Rechnerarchitekturen 1*

Rechnerkomponenten

Prof. Dr. Alexander Auch

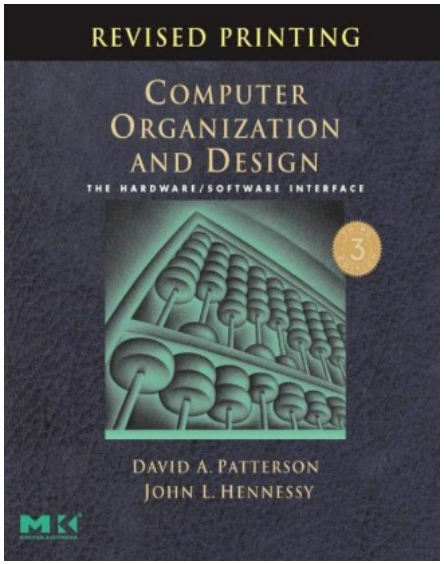
*Teilweise entnommen aus “Mikrocomputercomputertechnik 1” von Prof.Dr-Ing. Ralf Stiehler

Ziele der Veranstaltung

- **Rechnerentwurf**:
  - Prozessor, Speicher, Ein-/Ausgabe
  - Entwurfs- und Optimierungsmöglichkeiten
- **Prozessorentwurf**:
  - Befehlsverarbeitung
  - Entwurfs- und Optimierungsmöglichkeiten
- **Rechnerkomponenten**:
  - Peripherie

