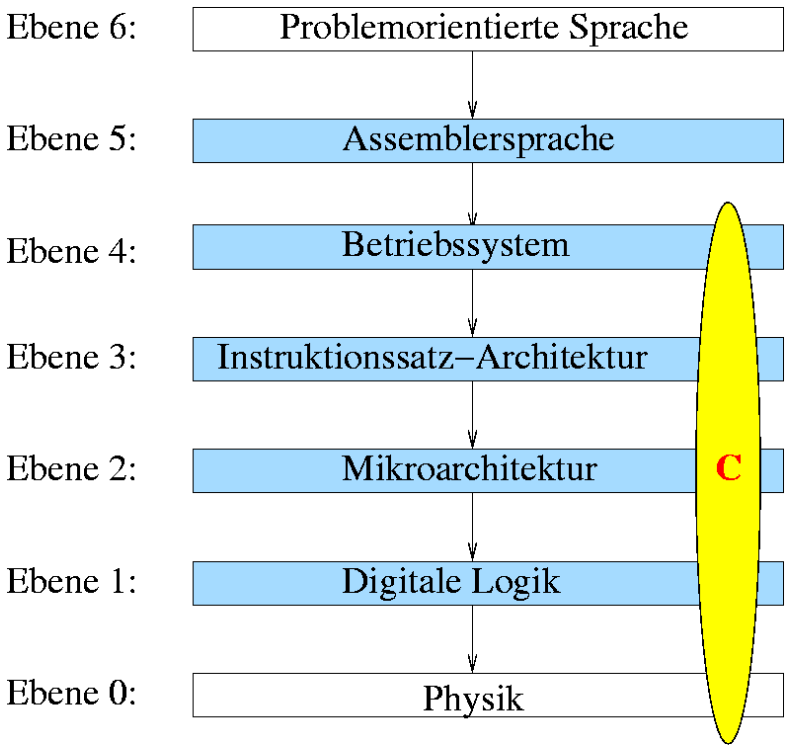
C Ein-Ausgabekonzepte und Bussysteme

1. Ein-/Ausgabe
2. Strategien der Ein-/Ausgabe
3. Techniken der Datenübertragung
4. Punkt-zu-Punkt Schnittstellen
5. Direct Memory Access (DMA)
6. Bussysteme
7. Gerätetreiber

C Ein-/Ausgabekonzepte und Bussysteme

- Einordnung in das Schichtenmodell:



1 Ein-/Ausgabe

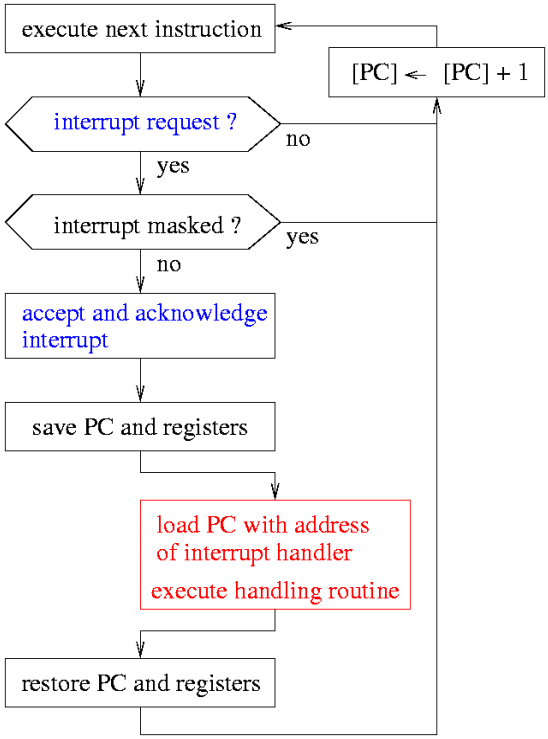
- zur Realisierung einer Ein-/Ausgabe werden benötigt:
  - eine **Strategie** der Ein-/Ausgabe
  - eine nach Möglichkeit **standardisierte Schnittstelle** (*Interface*) zur Ein-/Ausgabe, z.B. seriell (RS-232) oder parallel (EPP):
    - Definition eines *Übertragungsprotokoll*
    - Definition von Steckern/Buchsen und Kabeln
    - Hardwareunterstützung durch entsprechenden E/A-Baustein (*I/O Controller*), ggf. separate *Schnittstellenkarte*
  - **E/A-Geräte** bzw. **Peripheriegeräte**
    - zur Umwandlung von elektrischen Signalen in eine verwertbare physikalische Form ...
      (z.B. Bildschirm, Drucker)
    - ... und umgekehrt
      (z.B. Maus, Tastatur, Scanner)

