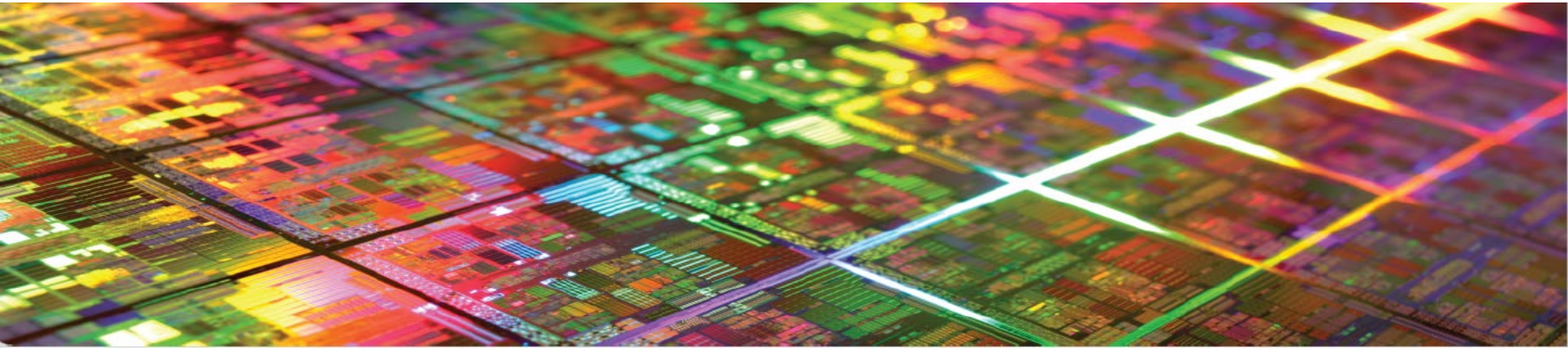
Karlsruher Institut für Technologie

Rechnerorganisation

Prof. Dr. Wolfgang Karl

Vorlesung im Wintersemester 2025/2026 – Foliensatz: RO25-FS15

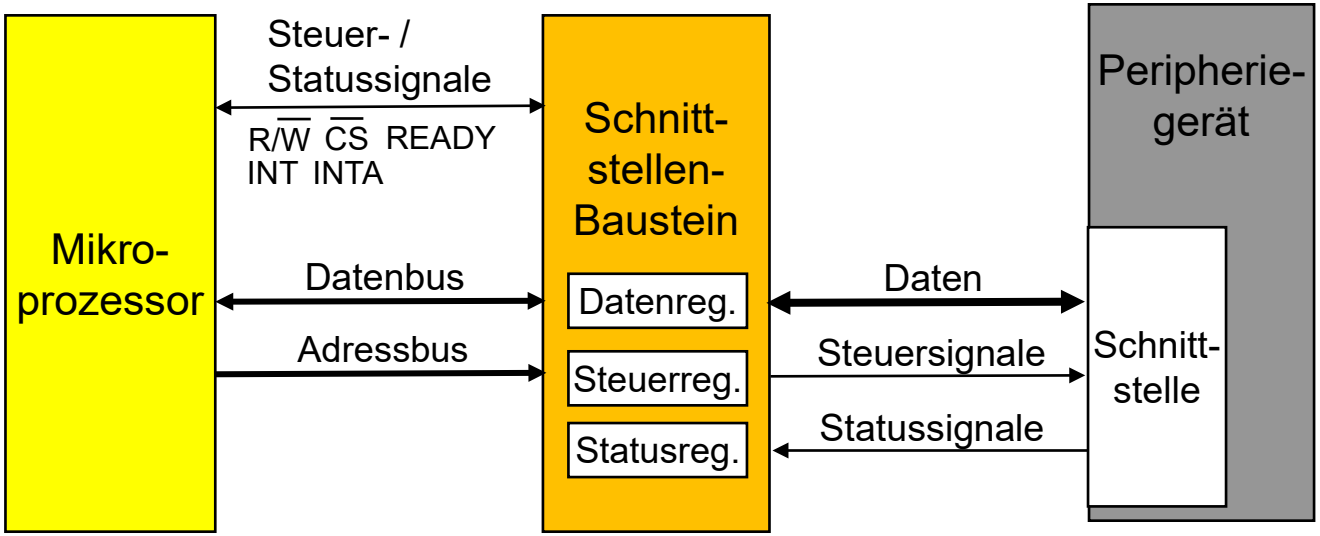


Kapitel 8
Ein-Ausgabewerk

- Peripheriemerkmale
- **Schnittstellenbausteine**
- Ein- / Ausgabeverfahren
- Adressierung der Systemkomponenten
- Universal Serial Bus

Schnittstellenbausteine

Anbindung eines peripheren Geräts



Anbindung des Peripherie-geräts:

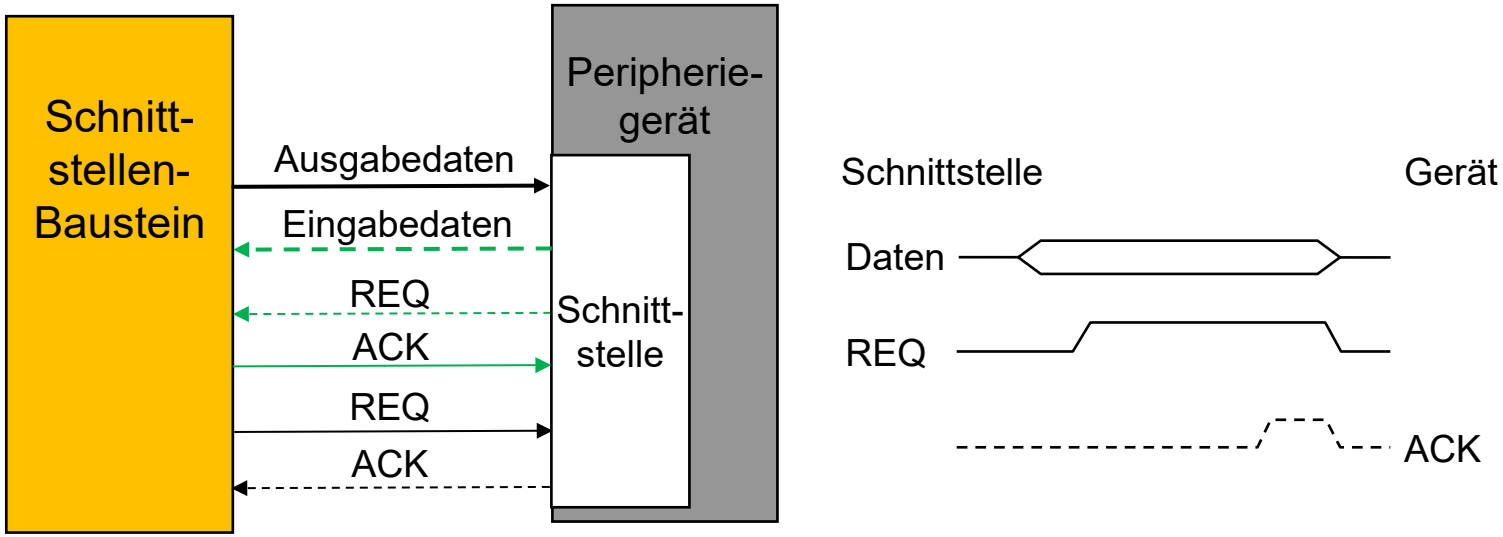
Datenleitungen: unidirektional, bidirektional, parallel/seriell, analog

Steuersignale: Ansteuern des Peripheriegeräts (Ein-/Ausschalten), Synchronisation der Übertragung, …

Statussignale: Informationen über den Zustand des Peripheriegeräts (Peripheriegerät bereit, Störung, …)

Schnittstellenbausteine

- **Anbindung eines peripheren Geräts**
  - **Synchronisation der Datenübertragung**
    - Erfolgt in Hardware- oder mit Software
    - Beispiel für eine Synchronisation in Hardware



Schnittstellenbausteine

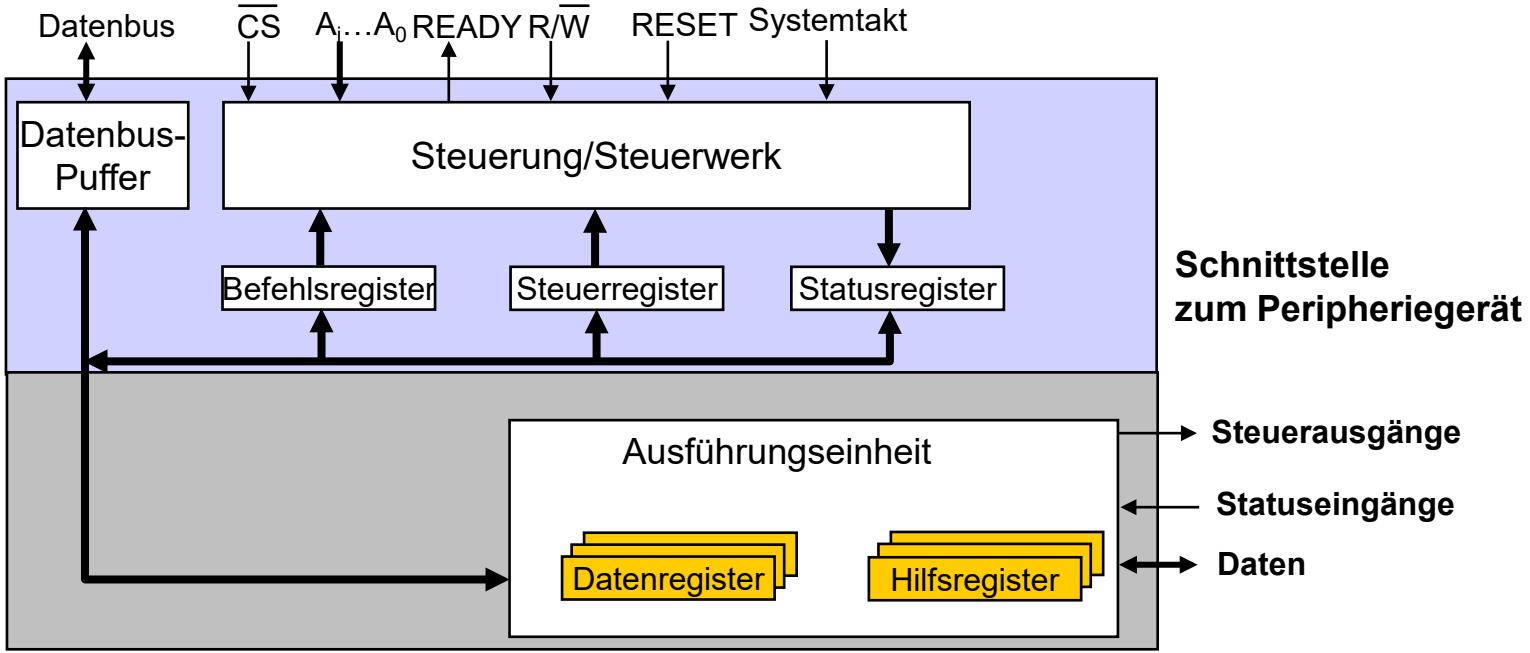
- **Anbindung eines peripheren Geräts**
  - **Synchronisation der Datenübertragung**
    - Beispiel für eine Synchronisation in Hardware
      - **Vollduplex-Betrieb (im Beispiel)**
        - Getrennte Datenwege für beide Übertragungsrichtungen
      - **Halbduplex-Betrieb**
        - Daten werden bidirektional über dieselben Leitungen ausgetauscht
      - **Für jede Richtung eine Request-Leitung (REQ) und eine Acknowledge-Leitung (ACK)**
        - Schnittstellenbaustein legt das Datum auf seine Datenleitungen; durch das REQ-Signal zeigt er, dass er die Übernahme der Daten durch das Peripheriegerät erwartet;
        - Das Gerät zeigt die Übernahme der Daten durch das ACK-Signal.