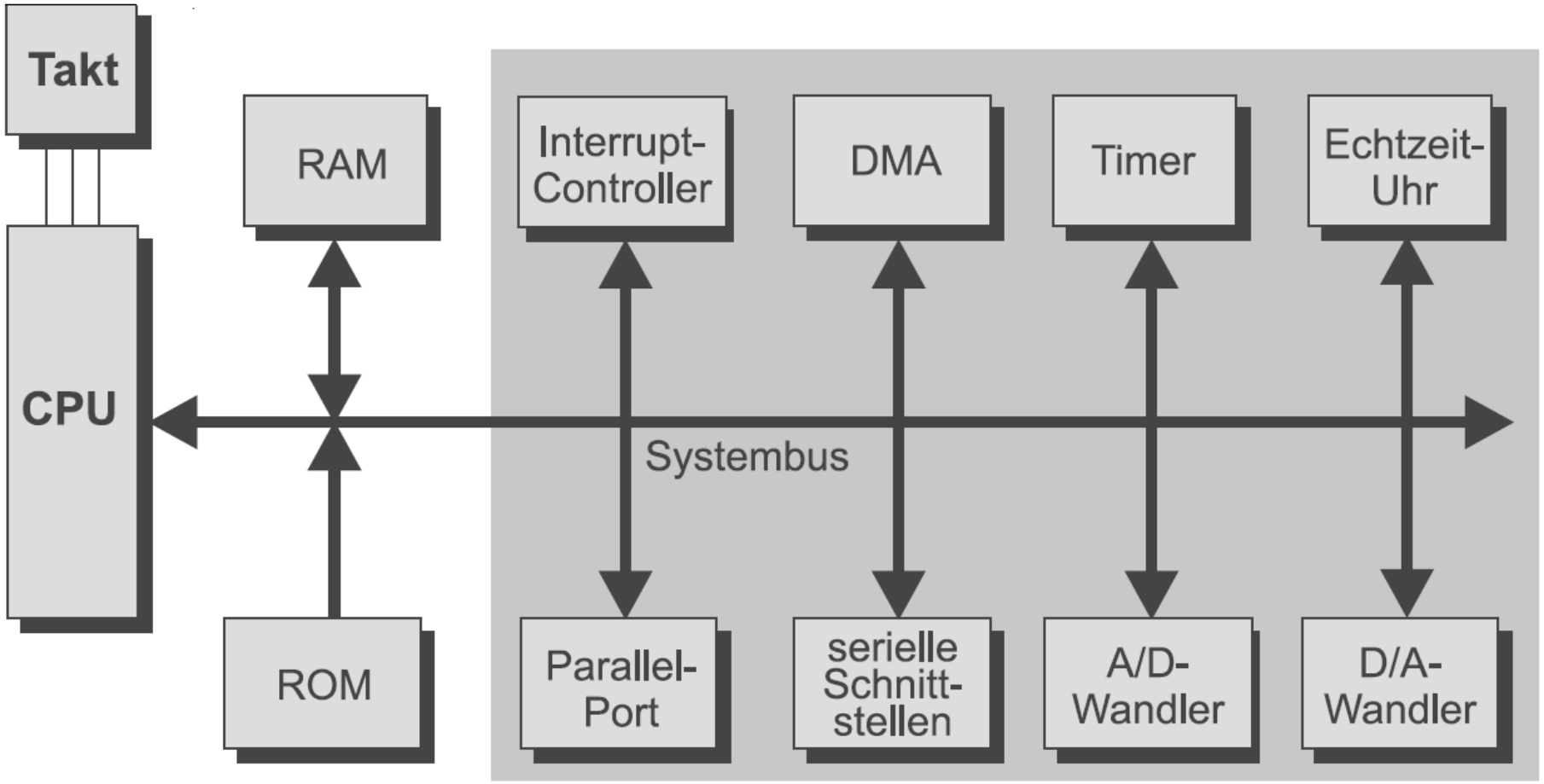
Inhalt

- Einführung
- Rechnerentwurf
- Rechnerarithmetik
- Prozessorentwurf



- **Rechnerkomponenten**

Systembausteine/-module



I/O-System

- **Alle Komponenten zusammen, d.h. I/O-Devices, Prozessor, DMA, Interrupt-Controller, Speicherhierarchie, Software usw. beeinflussen *Zuverlässigkeit*, *Erweiterbarkeit* und *Performance* von I/O-Aufgaben**

- **Anforderungen an ein I/O-System**

→ **Unterstützung unterschiedlicher Eigenschaften von I/O-Geräten**
→ ***Throughput / Bandbreite* : Datenmenge pro Zeiteinheit**
→ ***Response Time* (oft das wichtigste Maß bei Embedded- Systemen)**

- **Arten von Modulen / Bausteinen**

→ ***Hilfsmodule* : einzelne Gatter, Register, Decoder, Multiplexer, Demultiplexer**
→ ***Systemmodule* : führen bestimmte Funktionen aus**
→ ***Schnittstellenmodule* : binden das Mikroprozessorsystem an Peripherie an**

Systemmodule

- **Nicht programmierbare**

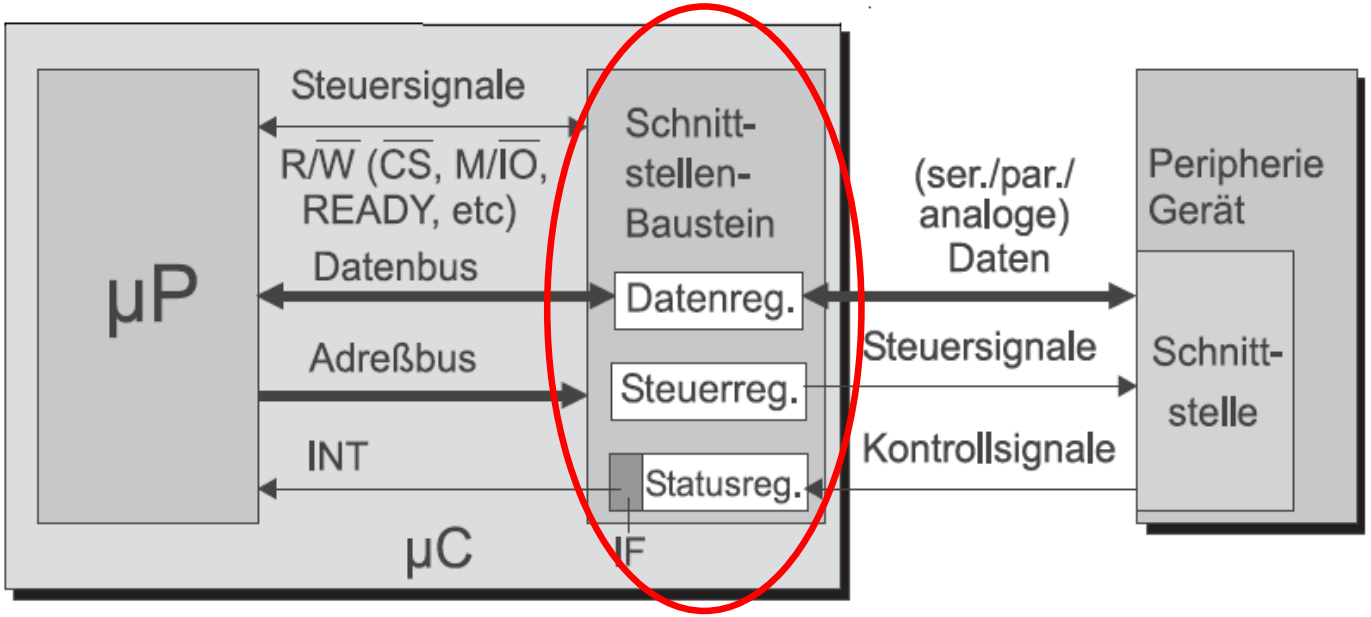
→ **dienen der Ausführung bestimmter fest vorgegebener Funktionen, evtl. sind einige wenige Betriebsalternativen *hardwaremäßig auswählbar*, die Auswahl ist in der Regel für die gesamte Einsatzdauer fest vorgegeben**
→ **meist auf dem Chip des Mikrocontrollers/-prozessors integriert**
→ **Beispiele : Taktgeneratoren, Frequenzteiler, Bus-Arbiter, DRAM-Controller**