2 Strategien der Ein-/Ausgabe

- **programmierte Ein-/Ausgabe**:
  - ein laufendes Programm (z.B. ein Anwendungsprogramm) legt explizit fest, *wann* eine Ein-/Ausgabe erfolgt
  - Falls der Ausgabebaustein noch nicht bereit ist, kann CPU in einer Warteschleife auf dessen Bereitschaft warten (*Busy Waiting*)
  - Eingabebausteine können zu definierten Zeitpunkten abgefragt werden, ob neue Eingabedaten anliegen (*Polling*)
- **Unterbrechungen** (*Interrupts*):
  - bei Eintreffen neuer Daten kann ein Eingabebaustein eine Unterbrechung anfordern (*Interrupt Request*)
  - Ausgabebaustein kann durch eine Unterbrechungsanforderung der CPU die erneute Bereitschaft signalisieren
  - sobald möglich, bestätigt CPU den Interrupt (*Interrupt Acknowledge*) und startet eine zugehörige Behandlungsroutine

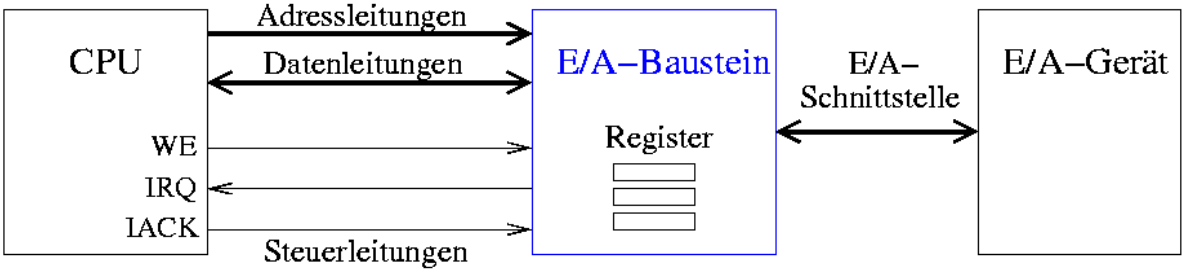
2 Strategien der Ein-/Ausgabe (2)

- Unterbrechungen erlauben eine schnelle Reaktion der CPU auf E/A-Ereignisse
  - allgemeiner **Ablauf** bei einer Unterbrechung:
  - Unterbrechung nur **nach** der Ausführung einer Instruktion möglich
  - fast alle Prozessoren bieten die Möglichkeit, Unterbrechungen zu verbieten (d.h. zu **maskieren**) bzw. wieder zu gestatten
  - verschiedene **Prioritätsstufen** bei mehreren E/A-Geräten sinnvoll



2 Strategien der Ein-/Ausgabe (3)

- Kopplung von CPU und E/A-Geräten erfolgt über spezielle **E/A-Bausteine** am Systembus:
  - Auswahl eines E/A-Bausteins über Adressleitungen
  - Datentransfer von/zu E/A-Baustein über Datenleitungen
  - Steuerleitungen z.B. für Richtungsauswahl, Unterbrechungsanforderung und Bestätigung einer Unterbrechung
  - zur Kommunikation mit CPU bietet E/A-Bausteine einige interne Register an