Schnittstellenbausteine

- **Kommunikation mit dem Prozessor**
  - Ein-/Ausgabekommando
    - Schnittstelle empfängt ein E-/A-Kommando vom Prozessor, z. B. über Steuerleitungen
  - Adressierung der Schnittstellenkomponenten
  - Datentransfer
  - Austausch von Statusinformationen
  - Synchronisation mit dem Schnittstellenbaustein

Schnittstellenbausteine

- **Grundlegender Aufbau eines Schnittstellenbausteins (E/A-Schnittstelle)**



Schnittstellenbausteine

- **Grundlegender Aufbau eines Schnittstellenbausteins (E/A-Schnittstelle)**
  - **Steuerwerk:**
    - Steuerung der internen Komponenten (Register, Multiplexer, …) und Datenpfade
    - Schnittstelle zum Prozessor
    - Register zur Bausteinprogrammierung:
      - Statusregister (Bausteinstatus),
      - Steuerregister (Betriebsart),
      - Befehlsregister (aktuelle Operation)
  - **Ausführungseinheit:**
    - Stellt die spezifischen Bausteinfunktionen zur Verfügung
    - Verschiedene Daten- und Hilfsregister (je nach Funktion)
    - Schnittstelle zum Peripheriegerät

Kapitel 8
Ein-Ausgabewerk

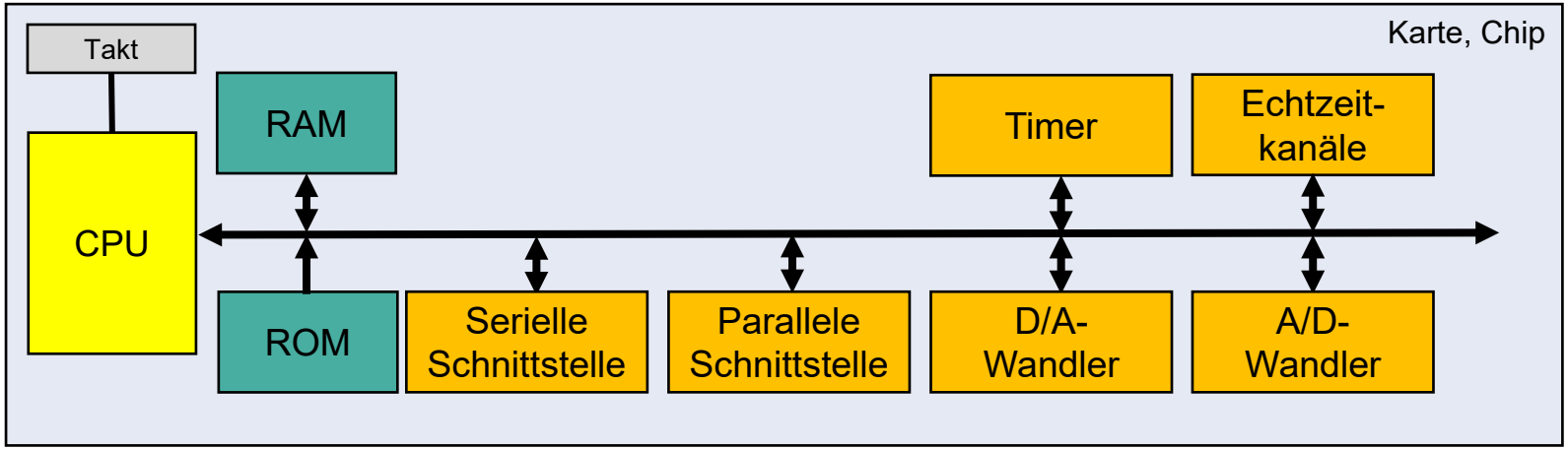
- Peripheriemerkmale
- Schnittstellenbausteine
- **Ein- / Ausgabeverfahren**
- Adressierung der Systemkomponenten
- Universal Serial Bus

Ein- / Ausgabewerk

- **Grundlegende Verfahren für die Ausführung von E/A-Aufgaben**
  - **Prozessorgesteuerte Ein-/Ausgabe**
    - Die Ausführung der Steuerungsaufgaben für einen Ein- oder Ausgabevorgang sowie der Datenübertragung obliegt dem Prozessor.
      - Ausgabe: Übertragen von Daten aus dem Hauptspeicher an die E/A-Schnittstelle
      - Eingabe: Übertragen der Daten von der E/A-Schnittstelle in den Hauptspeicher
    - **Unterscheidung bezüglich der Synchronisation**
      - **Programmierte Ein-/Ausgabe**
      - **Interrupt-gesteuerte Ein-Ausgabe**
  - **Direct Memory Access (DMA)**
    - Die Ausführung der Steueraufgaben für einen Ein- oder Ausgabevorgang sowie der Datenübertragung wird vom Prozessor an den DMA-Controller delegiert.

Ein- / Ausgabewerk

- Grundlegende Verfahren für die Ausführung von E/A-Aufgaben
  - Prozessorgesteuerte Ein-/Ausgabe
    - Programmierte E/A (programmed I/O, PIO)



Ein- / Ausgabewerk

- **Programmierte E/A (programmed I/O, PIO)**
  - Der Prozessor führt ein Programm aus, mit welchem der Ein-/Ausgabevorgang gesteuert wird.
  - Prozessor gibt dabei
    - die Adresse, die die entsprechende E-/A-Schnittstelle identifiziert, und
    - ein E-/A-Kommando, z. B. eine Eingabe oder eine Ausgabe, aus.
  - E/A-Schnittstelle führt die entsprechenden Aktionen aus und setzt die Bits im Statusregister
  - Prozessor überprüft, ob das E-/A-Kommando von dem Schnittstellenbaustein ausgeführt worden ist
    - Synchronisation durch Busy-Waiting
    - Zyklische Abfrage des Statusregisters des Schnittstellenbausteins bis durch den Eintrag des peripheren Geräts das Ende der Bearbeitung signalisiert wird
    - Zeitaufwendiger Prozess, da der Prozessor ständig den Status abfragen muss

Ein- / Ausgabewerk

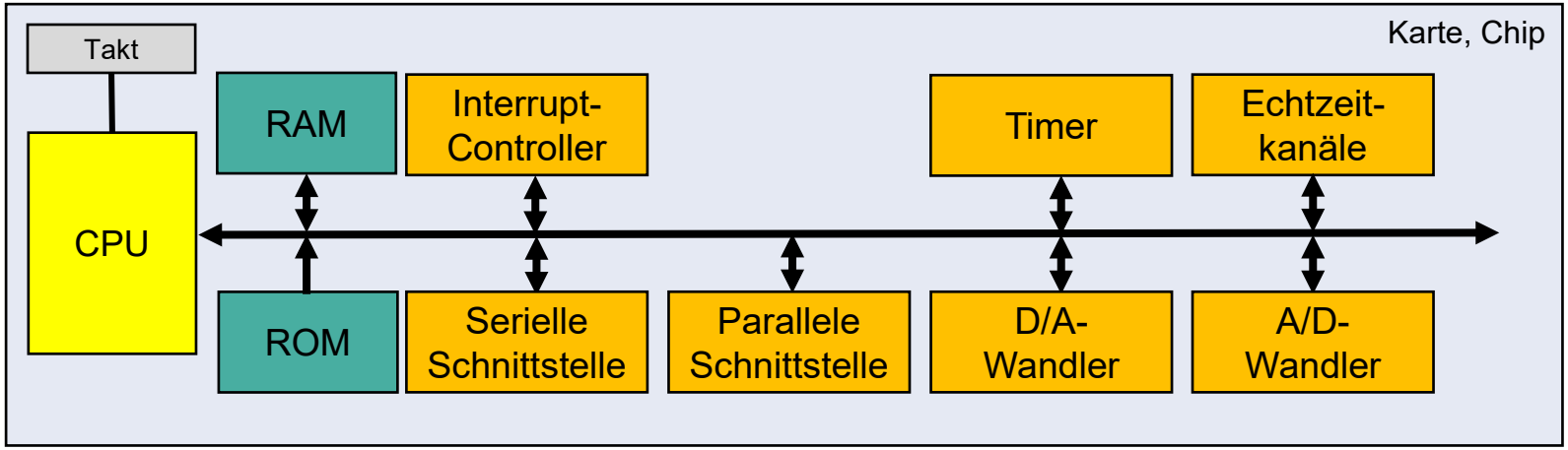
- **Programmierte E/A (programmed I/O, PIO)**
  - **Arten von E-/A-Aufgaben**
    - **Steuer-Kommandos**
      - Aktivierung eines peripheren Geräts
      - Ausführungsanweisungen für das periphere Gerät
      - Zugeschnitten auf den speziellen Typ des peripheren Geräts
    - **Test-Kommandos**
      - Abfrage von Statusbedingungen der E-/A-Schnittstelle und des peripheren Geräts
      - Ist das Gerät eingeschaltet?
      - Ist die E-/A-Operation ausgeführt?
      - Ist ein Fehler aufgetreten?