- **Programmierbare**

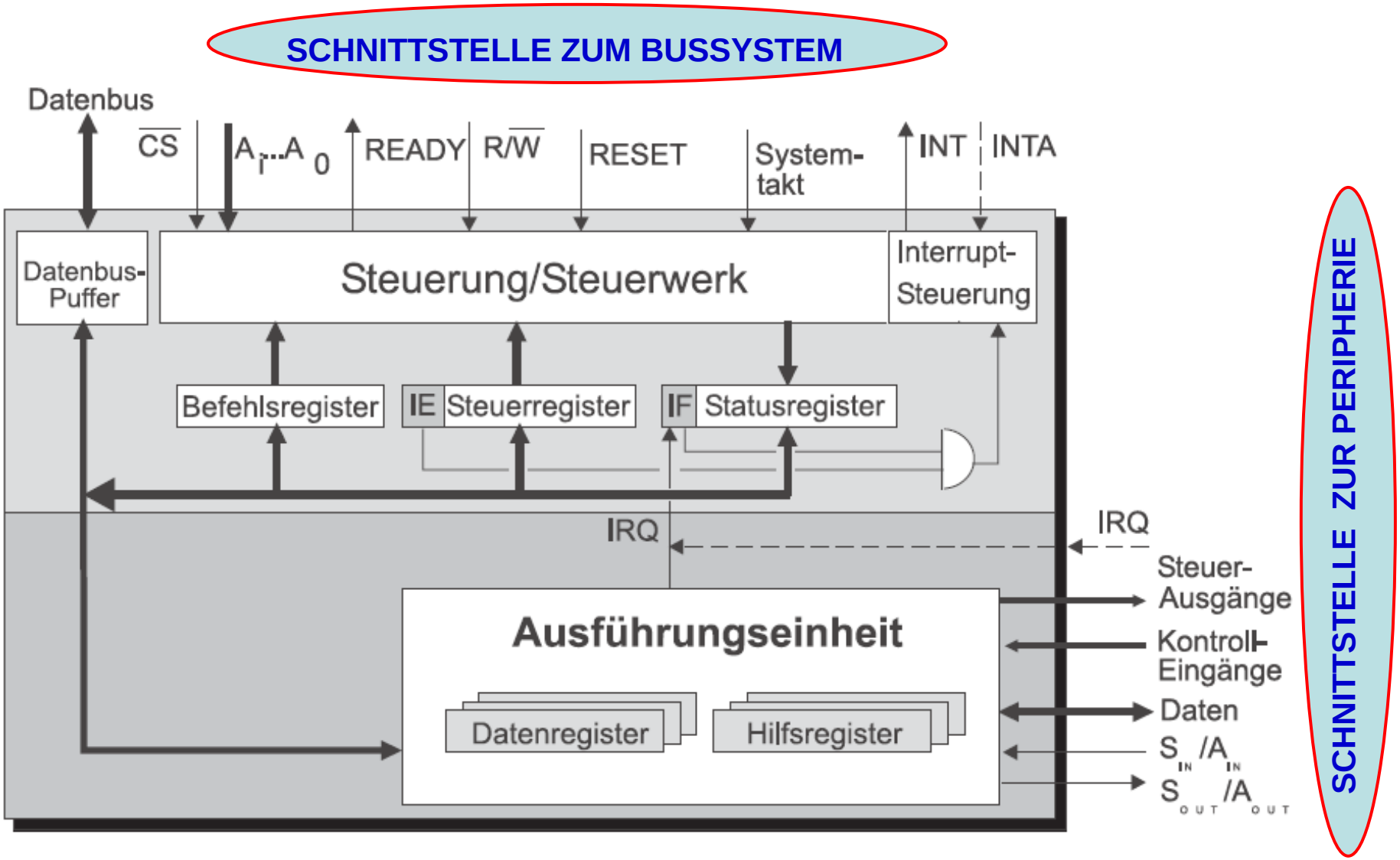
→ **Verschiedene Arbeitsmodi während des Betriebs möglich**
→ **programmierbar bedeutet: Prozessor bestimmt durch Einschreiben eines Steuerworts in ein Register den Arbeitsmodus**
→ **Beispiele : DMA-Controller, Interrupt Controller, Cache-Controller, Timer- Module, Echtzeituhren**