2 Strategien der Ein-/Ausgabe (4)

- ein E/A-Baustein verfügt über drei Arten interner Register:
  - **Kontrollregister**
    - zur Initialisierung und Parameterwahl durch CPU
  - **Datenregister**
    - zur Zwischenpufferung von einzulesenden oder auszugebenden Daten (nötig, da E/A-Geräte zumeist langsamer als CPU sind und zudem asynchron zur CPU arbeiten)
  - **Statusregister**
    - zum Austausch von Statusinformationen zwischen E/A-Baustein und CPU (z.B. Verfügbarkeit eines neuen Eingabewertes, oder Ausgabegerät hat Zeichen aus Datenregister gelesen)
    - E/A-Baustein setzt/löscht entsprechende Bits im Statusregister selbständig
    - bei Verwendung der Strategie *Polling* wird Statusregister in einer Schleife abgefragt

2 Strategien der Ein-/Ausgabe (5)

- zwei Möglichkeiten für den Zugriff der CPU auf die internen Register der E/A-Bausteine:
  - **speicherbezogene Adressierung** (*Memory-Mapped I/O*):
    - Register sind an bestimme Speicheradressen in den physikalischen Adressraum der CPU eingeblendet
    - E/A-Adressen müssen vom Caching ausgenommen werden!
    - Zugriff mit normalen `load`- und `store`-Befehlen, Zugriffsschutz nur über Speicherverwaltung
  - **separate Ein-/Ausgabeadressen**
    - separater, oft kleiner E/A-Adressraum
    - spezielle, i.a. privilegierte Befehle (z.B. `in`, `out`) für Lesen und Schreiben im E/A-Adressraum
    - zusätzliche Steuerleitung IO/$\overline{\text{Memory}}$ zur Selektion von Speicher- oder E/A-Adressraum

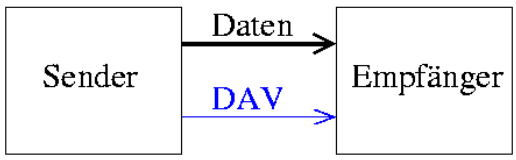
3 Techniken der Datenübertragung

- zwischen Sender (z.B. Rechner) und Empfänger (z.B. Peripheriegerät) existieren Leitungen für die Übertragung von **Daten** und **Steuersignalen**
- drei verschiedene Übertragungsprotokolle:
  1) **Open-Loop** Datenübertagung
     - keine Rückmeldung bei der Übertragung
  2) **Closed-Loop** Datenübertagung
     - Quittierung der Datenannahme über Steuersignal
     - auch als *Handshaking* bezeichnet
  3) **Fully-Interlocked** Datenübertragung
     - Quittierung sowohl der Datenannahme als auch aller Steuersignale
     - auch als *Fully-Interlocked Handshaking* bezeichnet

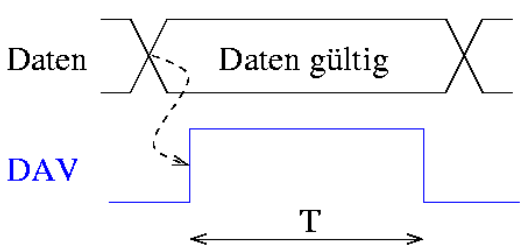
3.1 Open-Loop Datenübertragung

- eine Steuerleitung:
  DAV = *Data Valid*
- Sender legt ein Datum auf Datenleitungen und setzt für eine Zeitdauer T das Signal **DAV = 1**
- Empfänger muss die Daten übernehmen, solange **DAV = 1** ist
- Sender und Empfänger müssen Zeitdauer T vereinbart haben

Signale:



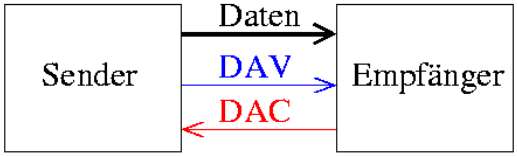
Zeitdiagramm:



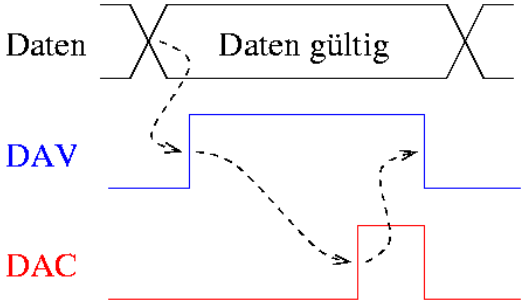
3.2 Closed-Loop Datenübertragung

- zwei Steuerleitungen:
  DAV = *Data Valid*
  DAC = *Data Accepted*
- nach Empfang eines Datums bestätigt Empfänger dies mit **DAC = 1**
- Sender nimmt Daten von Datenleitungen zurück, sobald er **DAC = 1** empfangen hat
- **einfache Flusskontrolle** (langsamer Empfänger kann Sender anhalten)
- ggf. Abbruch nach Wartezeit (**Timeout**)

Signale:



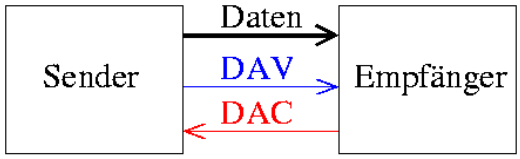
Zeitdiagramm:



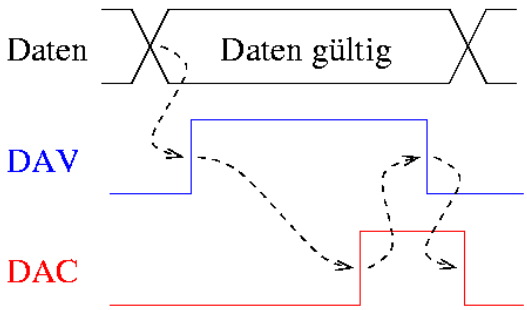
3.3 Fully-Interlocked Datenübertragung

- zwei Steuerleitungen, wie bei Closed-Loop
- Unterschiede zu Closed-Loop:
  - Empfänger hält **DAC = 1**, bis der Sender **DAV = 0** setzt
  - Empfänger bestätigt somit, die Deaktivierung des Signals DAV gesehen zu haben
  - Sender gibt erst neue Daten aus, wenn **DAC = 0** vorliegt
- **verbesserte Flußkontrolle** (Empfänger kann auch nach Empfang von Daten den Sender aufhalten)

Signale:



Zeitdiagramm:



4 Punkt-zu-Punkt Schnittstellen

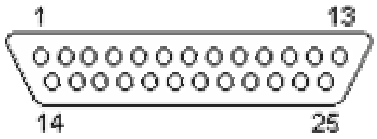
- Bei Verwendung einer **Punkt-zu-Punkt Schnittstelle** lässt sich nur ein einzelnes Peripheriegerät an eine Schnittstelle anschließen
- Gegenteil: **Bussysteme** (→ Abschnitt 6)
- Punkt-zu-Punkt Schnittstelle
  - **unidirektional**
    (es gibt einen Sender und einen Empfänger)
  - **bidirektional**
    (beide Seiten können als Sender oder Empfänger arbeiten)
- Beispiele für standardisierte Punkt-zu-Punkt Schnittstellen
  1) parallele Schnittstelle (IEEE 1284)
  2) serielle Schnittstelle (RS-232)

4.1 Parallele Schnittstelle

- **8-Bit** Datenworte werden **parallel** vom Sender zum Empfänger übertragen
- früher: *Centronics* Schnittstelle
  - nur unidirektionaler Betrieb: Ausgabe von Daten an ein Peripheriegerät
  - Transferrate bis zu 150 KByte/s
- heute: IEEE 1284 Schnittstelle, unterstützt u.a. 3 Modi:
  - *Enhanced Parallel Port* (**EPP**) gestattet einen bidirektionalen Betrieb: bei Anschluß eines Druckers kann dieser z.B. auch Statusmeldungen (kein Papier, Toner leer, ...) übertragen
  - *Extended Capability Port* (**ECP**) leistet zusätzlich Datenkomprimierung und Auswahl mehrerer logischer Geräte
  - *Standard Parallel Port* (**SPP**) definiert einen unidirektionalen Betrieb (kompatibel zur *Centronics* Schnittstelle)
  - mit EPP ist eine Transferrate bis zu ca. 2 MByte/s möglich