Ein- / Ausgabewerk

- **Programmierte E/A (programmed I/O, PIO)**
  - **Arten von E-/A-Aufgaben**
    - **Lese-Kommandos**
      - Daten, die von dem peripheren Gerät in den Datenpuffer der E/A-Schnittstelle geladen worden sind, können vom Prozessor über den Datenbus gelesen werden
    - **Schreib-Kommando**
      - Daten werden von der E-/A-Schnittstelle vom Datenbus genommen und an das periphere Gerät übertragen

Ein- / Ausgabewerk

- Grundlegende Verfahren für die Ausführung von E/A-Aufgaben
  - Prozessorgesteuerte Ein-/Ausgabe
    - Interrupt-gesteuerte E/A

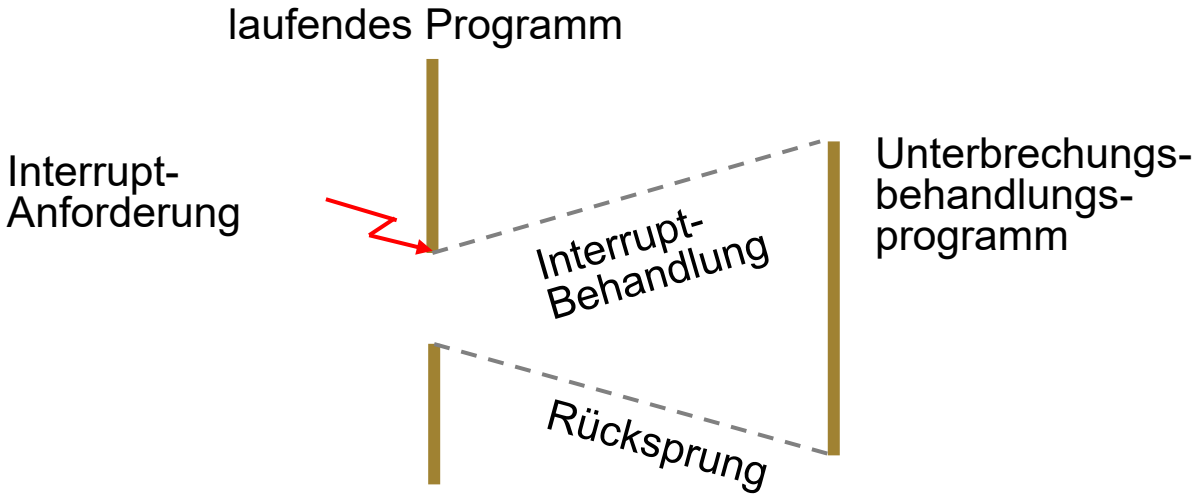


Ein- / Ausgabewerk

- **Interrupt-gesteuerte E/A**
  - Die Synchronisation zwischen Prozessor und Schnittstellenbaustein erfolgt durch eine Programmunterbrechung.
  - Der Prozessor sendet ein Ein- oder ein Ausgabe-Kommando an den Schnittstellenbaustein und fährt mit der Ausführung seines Programms fort.
  - Wenn die Schnittstelle für den Datenaustausch mit dem Prozessor bereit ist, sendet diese eine Interrupt-Anforderung an den Prozessor.
  - Wenn der Prozessor der Interrupt-Anforderung stattgibt, führt er im Rahmen einer Unterbrechungsbehandlung den Ein-/Ausgabevorgang durch.
  - Nach der Datenübertragung nimmt der Prozessor die Ausführung des unterbrochenen Programms wieder auf.

Ein- / Ausgabewerk

- **Interrupt-gesteuerte E/A**
  - **Interrupt-Behandlung**



Ein- / Ausgabewerk

- **Interrupt-gesteuerte E/A**
  - **Interrupt-Behandlung**
    - Wenn der Mikroprozessor einer Unterbrechung stattgibt, wird das laufende Programm unterbrochen.
    - Der Programmkontext wird gerettet.
    - Das Unterbrechungsbehandlungsprogramms (Interrupt Service Routine) wird aufgerufen.
      - Die Vektornummer wird von Interrupt-Controller bzw. Schnittstellenbaustein geliefert.
      - Die Vektornummer ist ein Index in Vektortabelle.
      - Die Einträge in der Vektortabelle sind die Einsprungadressen der Unterbrechungsbehandlungsprogramme.
    - Nach Abarbeitung des Unterbrechungsbehandlungsprogramms wird der Kontext des unterbrochenen Programms wieder hergestellt
    - Fortführung des unterbrochenen Programms.

Ein- / Ausgabewerk

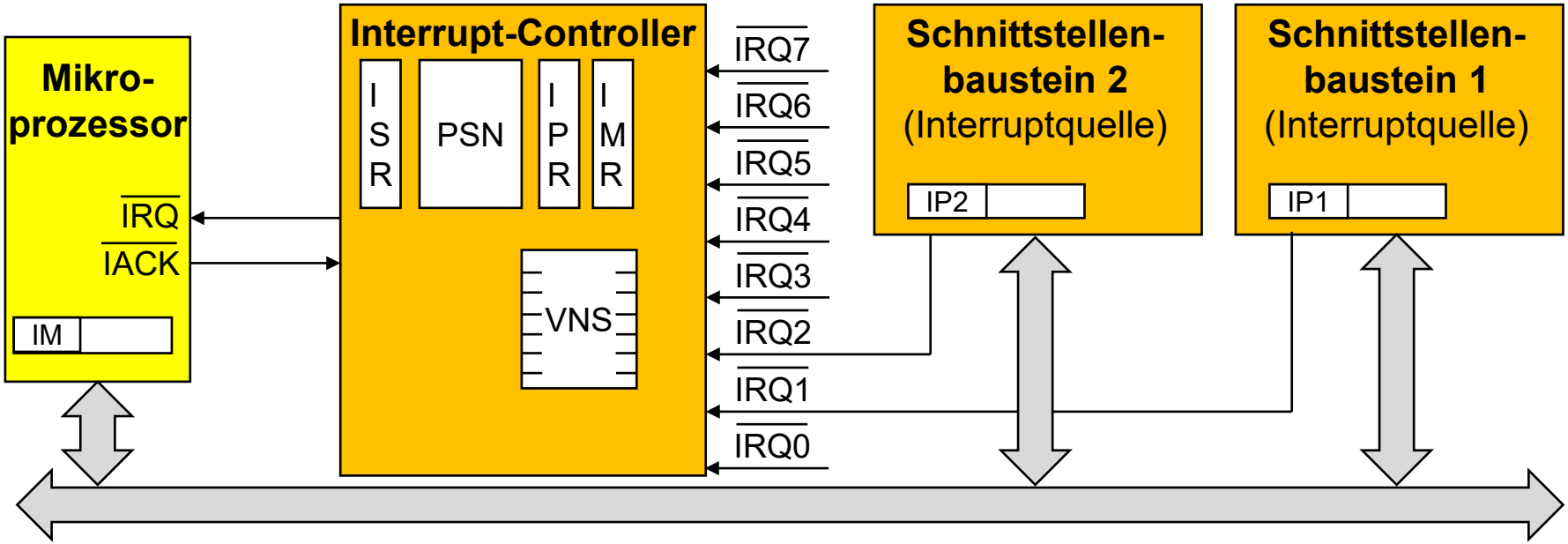
- **Interrupt-gesteuerte E/A**
  - **Beispiel: Eingabe aus Sicht des Schnittstellenbausteins:**
    - Bei einer Eingabe erhält der Schnittstellenbaustein vom Prozessor ein Lese-Kommando.
    - Die Daten werden vom peripheren Gerät an den Schnittstellenbaustein geliefert und dort im Datenpuffer abgelegt
    - Schnittstellenbaustein zeigt seine Bereitschaft zur Datenübertragung mit Hilfe eines Interrupt-Signals ($\overline{\text{IRQ}}$-Signal an Prozessor) an.
    - Der Schnittstellenbaustein wartet bis die Daten dann vom Prozessor angefordert werden.
    - Wenn die Bereitschaft vom Prozessor angezeigt wird, legt der Schnittstellenbaustein die Daten auf den Datenbus.

Ein- / Ausgabewerk

- **Interrupt-gesteuerte E/A**
  - **Beispiel: Eingabe aus Sicht des Prozessors**
    - Der Prozessor sendet eine Lese-Kommando an den Schnittstellenbaustein.
    - Der Prozessor wartet nicht auf die Bereitschaft des Schnittstellenbausteins, sondern kann die Zeit, bis die Schnittstelle zur Datenübertragung bereit ist, für andere Aufgaben nutzen.
    - Der Prozessor erkennt, ob eine Interrupt-Anforderung anliegt und prüft, ob der Interrupt-Anforderung stattgegeben werden kann.
    - Wenn der Prozessor die Interrupt-Anforderung ($\overline{\text{IACK}}$-Signal an den Schnittstellenbaustein) akzeptiert, reagiert er mit einer Programmunterbrechung und verzweigt an ein Unterbrechungsbehandlungsprogramm, das die E/A-Befehlsfolge enthält und den Ein-/Ausgabevorgang durchführt.
      - Der Prozessor liest die Daten vom Schnittstellenbaustein und legt sie im Speicher ab.

