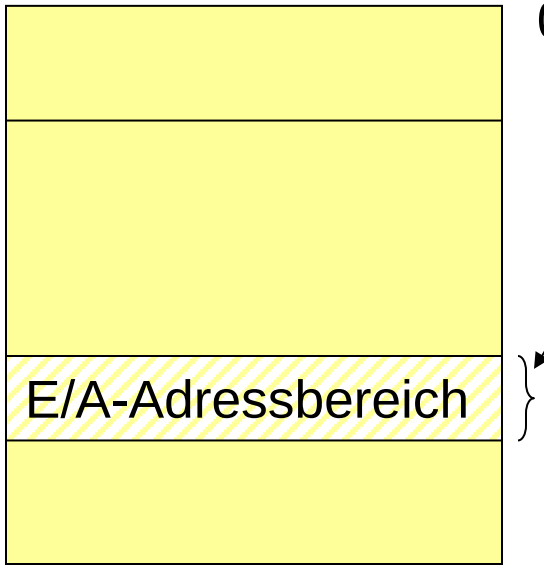


Ein-/Ausgabe aus Sicht der CPU (3)

Speicherbezogene Adressierung, *Memory-Mapped* I/O

- Einblenden des Ein-/Ausgabebausteins in den physikalischen Adressraum
  - Für Prozessoren ohne separate Speicher- und E/A-Schnittstelle
  - Spezielle Adressdecodierung vor dem I/O-Baustein durch Busadapter
- Zugriff mit konventionellen *load*- und *store*-Befehlen





load = Eingabe;
store = Ausgabe

Muss von *Caching* ausgenommen werden.

Beispiel: MIPS-Maschine

Ein-/Ausgabe aus Sicht der CPU (4)

Eigenschaften speicherbezogener Adressierung

- Vorteile:
  - Großer I/O-Adressraum aufwandsarm
  - Wenig Festlegung beim Prozessorentwurf
  - Keine separaten I/O-Befehle notwendig, kleiner Befehlssatz

- Nachteile:
  - Besonderes I/O-Timing nur über Busadapter möglich
  - Zugriffsschutz auf I/O-Geräte nur über Speicherverwaltung

Ein-/Ausgabe aus Sicht der CPU (5)

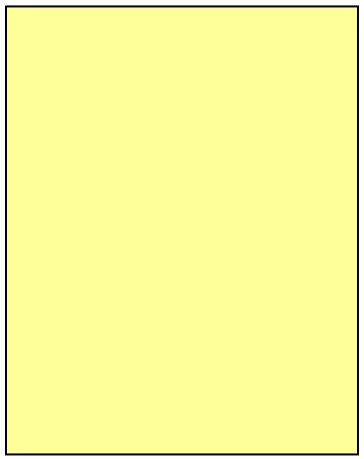
Spezielle Ein-/Ausgabebefehle

- Prozessor hat spezielle Befehle für Ein-/Ausgabe
  - Meist privilegierte Befehle
  - Angabe einer Adresse im I/O-Adressraum
- CPU-Bus hat zusätzliche Adressleitungen zur Selektion des I/O-Adressraums

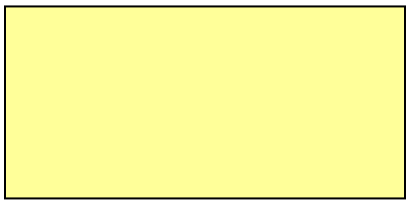
Ein-/Ausgabe aus Sicht der CPU (6)

Spezielle Ein-/Ausgabebefehle (fortges.)

Speicheradressraum



I/O-Adressraum

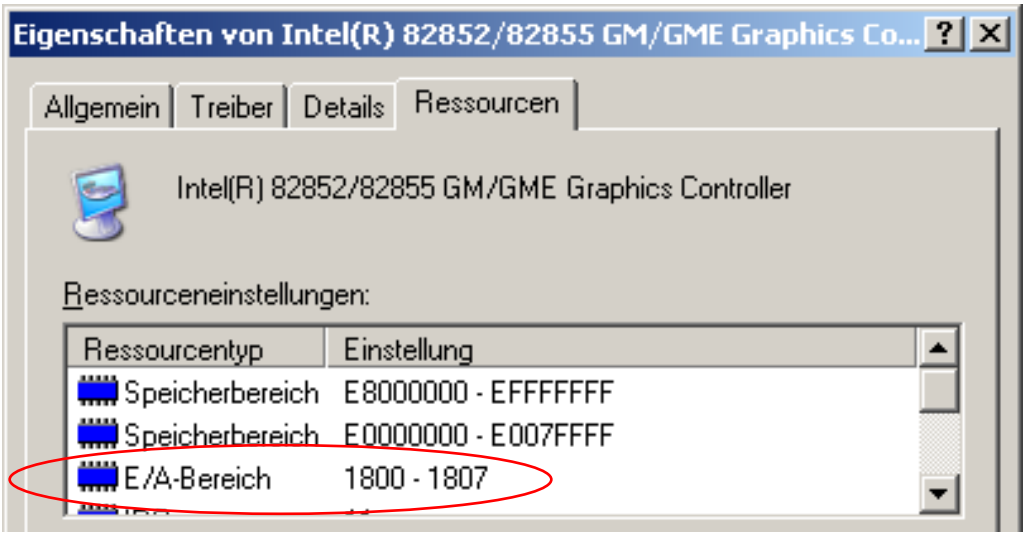


in = Eingabe;
out = Ausgabe

Umkehrung von Vor- und Nachteilen der speicherbezogenen Adressierung.

Prozessoren mit I/O-Adressen können auch speicherbezogene Adressierung nutzen.

Beispiel: Intel x86/Pentium-Familie.



Ein-/Ausgabe aus Sicht der CPU (7)

CPU ist treibende Kraft

– Nur wenn CPU agiert, wird Ein-/Ausgabe betrieben

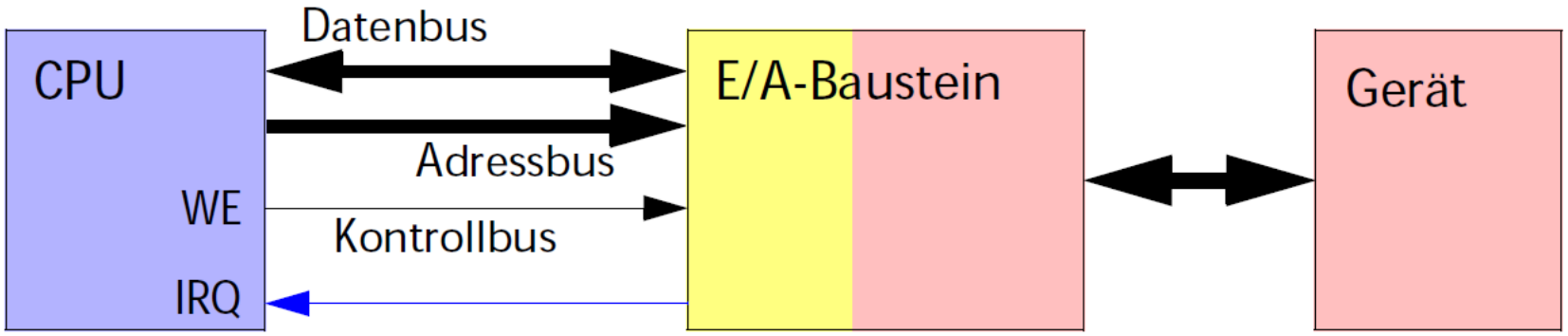
Aktion vom Gerät aus?

☞ **Unterbrechungen *(Interrupts)***

– Rückkanal vom Gerät über den I/O-Baustein zur CPU

– Anforderung einer Behandlung

  – z.B. Taste wurde gedrückt



Prinzipien der Datenübergabe

Annahme

- Ein Sender, ein Empfänger
- Leitungen zur Daten- und Adressübertragung und für Kontrollsignale

Übertragungsprotokolle

- Synchrone Busse
  - Keine Rückmeldung bei der Übertragung
  - Unidirektionales *Timing*
- Asynchrone Busse
  - Rückmeldung der Datenannahme
  - Bidirektionales *Timing*
- *Fully-Interlocked Handshaking* (vollständig verschränkte Signalisierung)
  - Spezialfall der asynchronen Datenübertragung

Synchrone und asynchrone Busse

Klassifizierung basiert auf Unterscheidung zwischen unidirektionalem *Timing* (bei synchronen Bussen) und bidirektionalem *Timing* (bei asynchronen Bussen).

Unidirektionales *Timing*

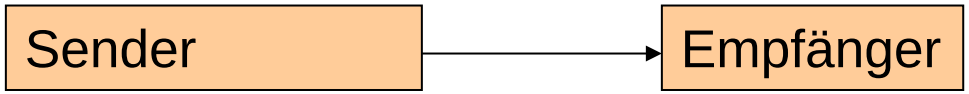
– Kommunikationspartner verlassen sich darauf, dass Partner innerhalb festgelegter Zeit reagieren.

Bidirektionales *Timing*

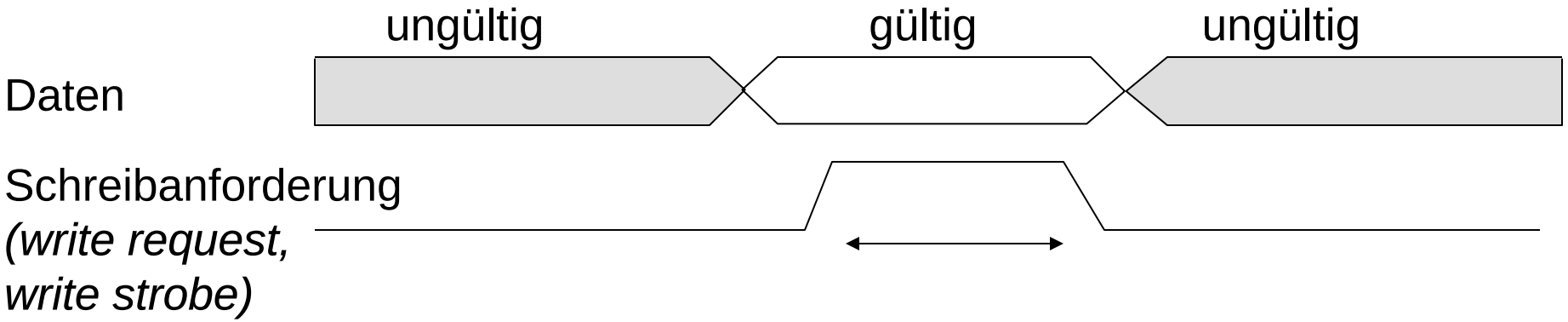
– Kommunikationspartner bestätigen per Kontrollsignal (senden ein *acknowledgment*), dass sie in der erwarteten Weise reagiert haben.

Schreiben bei synchronem Bus

Prinzip



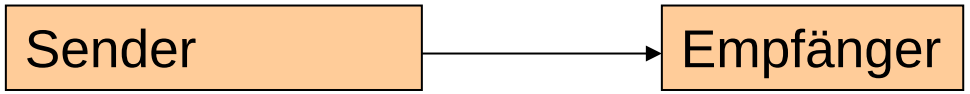
Zeitdiagramm der Übertragung



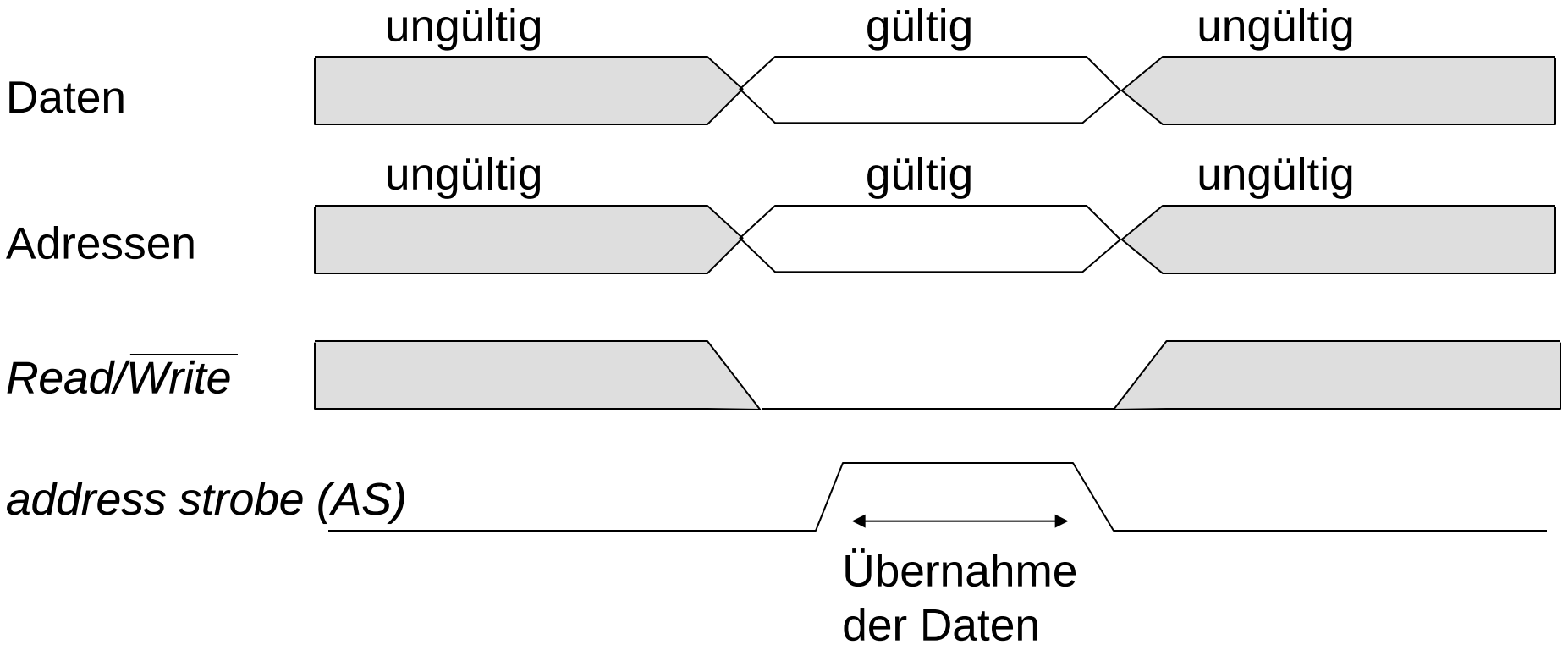
- Solange *write request* = 1, kann Empfänger Daten übernehmen
- Datenübernahme innerhalb maximaler Zeit nach Flanke auf *write request*
- Sender und Empfänger müssen Zeitdauer „vereinbart“ haben, Sender verlässt sich darauf, dass der Empfänger die Daten annimmt
- Alle Signale laufen vom Sender zum Empfänger ☞ schnell.

Schreiben bei synchronem Bus – mit Adressleitungen

Prinzip

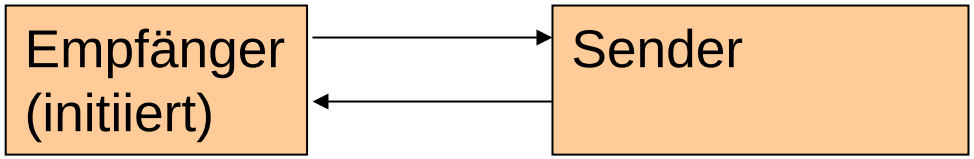


Zeitdiagramm der Übertragung

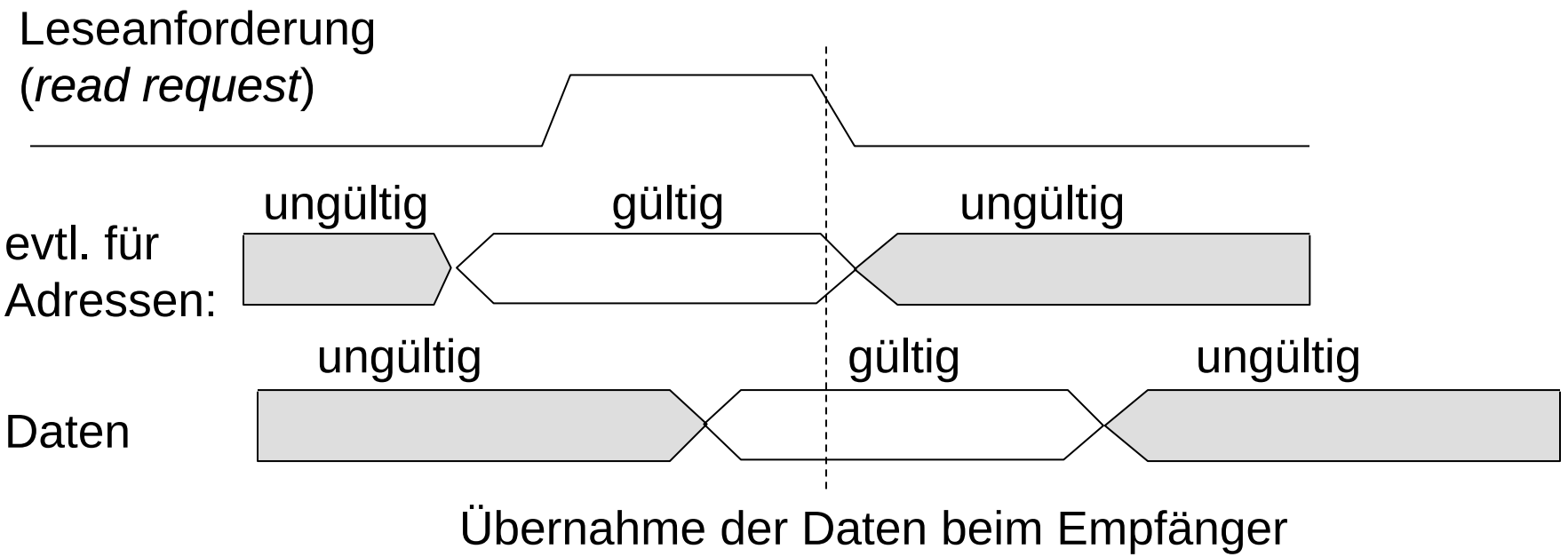


Lesen bei synchronem Bus

Prinzip



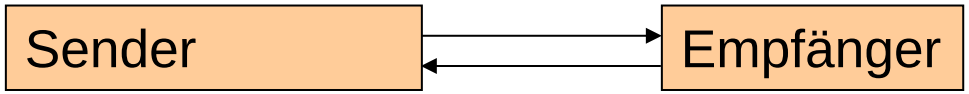
Zeitdiagramm der Übertragung



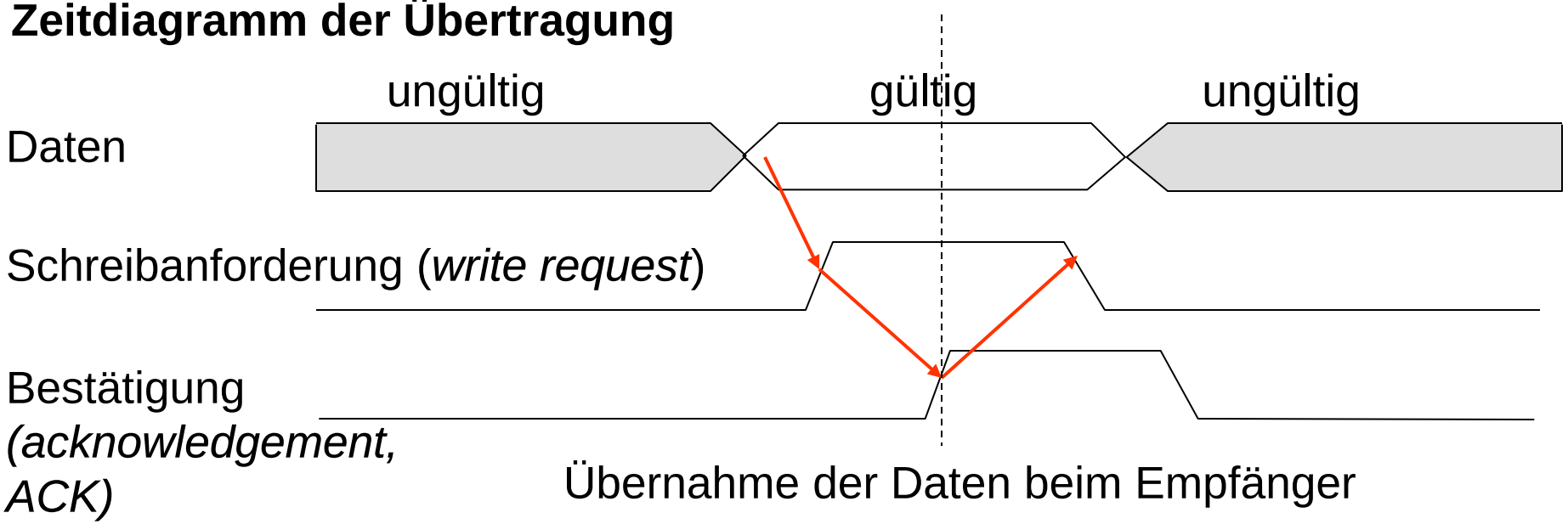
– Empfänger und Sender verlassen sich auf verabredete Zeiten.
  2 Signallaufzeiten