4.1 Parallele Schnittstelle (2)

- **25-polige** Steckerverbindung (Sub-D):
- Bedeutung der Steckerpins bei Einsatz im **SPP-Modus**:

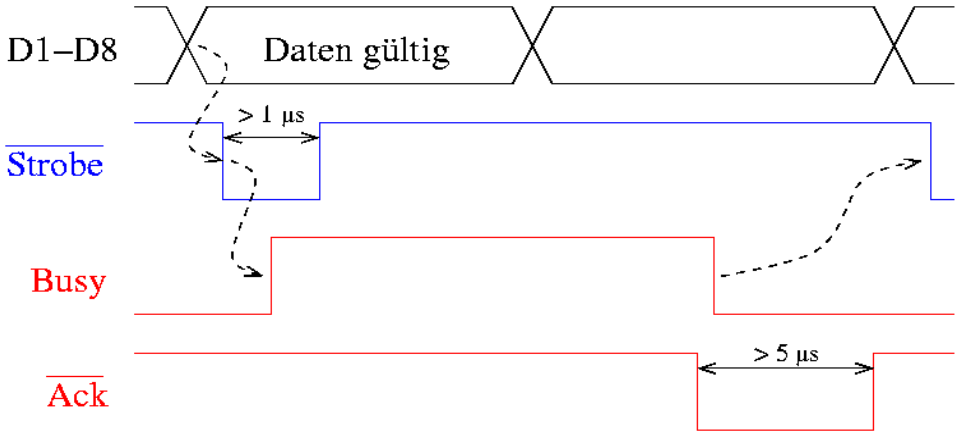


1 /Strobe	A	Daten gültig	10 /Ack	E	Bestätigung
2 D1 (LSB)	A	Datenbit 0	11 Busy	E	Endgerät beschäftigt
3 D2	A	Datenbit 1	12 PE	E	Papierende
4 D3	A	...	13 Select	E	Endgerät offline/online
5 D4	A		14 /Auto Feed	A	autom. Zeilenvorschub
6 D5	A		15 /Error	E	Fehler
7 D6	A		16 /Reset	A	Endgerät zurücksetzen
8 D7	A	Datenbit 6	17 /Select In	A	Auswahlleitung
9 D8 (MSB)	A	Datenbit 7	18-25 GND	–	Abschirmung

- einige Steuersignale (mit / markiert) mit sind „*low active*“
- heute z.T. auch 36-polige Steckerverbindung üblich

4.1 Parallele Schnittstelle (3)

- Zeitdiagramm für die Ausgabe eines Bytes zum Peripheriegerät:



- TTL-Pegel, d.h. +5V für logisch 1, 0V für logisch 0
- Closed-Loop Datenübertragung mit Steuersignalen /Strobe und Busy; das Signal /ACK liefert eine zusätzliche Bestätigung und kann zur Unterbrechungsanforderung verwendet werden

4.1 Parallele Schnittstelle (4)

- Bedeutung der Steckerpins bei Einsatz im **EPP-Modus**:

1 /Write	A	**Richtung**	10 /Intr	E	Interrupt
2 D1 (LSB)	**E/A**	Datenbit 0	11 /Wait	E	Endgerät bereit
3 D2	**E/A**	Datenbit 1	12 UserDef0	E	**frei belegbar**
4 D3	**E/A**	...	13 UserDef1	E	**frei belegbar**
5 D4	**E/A**		14 /DStrobe	A	Daten gültig
6 D5	**E/A**		15 UserDef2	E	**frei belegbar**
7 D6	**E/A**		16 /Reset	A	Endgerät zurücksetzen
8 D7	**E/A**	Datenbit 6	17 /AStrobe	A	Adressen gültig
9 D8 (MSB)	**E/A**	Datenbit 7	18-25 GND	–	Abschirmung