Ein- / Ausgabewerk

- **Interrupt-gesteuerte E/A**
  - **Beispiel: Systemaufbau mit Interrupt-Controller**



Ein- / Ausgabewerk

- **Interrupt-gesteuerte E/A**
  - **Beispiel: Systemaufbau mit Interrupt-Controller**
    - Interrupt-Anforderung wird durch Schnittstellenbaustein mit Hilfe eines $\overline{\text{IRQi}}$-Signals an Interrupt-Controller gemeldet
      - Im Schnittstellenbaustein i weist ein Statusbit IPi (Interrupt-Pending) eine Interrupt-Anforderung aus.
      - Hieraus wird das Anforderungssignal $\overline{\text{IRQi}}$=0 gebildet
    - Mehrere Interrupt-Quellen können Interrupt-Anforderungen an Interrupt-Controller melden
    - Interrupt-Controller wählt bei gleichzeitigem Anliegen von mehreren Interrupt-Anforderungen gemäß einer festgelegten Priorisierung eine Interrupt-Quelle aus
    - Interrupt-Controller sendet Interrupt-Anforderung $\overline{\text{IRQ}}$=0 an Prozessor

Ein- / Ausgabewerk

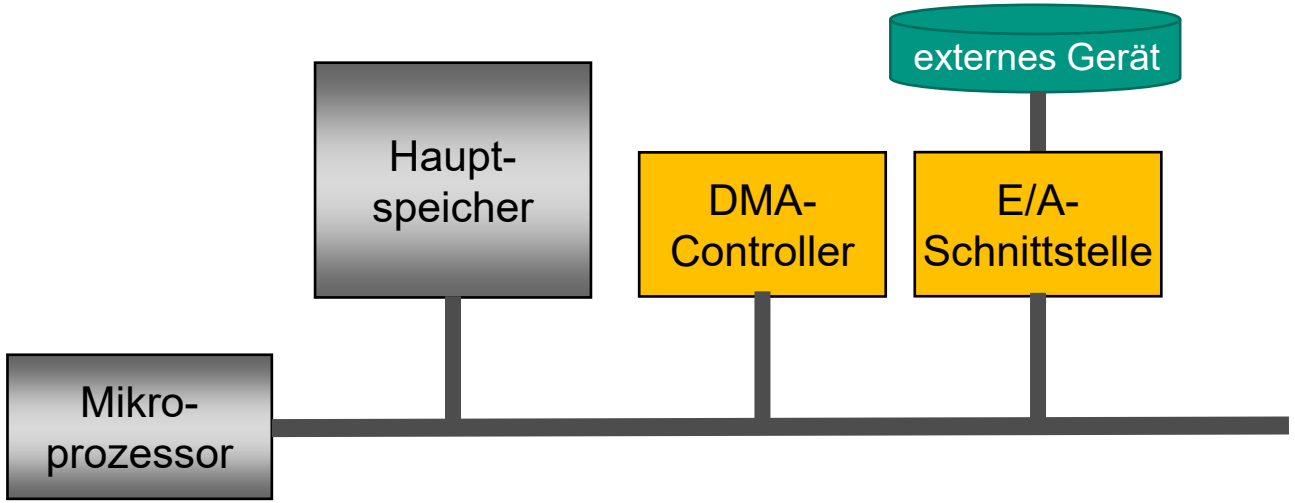
- **Interrupt-gesteuerte E/A**
  - **Beispiel: Systemaufbau mit Interrupt-Controller**
    - **Aufbau Interrupt-Controller**
      - An den Interrupt-Eingängen $\overline{\text{IRQ0}}$ bis $\overline{\text{IRQ7}}$ können bis zu acht Schnittstellenbausteine (Interrupt-Quellen) angeschlossen werden
      - Im Interrupt-Controller werden diesen acht Eingängen unterschiedliche Prioritäten (Prioritätsebenen) zugeordnet
      - Interrupt-Pending-Register (IPR) speichert die an den Eingängen anliegenden Interrupt-Anforderungen
      - Interrupt-Masken-Register (IMR) kann bestimmte Interrupt-Anforderungen blockieren
      - Interrupt-Service-Register (ISR) speichert die bereits in Bearbeitung befindlichen Anforderungen

Ein- / Ausgabewerk

- **Interrupt-gesteuerte E/A**
  - **Beispiel: Systemaufbau mit Interrupt-Controller**
    - **Aufbau Interrupt-Controller**
      - IPR und IMR wirken auf das Prioritätenschaltnetz (PSN):
        - Ein Eintrag im ISR blockiert sämtliche im IPR gespeicherten und noch nicht in Bearbeitung befindlichen Anforderungen gleicher und niedriger Prioritäten.
        - Unterbrechbarkeit durch Anforderungen mit höherer Priorität möglich:
      - Wenn einer Interrupt-Anforderung stattgegeben wird, wird die zu dieser Anforderung gehörige Vektornummer auf den Datenbus ausgegeben und an den Prozessor geschickt.
      - Mit Hilfe der Vektornummer kann der Prozessor die Adresse des Unterbrechungsbehandlungsprogramms (Interrupt Service Routine) zur Durchführung der Ein-/Ausgabe finden.
      - Der Vektornummernspeicher (VNS) speichert die den Interrupt-Eingängen zugeordneten Vektornummern.

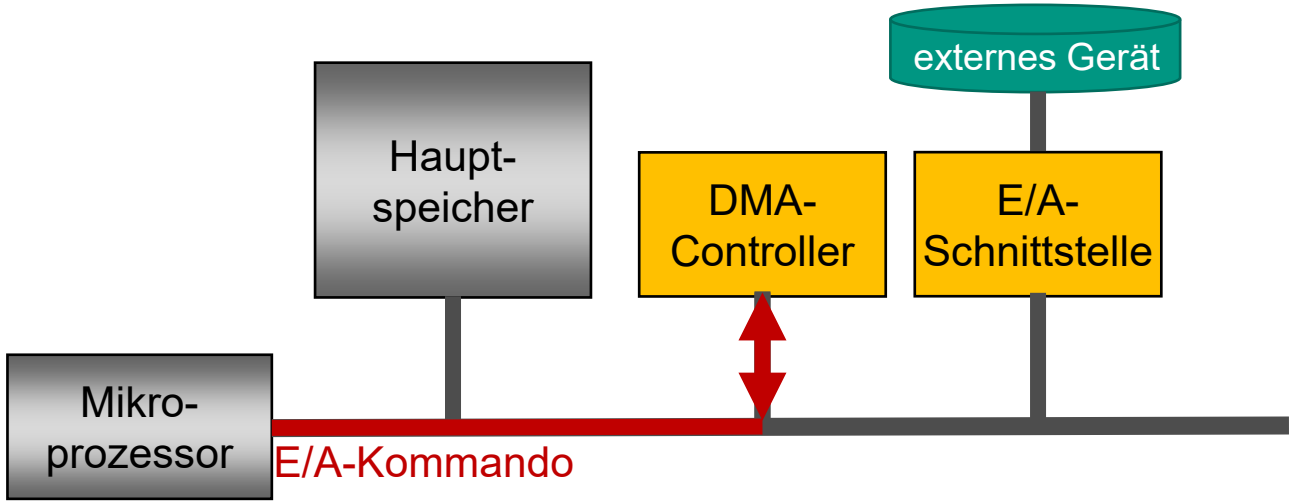
Ein- / Ausgabewerk

- **Grundlegende Verfahren für die Ausführung von E/A-Aufgaben**
  - **Direkter Speicherzugriff (Direct Memory Access, DMA)**
    - Datenübertragung zwischen Hauptspeicher und Hintergrundspeicher erfolgt unabhängig vom Prozessor
    - **Einsatz eines DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**



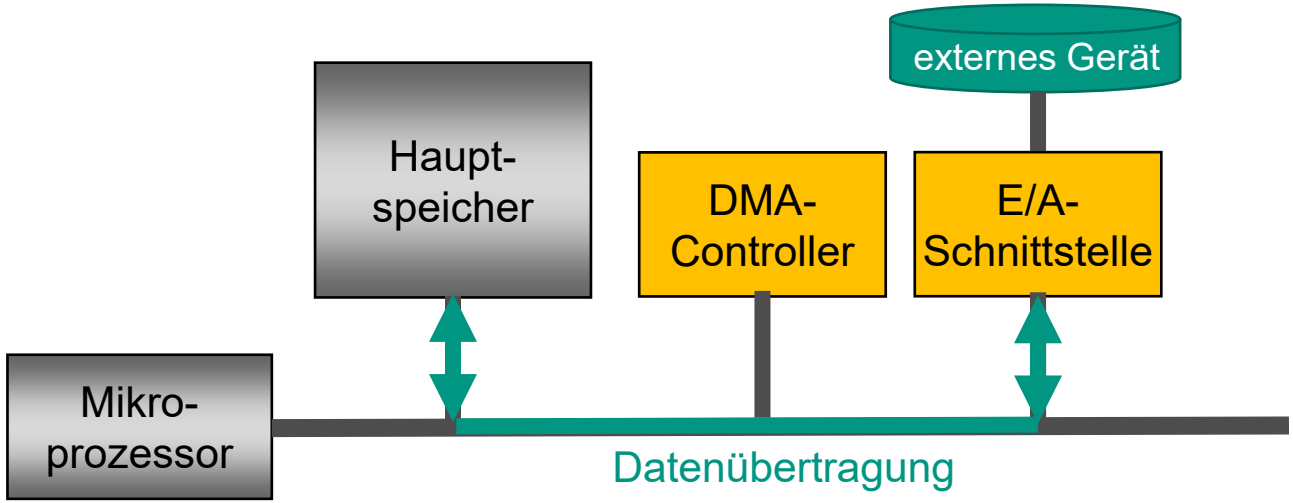
Ein- / Ausgabewerk

- **Direkter Speicherzugriff (Direct Memory Access, DMA)**
  - **Einsatz eines DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
    - **Schritte:** E/A-Kommando von Prozessor an DMA-Controller
      - Prozessor delegiert die Ein-/Ausgabeaufgabe an DMA-Controller