Systemmodule

- →z.B. I/O Bausteine, I/O-Controller auf dem Mikrocontrollerchip
- →sind Bindeglied zwischen Mikrocontroller/-prozessor und Peripherie
- →dienen der Pufferung von Ein-/Ausgabedaten
- →passen unterschiedliche Arbeitsgeschwindigkeiten an (System, Peripherie)
- →wandeln Daten um (Seriell/Parallel, Digital-Analog, etc)
- →erzeugen Steuersignale für Peripheriegeräte
- →synchronisieren die Übertragung zwischen Gerät und Schnittstellenbaustein
- →nehmen Interrupts der Peripheriegeräte auf (oder erzeugen selbst welche) und leiten sie an den Prozessor weiter



Schnittstellenmodul: Allgemein



Schnittstellenmodul: Allgemein

- Steuerwerk ist bei komplexen Bausteinen ein Zustandsautomat / FSM
- Quelle und Ziel eines Datentransports sind interne Register des Bausteins
- Steuerregister (Mode-Control-Register) werden durch den Prozessor (meist beim Initialisieren programmiert)
- Steuerung beschreibt Statusregister mit best. Informationen (z.B. Betriebs-modus), diese Statusregister können dann von der CPU ausgelesen werden
- durch eine Interrupt-Anforderung wird das Interrupt Flag Bit (IF) gesetzt
- wenn das Interrupt Enable Bit (IE) gesetzt ist, wird der Interrupt weitergeleitet
- im Befehlsregister legt der Prozessor Steuerungsbits für die gewünschte Operation ab
- die Interrupt-Steuerung erzeugt das INT-Signal
- die Ausführungseinheit stellt die eigentliche I/O-Funktionalität zur Verfügung, Datenregister (z.B. FIFOs) können vom Prozessor gelesen oder beschrieben werden

Schnittstellenmodul: Allgemein

Für die CPU erscheinen System- und Schnittstellenmodule wie ein kleiner Satz von Registern. Man unterscheidet speicherbezogene und isolierte Adressierung

Memory Mapped I/O (speicherbezogene Adressierung)

- **→Register-Adressblock ist zusammen mit allen Speicheradressen in einem gemeinsamen Adressraum untergebracht**
- **→CPU sieht zwischen Speicherzelle und Register keinen Unterschied**
- **→Gleiche LOAD- und STORE-Befehle für Zugriff auf System- und Schnittstellenmodule wie auf Speicher**