Schreiben bei asynchronem Bus

Anordnung



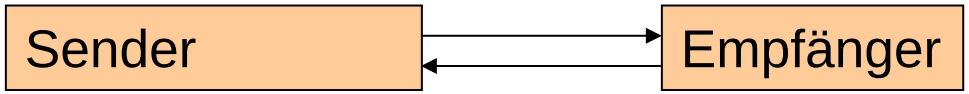
Zeitdiagramm der Übertragung



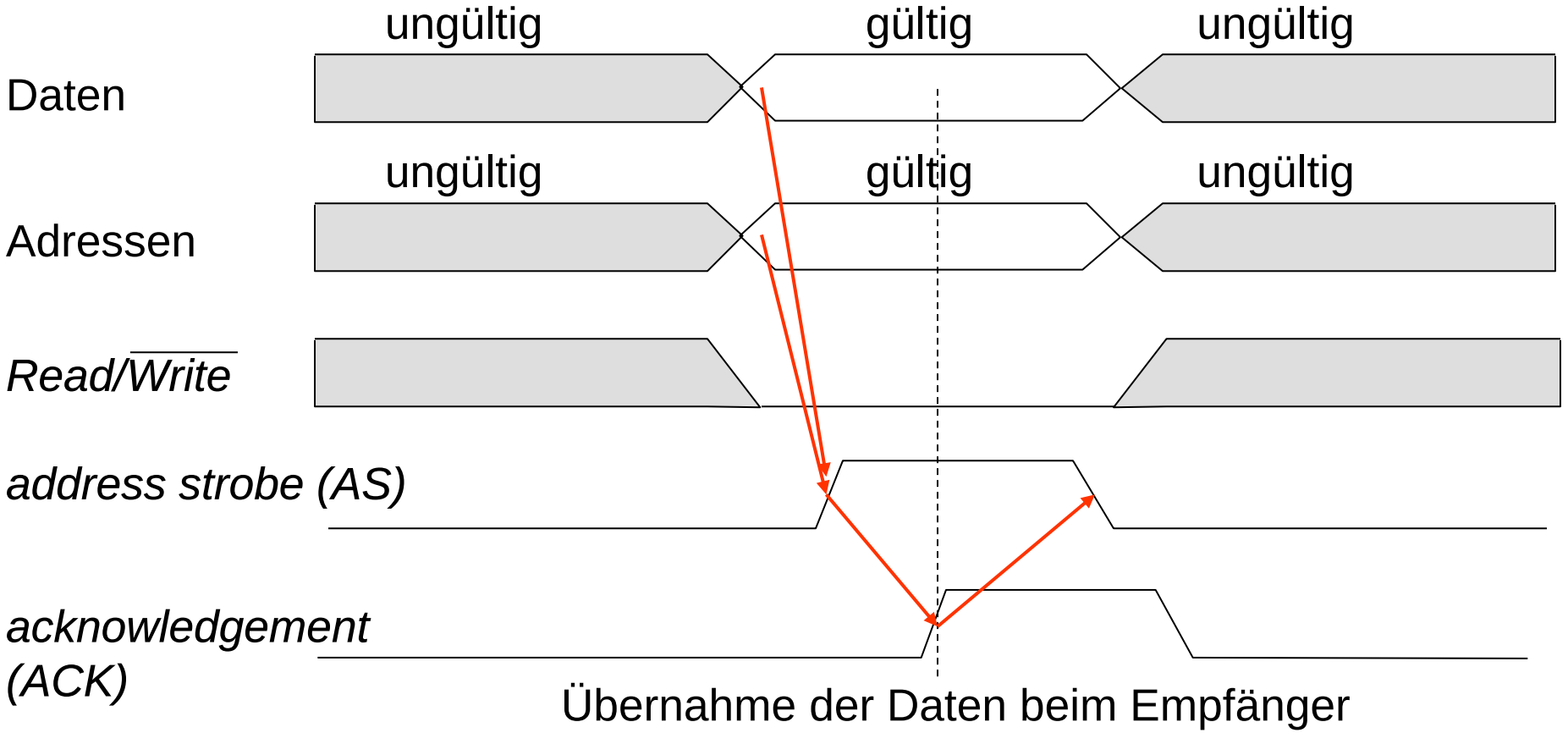
- Der Sender muss Daten gültig halten, bis Bestätigung eintrifft
- 2 Signallaufzeiten ☞ langsam

Schreiben bei asynchronem Bus – mit Adressleitungen

Prinzip

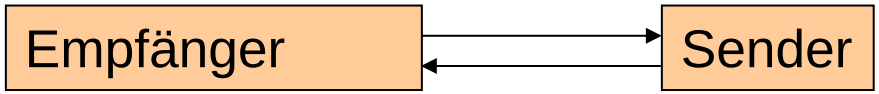


Zeitdiagramm der Übertragung

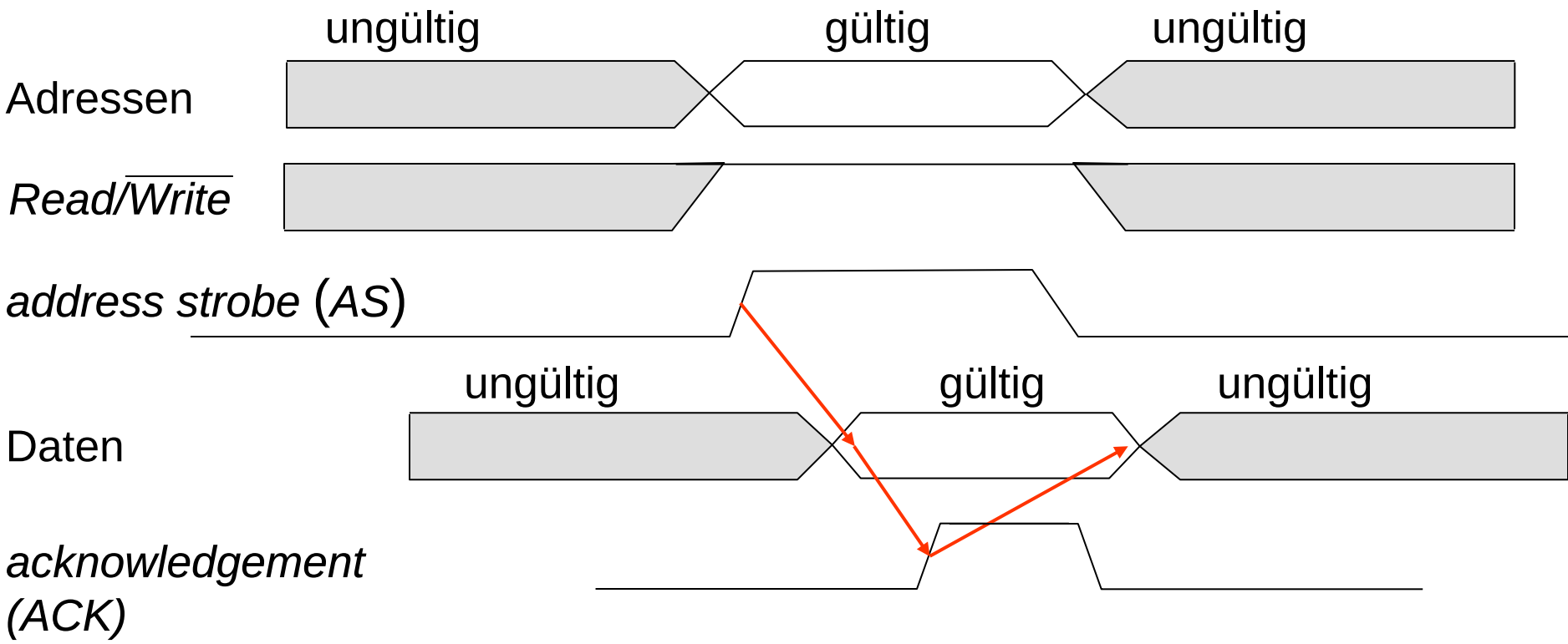


Lesen bei asynchronem Bus – mit Adressleitungen

Prinzip



Zeitdiagramm der Übertragung

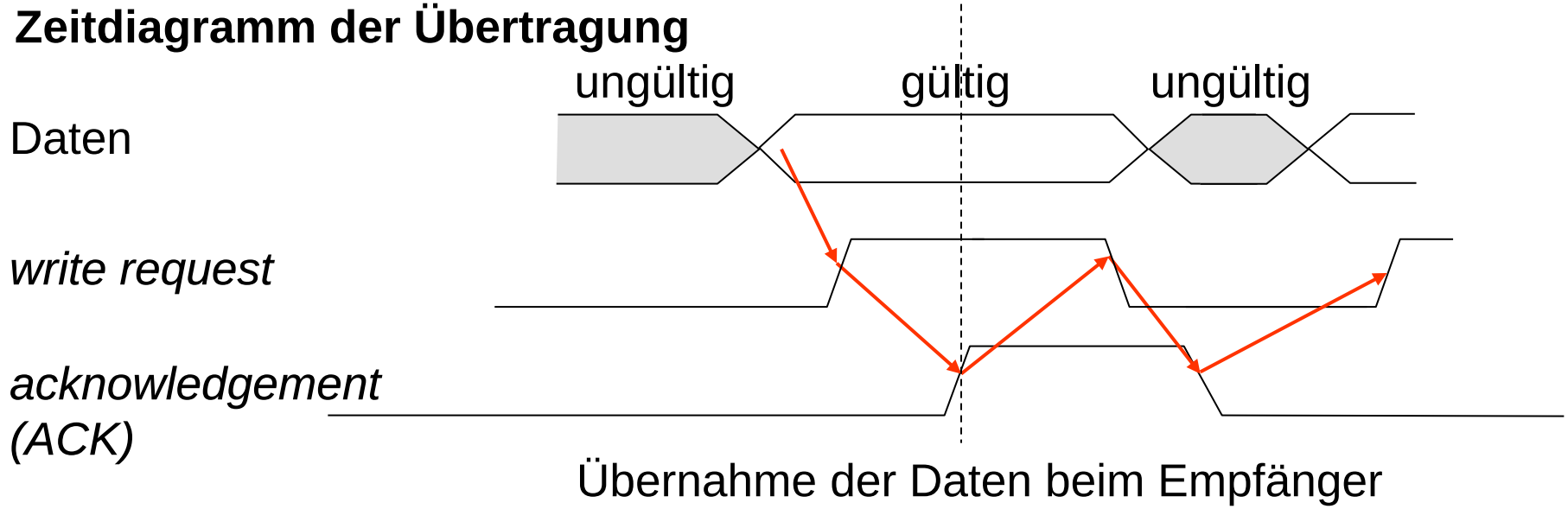


– Sender muss Daten gültig halten, bis ACK eintrifft. Mehrere Laufzeiten

Schreiben bei *Fully-Interlocked Handshaking*

Prinzip wie bei asynchronem Bus

Zeitdiagramm der Übertragung



- Ablauf wie bei asynchronem Bus
- Zusätzlich: ACK wird erst auf 0 gesetzt, wenn *write request* 0 ist
- Neue Daten werden erst ausgegeben, wenn ACK = 0
- Empfänger kann nach Datenempfang Sender aufhalten, 4 Buslaufzeiten

Vergleich der Methoden

	unidirektional	**bidirektional**
Vorteile	einfach; bei konstanten Antwortzeiten schnell	passt sich unterschiedlichen Geschwindigkeiten an; hoher Grad an Flusskontrolle möglich
Nachteile	Kommunikationspartner muss in bestimmter Zeit antworten	komplexer; Zeitüberwachung notwendig; evtl. langsam
Einsatz-gebiet	synchrone Busse, Speicherbusse	asynchrone Busse, E/A- und Peripheriebusse

Synchronisation zwischen Prozessor und Geräten

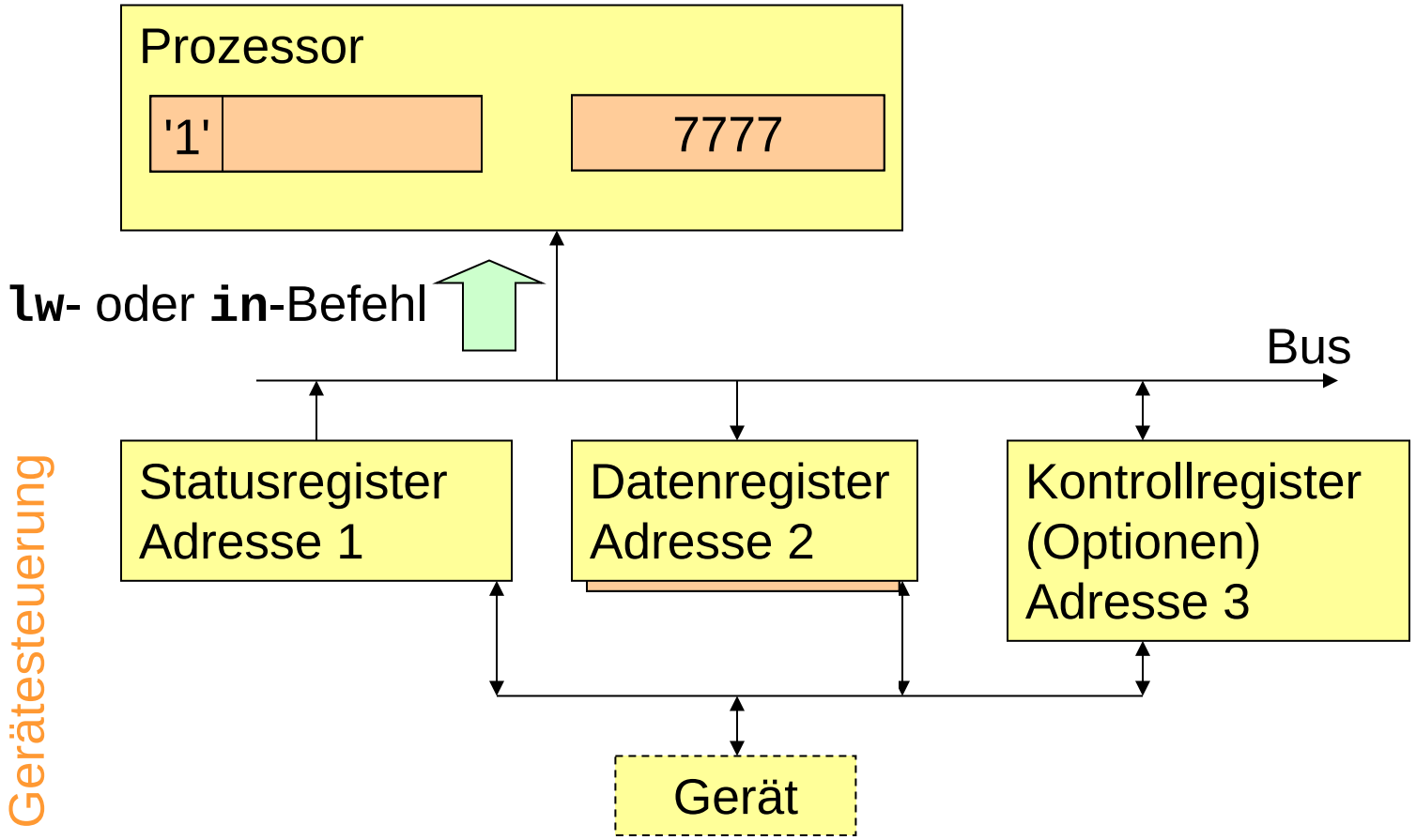
Wie stimmt man die Geschwindigkeit der I/O-Geräte mit der des Prozessors ab?

- Status-Methode *(busy waiting)*
- *Polling*
- Unterbrechungen *(Interrupts)*
- *Direct memory access (DMA)*

Status-Methode *(busy waiting)* – Hardwarestruktur

Das Gerät bzw. die Steuerung stellt Bereit-Bit zur Verfügung

Beispiel:

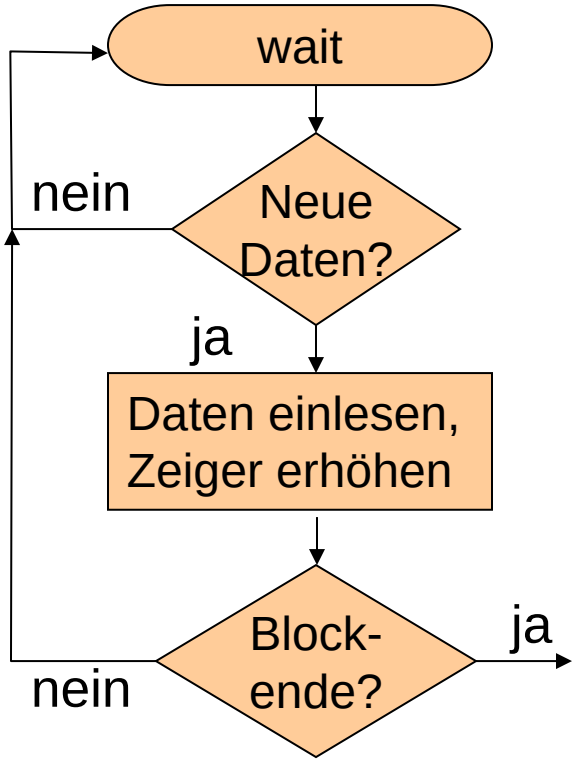


Status-Methode *(busy waiting)* – Warteschleife

Das „Bereit“-Bit wird in einer Warteschleife abgeprüft, bis nach Anzeige eines entsprechenden Werts die Ein- bzw. Ausgabe erfolgen kann.