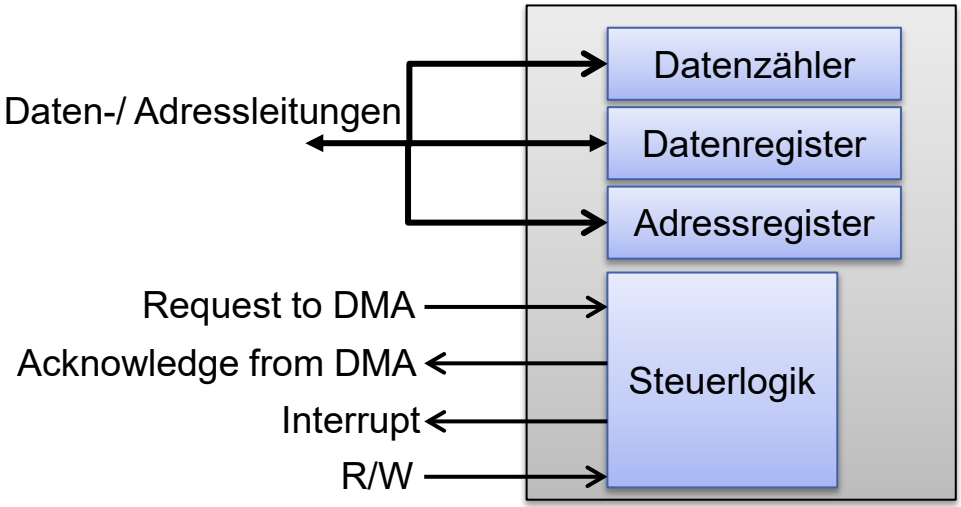
Ein- / Ausgabewerk

- **Direkter Speicherzugriff (Direct Memory Access, DMA)**
  - **Einsatz eines DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
    - **Schritte:** Blockweise Datenübertragung mit jeweils einem Wort zu einem Zeitpunkt vom und zum Hauptspeicher



Ein- / Ausgabewerk

- **DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
  - **Aufbau**



Ein- / Ausgabewerk

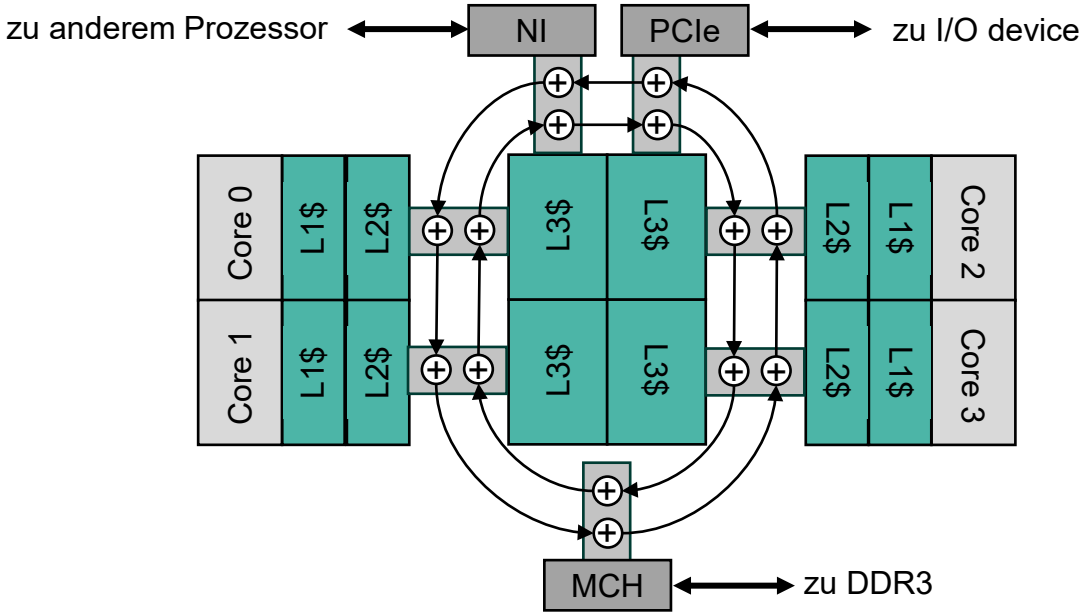
- **DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
  - **Arbeitsweise**
    - Wenn der Prozessor einen Block von Daten lesen oder schreiben möchte, initiiert er ein entsprechendes Kommando an die DMA-Einheit und sendet folgende Informationen:
      - Die Lese- oder Schreib-Anforderung über die Lese-/Schreib-Steuerleitung;
      - Die Adresse der E/A-Schnittstelle, die betroffen ist, über die Datenleitungen;
      - Die Startadresse im Speicher, ab der gelesen oder geschrieben werden soll, wird über die Datenleitungen übertragen und im Adressregister der DMA-Einheit gespeichert.
      - Die Anzahl der zu lesenden oder zu schreibenden Wörter, die ebenfalls über die Datenleitungen geliefert und im Datenzähler der DMA-Einheit abgelegt wird.

Ein- / Ausgabewerk

- **DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
  - **Arbeitsweise**
    - Der Prozessor hat die Datenübertragung an DMA-Einheit delegiert und fährt unabhängig mit seiner Programmausführung fort.
    - Die DMA-Einheit überträgt den Datenblock direkt vom bzw. zum Hauptspeicher.
    - Die DMA-Einheit informiert den Prozessor nach der Datenübertragung, z. B. über ein Interrupt-Signal.

Ein- / Ausgabewerk

- **DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
  - **DMA-Zugriff bei Multicore-Prozessoren**



L3$: gemeinsamer Cache
L1$, L2$: privater Cache für jeden Kern

NI: Netzwerkschnittstelle
PCIe: PCIexpress (E/A-Schnittstelle)
MCH: Memory Controller Hub

Ein- / Ausgabewerk

- **DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
  - **DMA-Zugriff bei Multicore-Prozessoren**
    - Ein bidirektionaler Ring verbindet die Prozessorkerne, den L3$, PCIe und MCH miteinander.
      - Jede am bidirektionalen Ring angeschlossene Komponente wird als Ring Agent betrachtet und enthält jeweils eine Logik zur Implementierung der Ring Agenten Logik.
      - Die Ring Agenten kooperieren über ein verteiltes Protokoll, um Zugriff auf den Ring anzufordern und in Form von Zeitscheiben zugeteilt zu bekommen.
      - Wenn ein Ring Agent Daten verschicken möchte wählt die Logik die Richtung mit dem kürzesten Pfad zum Empfänger und überträgt die Daten, wenn eine Zeitscheibe verfügbar ist.

Ein- / Ausgabewerk

- **DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
  - **DMA-Zugriff bei Multicore-Prozessoren**
    - DMA-Operation:
      - Daten werden zwischen Hauptspeicher und einem E-/A-Gerät mit Hilfe der Verbindungsstruktur ausgetauscht.
      - Eine E-A/-Treiberroutine, die auf einem Core ausgeführt wird, sendet ein E-/A-Kommando zusammen mit der Adresse und der Größe des Puffers im Hauptspeicher, an den PCIe-Controller (E-/A-Schnittstelle). Der Puffer enthält oder empfängt die zu übertragenden Daten.
      - Der E-/A-Controller stößt eine Lese-Anforderung an, die an den MCH gesendet wird. Der MCH greift auf die Daten im DDR3-Speicher zu und legt sie für die E-/A-Schnittstelle auf den Ring.
      - In ähnlicher Weise werden ankommende Daten von der E-/A-Schnittstelle an den MCH geliefert. Cache-Zeilen mit den betroffenen aktualisierten Speicherstellen müssen im L3$ invalidiert werden.

Ein- / Ausgabewerk

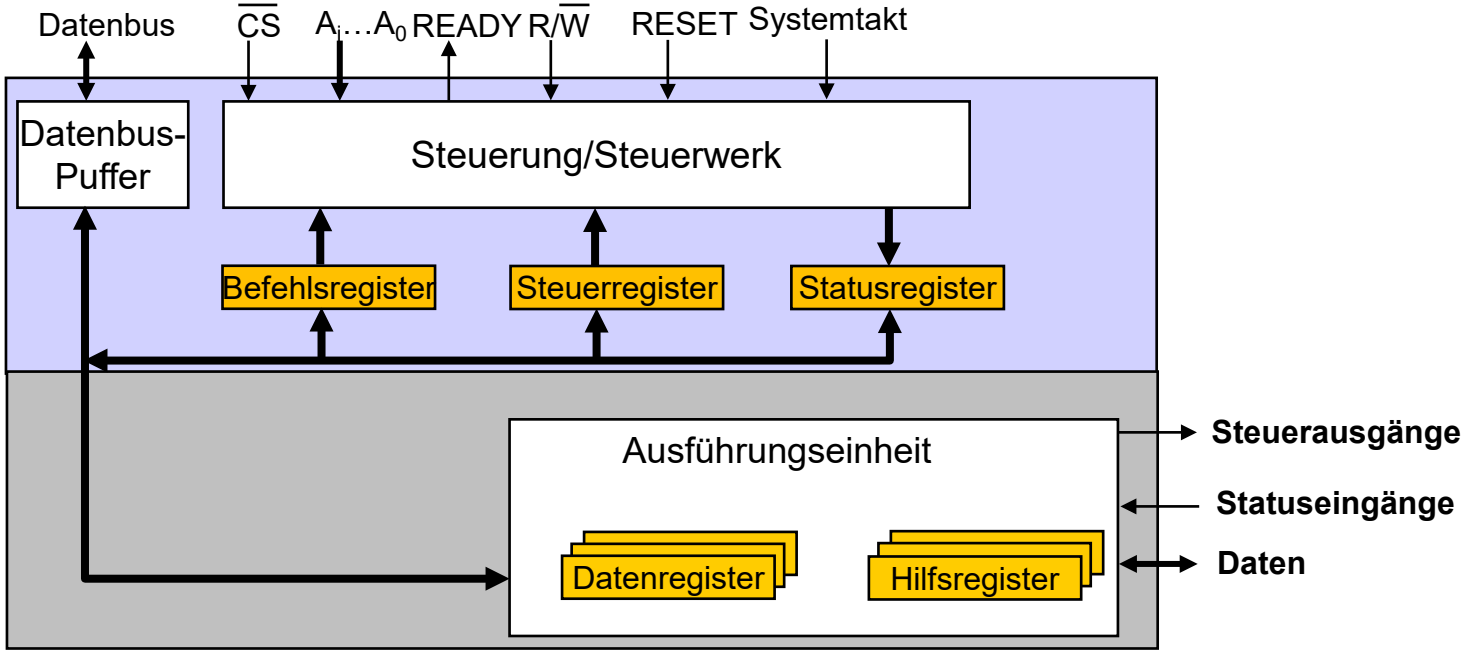
- **DMA-Controllers (DMA-Steuerbaustein, DMA-Einheit):**
  - **DMA-Zugriff auf Last-Level Cache (L3$)**
    - DMA-Operation
      - Bei einer Ausgabe stößt die E-/A-Schnittstelle eine Lese-Anforderung an. Der MCH prüft, ob die angeforderten Daten im L3$ enthalten sind.
      - In diesem Fall werden die Daten direkt vom L3$ an die E-/A-Schnittstelle. Es ist kein Hauptspeicherzugriff notwendig.
      - Wenn die E-/A-Schnittstelle eine Lese-Anforderung hat, werden die Daten im L3$ entfernt.
      - Die Eingabe erfolgt nicht in den L3$ sondern direkt in den Hauptspeicher.