Isolated I/O

- **→ Getrennte Adressräume für Speicher und Schnittstellenbausteinen**
- **→ Auswahl durch ein zusätzliches Signal nötig (z.B. M/IO), alle Module und der Speicher müssen dieses Signal auswerten**
- **→ zusätzliche Befehle neben Load/Store im Befehlssatz der CPU nötig**

Interrupt-Controller

→Periphere Geräte oder Systemsteuerbausteine können über spezielle Eingänge asynchron sog. Interrupts an die CPU abgeben, z.B. wenn ein bestimmtes Datum oder ein bestimmter Zustand vorliegt
→der Interruptanforderung wird von der CPU frühestens nach der vollständigen Beendigung des aktuellen Befehls stattgegeben
→die Reaktion des Prozessors ist das Starten einer Interrupt-Routine

→Interrupteingänge:

NMI (Non maskable Interrupt)
→ werden unbedingt durchgeführt
→ während der Abarbeitung ist kein weiterer Interrupt zugelassen

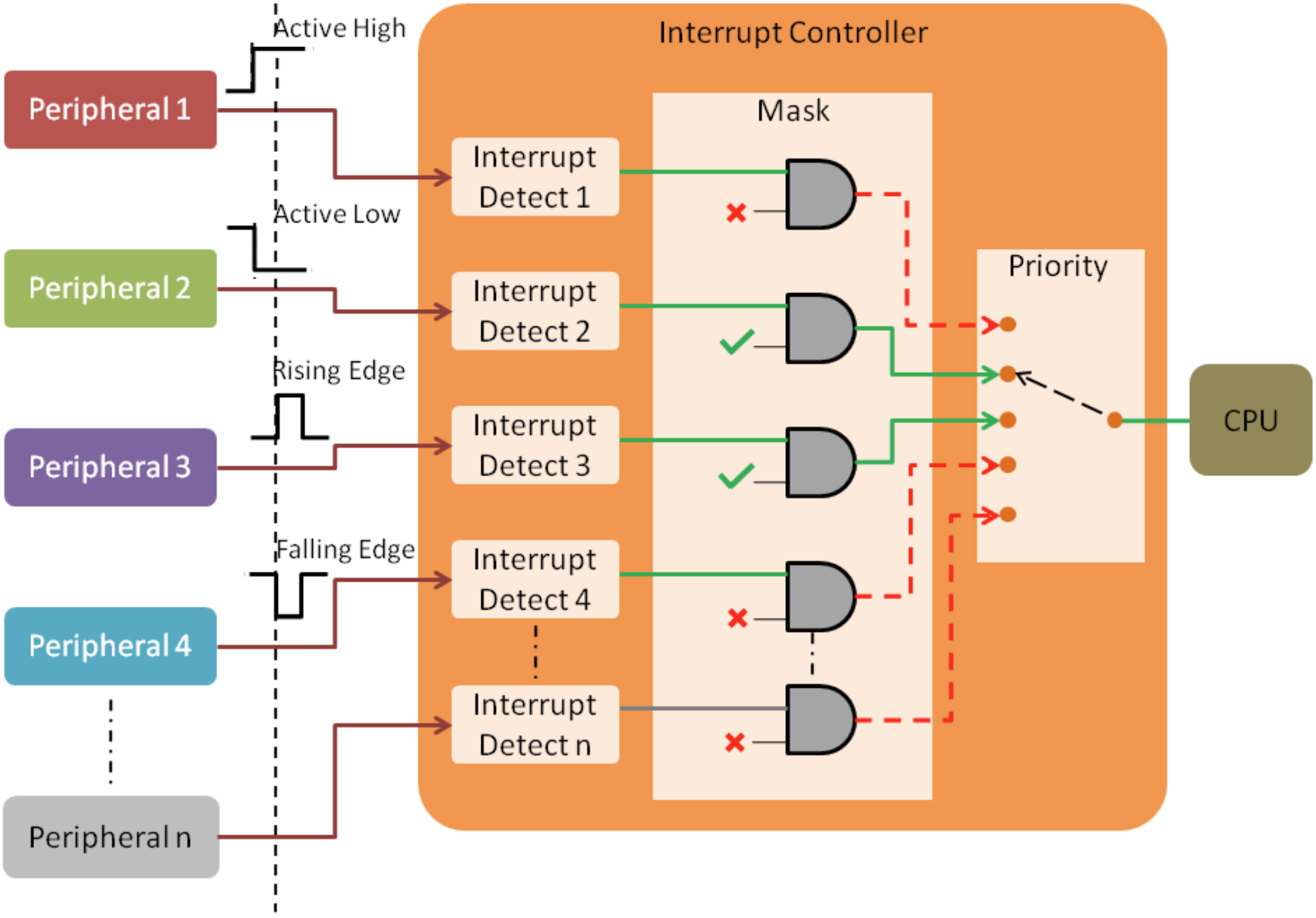
IRQ (Interrupt Request)