Prinzip:

```
REPEAT
  REPEAT
    lese Statuswort des Gerätes;
  UNTIL Bereit-Bit im Statuswort ist gesetzt;
  lese Datenwort;
  erhöhe Blockzeiger;
UNTIL Ende des Blocks ist erreicht;
```



Prozessor bleibt während des Wartens beschäftigt *(busy)*.

Eigenschaften von *Busy Waiting*

Nachteile

– Keine Möglichkeit der verzahnten Bearbeitung anderer Aufgaben
– Geringe Übertragungsgeschwindigkeit

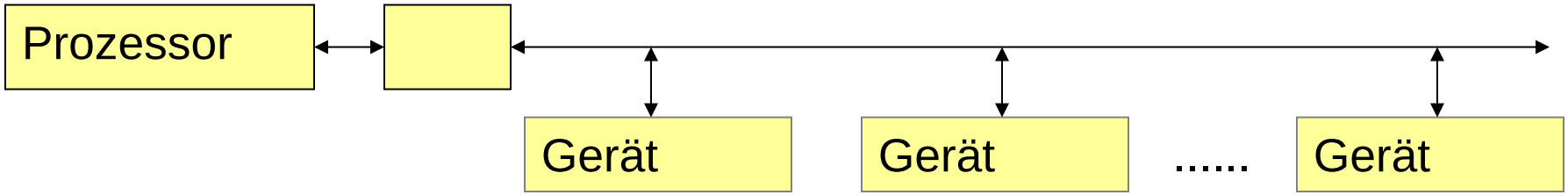
Anwendung

– Wenn keine anderen Aufgaben vorliegen
– Wenn das Gerät so schnell ist, dass die Schleife selten durchlaufen wird und für die Umschaltung auf andere Aufgaben keine Zeit bleibt
– Wenn das Gerät zu keiner anderen Methode fähig ist

Polling

Polling = gelegentliches Prüfen des Bereit-Bits mit verzahnter Bearbeitung anderer Aufgaben

Anwendungsbeispiel: Erfassung von Übertragungswünschen einer großen Anzahl von Geräten:



Erlaubt eine gerechte Bedienung vieler Geräte (z.B. Terminals oder USB-Geräte). Vermeidet Überlastungen durch viele Übertragungswünsche. Vorteilhaft bei *denial-of-service* Attacken, unter Umständen große Reaktionszeit.

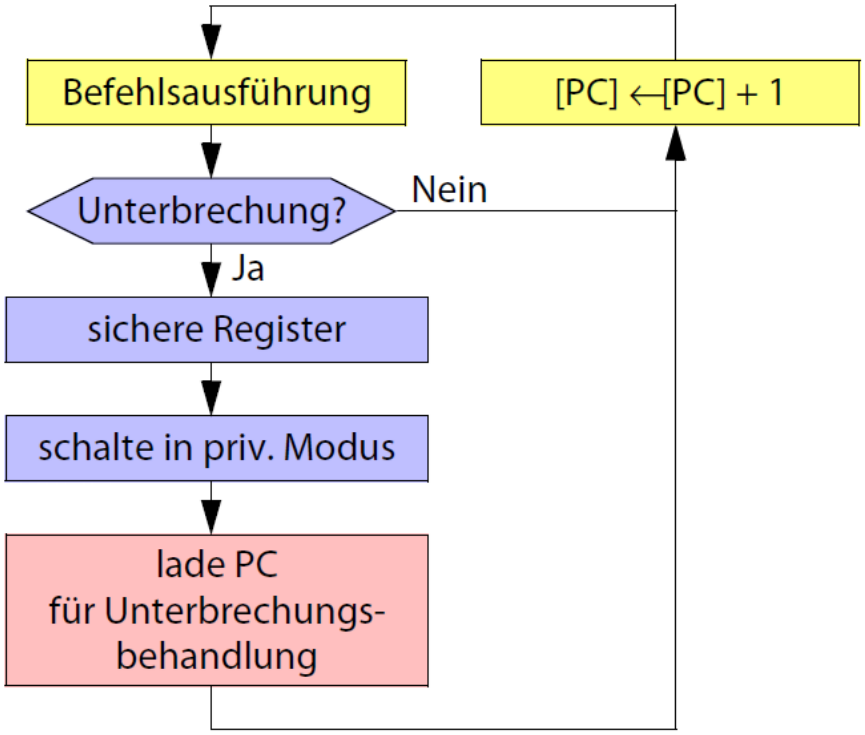
Unterbrechungen / *Interrupts* (1)

Sinn und Zweck von Interrupts

- Geräteabfrage bzw. *Polling* vermeiden
- Behandlung externer Ereignisse
  - Datenaustausch mit externen und internen Geräten (Serielle/parallele Schnittstelle, Festplattencontroller, Netzwerkkarten, Tastatur, ...)
- Behandlung interner Fehlersituationen
  - Allgemeine Speicherschutzverletzung
  - Ungültige Maschineninstruktion
  - Ausgelagerte Speicherseite
  - Arithmetische Fehlersituation
  - ...

Unterbrechungen / *Interrupts* (2)

Schematische Befehlsabwicklung



Unterbrechungen / *Interrupts* (3)

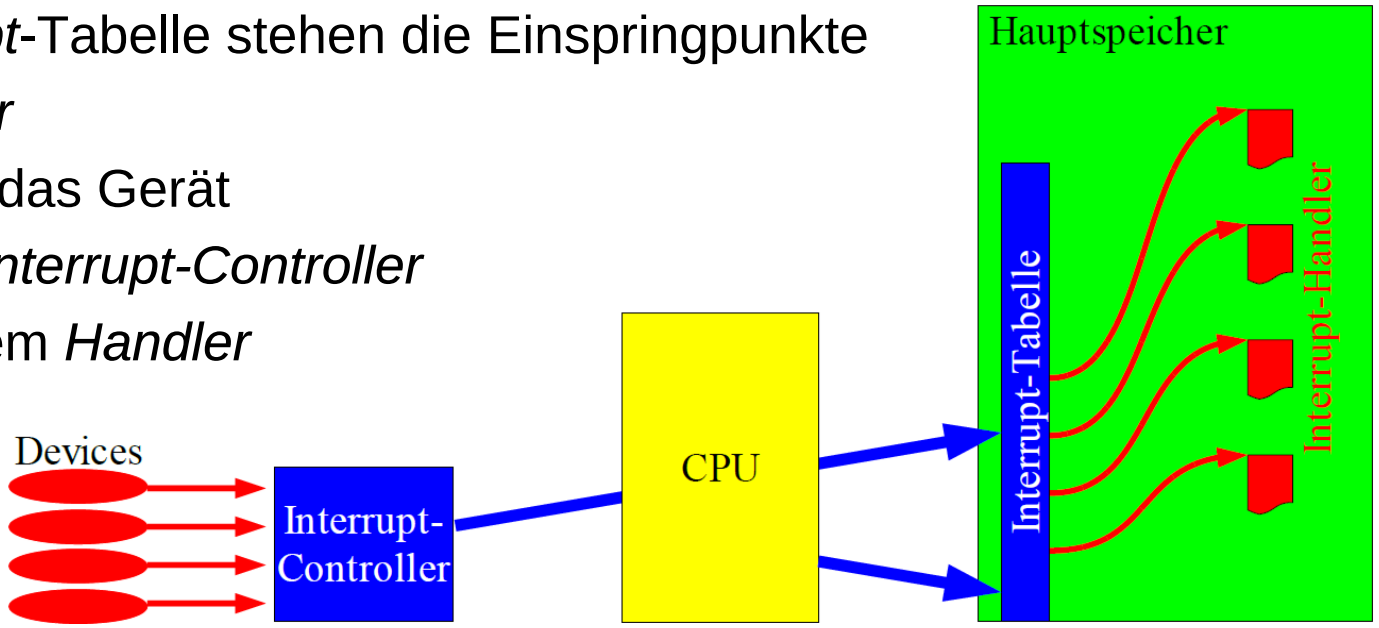
Ablauf der Unterbrechung

- Unterbrochene Befehlsfolge bleibt in der Regel unberührt
  - Unterbrechung ist transparent
- Verschachtelte Unterbrechungen möglich
  - Unterbrechung der Unterbrechungsbehandlung
  - Koordinierte Handhabung der gesicherten Register

Unterbrechungen / *Interrupts* (4)

Ablauf eines HW-Interrupts

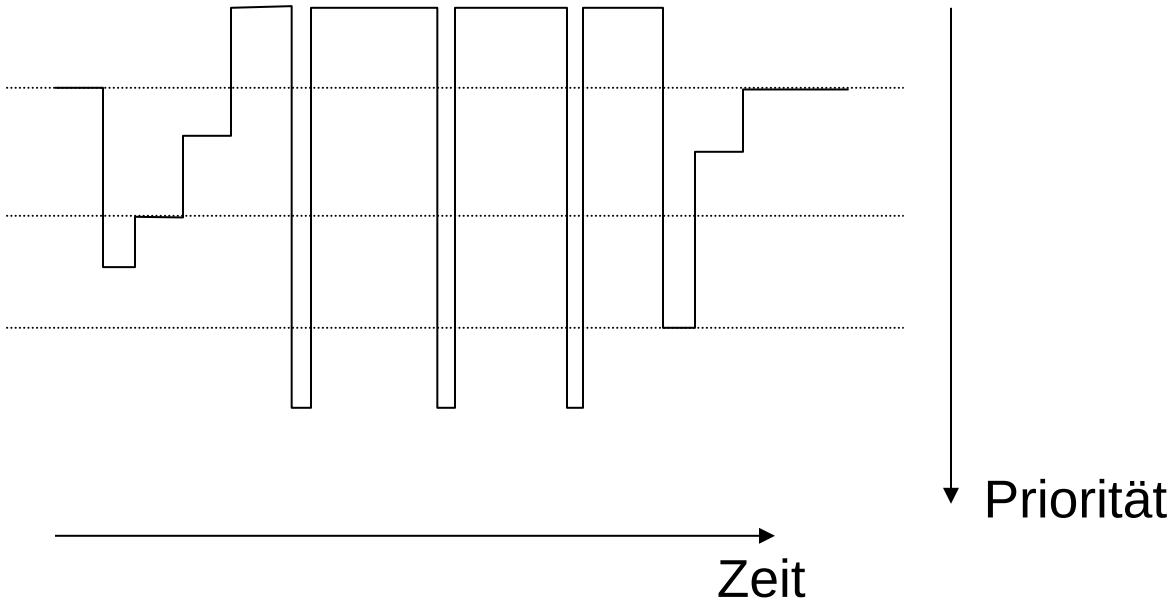
- Gerät meldet IRQ *(Interrupt Request)* an *Interrupt-Controller*
- *Interrupt-Controller* fordert Behandlung des IRQ bei der CPU
- CPU bestätigt die Anforderung und verlangt den *Interrupt*-Vektor
- *Controller* sendet *Interrupt*-Vektor an CPU
- CPU wählt einen *Interrupt-Handler* aus der *Interrupt*-Tabelle
  - ☞ In der *Interrupt*-Tabelle stehen die Einspringpunkte für die *Handler*
- *Handler* bedient das Gerät
- Bestätigung an *Interrupt-Controller*
- Rückkehr aus dem *Handler*



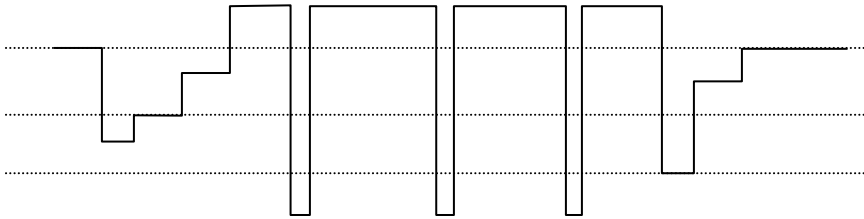
Unterbrechungen / *Interrupts* (5)

Gerätesteuerung unterbricht den Prozessor, falls Datentransport(e) erforderlich werden oder falls Fehler auftreten.

Neuer Prozess
Prozess A
Dispatcher
Betriebssystem
BS-Aufruf
E/A-Ende-*Interrupt*
E/A-Daten-*Interrupt*
....
Stromausfall



Unterbrechungen / *Interrupts* (6)



Ablauf

- Prozess A überträgt I/O-Auftrag an das Betriebssystem
- *Dispatcher* schaltet auf neuen Prozess um
- Gerät sendet *Interrupt*
- Falls *Interrupt* nicht gesperrt ist und die Priorität ausreichend hoch ist und der laufende Maschinenbefehl abgeschlossen ist:
- Sicherung des Zustandes
- Feststellen der *Interrupt*-Ursache
- Datentransport: Ein-/Ausgabe, Lesen/Schreiben Speicher, Pufferzeiger erhöhen, Test auf Blockende
- Restaurieren des Kontexts, *return from interrupt*
- Prozess fährt in Bearbeitung fort
- Nach einer Reihe von Datenübertragungs-*Interrupts* erfolgt ein Blockende-*Interrupt*, welcher den *Dispatcher* startet

Unterbrechungen / *Interrupts* (7)

Eigenschaften

– Wegen des *Overheads* nicht für die schnelle Datenübertragung geeignet

Zu programmieren

– Starten der Geräte vor der Übertragung
– Übertragung per *Interrupt-Handler* oder *Interrupt Service Routine (ISR)*

Anwendungen

– Häufig zur Übertragung zum Drucker oder von Tastatur eingesetzt
– Für Rechner ohne *direct memory access* ist diese Methode erforderlich
– Zum Abfangen von Laufzeitfehlern, die durch Software erzeugt werden (z.B. Division durch 0, Verletzung von Segmentgrenzen, ...)
– Systemaufrufe *(Traps)*: Spezialbefehl, der SW-*Interrupt* auslöst; Betriebssystem führt *Trap-Handler* aus, dem Parameter übergeben werden können ☞ Implementierung der Betriebssystem-Schnittstelle

Direct Memory Access, DMA (1)

Bislang: Datentransport zwischen Speicher und I/O-Bausteinen

- Nur der Prozessor hat die volle Kontrolle über den Bus (legt Adressen an, erzeugt Kontrollsignale, ...)
- Andere Einheiten dürfen nur auf Anforderung des Prozessors tätig werden
- Prozessor liest aus Speicher und übergibt Daten an I/O-Baustein oder liest Daten vom I/O-Baustein und schreibt in Speicher
  - Üblicherweise relativ viele Daten zu übertragen (z.B. Plattentransfer)
  - Ständige Wartezeiten durch relativ langsame I/O-Geräte

☞ ***Nur der Prozessor hat bislang Funktion des Bus-Masters.***

Direct Memory Access, DMA (2)

Beim DMA werden die Datentransporte *direkt* zwischen Gerätesteuerung und Speicher durchgeführt, *ohne* Beteiligung des Prozessors.

- Spezielle Hardware tritt als Aktor auf dem Bus auf
  - Liest Daten und übergibt sie an I/O-Baustein, oder umgekehrt
  - CPU ist währenddessen frei (lediglich Bus teilw. durch DMA belegt)
- Programmierung des DMA-Systems
  - Länge, Speicheradresse, I/O-Adresse von Ein-/Ausgaberegister
  - Unterbrechung zur Signalisierung des Transferendes
  - Unterbrechung zur Signalisierung von Fehlern
  - ☞ Aus Sicht des Programmierers ist DMA die einfachste Technik.