- Datenleitungen nun **bidirektional**, zusätzliche Steuerleitung zur **Richtungsangabe** (/Write = 0 : Ausgabe, /Write = 1: Eingabe)
- EPP stellt eine Multifunktionsschnittstelle dar
  (⇒ Adressübertragung, benutzerdefinierbare statt druckerspezifische Signale)

4.1 Parallele Schnittstelle (5)

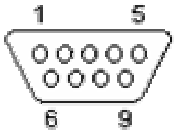
- Übertragungsprotokoll bei **SPP** in Software:
  - CPU muss Status des Empfängers (Busy) mittels Statusregister auswerten,
  - ein Byte in das Datenregister des E/A-Bausteins schreiben,
  - Strobe-Signal mittels bestimmter Bits im Kontrollregister aktivieren und ca. 1 µs später wieder deaktivieren, und
  - Bestätigung des Empfängers (Ack) im Statusregister abwarten
- Übertragungsprotokoll bei **EPP** in Hardware:
  - CPU schreibt/liest lediglich ein Byte in/aus Datenregister
- **EPP** stellt heute eine einfache, preiswerte Schnittstelle für Geräte mit relativ hoher Datenübertragungsrate dar:
  - Drucker und Scanner
  - Zip-Laufwerk
  - ...
- wird jedoch von anderen Schnittstellen (z.B. **USB**) verdrängt

4.2 Serielle Schnittstelle

- serielle, **asynchrone** Punkt-zu-Punkt Verbindung
  - jedes zu übertragene Zeichen wird mit Start-, Stop- und ggf. Paritätsbit in ein Paket verpackt, das asynchron (ungetaktet) und bitseriell über eine Leitung übertragen wird
  - mit minimal 3 Leitungen bereits funktionsfähig (⇒ geringe Kabelkosten)
  - verbindet **DTE** (*Data Terminal Equipment*, z.B. PC) **mit DCE** (*Data Communication Equipment*, z.B. Modem),
  - kann auch **DTE mit DTE** verbinden
  - Sender und Empfänger müssen sich für die Übertragung eines jeden Pakets erneut synchronisieren
- Standard: **RS-232** (*Electronic Industries Association*, 1969)
  - auch als V.24 bezeichnet (CCITT)

4.2 Serielle Schnittstelle (2)

- **9-polige** Steckerverbindung (Sub-D):
- Pinbelegung aus Sicht eines DTE:
  (mit E = Eingangsleitung, A = Ausgangsleitung)

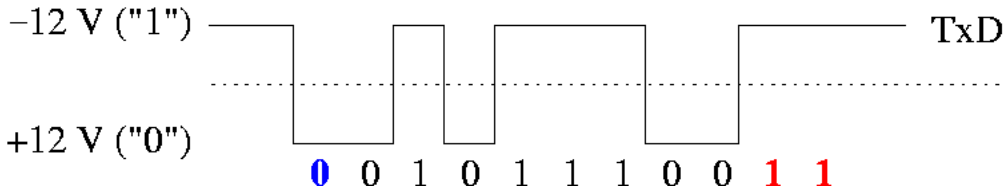


1 DCD	E	Trägererkennung (*Data Carrier Detect*)
2 RxD	**E**	**Empfangsleitung** (*Receive Data*)
3 TxD	**A**	**Sendeleitung** (*Transmit Data*)
4 DTR	A	DTE ist betriebsbereit (*Data Terminal Ready*)
5 GND	–	Masse, Abschirmung (*Ground*)
6 DSR	E	DCE ist betriebsbereit (*Data Set Ready*)
7 RTS	**A**	**DTE ist bereit zum Datenaustausch** (*Request to Send*)
8 CTS	**E**	**DCE ist bereit zum Datenaustausch** (*Clear to Send*)
9 RI	E	ankommender Anruf (*Ring Indicator*)