Kapitel 8
Ein-Ausgabewerk

- Peripheriemerkmale
- Schnittstellenbausteine
- Ein- / Ausgabeverfahren
- **Adressierung der Systemkomponenten**
- Universal Serial Bus

Ein- / Ausgabewerk

- **Adressierung der Systemkomponenten**
  - **Wie werden die Komponenten eines Schnittstellenbausteins angesprochen?**



Ein- / Ausgabewerk

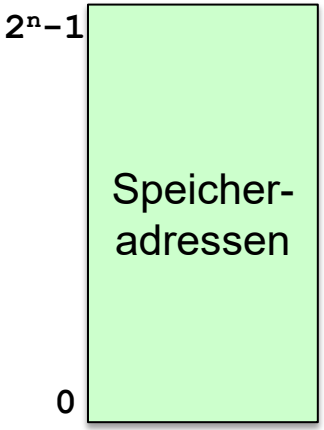
- **Adressierung der Systemkomponenten**
  - Jeder programmierbare Schnittstellenbaustein erscheint dem Prozessor wie ein kleiner Satz von Registern (**I/O Ports**), die über einen zusammenhängenden Block von Adressen (**Port-Adressen**) angesprochen werden können.

  - **Adressierungstechniken**
    - **Isolierte Adressierung (Isolated IO):**
      - zwei getrennte Adressräume für Speicher und Ein-/Ausgabe;
      - die Auswahl des Adressraumes erfolgt durch ein zusätzliches Signal (memory/input-output: M/$\overline{IO}$); z. B. bei Intel 80x86-Prozessoren;

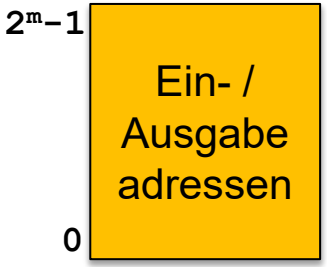
    - **Speicherbezogene Adressierung (Memory Mapped I/O):**
      - ein gemeinsamer Adressraum für Speicher und Schnittstellenbausteine;

Ein- / Ausgabewerk

- **Adressierung der Systemkomponenten**
  - **Isolierte Adressierung (Isolated IO)**
    - Eigener Adressraum für den Hauptspeicher (**n** Speicheradressen)
    - Eigener Adressraum für die Ein-/Ausgabe (**m** Ports)
    - **Beispiel: Adressvergabe für n=32 und m=16**



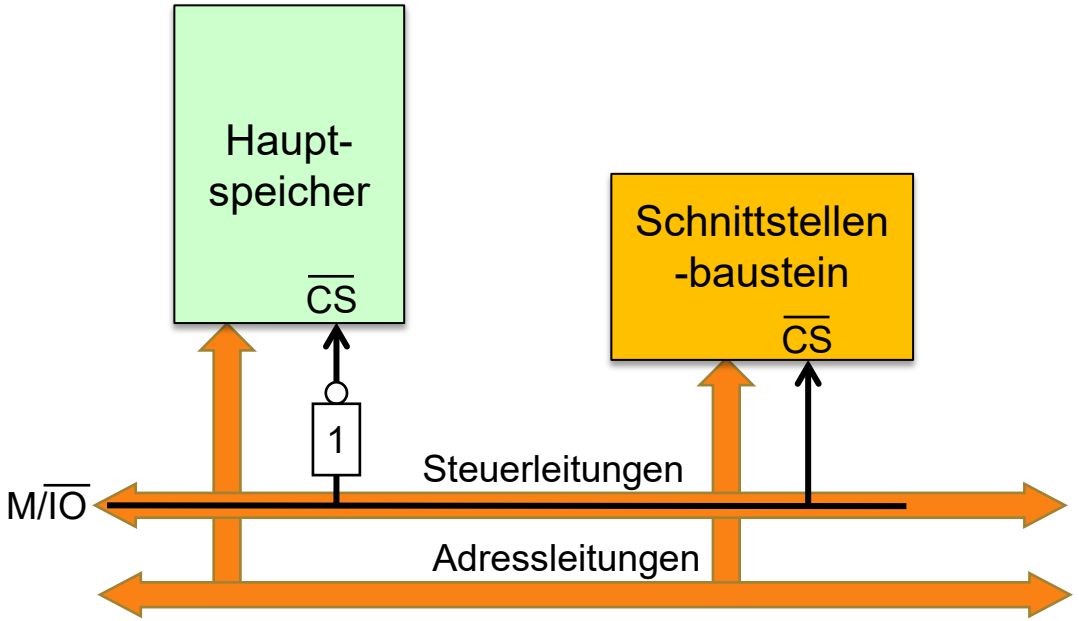
Der Adressraum für den Hauptspeicher besteht aus 4G (2^{32}) Speicheradressen



Der I/O-Adressraum besteht aus 64K (2^{16}) adressierbaren 8-Bit Ports

Ein- / Ausgabewerk

- **Adressierung der Systemkomponenten**
  - **Isolierte Adressierung (Isolated IO)**



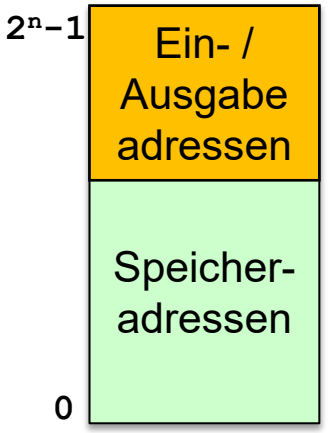
Der Datentransport wird mit speziellen Ein-/Ausgabebefehlen (**`in`**, **`out`**) durchgeführt, wobei die Adresse des Ports als Operand im Befehl steht.

M/$\overline{IO}$ zeigt an, ob es sich um eine Speicheradresse oder um eine Adresse des Schnittstellen-bausteins handelt.

$\overline{CS}$: Chip select

Ein- / Ausgabewerk

- **Adressierung der Systemkomponenten**
  - **Speicherbezogene Adressierung (memory mapped IO)**
    - Der Adressraum ist in einen Bereich für Speicher und einen Bereich für die Ein-/Ausgabe-Einheiten aufgeteilt.
    - Die Register der Schnittstellenbausteine werden genauso wie Speicherzellen adressiert.
      - Es sind keine speziellen Ein-/Ausgabebefehle notwendig.
      - Die Steuerleitung M/$\overline{IO}$ entfällt.



Kapitel 8
Ein- / Ausgabewerk

- Peripheriemerkmale
- Schnittstellenbausteine
- Ein- / Ausgabeverfahren
- Adressierung der Systemkomponenten
- **Universal Serial Bus**

Ein- / Ausgabewerk

- **Universal Serial Bus (USB)**
  - Datenübertragungssystem zum Anschluss von peripheren Geräten an den Computer
  - Standardisierungsgremium: **https://www.usb.org/**
  - Ziel: Vereinheitlichung von Steckverbindungen
    - Unterschiedliche Stecker zum Anschluss von peripheren Geräten an den Computer werden durch USB ersetzt
    - Verschiedene USB-Stecker und Buchsen

Diverse USB-1.0-/2.0-Stecker

USB 3.1 Typ-C auf Typ-A Kabel 10Gbps PD 60W

https://commons.wikimedia.org/wiki/File:USB_3.1_Gen2_Typ-C_to_Typ-A_cable_10Gbps_PD_60W.jpg#filelinks

Universal Serial Bus (USB)