☞ ***DMA macht es erforderlich, dass Gerätesteuerungen Bus-Master werden können.***

Direct Memory Access, DMA (3)

Arbeitsmodi

- *Burst*-Modus: DMA-Baustein übernimmt Systembus für die Dauer des vollständigen Transfers
  - I/O-Baustein benötigt internen Pufferspeicher
  - Vorteil: hohe Transferrate möglich
  - Nachteil: CPU für lange Zeit am Speicher-/Buszugriff gehindert
- *Cycle Stealing:* Mischen von DMA- und CPU-Buszyklen
  - Fester Anteil an Buszyklen (z.B. jeder zweite) wird vom DMA-Baustein der CPU „gestohlen“
  - DMA kann zusätzlich alle von der CPU nicht benötigten Buszyklen nutzen (z.B. nach *Execute*-Phase von Register/Register-Befehlen)
  - Vorteil: CPU nur geringfügig behindert
  - Nachteil: geringere Transferrate

Zusammenfassung

Prinzipien der Datenübergabe

- Wie adressiert man I/O-Geräte und deren Steuerungen
  Memory-Mapped I/O,
  Eigene I/O-Adressen
- Busse mit oder ohne Bestätigung?
  Synchrone Busse / Unidirektionales *Timing*,
  Asynchrone Busse / Bidirektionales *Timing*,
  Fully-interlocked Handshaking
- Wie synchronisiert man Geräte und Prozessoren?
  Busy waiting,
  Polling,
  Interrupts,
  DMA

Roter Faden

8. Ein-/Ausgabe

- Einleitung
  - Datentransport von/zu Speicher oder E/A-Bausteinen
  - *Memory-mapped* I/O
  - Spezielle Ein-/Ausgabebefehle
  - *Interrupts*
- Prinzipen der Datenübergabe
  - Synchrone Busse / unidirektionales *Timing*
  - Asynchrone Busse / bidirektionales *Timing*
  - *Fully-Interlocked Handshaking*
  - Synchronisation Prozessor ↔ Geräten: *Busy Waiting*, *Polling*, *Interrupts*, DMA
- *Point-to-Point* Verbindungen
- Busse
- Fehlererkennung mit CRC-Zeichen

Point-to-Point Verbindungen

***Point-to-Point:* Ein Sender, ein Empfänger**

Typische *Point-to-Point* Verbindungen

- Serielle Schnittstelle (RS-232)
- Parallele Schnittstelle (IEEE 1284)

Fallstudie: RS-232 Schnittstelle (1)

Asynchrone serielle Schnittstellen sind

– *seriell* (d.h. die Bits werden nacheinander über *eine* Leitung übertragen), und
– *asynchron* (das bedeutet hier: der zeitliche Abstand zwischen aufeinanderfolgenden *Zeichen* ist variabel)

Nacheinander werden folgende Werte übertragen

– das *Startbit:* dies ist immer eine ‚0'. Sie dient zur Synchronisierung zwischen Sender und Empfänger
– 7 oder 8 D*atenbits*, *LSB (least significant bit) travels first*
– *Paritätsbit:* gerade, ungerade oder keine Parität
– 1-2 *Stoppbits:* Mindestlänge der Pause zwischen zwei Zeichen

Fallstudie: RS-232 Schnittstelle (2)

Paritätsbit

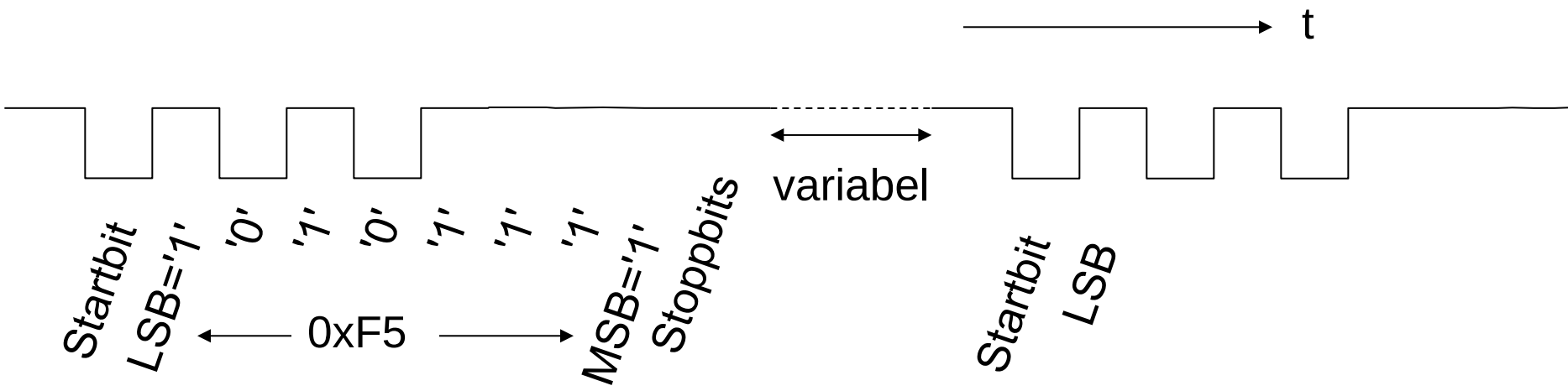
- Zusätzlich als Datenbit versandtes Bit
  - *Even Parity:* Summe der 1-Datenbits gerade
  - *Odd Parity:* Summer der 1-Datenbits ungerade

Typische Parametrisierung

- 8N1 bedeutet: 8 Datenbits, *None Parity*, 1 Stoppbit

Fallstudie: RS-232 Schnittstelle (3)

Beispiel (8 Bit, ohne *parity*)



Fallstudie: RS-232 Schnittstelle (4)

Eigenschaften

- Die Zeit für jeweils ein Bit richtet sich nach der Datenübertragungsrate, gemessen in Bit/s.
- Symboltakt sind bei Sender und Empfänger gleich einzustellen (Baud-Rate)
- Symbol hier gleich 1 Bit (Bitrate = Baud-Rate)
- Typische Bitraten: 300, 1.200, 2.400, …, 19.200, 38.400, 57.600
- Bei 57.600 Bit/s ist ein Bit etwa 17 µs lang
- Auf kurzen Entfernungen (im Raum) sind Übertragungsraten bis knapp oberhalb von 100 kbit/s erreichbar.
- Keine Taktleitung: asynchrone Übertragung

Fallstudie: RS-232 Schnittstelle (5)

Unterscheidung bezüglich Übertragungsrichtungen

- *Voll-Duplex-Betrieb:* Übertragung in beide Richtungen gleichzeitig
- *Halb-Duplex-Betrieb:* Übertragung in beide Richtungen, aber nur in eine zur Zeit
- *Simplex-Betrieb:* Übertragung in nur eine Richtung

Fallstudie: RS-232 Schnittstelle (6)

V.24-Standard = RS-232 Standard

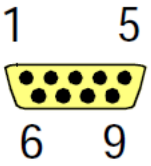
Normierung der asynchronen Übertragung zwischen

- einem Daten-Endgerät (deutsch: DEE, englisch *Data Terminal Equipment DTE*), d.h. Rechner oder Terminal
- und einer Datenübertragungseinrichtung DÜE (englisch *Data Communication Equipment DCE*), z.B. einem Modem (Modulator/Demodulator)
- oder zwischen DEE und DEE.

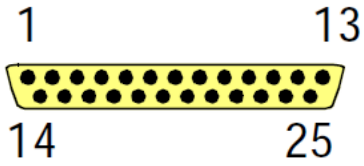
- 9- oder 25-poliger Stecker (9-polig eher üblich)
  - Im Minimalfall mit drei Leitungen betreibbar

Fallstudie: RS-232 Schnittstelle (7)

Pinbelegung der Steckverbindungen (Steckeraufsicht, *Male*)



1	DCD
2	RxD
3	TxD
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI



1	GND	14
2	TxD	15
3	RxD	16
4	RTS	17
5	CTS	18
6	DSR	19
7	GND	20 DTR
8	DCD	21
9		22 RI
10		23
11		24
12		25
13		

GND *(Ground)*: Masse
TxD *(Transmit Data)*: Sendedaten
RxD *(Receive Data)*: Empfangsdaten
RTS: DEE möchte Daten senden
CTS: Partner kann Daten empfangen
DSR: Partner ist betriebsbereit
DTR: DEE ist betriebsbereit
DCD: Partner empfängt analoges Trägersignal
RI *(Ring Indicator)*: Rufsignal

25-poliger Verbinder: weitere Pins belegt, aber hier nicht dargestellt!

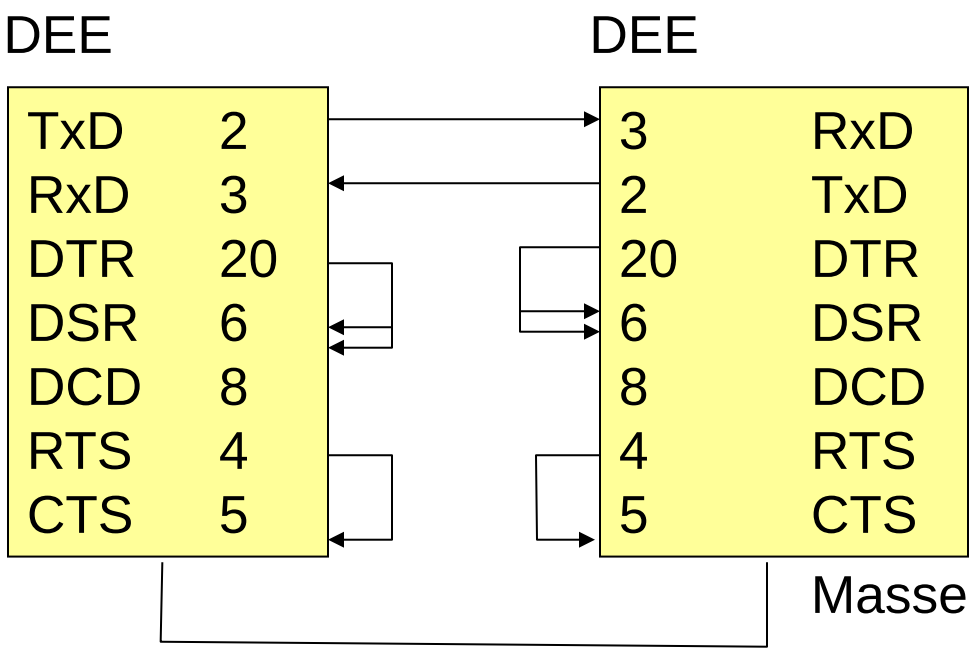
Fallstudie: RS-232 Schnittstelle (8)

Logik aller Leitungen

- Wert „Logisch 1“: Spannung zur Masse -12 V
- Wert „Logisch 0“: Spannung zur Masse +12 V
- Werterkennung auf Empfängerseite von +/-3 V bis +/-15 V
- Ermöglicht lange Leitungen und guten Störabstand

Fallstudie: RS-232 Schnittstelle (9)

RS-232 über drei Leitungen



Spezielle Kabel (Pins 2/3 vertauscht!)

Interpretation:

Wenn die DEE eingeschaltet ist, nimmt sie an, der Partner ist es auch.

Bei Sendewunsch über RTS erfolgt über CTS sofort die Bestätigung.

☞ keine Prüfung auf Überlastung.

Zur Absicherung gegen Überlastung Benutzung einer Software-Flusskontrolle (*handshaking*, z.B. mit Ctrl-S und Ctrl-Q).

Fallstudie: RS-232 Schnittstelle (10)

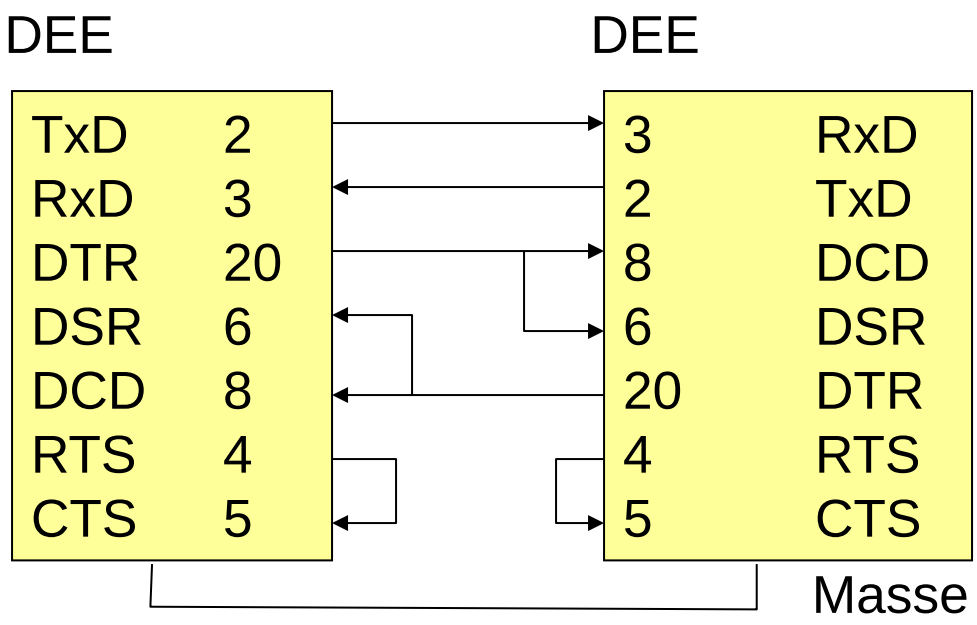
Flusskontrolle

- Software-Flusskontrolle
  - Senden des Steuerzeichens Ctrl-S (0x13) stoppt den Datentransport
  - Senden des Steuerzeichens Ctrl-Q (0x11) hebt den Stopp auf
  - Steuerzeichen dürfen nicht in Daten vorkommen
  - XON = *Transmission on*, XOFF = *Transmission off*
  - ☞ Software-Flusskontrolle = XON/XOFF Flusskontrolle

- Hardware-Flusskontrolle
  - z.B. über RTS- und CTS-Leitungen

Fallstudie: RS-232 Schnittstelle (11)

RS-232 mit Hardware-Flusskontrolle



Interpretation:

Senden setzt voraus, dass Partner über DTR seine Bereitschaft zum Empfang von Daten signalisiert.

CTS, RTS: Mitteilung eines Übertragungswunsches und der Empfangsbereitschaft; Pins z.T. offen oder kreuzweise verbunden.

Hier: sog. Nullmodemkabel, d.h. Pins 2 und 3 sind jeweils über kreuz verbunden.
Für Kommunikation DEE / DÜE: an Pins 2 und 3 durchverbundene Kabel („Verlängerungskabel“)

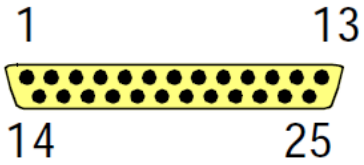
Fallstudie: Paralleler Port (1)

Die parallele PC-Schnittstelle

- Parallel Übertragung von 8 Bits
- Früher: Centronics-Schnittstelle (Druckerschnittstelle)
  - Nur Ausgabe von Daten
  - Transferrate 150 kByte/s
- Heute meist: IEEE 1284 Schnittstelle
  - *SPP – Standard Parallel Port:* unidirektionaler Betrieb, kompatibel mit Centronics
  - *EPP – Enhanced Parallel Port*, *ECP – Enhanced Capability Port:* Ein- und Ausgabe von Daten möglich
  - Transferrate bis 2 MByte/s
- Verbindungsstecker mit 25 oder 36 Pins

Fallstudie: Paralleler Port (2)

25-polige SPP-Pinbelegung (Steckeraufsicht, *Male*)



1 Strobe	14 Auto Feed
2 D1	15 Error
3 D2	16 Init
4 D3	17 Select In
5 D4	18 GND
6 D5	19 GND
7 D6	20 GND
8 D7	21 GND
9 D8	22 GND
10 Ack	23 GND
11 Busy	24 GND
12 PE	25 GND
13 Select	

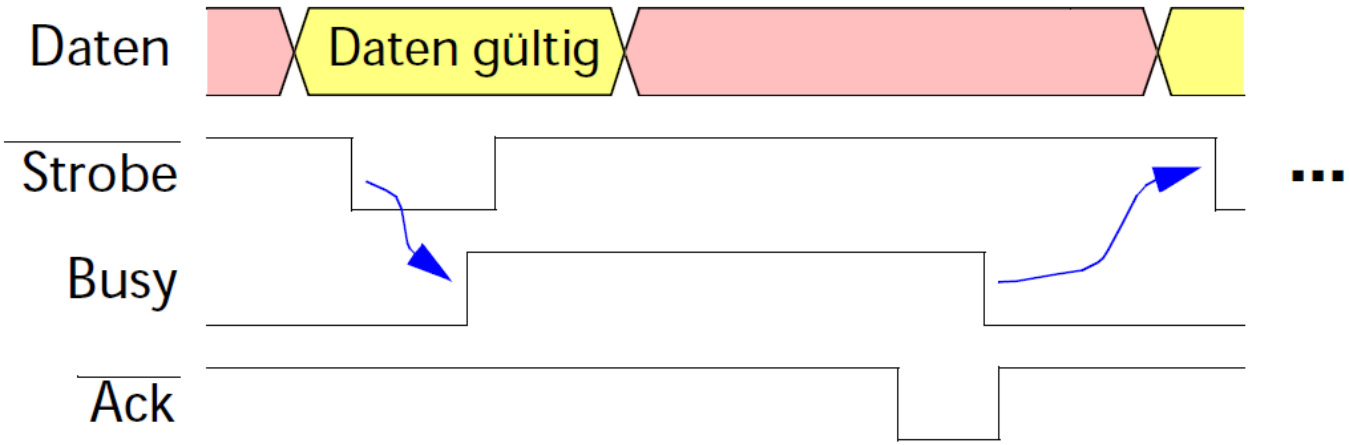
GND *(Ground)*: Masse
D1-8: Datenleitungen
Strobe: Daten gültig
Ack: Bestätigung des Empfangs
Busy: Endgerät beschäftigt
PE *(Paper End)*: Endgerät nicht bereit
Select: Zeigt Druckerstatus an (*on-* oder *offline*)
Auto Feed: Papiervorschub
Error: Fehler
Init: Rücksetzen
Select In: Teilt dem Drucker mit, dass er angesprochen ist

Fallstudie: Paralleler Port (3)

TTL-Logik

– +5 V (Logisch 1) und 0 V (Logisch 0) zur Masse

Datenübertragung



– Asynchroner Bus / bidirektionales *Timing*
– Ack dient als zusätzlicher Unterbrechungsauslöser
  – Bestätigt Empfang der Daten

Fallstudie: Paralleler Port (4)

Einsatz

- Drucker
- Billiger Geräteanschluss für
  - Scanner
  - Zip-Drives
  - …

Heute weitgehend Verdrängung durch USB

Roter Faden

8. Ein-/Ausgabe

- Einleitung
- Prinzipien der Datenübergabe
- *Point-to-Point* Verbindungen
  - Fallstudie: Serielle Schnittstelle (RS-232 / V.24)
  - Fallstudie: Paralleler Port (Centronics / IEEE 1284)
- Busse
- Fehlererkennung mit CRC-Zeichen

Busse (1)

Zugang für mehrere Einheiten an die gleichen Transport- und Kontrollleitungen

- Vorteil: Einsparung von Verbindungen und Leitungen im Gegensatz zu *Point-to-Point* Verbindungen zwischen mehreren Einheiten
- Beispiel: CPU-Bus zur Verbindung von CPU und Speicherbausteinen

Interne Busse

- Bussysteme zur Verbindung rechnerinterner Einheiten
  - z.B. Prozessoren, Speicher, I/O-Bausteine
  - „Verlängerung“ des CPU-Busses

Externe Busse

- Bussysteme zur Verbindung von externen Geräten
  - z.B. Festplattensysteme, Netzwerke

Busse (2)

Motivation für standardisierte Busse

- Unabhängige Geräte und Komponentenentwicklung
  - Hersteller- und Plattformunabhängigkeit
  - z.B. Steckkarten für PC (interner Bus)
  - z.B. externe Festplattensysteme (externer Bus)
- Erforderliche Standardisierung
  - Signale und deren Spannungen
  - Steckverbindungen und Signalbelegungen
  - Elektrische und zeitliche Verhaltensspezifikation

Buszugang (1)

Problem: Mehrere Sender an einem Bus

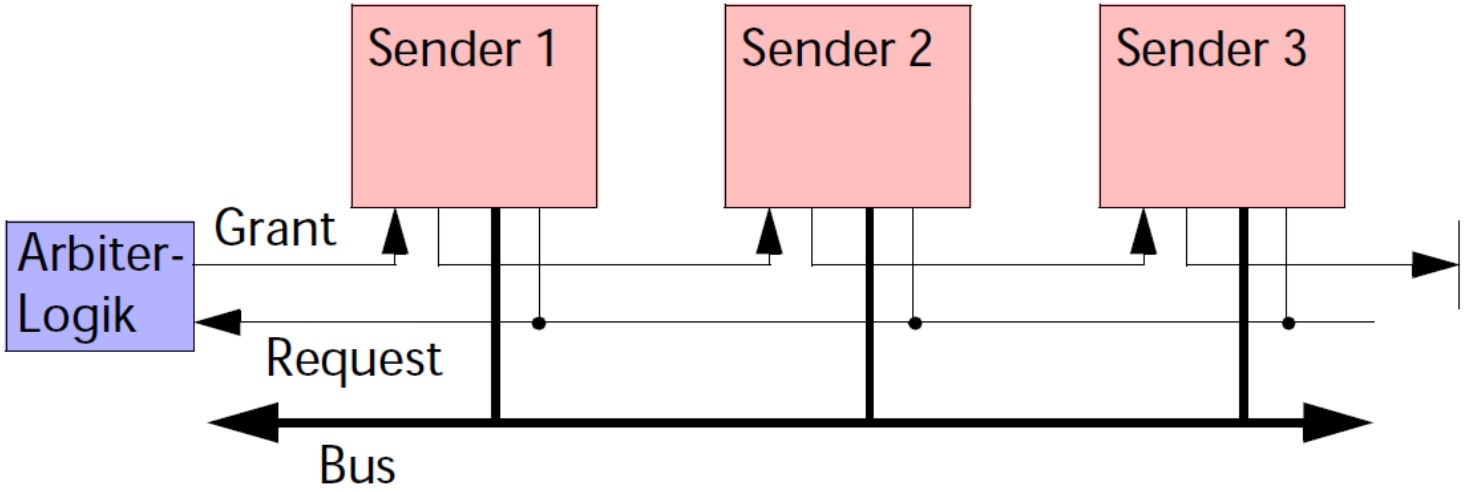
- z.B. mehrere CPUs am selben CPU-Bus
- Elektrische Probleme
  - Kurzschlüsse bei gleichzeitiger Bus-Nutzung durch mehrere Sender
- Konsistenzprobleme
  - Übertragungsprotokolle nicht abgesichert gegen mehrere Sender
  - Störung des Protokolls

Zugangsregelung für Busse mit mehreren Sendern: Busarbitrierung

- Gängige Verfahren
  - *Daisy Chain*
  - Parallele und zentrale Arbitrierung
  - Verteilte Arbitrierung

Buszugang (2)

Daisy Chain Technik



- Über *Request*-Leitung signalisiert Sender die Busnutzung
- Freigabe wird über *Grant*-Signal verbreitet
  - Weitergabe des *Grant*-Signals von Sender zu Sender *(Daisy Chain)*
  - Unterbrechung des *Grant*-Signals, wenn Sender Bus nutzen will
  - Üblicherweise Verbreitung eines Besetztsignals durch Arbiterlogik an alle

Buszugang (3)

Daisy Chain Technik (fortges.)