- auch **25-polige** Steckerverbindung (zusätzliche Steuersignale)

4.2 Serielle Schnittstelle (3)

- **Spannungspegel** auf den Leitungen:
  - Logisch 1: $< -3V$ (typisch $-12V$, maximal $-15V$)
  - Logisch 0: $> +3V$ (typisch $+12V$, maximal $+15V$)
  - hoher Störabstand $\Rightarrow$ maximale Leitungslänge: ca. 20m
- serielle Datenübertragung
  - Ruhezustand: TxD = 1
  - **1 Startbit** (TxD = 0)
  - 5,6,7 oder 8 Datenbits (niedrigstwertiges Datenbit zuerst)
  - **1** oder **2 Stopbits** (TxD = 1)
- Beispiel: serielle Übertragung des Byte $3A_{16} = 00111010_2$



4.2 Serielle Schnittstelle (4)

- optional zusätzliches **Paritätsbit :**
  - **gerade Parität** (*Even Parity*): Summe der 1-Datenbits ist gerade
  - **ungerade Parität** (*Odd Parity*): Summe der 1-Datenbits ist ungerade
- typische Parameterwahl:
  - 8 Datenbits, keine Parität (*None Parity*), 1 Stopbit $\Rightarrow$ Kurzform: 8N1
- Übertragungsgeschwindigkeit wird in **Baud** angegeben:
  - Baud-Rate = Anzahl Bit je Sekunde
  - typische Baud-Raten: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
  - bei einer Baudrate von 115200 dauert jedes Bit 8.6 µs
- Auf Sende- und Empfangsseite sind Baud-Rate sowie Parameter (Parität, Anzahl Daten- und Stop-Bits) **identisch** einzustellen

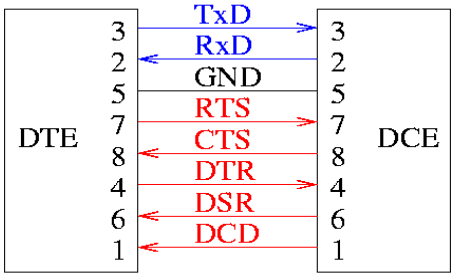
4.2 Serielle Schnittstelle (5)

- keine Empfangsbestätigung $\Rightarrow$ Open-Loop Datenübertragung
- Anpassung unterschiedlicher Geschwindigkeiten von DTE und DCE mittels **Flußkontrolle**:
  - entweder mittels **Hardware** (*RTS/CTS protocol*):
    - wenn **DCE** keine Daten mehr aufnehmen kann, erhält DTE auf der Eingangsleitung **CTS** den Pegel 0
    - wenn **DTE** keine Daten mehr aufnehmen kann, wechselt DTE den Pegel auf der Ausgangsleitung **RTS** von 1 auf 0
  - oder mittels **Software** (*Xon/Xoff protocol*):
    - Senden des Steuerzeichens *Xoff* (Ctrl-S, ASCII 19h) stoppt den Datentransport in die entgegengesetzte Richtung
    - Senden des Steuerzeichens *Xon* (Ctrl-Q, ASCII 17h) hebt den Stop wieder auf
    - beide Steuerzeichen dürfen nicht im Datenstrom vorkommen!

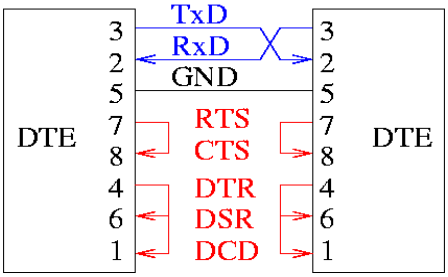
4.2 Serielle Schnittstelle (6)

- verschiedene Kabel zur Verbindung von Geräten mittels RS-232:

Modem–Kabel (*straight cable*):
(für RTS/CTS Flußkontrolle)



Nullmodem–Kabel:
(für Xon/Xoff Flußkontrolle)



Nullmodem–Kabel:
(für RTS/CTS Flußkontrolle)