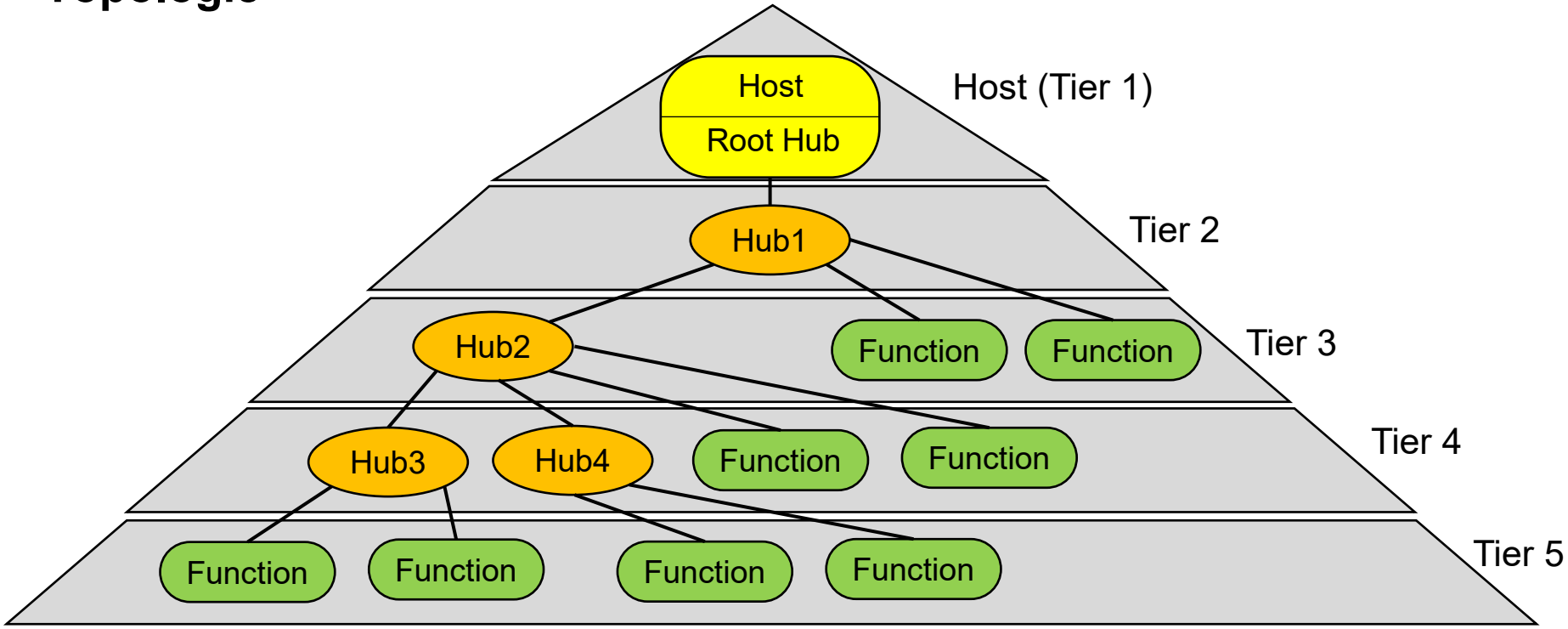
- **Allgemeine Eigenschaften**
  - **Datenübertragungsraten**
    - USB 1.0: 1,5 Mbits/s (Low Speed, Spezifikation 1996)
    - USB 1.0: 12 Mbits/s (Full Speed, Spezifikation 1996)
    - USB 2.0: 480 Mbits/s (Hi Speed Spezifikation 2000)
    - USB 3.1: 10 Gbits/s (SuperSpeed, Spezifikation 2014)
    - USB 3.2: 20 Gbits/s (SuperSpeed+, Spezifikation 2017)
    - …

Universal Serial Bus (USB)

- **Allgemeine Eigenschaften**
  - Mit USB ist eine Stromversorgung von Geräten möglich.
  - **Hot Plug-in**
    - Geräte können im laufenden Betrieb an den Computer angeschlossen und entfernt werden.
      - Die Geräte und deren Eigenschaften werden automatisch erkannt.

Universal Serial Bus (USB)

Topologie



Universal Serial Bus (USB)

- **Topologie**
  - **Stern-/Strangstruktur (tiered star)**
    - Physikalische Struktur: Baum mit Host an der Wurzel
    - Logische Struktur: Stern: Host steuert alles
    - Maximal 7 Ebenen (tiers)
  - **USB-Device (USB-Gerät)**
    - Hub
    - Function

Universal Serial Bus (USB)

- **USB Host, Root Hub**
  - Es gibt nur einen Host im System.
  - Der USB Host bildet die Schnittstelle zum Rechner und besteht aus:
    - **Host Controller** (Host Steuereinheit);
    - **Root Hub,** der mehrere Anschlüsse nach außen anbietet.

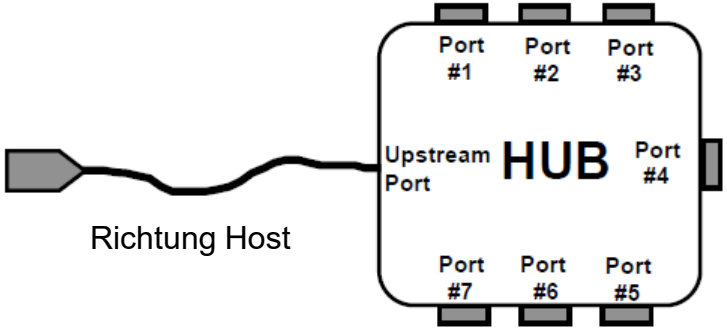
  - **Aufgaben:**
    - Erkennen, dass USB-Geräte angeschlossen oder entfernt wurden;
    - Verwaltung des Kontrollflusses zwischen dem Host und den USB-Geräten;
    - Verwaltung des Datenflusses zwischen dem Host und den USB-Geräten;
    - Sammeln Status- und Aktivitätsinformationen;
    - Stromversorgung für die angeschlossenen USB-Geräte;

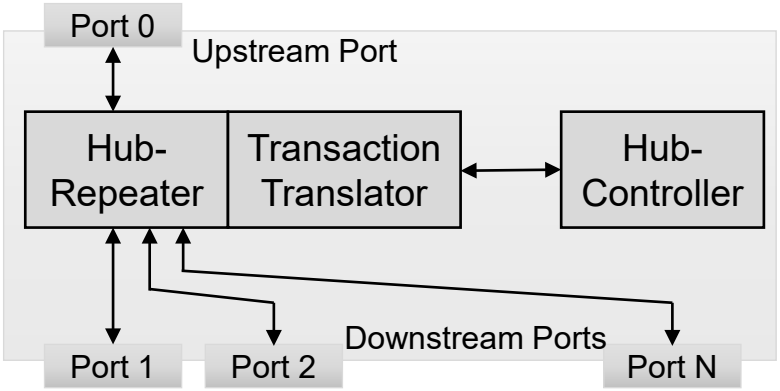
Universal Serial Bus (USB)

- **USB Host, Root Hub**
  - **USB System Software** auf dem Host verwaltet das Zusammenspiel zwischen einem USB-Gerät und der auf dem Host laufenden **gerätespezifischen Software**:
    - Gerätenummerierung und Konfiguration;
    - Isochroner Datentransfer;
    - Asynchroner Datentransfer;
    - Stromverwaltung;
    - Geräte- und Bus-Verwaltungsinformationen;

Universal Serial Bus (USB)

- **Hubs: Sternverteiler**
  - haben einen oder mehrere **Anschlüsse (ports).**
  - Von den Anschlüssen gehen jeweils **Punkt-zu-Punkt-Verbindungen** aus, so dass an einen Anschluss ein weiterer Hub oder USB-Gerät angeschlossen werden kann.





Quelle: Universal Serial Bus Specification Revision 2.0
https://www.usb.org/document-library/usb-20-specification

Universal Serial Bus (USB)

- **Hubs: Sternverteiler**
  - **Hub Repeater**
    - Schalter, der mit Hilfe eines Protokolls den Upstram Port mit einem der Downstream Ports verbindet;
    - enthält HW-Logik für die Signale zum Reset und den Zustandsübergängen in die Suspend- bzw. Resume-Zustände eines USB-Geräts.
  - **Hub Controller**
    - ist verantwortlich die Kommunikation vom und zum Host.
    - Hub-spezifische Status- und Steuerkommandos erlauben es dem Host, einen Hub zu konfigurieren, zu beobachten und seine Anschlüsse zu steuern.
  - **Transaction Translator**
    - stellt die Mechanismen zur Unterstützung der Full-/ oder Low-speed-Geräte und für die Datenübertragungen zwischen den USB-Geräten und dem Host bereit.

Universal Serial Bus (USB)

- **Functions**
  - können Daten oder Kontrollinformationen übertragen oder erhalten;
  - werden mit Hilfe eines Kabels in einen Hub-Port angeschlossen;
  - enthalten Informationen über die jeweilige Konfiguration, die die Fähigkeiten und Anforderungen der jeweiligen USB-Geräte beschreiben.
  - Eine Function muss konfiguriert werden, bevor sie verwendet werden kann:
    - Zuweisung der USB-Bandbreite
    - Auswahl funktionsspezifischer Konfigurationsoptionen.
  - Beispiele:
    - Tastaturen, Mäuse, Joysticks, …
    - Drucker, Scanner, Kameras, …
    - Massenspeicher, …

Universal Serial Bus (USB)

- **Systemkonfiguration**
  - USB-Geräte können jederzeit angeschlossen und wieder entfernt werden.
  - Über Statusbits in den Hubs wird das Anschließen oder das Entfernen eines USB-Geräts angezeigt:
    - Statusbits können vom Host abgefragt werden.
  - Der Host weist dem Gerät eine eindeutige USB-Adresse zu und erkennt, ob das neue USB-Gerät ein Hub oder eine Function ist.
  - Anlegen eines Übertragungskanals für die Steuerkommandos.

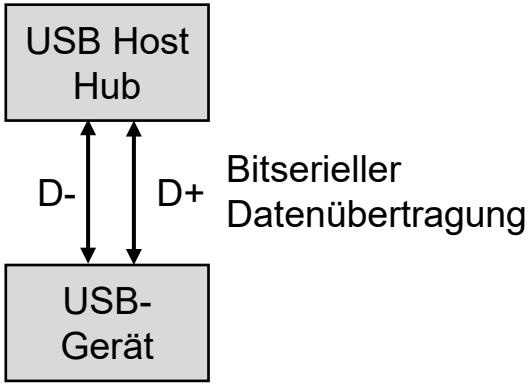
Universal Serial Bus (USB)

- **Datentransfer**
  - zwischen USB Host und USB-Device erfolgt mit Hilfe von uni-direktionalen oder bidirektionalen Pipes;
  - **Pipe:**
    - Logischer Datenkanal als Verbindung zwischen der Host Software und einem bestimmten **Endpoint** eines USB-Geräts.
    - Ein USB-Gerät kann mehrere Pipes haben.
    - Über eine Pipe kann nur eine der vier möglichen Arten von Datentransfers durchgeführt werden.
  - **Endpoint**
    - ist der eindeutig adressierbare Teil eines USB-Geräts, der als Quelle oder Senke für die Informationen bei einer Kommunikation zwischen Host und USB-Gerät dient.
    - Bis zu 16 Endpoints pro Function;