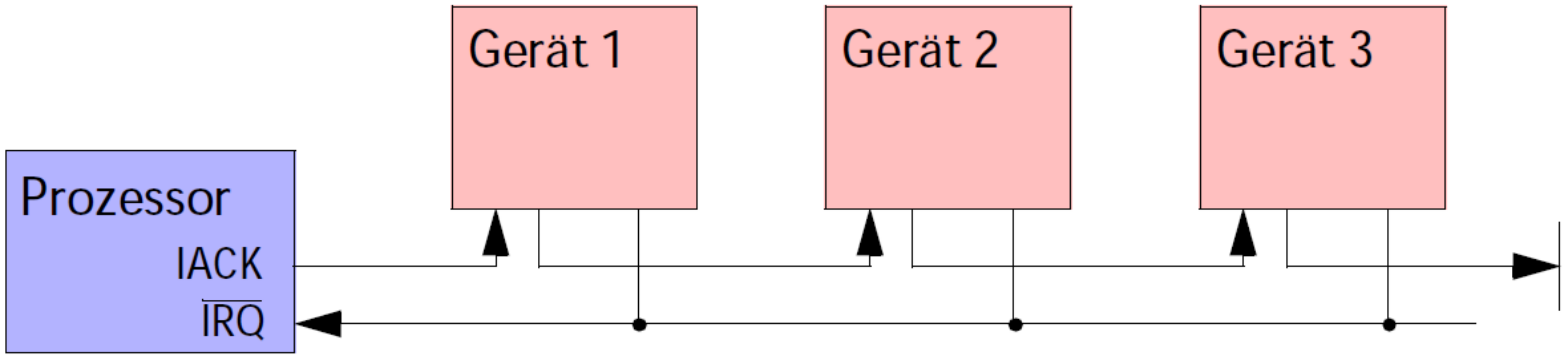
- Realisierung bei Steckkarten
  - Eingabe- und Ausgabeleitung der *Daisy Chain*
  - Verbindung der beiden Leitungen bei unbelegter Karte (z.B. durch *Jumper* oder Schalter)
- Priorisierung der Sender durch Anordnung in der *Daisy Chain*
  - Unfaire Busvergabe
  - Aushungerungsproblem *(starvation)*

Typischer Einsatz von _Daisy Chain_ Arbitern

- VME-Bus
- *Interrupt-Acknowledge* Logik bei mehreren *Interrupt*-Quellen gleicher Priorität

Buszugang (4)

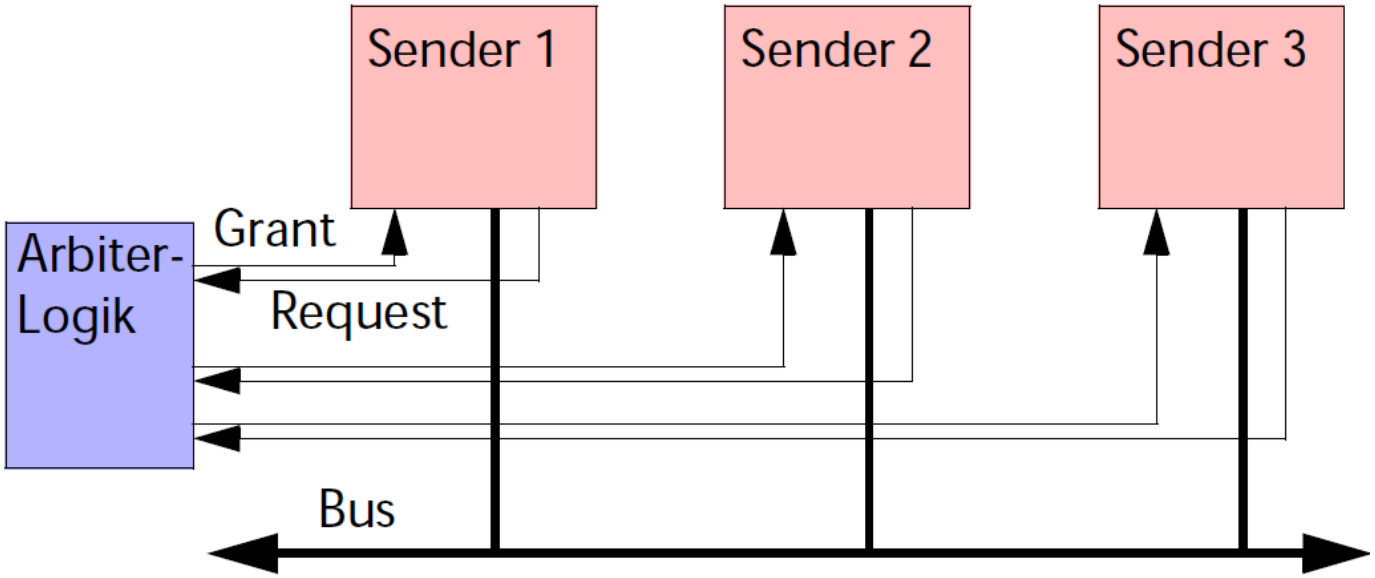
Fallstudie: *Interrupt-Acknowledgment*



- Gerät signalisiert Unterbrechung durch IRQ-Leitung
- Prozessor bestätigt Unterbrechung durch IACK-Leitung
  - Erstes Gerät in der *Chain* nimmt IRQ-Leitung zurück
  - Andere Geräte warten, bis sie an der Reihe sind

Buszugang (5)

Parallele, zentrale Arbitrierung



- Über eigene *Request*-Leitung signalisiert Sender die Busnutzung
- Eigene *Grant*-Leitung signalisiert Sender die Freigabe
- Beliebige, aber komplexere Arbiter-Logik
  - Verschiedene Priorisierungen und Fairness-Strategien möglich

Buszugang (6)

Parallele, zentrale Arbitrierung (fortges.)

- Schnelle *Grant*-Signalisierung
  - Weniger Gatterlaufzeiten als bei *Daisy Chain*
- Zentraler Arbiter könnte aber zum Flaschenhals werden, der Busnutzung limitiert

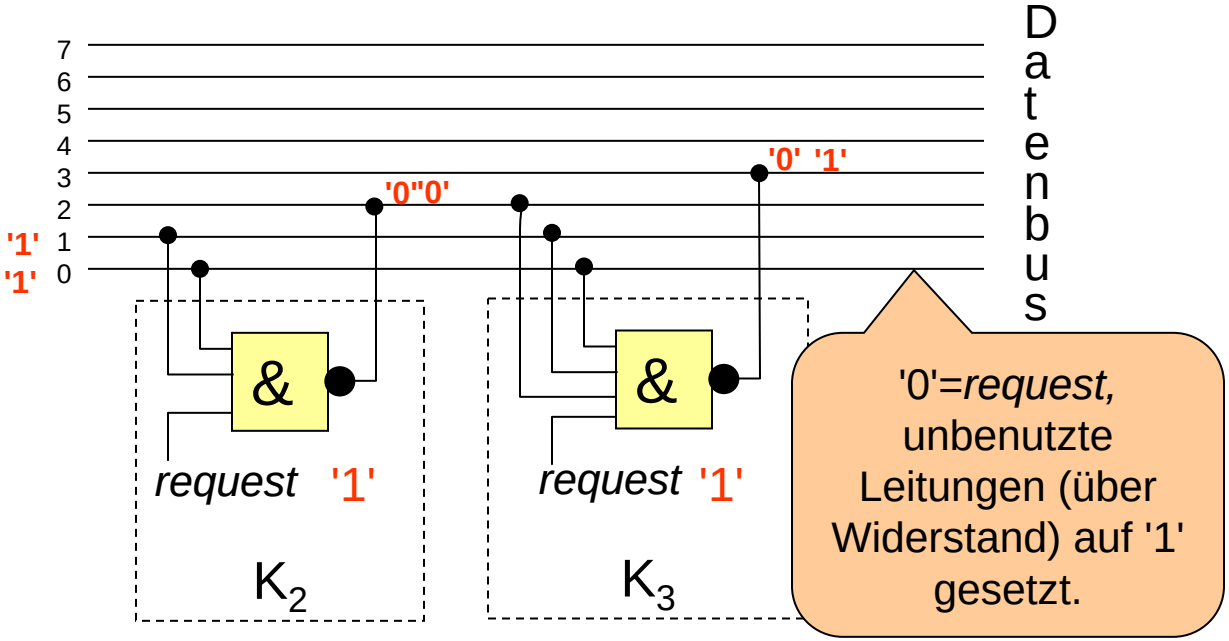
Typischer Einsatz von paralleler, zentraler Arbitrierung

- Alle heute gängigen internen Bussysteme
  - z.B. PCI-Familie von Bussen (ausgewählter Sender wird Bus-*Master*)

Buszugang (7)

Verteilte Arbitrierung *(self selection)*

„These schemes also use multiple request lines, but the devices requesting bus access determine who will be granted access.“ [Hennessy/Patterson]



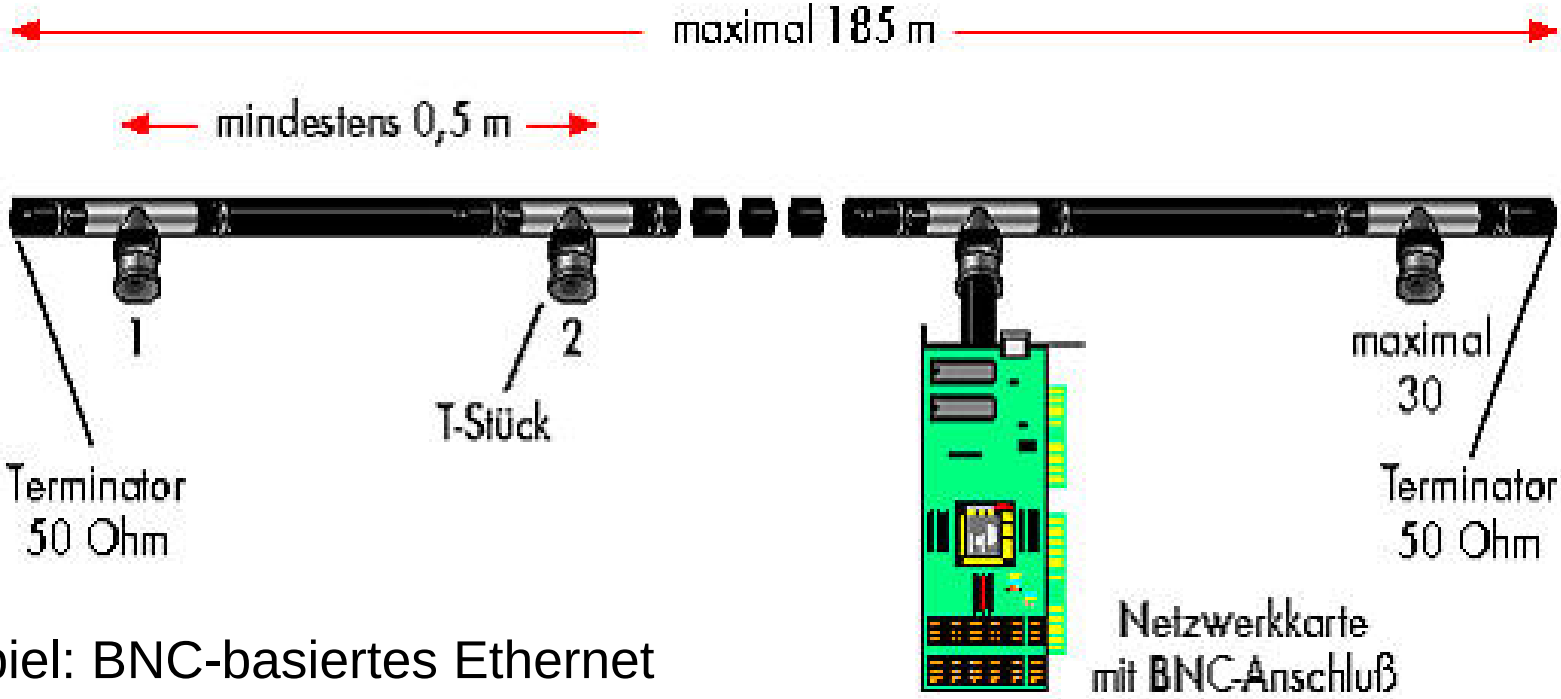
Phase, in der Datenbus zur Arbitrierung benutzt wird. Ein eigener *Request* erscheint nur dann als ‚0‘ auf dem Bus, wenn kein Wunsch einer höheren Priorität auf dem Bus angemeldet ist.

Max. Anzahl der *Controller* = Anzahl der Datenleitungen. Beispiel: SCSI

Buszugang (8)

Verteilte Arbitrierung *(collision detection)*

„In this scheme, each device independently requests the bus. Multiple simultaneous requests result in a collision. The collision is detected and a scheme for selecting among the colliding parties is used.“
[Hennessy/Patterson]

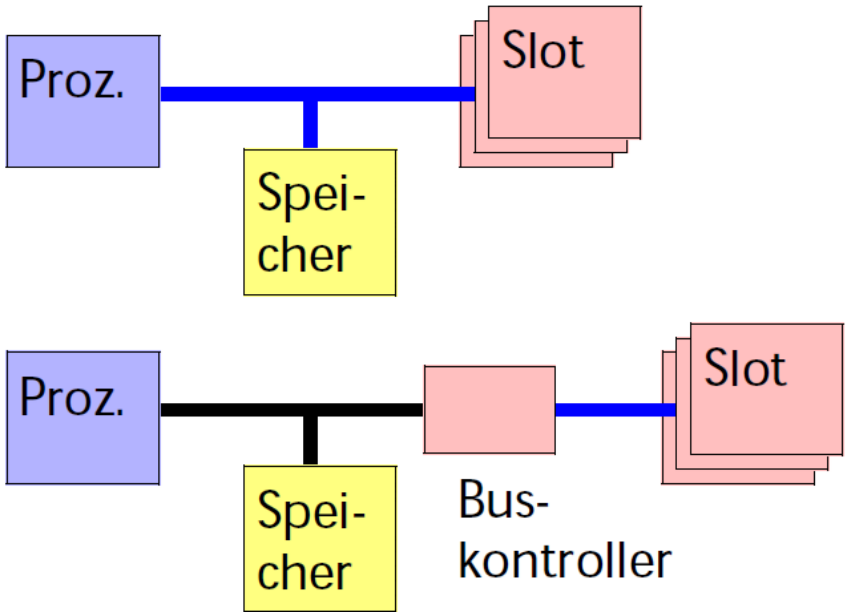


Beispiel: BNC-basiertes Ethernet

Interne Busse (1)

Typische interne Busse

- Spezifikation der Pins und Signale
- Spezifikation der Steckerverbindung für Einsteckkarten *(Slots)*
- Erweiterung des CPU-Busses
  - Übernahme des CPU-Busses, Hinzunahme weiterer Leitungen (z.B. für Unterbrechungssteuerung)
  - Umsetzung des CPU-Busses auf internen Bus über Buskontroller-Chipsatz

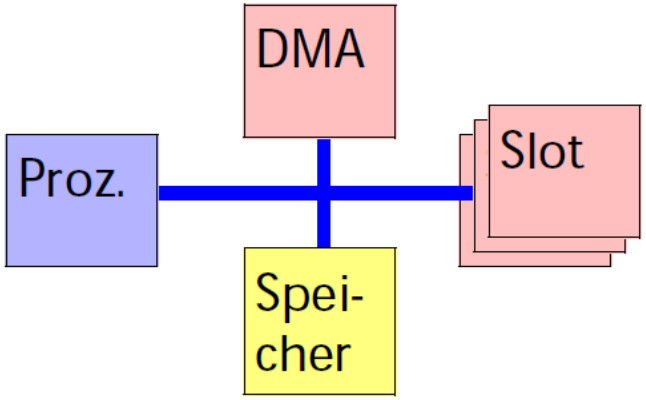


Interne Busse (2)

Interne Busse mit DMA-Controller

- Evtl. mehrere DMA-Kanäle
  - Mehrere Transfers gleichzeitig aktivierbar

Beispiel: DMA-Controller am CPU-Bus



- Programmierung über spezielle DMA-Register

Fallstudie: PCI-Bus (1)

***Peripheral Component Interconnection* (PCI von Intel)**

– Seit etwa 1992

Busarchitektur

– Synchroner Bus
– Taktrate 33 bzw. 66 MHz
  (133, 266 und 533 MHz sind definiert aber meist noch nicht genutzt)
– Datenbreite 32 Bit (64 Bit sind definiert aber selten genutzt)
  – Datenübertragung im *Burst*-Mode 133 MByte/s (266 MByte/s)
– Adressbreite 32 Bit
  – 32 Bit Speicheradressraum
  – 32 Bit I/O-Adressraum
– Zusätzlich: Konfigurationsadressraum pro Busteilnehmer

Fallstudie: PCI-Bus (2)

Buselektrik

- Meist 5 V
- Tendenz zu geringeren Spannungen 3,3 V oder 1,5 V
  - Höhere Busgeschwindigkeit möglich