Interrupt-Controller: Behandlung

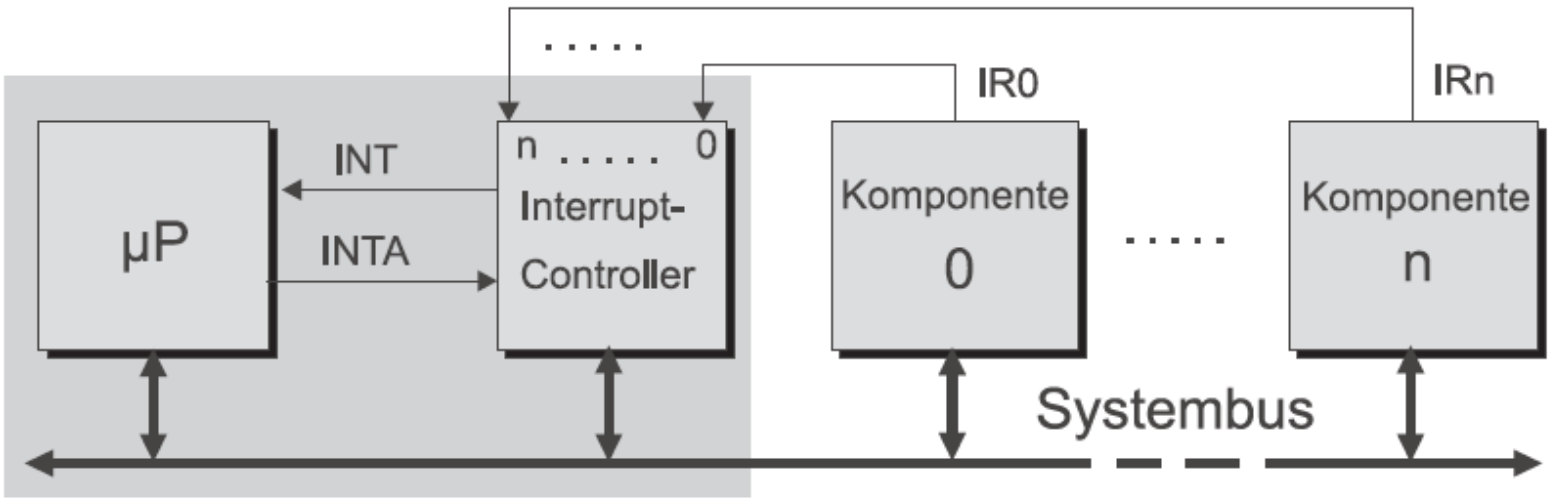
- **Statusregister: (single entry point for all exceptions)**
  - **Codiert Ursache einer Exception**

- **Vectored Interrupt:**
  - **→ Die Verzeigungsadresse wird durch die Ursache der Exception bestimmt,d.h. es gibt für jede mögliche Exception eine festgelegte Speicheradresse, zu der dann verzweigt wird. Hier sollte dann die entsprechende Exception-Handling-Routine stehen.**
  - **→ Der Exception Handler „kennt“ den Grund der Exception nur durch die Adresse zu der verzweigt wurde.**

