

Lernzettel

Eingabe/Ausgabe und Peripherie: Busse,
Interrupts, DMA, I/O-Subsysteme

Universität: Technische Universität Berlin
Kurs/Modul: Technische Grundlagen der Informatik
Erstellungsdatum: September 19, 2025



Zielorientierte Lerninhalte, kostenlos!
Entdecke zugeschnittene Materialien für deine Kurse:

<https://study.AllWeCanLearn.com>

Technische Grundlagen der Informatik

Lernzettel: Eingabe/Ausgabe und Peripherie

(1) Grundlagen. Die Eingabe/Ausgabe (I/O) umfasst die Interaktion des Computers mit Peripherie wie Druckern, Netzwerken, Festplatten und Sensoren. Zentral sind dabei Schnittstellen, Pufferung, Protokolle und Timings, damit Daten zuverlässig und effizient übertragen werden. Die I/O-Subsysteme koordinieren die Kommunikation zwischen CPU, Speicher und Peripherie über Busse und Controller.

(2) Busse: Daten-, Adress- und Steuerbus. Busse übertragen Daten, Adressen und Steuerinformationen zwischen Komponenten. Typische Merkmale: - Adressbus: adressiert Peripherie oder Speicherzellen (Breite in Bit, z. B. 32-Bit). - Datenbus: transportiert die eigentlichen Nutzdaten (Breite in Bit, z. B. 64-Bit). - Steuerbus: Signale wie READY/WAIT, Interrupt-Anforderungen, Takte. Der Buszugriff erfolgt oft über einen Bus-Arbiter. Ein Master steuert den Zugriff, während Slaves reagieren.

(3) Interrupts: asynchrone Ereignisse und ISR. Ein Interrupt signalisiert der CPU, dass eine Peripherie eine Aufgabe fertiggestellt hat oder Aufmerksamkeit benötigt. Typischer Ablauf: - Ereignis: Peripherie löst Interrupt aus (IRQ). - CPU speichert Kontext (Program Counter, Register). - Sprung zur Interrupt-Service-Routine (ISR). - ISR bearbeitet das Ereignis, beendet mit RETI (Return from Interrupt). Vorteile: geringe Idle-Time, bessere Reaktionsfähigkeit; Nachteile: Kontextwechsel-Overhead.

(4) Direct Memory Access (DMA). DMA entlastet die CPU, indem Peripherie Daten direkt in/aus dem Hauptspeicher transferiert. Typischer Ablauf: - Peripherie fordert DMA-Transfer an. - DMA-Controller oder DMAC übernimmt die Adressierung und den Datentransfer. - Nach Abschluss wird ein Interrupt an die CPU ausgelöst (optional). Vorteile: höherer Durchsatz, geringe CPU-Belastung; Nachteile: zusätzliche Hardware und Komplexität.

(5) I/O-Subsysteme: Controller, Bridges und Protokolle. I/O-Subsysteme umfassen Controllerboards, Bridge-Chips (z. B. Northbridge/Southbridge-Konzept in alten Architekturen oder moderne SoCs), Speicherschnittstellen, Puffer, DMA-Module und Protokolle (z. B. PCIe, USB, SATA). Zwei gängige Konzepte: - Memory-Mapped I/O: Peripherie ist an denselben Adressraum wie RAM adressierbar; Steuerbefehle als Lese/Schreibe-Operationen. - I/O-Mapping (Port-Mapped I/O): Separate Adressräume, oft mit speziellen Befehlen / In/Out-Instruktionen. Die effiziente Nutzung erfordert Pufferung, Interrupt-Steuerung, und Timing-Benchmarks.

(6) Synchronisation und Timings in I/O. Arbitration-Algorithmen regeln den Zugriff mehrerer Geräten auf gemeinsame Busse: Round-Robin, Prioritätsbasierte Verfahren, oder kombinierte Ansätze. Timing-Konzepte: - Setup-Time, Hold-Time: Zeitfenster, das ein Signal stabil bleiben muss. - Setup/Duration von Transfers: Wann Daten anliegen und wann sie gelesen werden müssen. - Durchsatz-Dimensionen: effektiver Takt pro Transfer vs. theoretischer Maximalwert. Formel (Beispiel):

$$\text{Durchsatz} = \frac{\text{Datenmenge}}{\text{übertragene Zeit}} = \frac{N \text{ Byte}}{T \text{ s}} \quad (\text{Byte/s})$$

(7) Beispiele und typische Szenarien. - USB-Datenübertragung: Massenspeicher oder Eingabegeräte nutzen Interrupts und DMA-Transfers, um CPU-Belastung gering zu halten. - PCIe-basiertes I/O:

Hochgeschwindigkeitsbussen mit asynchronem Interrupt-Modell und optionalem DMA. - Drucker im Druck-Cache: Pufferung und Flusskontrolle, um Staus zu vermeiden.