Busarbitrierung

- Parallele, zentrale Arbitrierung durch Buscontroller
- Bus-*Master* kann PCI-Bus für einen Zugriff benutzen
- In den Anfängen kein DMA-Controller
  - Stattdessen überträgt Bus-*Master* Daten selbst

Fallstudie: PCI-Bus (3)

Buszyklen

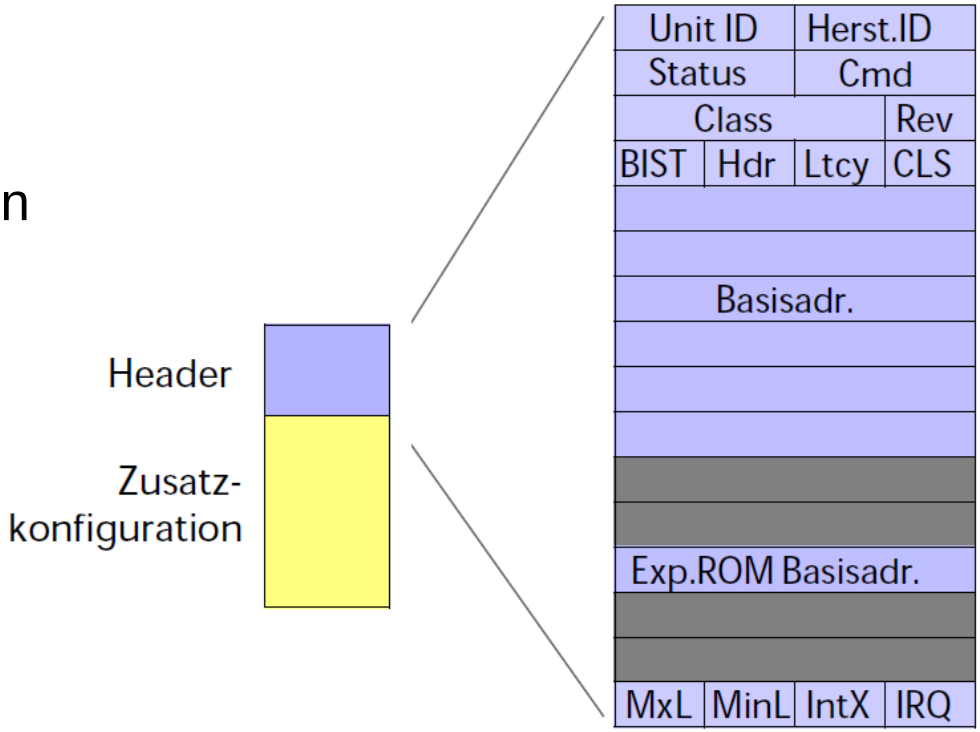
- Speicherzugriff (Lesen, Schreiben)
  - Adresse und Daten im Zeitmultiplex über gleiche Leitungen
- I/O-Zugriff (Lesen, Schreiben)
- Konfigurationszugriff (Lesen, Schreiben der Konfigurationsparameter)
- *Burst*-Mode Zugriffe
  - Eine Adressübertragung
  - Mehrfache Datenzugriffe

Fallstudie: PCI-Bus (4)

Konfiguration

- Jeder Bus-Teilnehmer

  256 Byte Konfigurationsdaten
  - 64 x 32 Bit
- Anordnung genormt
  - 64 Byte genormter *Header*
  - 192 Byte Zusatzkonfiguration



Fallstudie: PCI-Bus (5)

Konfiguration (fortges.)

- Codes für Hersteller und Gerätenummer des Herstellers
  - Wird für *Plug-and-Play*-Betrieb benötigt
- Code für Geräteart und Revisionsnummer
  - Z.B. Busteilnehmer ist ein *Floppy-Controller*
- Diverse Konfigurationsdaten zu
  - Selbsttest
  - Struktur der Zusatzkonfigurationsdaten, etc.
- Adressregister für das Einblenden der I/O-Register und eines ROMs in die Busadressräume
- Latenzinformationen bei Buszugriffen
- Zuordnung der Unterbrechungsleitung des Busteilnehmers zu PCI-Unterbrechung und Prozessorunterbrechung

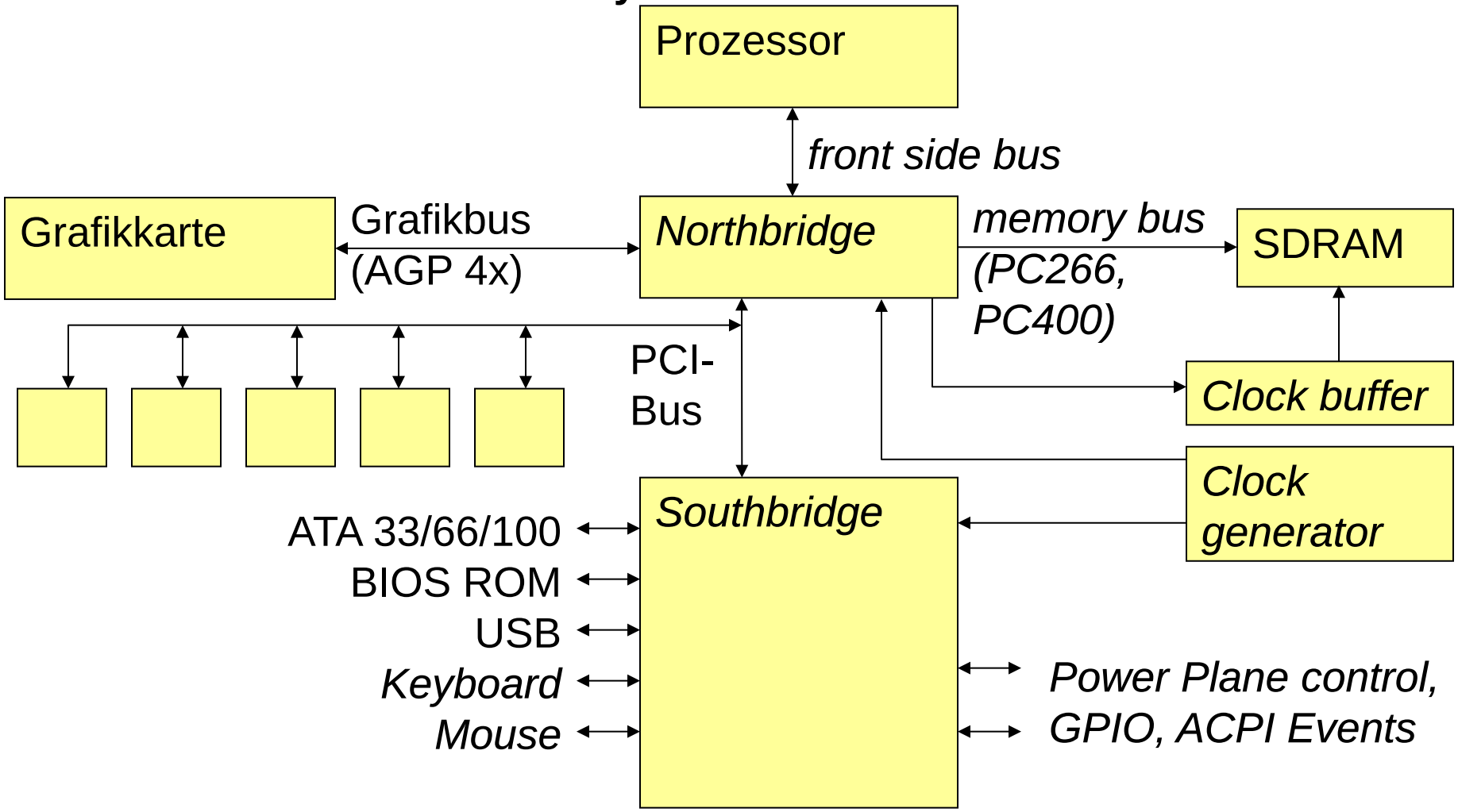
Fallstudie: PCI-Bus (6)

Konfiguration (fortges.)

- Initialisierung beim Hochfahren des Betriebssystems
- Selektierung der einzelnen Busteilnehmer (Karten) durch IDSEL-Signal
  - Konfiguration der I/O- und Speicheradressen
  - Automatische Konfliktauflösung
    (sensible Kartentypen können bevorzugt werden)
  - Prozessor-IRQ-Leitungen können gemeinsam benutzt werden
- Keine manuelle Kartenkonfiguration (wie bspw. bei alten ISA-Bussen) mehr notwendig

Fallstudie: PCI-Bus (7)

Grundstruktur PCI-basierter Systeme



Fallstudie: PCI-Bus (8)

***Front-Side*-Bus**

– Systembus zum Chipsatz
– Typisch: 64 Bit Daten, 32 Bit Adressen, 200 MHz, mehrere Datentransfers pro Taktzyklus

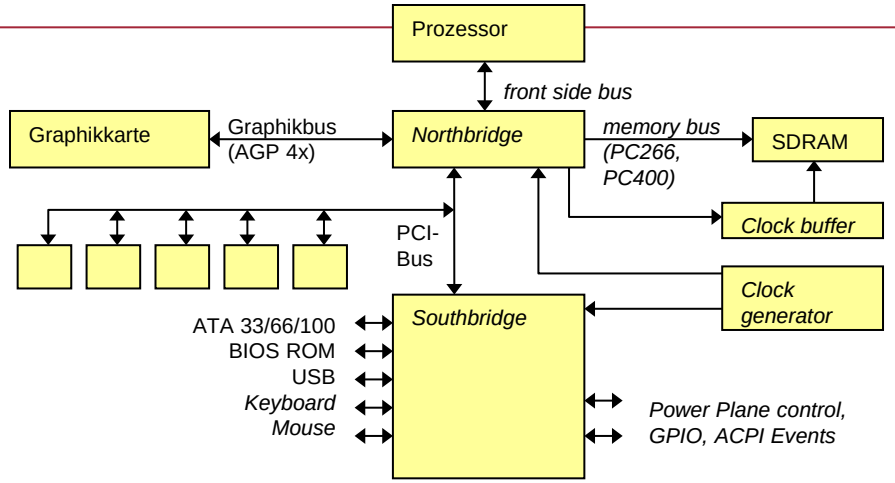
Northbridge

– Anbindung des Hauptspeichers und schneller Grafikkarte

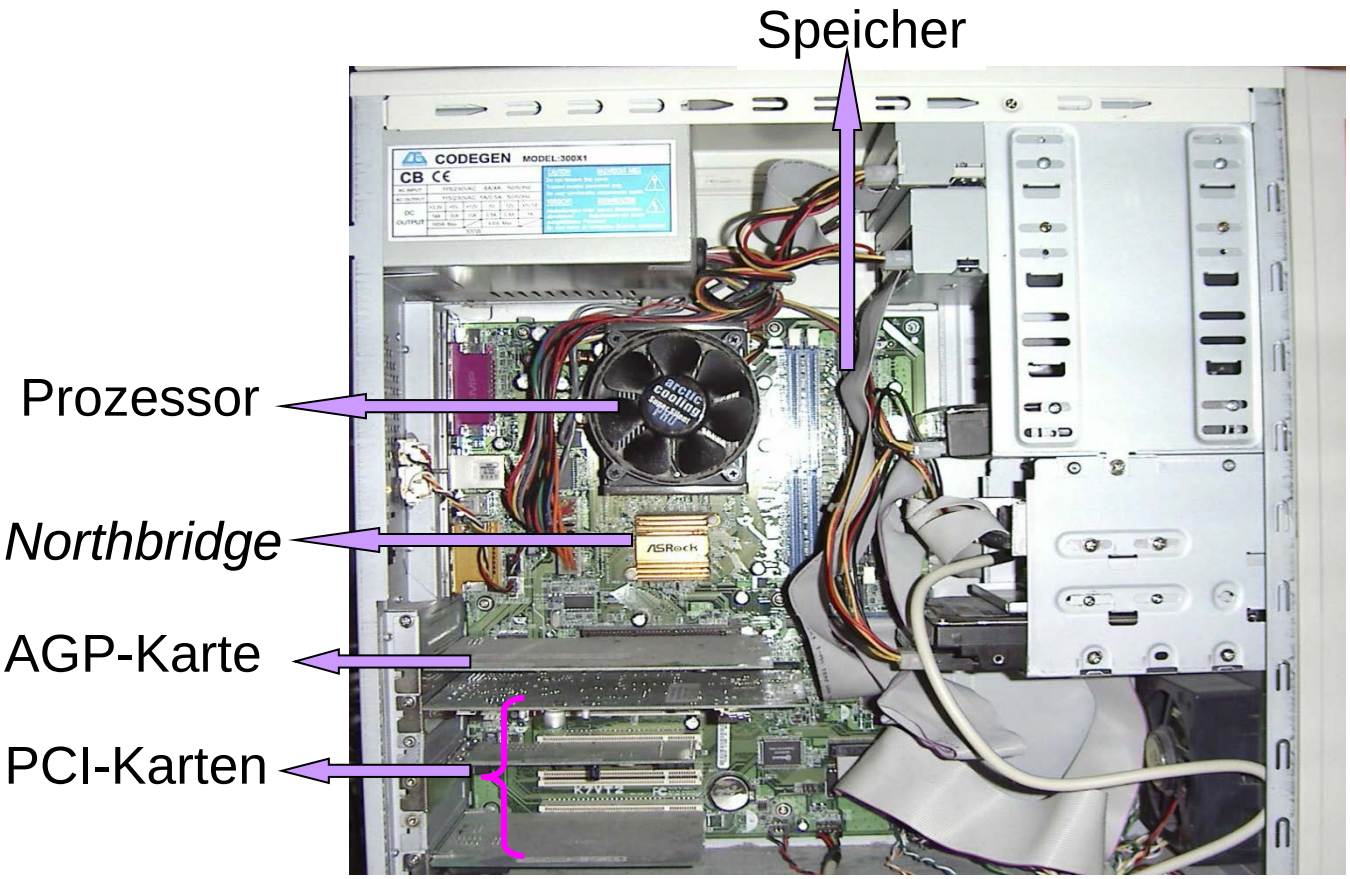
Southbridge

– Umsetzung auf PCI-Bus
– Anbindung von Legacy-I/O (RS-232, SPP, EPP etc.)

Fallstudie: PCI-Bus (9)



Wo sind diese Komponenten auf dem Board?



Fallstudie: PCI-Bus (10)

Heutiger Standard (PCI 2.0) für hohe Datenraten nicht ausreichend

- Z.B. für schnelle Grafikkarten oder Gigabit-Ethernet-Netzwerkkarten

PCI-X

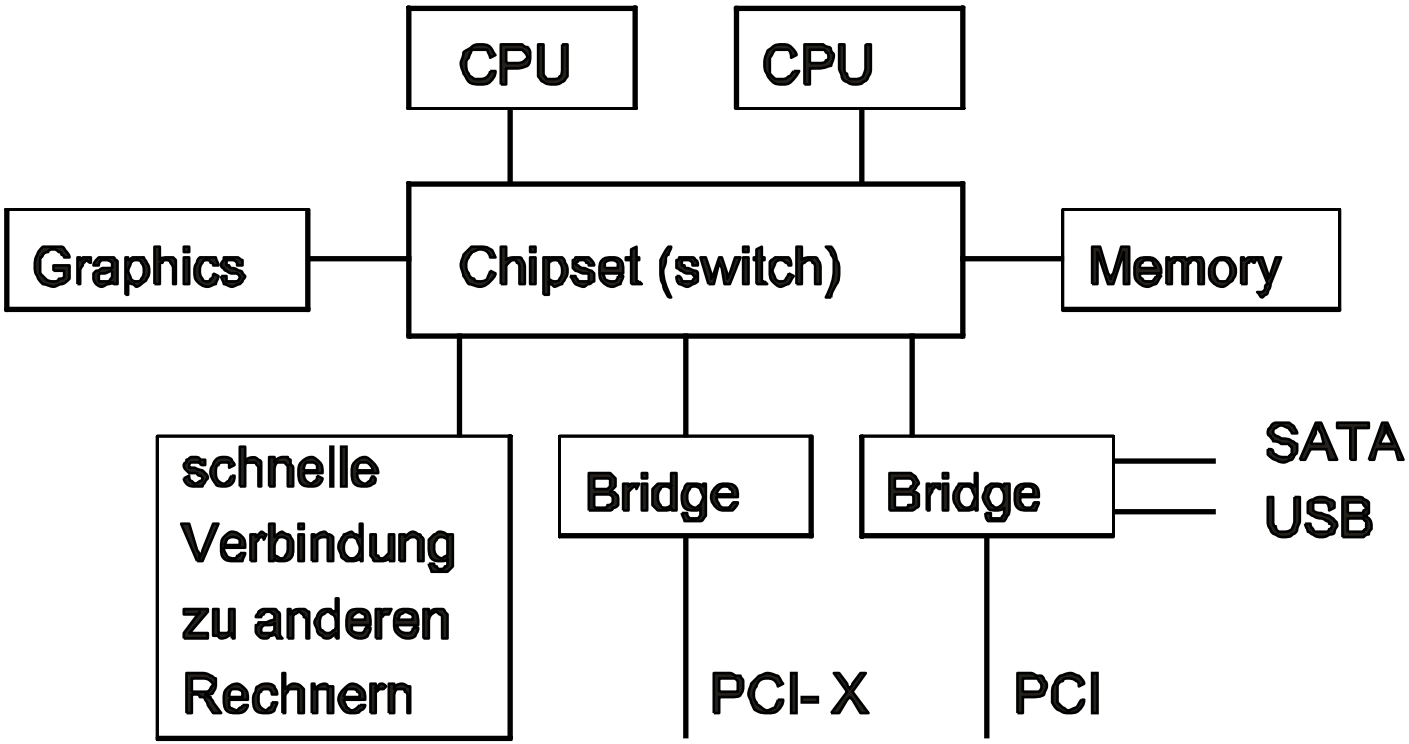
- Erweiterung auf 64 Bit Datenbreite

PCI-Express

- Serielle *Point-to-Point* Verbindungen *(Lanes)* zwischen den Slots
- Max. Übertragungsrate 2 GBit/s pro *Lane*
  - Mehrere *Lanes* pro Karte möglich
- Konfigurierter *Switch* steuert Verbindungen
  - Mehrere *Lanes* pro Verbindung schaltbar
  - Max. Übertragung 9,5 GByte/s
- Software kann kompatibel bleiben

Fallstudie: PCI-Bus (11)

PCI-Express (fortges.)