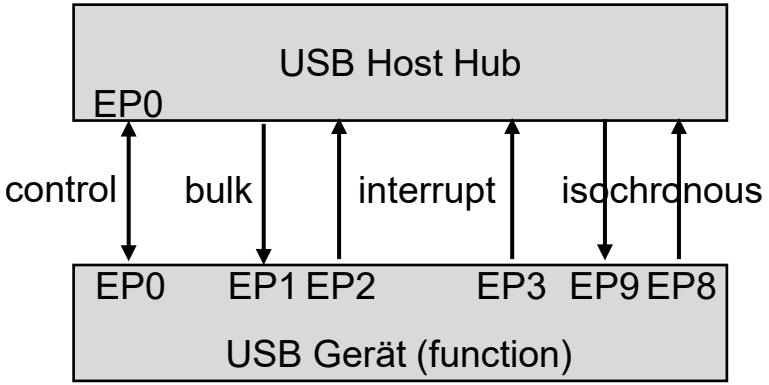
Universal Serial Bus (USB)

Datentransfer

Physikalische Sicht



Logische Sicht



Universal Serial Bus (USB)

- **Arten von Datentransfers**
  - **Control-Transfer**
    - Wird für die Konfiguration eines USB-Geräts verwendet, wenn es angeschlossen wird;
    - Kann auch für weitere gerätespezifische Zwecke, wie die Steuerung anderer Pipes verwendet werden;
  - **Bulk Data Transfer**
    - Datenübertragung ohne Latenz- oder Echtzeitanforderungen;
  - **Interrupt Data Transfer**
    - Zuverlässiges Übertragen von Daten bei einer Mensch-Maschine-Interaktion;
  - **Isochronous Data Transfer (streaming real-time transfer)**
    - Datenübertragung mit garantierter Latenz / Bandbreite
      - Z. B. Audiodaten;

Universal Serial Bus (USB)

- **USB-Protokoll**
  - **Polling-Verfahren**
    - Der Host-Controller fragt die angeschlossenen Geräte nacheinander ab.
    - Der Host-Controller verschickt ein sogenanntes Token Packet mit Informationen über die Art und Richtung der Transaktion, den Geräte- und Endpoint-Adressen, womit dem Gerät signalisiert wird, dass es den Bus benutzen darf.
  - **Datenübertragung**
    - erfolgt ebenfalls als Paket mit einer Menge von Bytes vom Host zum Gerät oder umgekehrt;
    - Erfolgt logisch über eine Pipe;
    - Übertragung von CRC-Prüfinformationen zur Fehlererkennung;
    - Flusskontrolle bei isochroner Übertragung;
    - Bestätigung der Übertragung;

Universal Serial Bus (USB)

- **USB-Protokoll**
  - **Eine USB-Transaktion** umfasst
    - ein **Token Packet** mit Verwaltungsinformationen,
    - ein optionales **Datenpaket** und
    - ein **Statuspaket** für Bestätigungen bei einer Übertragung und für die Fehlerkorrektur.

Universal Serial Bus (USB)

- **USB-Protokoll**
  - **Felder in einem Paket**
    - **Sync:**
      - jedes Paket startet mit einem Sync-Feld;
    - **PID (Packet ID)**
      - Gibt den Pakettyp an;
      - 4 Bits für Pakettyp + 4 Bits invertierte Bits;
    - **ADDR**
      - Spezifiziert die Zieladresse
      - 7 Bits: 127 Geräte können angesprochen werden
    - **ENDP**

      Angabe des Endpoints

      4 Bits: 16 Endpoints können in einem Gerät angesprochen werden

Universal Serial Bus (USB)

- **USB-Protokoll**
  - **Felder in einem Paket**
    - **CRC (Cyclic Redundancy Checks)**
      - Token-Pakete haben 5 Bit CRC
      - Datenpakete haben 16 Bit CRC
    - **EOP**
      - End of Packet

Universal Serial Bus (USB)

- **USB-Protokoll**
  - **Token Pakete**
    - **Es gibt 3 Token Pakettypen:**
      - **In:** informiert das USB-Gerät, dass der Host Informationen lesen möchte
      - **Out:** informiert das Gerät, dass der Host Informationen senden will
      - **Setup:** wird für die Übertragung von Steuerinformationen genutzt
    - **Format:**

Sync	PID	ADDR	ENDP	CRC5	EOP

Universal Serial Bus (USB)

- **USB-Protokoll**
  - **Daten-Pakete**
    - **Es gibt 2 Daten-Pakettypen:**
      - Es können bis zu 1024 c Bytes gesendet warden
      - **Data0**
      - **Data1**
    - **High Speed Modus spezifiziert zwei weitere Pakettypen:**
      - **DATA2**
      - **MDATA.**
    - **Format**

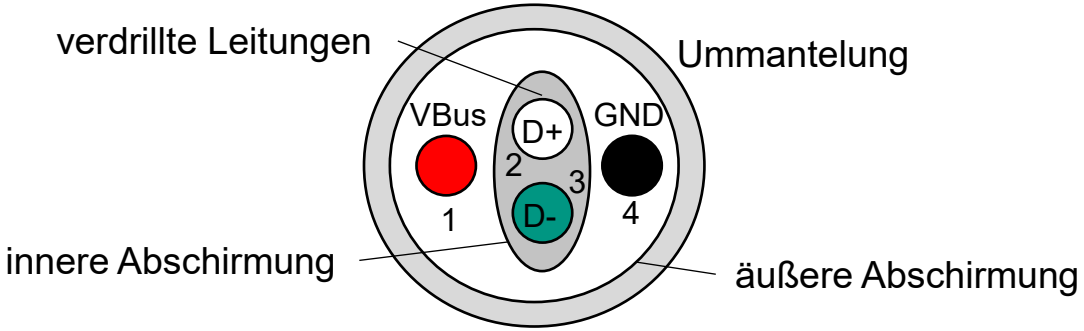
Sync	PID	Data	CRC16	EOP

Universal Serial Bus (USB)

- **USB-Protokoll**
  - **Handshake Pakete**
    - **Es gibt 3 Handshake-Pakettypen**
      - **ACK:** Bestätigung, dass das Paket erfolgreich empfangen worden ist
      - **NAK:** Gerät kann im Moment nicht senden oder empfangen
      - **STALL:** Gerät befindet sich in einem Zustand, in dem der Host eingreifen muss
      - **Format**

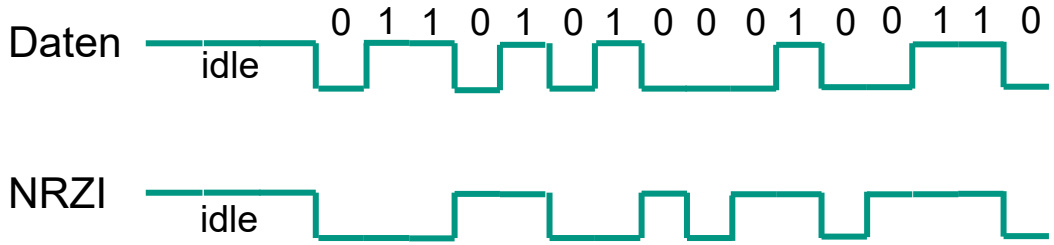
Universal Serial Bus (USB)

- **Datenübertragung**
  - **Verbindungskabel**
    - 2 Kabel für differentielle Signalübertragung (D+, D-)
    - 2 Kabel für Stromversorgung (VBus, GND)
    - Twisted Pair für mittlere und höhere Übertragungsraten
    - Unverdrillte und nichtabgeschirmte Kabel für Low-Speed Übertragungen
    - Kabellänge auf 5m begrenzt



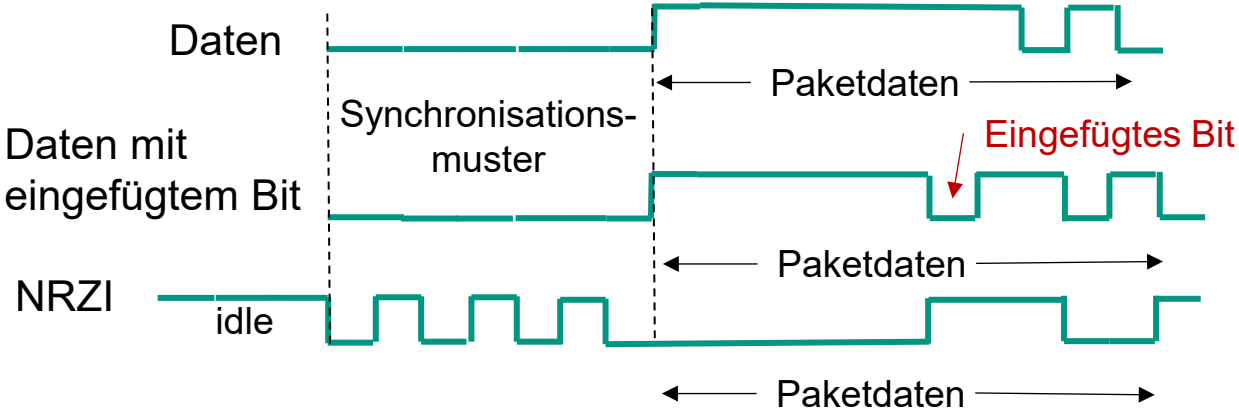
Universal Serial Bus (USB)

- **Datenübertragung**
  - **NRZI Kodierung (non return to zero with interchange)**
    - Eine logische 1 wird durch gleichbleibende Polarität dargestellt
    - Eine logische 0 wird durch Wechsel der Polarität dargestellt



Universal Serial Bus (USB)

- **Datenübertragung**
  - **Synchronisation mit Empfänger**
    - Voranstellen eines Synchronisationsmusters
    - **Bit-Stuffing**
      - Einfügen einer 0 nach sechs aufeinanderfolgenden 1en in einem Datenstrom
    - Empfänger muss NRZI Daten dekodieren, das zusätzliche eingefügte (“stuffed”) Bit erkennen und entfernen



Universal Serial Bus (USB)

- **Literatur**
  - USB 2.0 Specification: https://www.usb.org/document-library/usb-20-specification
  - USB in a nutshell: https://www.beyondlogic.org/usbnutshell/usb1.shtml

Universal Serial Bus (USB)

- **Literatur**
  - USB 2.0 Specification: https://www.usb.org/document-library/usb-20-specification
  - USB in a nutshell: https://www.beyondlogic.org/usbnutshell/usb1.shtml