– Möglichkeit der Vermeidung eines Busses, an den mehrere Karten angeschlossen sind und die gegenseitig die Geschwindigkeit begrenzen

Externe Busse

Busse für spezielle Anwendungen

- Verbindung bestimmter Gerätetypen und/oder
- Verbindung gehäuseexterner Geräte
- Keine Verlängerung des CPU-Busses

Beispiele

- Busse zur Ansteuerung von Festplatten und ähnlichen Geräten
  - Gehäuseintern und -extern
  - z.B. IDE und SCSI
- Busse zur allgemeinen Ansteuerung externer Geräte
  - z.B. USB

Fallstudie: IDE (1)

Integrated Drive Electronics (IDE)

- Seit 1986
- Uneinheitliche Namensgebung
  - ATA, ATA-1 (IDE), ATA-2 (EIDE), ATA-3, ATA-4 (ATAPI, UDMA 33), ATA-5 (UDMA 66), ATA-6 (UDMA 100), ATA-7 (UDMA 133)
- Verschiedene historische Erweiterungen

IDE-Spezifikation

- Kabel und Steckverbinder für Festplatten
- Signalleitungen im Kabel
- Registersatz des I/O-Bausteins zur Plattenansteuerung und dessen Semantik
  - Festplattencontroller auf der Festplatte

Fallstudie: IDE (2)

Ansteuerung

- Anbindung der Festplatte über PCI-basierten *Controller*
  - Vorgeschalteter *Controller* tritt als PCI-Busteilnehmer auf
- Zwei Festplatten pro *Controller* ansprechbar
  - *Primary* und *Secondary Device*
  - Zum Teil im *Daisy Chain* Verfahren verknüpft
- Datenübertragung in der Regel per DMA
  - Verschiedene Erweiterungen: bis 133 MByte/s (Ultra DMA 133)

Fallstudie: IDE (3)

Kommandos

- *Identify Drive:* ermittelt Laufwerksparameter
- *Read:* lese Plattenblock per 16 Bit I/O-Register
- *Read DMA:* lese Plattenblock per DMA
- *Write DMA:* schreibe Plattenblock per DMA
- und einige mehr

Plattencontroller

- U.U. überlappende Bearbeitung von Kommandos
- Interne Plattensteuerung
  - Schreib-/Lesekopfbewegungen
  - Datenaufzeichnung und Sektorerkennung

Fallstudie: IDE (4)

Benennung von Plattenblöcken

- Zylinder- (16 Bit), Kopf- (4 Bit) und Sektornummer (8 Bit)
  - CHS = *Cylinder, Head, Sector*
  - max. 127 GB
  - Heute meist Umsetzung der Angaben durch den Plattencontroller (CHS-Info stimmt nicht mit tatsächlicher Plattengeometrie überein)
- *Logical Block Addressing (LBA)*
  - 28 Bit lineare Blocknummern, max. 128 GB
  - 48 Bit lineare Blocknummern, max. 128 PB (Petabyte)

Fallstudie: IDE (5)

ATAPI-Erweiterung

- *ATA Packet Interface*
- Übertragung von Datenpaketen ungleich 512 Byte (Standardsektorgröße)
- Geeignet für Wechselmedien, CD- und DVD-ROM-Laufwerke

Fallstudie: IDE (6)

SATA – *Serial ATA*

- Serielle Übertragung
  - Differenzielle Übertragung:
    Normalerweise werden Werte über Spannungen zwischen einer Signalleitung und Masse codiert (*single-ended*, asymmetrisch). Bei differenziellen Signalen benötigt jedes Signal zwei Leitungen. Ist eine positiv gegenüber der anderen bedeutet dies eine ‚0', ist sie negativ, eine ‚1'.
    ☞ Bessere Störunempfindlichkeit.
  - NRZ-Codierung *(Non-Return-to-Zero)*, d.h. jedem Bitwert ist ein fester Signalpegel zugeordnet
- Weniger Kabel, weniger Energieverbrauch
- Transferraten bis 150 MByte/s bzw. 300 MByte/s (SATA II)

Fallstudie: SCSI (1)

Small Computer's System Interface (SCSI)

- Seit 1979 (Shugart)
- Ursprünglich zum Anschluss von Platten konzipiert, heute Bus zur Ansteuerung von
  - Festplatten und optischen Speichermedien
  - Bandgeräten
  - Scannern
  - u.a.

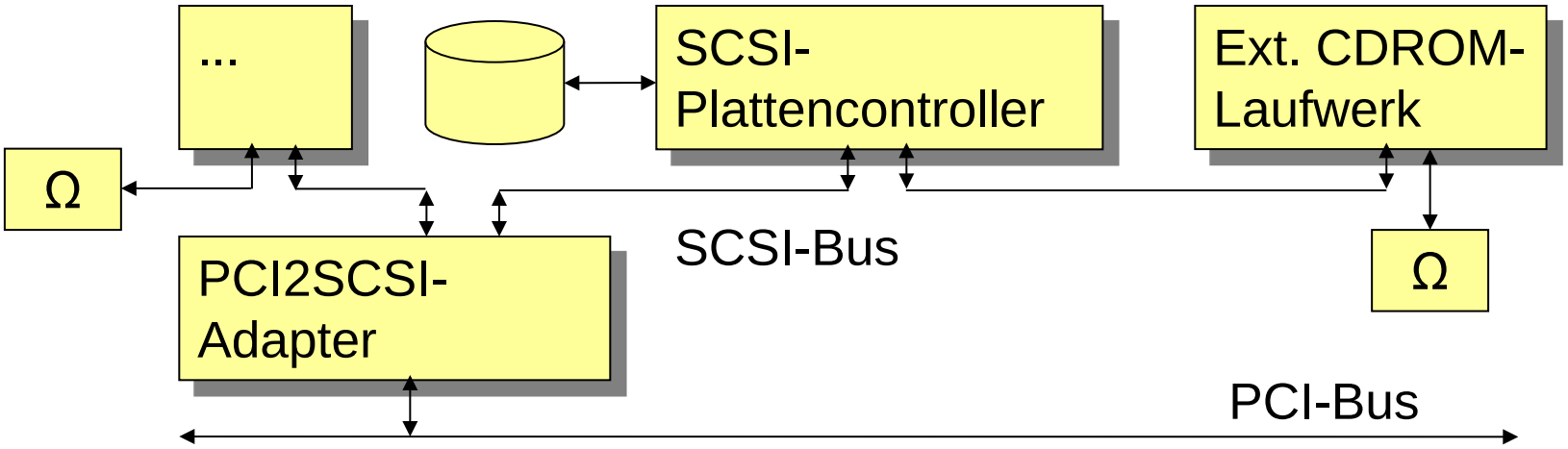
Echter Bus

- Bis zu 8 oder 16 Busteilnehmer (einschließlich Rechner)
- Normalerweise ein Rechner (aber auch mehrere möglich)
- Jeder Teilnehmer kann Übertragung initiieren
- Datenbreite 8 oder 16 Bit plus Paritätsbit(s)

Fallstudie: SCSI (2)

Bus

- Alle Geräte werden an Bus gehängt
  - Typisch: pro Gerät zwei Stecker zur Verkettung der Geräte
- Busterminierung
  - Aus elektrischen Gründen muss letztes Gerät einen Abschlusswiderstand auf den Bus setzen (sonst Signalechos oder falsche Signalisierung möglich)



Fallstudie: SCSI (3)

Verschiedene Übertragungsstandards (ausgewählt)

- *Fast SCSI:* 10 MHz Bustakt, 8 Bit, 10 MByte/s Übertragung, max. 7 Geräte
- *Fast and Wide SCSI:* 10 MHz Bustakt, 16 Bit, 20 MByte/s Übertragung, max. 15 Geräte
- *Ultra Wide SCSI:* 20 MHz Bustakt, 16 Bit, 40 MByte/s Übertragung, max. 4 oder 7 Geräte
- *Ultra-3 Wide SCSI:* 80 MHz Bustakt, 16 Bit, 160 MByte/s Übertragung, differenzielle Signale

Fallstudie: SCSI (4)

Busarbitrierung

- BSY-Signal zeigt Busbelegung an
- Buszugang durch *Distributed Arbitration with self-selection*
  - Pro Gerät eigene SCSI-ID (0-7, 0-15)
  - Datenleitung mit ID-Nr. wird gesetzt
  - Konfliktauflösung: Gerät mit niedrigerer ID gewinnt Buszugang
- Bus belegt bis explizit Freigabe durch belegendes Gerät

Kommandos

- Kommandodaten ähnlich wie bei IDE, Kommandoerweiterungen möglich
- Datentransport zwischen den Geräten untereinander möglich
  - Z.B. mehrere Rechner am Bus
  - Z.B. Datenübertragung von Festplatte zu Festplatte

Fallstudie: SCSI (5)

Vergleich IDE

- SCSI aufwändiger und teurer als IDE
- Häufiger Einsatz im *High-End* Bereich
  - Große und schnelle Festplatten meist mit SCSI-Anschluss
  - „Langlebigere“ Platten
  - Erlaubt durch standardisierte Befehlsschnittstellen weitgehende Abstraktion vom physikalischen Medium und Entlastung des Prozessors
  - Flexible Kommandostruktur geeignet für externe RAID-Systeme (Festplatten-Arrays mit redundanter Datenspeicherung)

Fallstudie: *Universal Serial Bus, USB* (1)

Moderner billiger Allzweckbus für externe Geräte

- Maus, Tastatur
- Drucker, Fax, Modems, ISDN-Adapter
- Disketten, Band-, CD-, Festplattenlaufwerke
- Kameras, Mikrofone

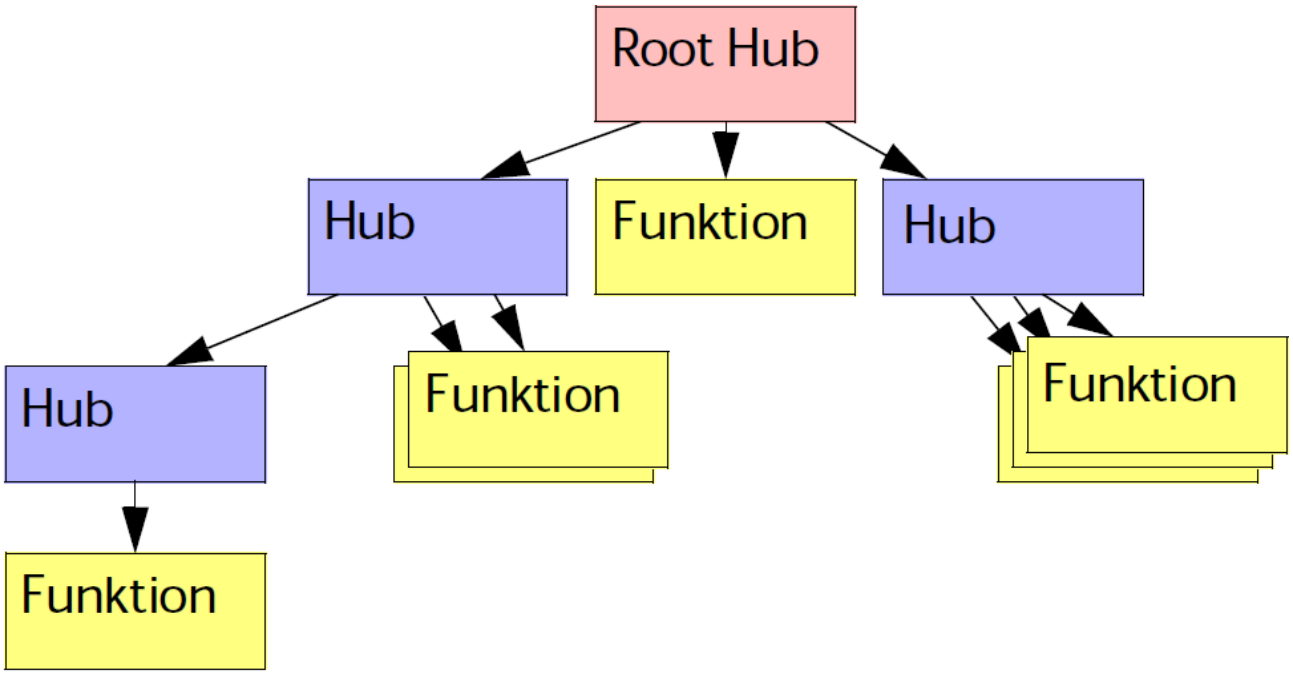
Busarchitektur

- Zentrale Komponente (Rechner)
- 4-adriges Kabel
  - Versorgungsspannung, Masse, positive und negative Datenleitung
  - Stromversorgung für einfache Geräte integriert
- Billige Stecker (adaptiert vom Gameboy-Linkkabel)

Fallstudie: *Universal Serial Bus, USB* (2)

Busarchitektur (fortges.)

- Sternverkabelung (Kabel nur Punkt zu Punkt)
  - *Hubs* als Verteiler
  - Max. 7 verschachtelte Ebenen, max. 127 Knoten bzw. Funktionen



Fallstudie: *Universal Serial Bus, USB* (3)

Datenverkehr

- Paketorientierte Datenübertragung
- USB setzt stets einen Master (Rechner/*Root Hub*) voraus
- Vier Datenraten
  - *Low Speed:* 1,5 MBit/s (USB 1.1)
  - *High Speed:* 12 MBit/s (USB 1.1)
  - *Full Speed:* 480 MBit/s (USB 2.0)
  - *Super Speed:* 4 Gbit/s (USB 3.0)
  - Ratenanpassung in den *Hubs* möglich

Fallstudie: *Universal Serial Bus, USB* (4)

Datenverkehr (fortges.)

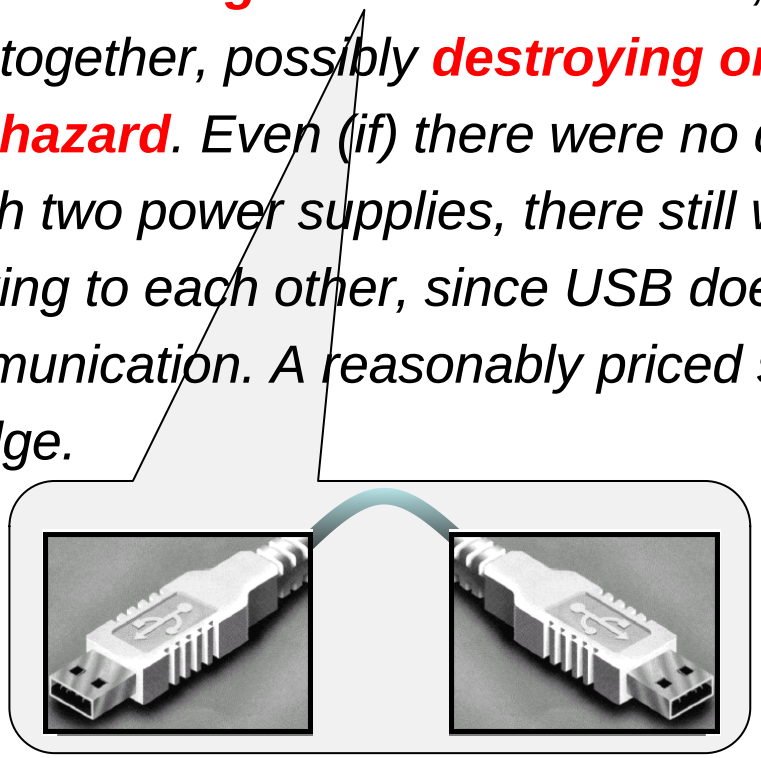
- Übertragungsmodi
  - Nachrichten (Daten- und Kontrollnachrichten)
    max. 54 Bytes, garantiert pro Zeitintervall
  - Strom (aufgeteilt in mehrere Pakete)
    nur wenn Bandbreite vorhanden ist
  - Isochroner Strom (garantierte Datenrate für Strom, z.B. für Kamera oder Mikrofon)
    zeitgenau, keine Fehlerbehandlung

Fallstudie: *Universal Serial Bus, USB* (5)

Aufbau von Netzwerken? Auszug aus FAQ, seinerzeit auf www.usb.org:

Q7: You mean I can't make a direct cable connection like a null modem?

A7: Correct. In fact, if you try this with an ***illegal A to A USB cable****, you'll short the two PCs' power supplies together, possibly* ***destroying one or both machines or causing a fire hazard****. Even (if) there were no danger to the machines from the problem with two power supplies, there still wouldn't be any way to get the two PCs talking to each other, since USB doesn't support that particular kind of communication. A reasonably priced solution to handle this need is the USB bridge.*



Fallstudie: *Universal Serial Bus, USB* (6)

Datencodierung

– Synchronisationsprobleme bei NRZ-Codierung und langen Datenpaketen
– Einsatz von NRZI-Codierung *(Non-Return-to-Zero-Inverted)* kombiniert mit *Bit-Stuffing*, um bei der seriellen Übertragung Taktung zu erzeugen

NRZI-Codierung

– Leitungspegel wechselt bei der Übertragung einer 0 und bleibt bei der Übertragung einer 1

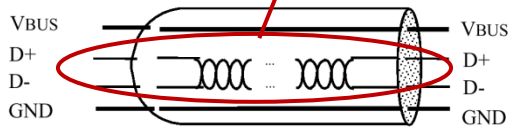
Bit-Stuffing

– Einfügen einer „künstlichen“ 0 nach (hier) sechs aufeinanderfolgenden 1
– *Bit-Stuffing* beim Senden *vor* NRZI-Codierung. Beispiel:

zu übertragen: 01100100011101001111111010
nach Bit-Stuffing: 01100100011101001111110**0**11010

Signal:

Twisted Pair



Roter Faden

8. Ein-/Ausgabe

- Einleitung
- Prinzipien der Datenübergabe
- *Point-to-Point* Verbindungen
- Busse
  - Arbitrierungsverfahren:
    Daisy Chain, parallele und zentrale Arbitrierung, verteilte Arbitrierung
  - Interne Busse
    - Fallstudien: PCI, PCI-X, PCI-Express
  - Externe Busse
    - Fallstudien: IDE, ATAPI, SATA, SCSI, USB
- Fehlererkennung mit CRC-Zeichen

Fehlererkennung mit CRC-Zeichen

Problem

- Datenübertragung über physikalisches Medium kann fehlerbehaftet sein
  - Auf Ebene der Netzwerkpakete:
    durch Netzwerkprotokoll detektiert und behandelt
  - Auf Ebene der Paketinhalte (fehlerhafte Übertragung einzelner oder mehrerer Bits):
    Möglichkeit zur Validierung empfangener Daten erforderlich

Prinzip: Redundante Codierung der Daten

- D.h. es werden mehr Bits übertragen als Nutzdaten

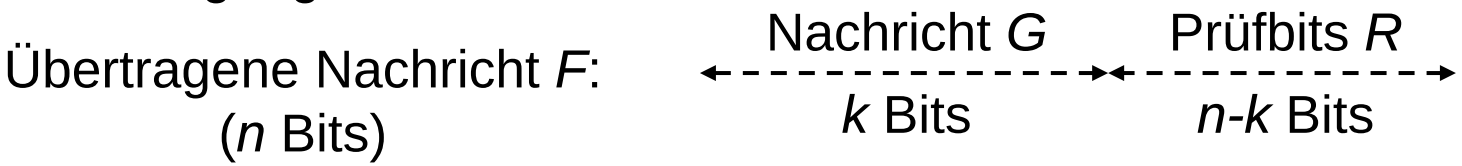
Verbreitete Methode: *Cyclic Redundancy Check (CRC)*

- Erkennt Vielzahl von Fehlertypen (je nach Parametrisierung)
- Ist *error-detecting code*, im Gegensatz zu *error-correcting code*

Bildung von CRC-Zeichen

Zu übertragen: Nachricht *G*, in *k* Bits codiert.
Ziel: Durch Übertragung von *n* Bits (*n* > *k*) Schutz gegen Fehler

Annahme: Systematischer Code – Übertragene Nachricht stimmt in *k* Bits mit der Ausgangsnachricht überein



Interpretation: An eine ursprüngliche Nachricht *G* wird eine gewisse Anzahl von Prüfbits *R* angehängt, und die resultierende *n* Bits lange Nachricht wird über ein Kommunikationsmedium übertragen.

Betrachtung der Bits a_i der Nachricht als Koeffizienten eines Polynoms *G*, des *Nachrichtenpolynoms*:

$$(1) \qquad G = \sum_{i=0}^{k-1} a_i * 2^i$$

Beispiel zur Polynom-Darstellung

Nachricht *G*: `11000101`

1. Bit von rechts hat Wertigkeit 1 = 2^0
2. Bit von rechts hat Wertigkeit 2 = 2^1
3. Bit von rechts hat Wertigkeit 4 = 2^2
4. Bit von rechts hat Wertigkeit 8 = 2^3
5. Bit von rechts hat Wertigkeit 16 = 2^4
6. Bit von rechts hat Wertigkeit 32 = 2^5
7. Bit von rechts hat Wertigkeit 64 = 2^6
8. Bit von rechts hat Wertigkeit 128 = 2^7

$G = \mathbf{11000101} = 2^7 + 2^6 + 2^2 + 2^0$

$= \mathbf{1}*2^7 + \mathbf{1}*2^6 + \mathbf{1}*2^2 + \mathbf{1}*2^0$

$= \mathbf{1}*2^7 + \mathbf{1}*2^6 + \mathbf{0}*2^5 + \mathbf{0}*2^4 + \mathbf{0}*2^3 + \mathbf{1}*2^2 + \mathbf{0}*2^1 + \mathbf{1}*2^0$

$= \sum_{i=0}^{k-1} a_i * 2^i$

Darstellung von Nachrichten

Verallgemeinerung: Ersetzen der Basis 2 durch Basis *x*

$$(2) \qquad G(x) = \sum_{i=0}^{k-1} a_i * x^i$$

Übertragene Nachricht *F(x)*

– Nehme ursprüngliches Nachrichtenpolynom *G(x)*

– „Mache Platz“ für *n-k* viele Bits des *Prüfbit-Polynoms* *R(x)*, indem *G(x)* um *n-k* Stellen nach links geschoben wird

☞ Links-Schieben einfach per Multiplikation von *G(x)* mit x^{n-k}

– Erzeuge zu übertragendes *Codepolynom* *F(x)*, indem einfach *R(x)* an nach links verschobene Nachricht angehängt wird, d.h. hinzuaddiert wird:

$$(3) \qquad F(x) = G(x) * x^{n-k} + R(x)$$

Beispiel zur Erzeugung der übertragenen Nachricht

Nachricht *G*: 11000101 **Prüfbits *R*: 110**

☞ $G(x) = \mathbf{1}*x^7 + \mathbf{1}*x^6 + \mathbf{0}*x^5 + \mathbf{0}*x^4 + \mathbf{0}*x^3 + \mathbf{1}*x^2 + \mathbf{0}*x^1 + \mathbf{1}*x^0$

Verschiebe *G(x)* um 3 Stellen nach links, da *R* 3 Bits lang ist:

$$G(x) * x^3 = (\mathbf{1}*x^7 + \mathbf{1}*x^6 + \mathbf{0}*x^5 + \mathbf{0}*x^4 + \mathbf{0}*x^3 + \mathbf{1}*x^2 + \mathbf{0}*x^1 + \mathbf{1}*x^0) * x^3$$
$$= \mathbf{1}*x^{10} + \mathbf{1}*x^9 + \mathbf{0}*x^8 + \mathbf{0}*x^7 + \mathbf{0}*x^6 + \mathbf{1}*x^5 + \mathbf{0}*x^4 + \mathbf{1}*x^3$$

Addiere Prüfbit-Polynom *R(x)* hinzu:

$$G(x) * x^3 + R(x) = \mathbf{1}*x^{10} + \mathbf{1}*x^9 + \mathbf{0}*x^8 + \mathbf{0}*x^7 + \mathbf{0}*x^6 + \mathbf{1}*x^5 + \mathbf{0}*x^4 + \mathbf{1}*x^3$$
$$+ \mathbf{1}*x^2 + \mathbf{1}*x^1 + \mathbf{0}*x^0$$

Zu übertragendes Codepolynom *F(x)*:

$$F(x) = \underbrace{\mathbf{1}*x^{10} + \mathbf{1}*x^9 + \mathbf{0}*x^8 + \mathbf{0}*x^7 + \mathbf{0}*x^6 + \mathbf{1}*x^5 + \mathbf{0}*x^4 + \mathbf{1}*x^3}_{\text{Bitmuster von } G} + \underbrace{\mathbf{1}*x^2 + \mathbf{1}*x^1 + \mathbf{0}*x^0}_{\text{Bitmuster von } R}$$

Berechnung der Prüfbits

Forderung: Übertragene Nachricht *F(x)* soll durch ein sog. *Generator-Polynom P(x)* teilbar sein, das sowohl dem Sender als auch dem Empfänger von *F(x)* bekannt ist:

$$(4) \qquad F(x) = P(x) * Q(x) = G(x) * x^{n-k} + R(x)$$

Solche Nachrichten-Codes heißen *zyklische Codes*.

Gleichung (4) nach *R(x)* auflösen

$$(5) \qquad R(x) = P(x) * Q(x) - G(x) * x^{n-k}$$

– Da wir Polynome über Bits betrachten, die nur Werte 0 und 1 annehmen können, können wir alle Rechenoperationen modulo 2 durchführen

☞ Wirkung von Addition (+) und Subtraktion (–) ist gleich!

☞ $$(6) \qquad R(x) = G(x) * x^{n-k} - P(x) * Q(x)$$

Rechenoperationen modulo 2

Wichtig: Bei CRC-Zeichen wird auf Polynomen über den Werten 0 und 1 gerechnet, und *nicht* auf Binärzahlen (bspw. im Zweierkomplement o.ä. Darstellungen aus Kapitel 5). Daher unterscheiden sich hier auch Addition und Subtraktion auf Polynomen von den Standard-Operationen auf Binärzahlen!

Wertetabellen: ($_{std}$ steht für Standard-Arithmetik auf Binärzahlen, $_2$ für Arithmetik modulo 2)

a	b	a $+_{std}$ b	a $+_2$ b	a $-_{std}$ b	a $-_2$ b	a XOR b
0	0	00	0	00	0	0
0	1	01	1	11	1	1
1	0	01	1	01	1	1
1	1	10	0	00	0	0

Identische Rechenoperationen!

Interpretation

Bedeutung von Gleichung (6)

$$(6) \qquad R(x) = G(x) * x^{n-k} - P(x) * Q(x)$$

– Die Prüfbits *R(x)* sind gleich dem Rest der Division von *G(x)* * x^{n-k} durch das Generator-Polynom *P(x)*!

Analogie: Darstellung des Divisionsrests bei ganzen Zahlen

– Divisionsrest von 42 geteilt durch 4 ist 2, weil
– 40 die nächstkleinere Zahl unterhalb von 42 ist, die glatt durch 4 teilbar ist, und
– die Differenz zwischen 42 und 40 gleich 2 ist

☞ 42 = 4 * 10 + 2 d.h. 2 = 42 – 4 * 10

– **Allgemein:** Rest *r* der Division von *g* durch *p* ist
r = *g* – *p* * *q* mit *r* < *p*

Beispiel zur Polynomdivision

$$(1x^6 + 0x^5 + 1x^4 + 1x^3 + 0x^2 + 0x + 0) : (1x^3 + 1x^2 + 0x + 1) = 1x^3 + 1x^2 + 0x + 0$$

$$\underline{1x^6 + 1x^5 + 0x^4 + 1x^3}$$
$$0x^6 + 1x^5 + 1x^4 + 0x^3 + 0x^2$$
$$\underline{1x^5 + 1x^4 + 0x^3 + 1x^2}$$
$$0x^5 + 0x^4 + 0x^3 + 1x^2 + 0x$$
$$\underline{0x^4 + 0x^3 + 0x^2 + 0x}$$
$$0x^4 + 0x^3 + 1x^2 + 0x + 0$$
$$\underline{0x^3 + 0x^2 + 0x + 0}$$
$$0x^3 + 1x^2 + 0x + 0$$

$G(x) = \mathbf{1011}$

$R(x) = \mathbf{100}$

$F(x) = \mathbf{1011\ 100}$

Beachte: Alle Subtraktionen während der Polynomdivision sind $-_2$!
Tipp: Einfach XOR anwenden.

Reduktion der Polynomdivision auf die Koeffizienten

$$(1 + 0 + 1 + 1 + 0 + 0 + 0) : (1 + 1 + 0 + 1) = 1 + 1 + 0 + 0$$

$$\underline{1 + 1 + 0 + 1}$$

$$0 + 1 + 1 + 0 + 0$$

$$\underline{1 + 1 + 0 + 1}$$

$$0 + 0 + 0 + 1 + 0$$

$$\underline{0 + 0 + 0 + 0}$$

$$0 + 0 + 1 + 0 + 0$$

$$\underline{0 + 0 + 0 + 0}$$

$$0 + 1 + 0 + 0$$

☞ Rest **100**

Überprüfung von Nachrichten auf Korrektheit

Sender verschickt zu übertragende Nachricht *F(x)*
Empfänger wird im Allgemeinen eine gestörte Nachricht *H(x)* erhalten:

$$(7) \qquad H(x) = F(x) + E(x)$$

E(x) heißt *Fehlerpolynom*. Jeder Term von *E(x)* kennzeichnet eine Fehlerstelle.

Erkennung von Fehlern

- *H(x)* nicht durch *P(x)* teilbar ☞ definitiv Übertragungsfehler erkannt.
- *H(x)* durch *P(x)* teilbar
  - ☞ Übertragung fehlerfrei *oder*
  - ☞ Übertragung fehlerhaft mit nicht erkennbarem Fehler (*P(x)* teilt *E(x)*)

Überprüfung von Nachrichten – Beispiel (1)

```
      ↓
(1 0 0 1 1 0 0) : (1 1 0 1) = 1 1 1 1
 1 1 0 1
 0 1 0 0 1
   1 1 0 1
   0 1 0 0 0
     1 1 0 1
     0 1 0 1 0
       1 1 0 1
       0 1 1 1
```

- ☞ Divisionsrest = **111**
- ☞ Übertragung war fehlerhaft

Überprüfung von Nachrichten – Beispiel (2)

```
   ↓       ↓ ↓
(1 1 1 1 1 1 1) : (1 1 0 1) = 1 0 1 1
 1 1 0 1
 0 0 1 0 1
   0 0 0 0
   0 1 0 1 1
     1 1 0 1
     0 1 1 0 1
       1 1 0 1
       0 0 0 0
```

- ☞ Divisionsrest = **0**
- ☞ Übertragung war fehlerfrei, *oder nicht erkannter Fehler*

Erkennbare Fehler – Einzelfehler

Einzelfehler sind Fehler, die in der übertragenen Nachricht genau ein Bit verfälschen.

Satz 1: Ein zyklischer Code, der durch ein Generator-Polynom erzeugt wird, das mehr als einen Term enthält, entdeckt alle Einzelfehler.

Beweis

– Einzelfehler besitzen ein Fehlerpolynom der Form $E(x) = x^i$

– Hat $P(x)$ mehr als einen Term (z.B. $P(x) = x^j + x^k$), so teilt es x^i nicht

Erkennbare Fehler – ungerade Fehlerzahlen (1)

Ungerade Fehlerzahl haben Fehlerpolynome mit ungerader Anzahl v. 1en.

Satz 2: Jedes durch 1 + *x* teilbare Polynom hat eine gerade Termzahl.

Beweis

– Polynom *P* sei durch (1 + *x*) teilbar: $P = \sum a_i x^i * (1 + x)$

$$\Leftrightarrow P = \sum a_i * (x^i + x^{i+1})$$

– Falls nie benachbarte a_i in *P* gleich 1 sind: *P* hat gerade Termzahl

Beispiel: *P* enthält 1 0 ... 0 1 ☞ $P = 1 * (x^j + x^{j+1}) + 1 * (x^k + x^{k+1})$

Position j k a_j a_k

– Falls benachbarte a_i gleich 1 sind: „Auslöschung“ einer geraden Termzahl $P = \cdots + a_i(x^i + \cancel{x^{i+1}}) + a_{i+1}(\cancel{x^{i+1}} + x^{i+2})$

$P = \cdots + a_i(x^i + x^{i+2})$ (wegen Rechnen in mod 2)

☞ Es ergibt sich stets eine gerade Termzahl für *P*

Erkennbare Fehler – ungerade Fehlerzahlen (2)

Ungerade Fehlerzahl haben Fehlerpolynome mit ungerader Anzahl v. 1en.

Satz 2: Jedes durch 1 + *x* teilbare Polynom hat eine gerade Termzahl.

Folgerung

- Mit einem Generator-Polynom *P*, das (1 + *x*) als Faktor enthält, findet man jede ungerade Anzahl von Fehlern.
- Begründung:
  Würde man mit einem solchen Generator-Polynom *nicht* alle ungeraden Fehlerzahlen entdecken, so wäre das Fehlerpolynom *E(x)* durch *P(x)* teilbar.
  Da (1 + *x*) ein Faktor von *P(x)* ist, wäre *E(x)* auch durch (1 + *x*) teilbar.
  Dann müsste laut Satz 2 *E(x)* jedoch eine gerade Termzahl haben, Widerspruch.

Erkennbare Fehler – Burstfehler (1)

Ein **Burstfehler** der Länge *b* ist ein Fehler, bei dem die falschen Symbole den Abstand *b* haben.

Beispiel: $E(x) = x^7 + x^4 + x^3 = 0010011000$.

Per Definition zählt man dies als Abstand $b = 5$, d.h. man zählt die Randsymbole mit.

Satz 3: Mit einem Generator-Polynom *P(x)* vom Grad *n-k* entdeckt man alle Burstfehler der Länge $b \leq n-k$, wenn *P(x)* den Term *x* nicht als Faktor enthält.

Erkennbare Fehler – Burstfehler (2)

Satz 3: Mit einem Generator-Polynom *P(x)* vom Grad *n-k* entdeckt man alle Burstfehler der Länge $b \leq n-k$, wenn *P(x)* den Term *x* nicht als Faktor enthält.

Beweis

– Sei x^i der Term in *E(x)* mit dem kleinsten Exponenten: $E(x) = x^i * E_1(x)$

– *E(x)* ist nicht durch *P(x)* teilbar, wenn beide Terme nicht teilbar sind:

1. x_i ist nicht teilbar, wenn *P(x)* nicht *x* als Faktor enthält.
2. Grad von $E_1(x)$ ist $\leq b-1$, d.h. $\leq n-k-1$
   - ☞ Grad von $E_1(x)$ < Grad von *P(x)*
   - ☞ *P(x)* teilt $E_1(x)$ nicht

Erkennbare Fehler – Burstfehler (3)

Satz 4: Die Anzahl der nicht erkannten Burstfehler der Länge $b > n\text{-}k$ ist, bezogen auf die Gesamtanzahl möglicher Burstfehler:

$$2^{-(n-k-1)}, \text{ wenn } b = n\text{-}k+1$$
$$2^{-(n-k)}, \text{ wenn } b > n\text{-}k+1$$

Beispiel

Generator-Polynom $P(x) = (1 + x^2 + x^{11})\ (1 + x^{11})$ erkennt:

- 1 Burst der Länge $b \leq 22$ (einschl. Einzelfehlern)
- Jede ungerade Anzahl von Fehlern (wegen Faktor $(x^q + 1)$)
- 99,99996% der Bursts der Länge $b = 23$
- 99,99998% der Bursts der Länge $b > 23$

In der Praxis sind viele der vorkommenden Fehler Burstfehler.

Sehr einfaches Beispiel

$P(x) = (1 + x)$, $k = 2$, $n = 3$

Aufgrund der o.a. Sätze

- Erkennung aller Einzelfehler (Sätze 1, 2 und 3)
- Für Burstfehler mit $b = 2$
  - Wahrscheinlichkeit, Fehler nicht zu erkennen: $2^{-(1-1)} = 2^0 = 1$
- Für Burstfehler mit $b = 3$
  - Wahrscheinlichkeit, Fehler nicht zu erkennen: $2^{-(1)} = ½$

Interaktive Berechnung

Unter http://einstein.informatik.uni-oldenburg.de/20911.html

- Simulationsprogramm
- Anleitung
- Theorie von CRC-Zeichen

Anwendung von CRC-Zeichen

In der Praxis werden u.a. folgende Generator-Polynome verwendet

- CRC-5 (USB)
  $x^5 + x^2 + 1$
- CRC-7 (SD/MMC-Card)
  $x^7 + x^3 + 1$
- CRC-16 (HDLC, X.25)
  $x^{16} + x^{12} + x^5 + 1$
- CRC-32 (Ethernet)
  $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Zusammenfassung (1)

Funktionsweise von Ein-/Ausgabe

- I/O geschieht entweder über reservierte E/A-Adressbereiche im Hauptspeicher oder über spezielle Ein-/Ausgabebefehle
- Datenübergabe mit unidirektionalem Timing (synchrone Busse): Kommunikationspartner verlassen sich darauf, dass Partner innerhalb festgelegter Zeit reagieren
- Datenübergabe mit bidirektionalem Timing (asynchrone Busse): Kommunikationspartner bestätigen per *acknowledgment*; Spezialfall *Fully-Interlocked Handshaking*
- Synchronisation: *busy waiting*, *polling*, *interrupts*, DMA

Zusammenfassung (2)

***Point-to-Point* Verbindungen**

- RS-232
  - Startbit, Datenbits, Paritätsbit, Stoppbits
  - Übertragungsrichtungen Voll-Duplex, Halb-Duplex, Simplex
  - Physikalischer Aufbau
  - Flusskontrolle: per Software (XON/XOFF), per Hardware (RTS/CTS)
- LPT

Busse

- Zugangsregelung: *Daisy Chain*, zentrale vs. verteilte Arbitrierung
- Interne PCI-basierte Busse: Aufbau, Konfiguration, heutige Standards
- Externe Busse (IDE, SCSI, USB): Aufbau und Verkabelung, Arbitrierung, Übertragungsmodi, Codierungen (NRZ, NRZI, *Bit-Stuffing*)

Zusammenfassung (3)

Fehlererkennung mit CRC-Zeichen

- Aufbau, Berechnung und Prüfung von CRC-Zeichen
- Erkennbare und nicht erkennbare Fehler