Interrupt-Controller: Behandlung

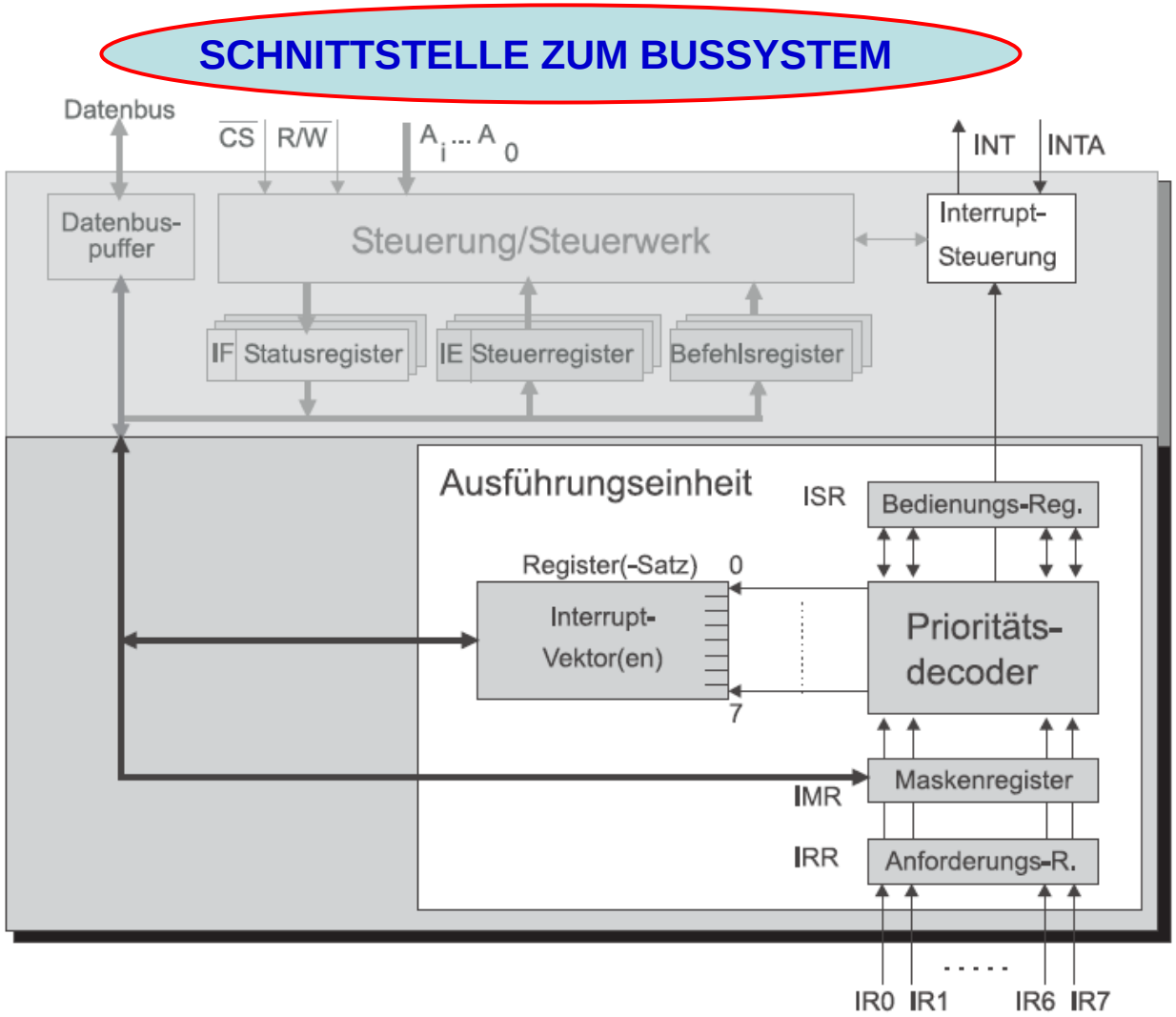


Interrupt-Controller: Modul



- **Verwaltet mehrerer Interrupt-Quellen**
- **IRi: eigener Unterbrechungswünsch von Systemkomponente i**
- **der Controller ermittelt die Interruptquelle höchster Priorität und gibt deren Anforderung über das Signal INT an den Prozessor weiter, der prüft (z.B. über Interrupt-Enable Bits – IE) ob Unterbrechungen zugelassen sind.**
- **bei Annahme des Interrupts unterrichtet der Prozessor den Interrupt Controller**

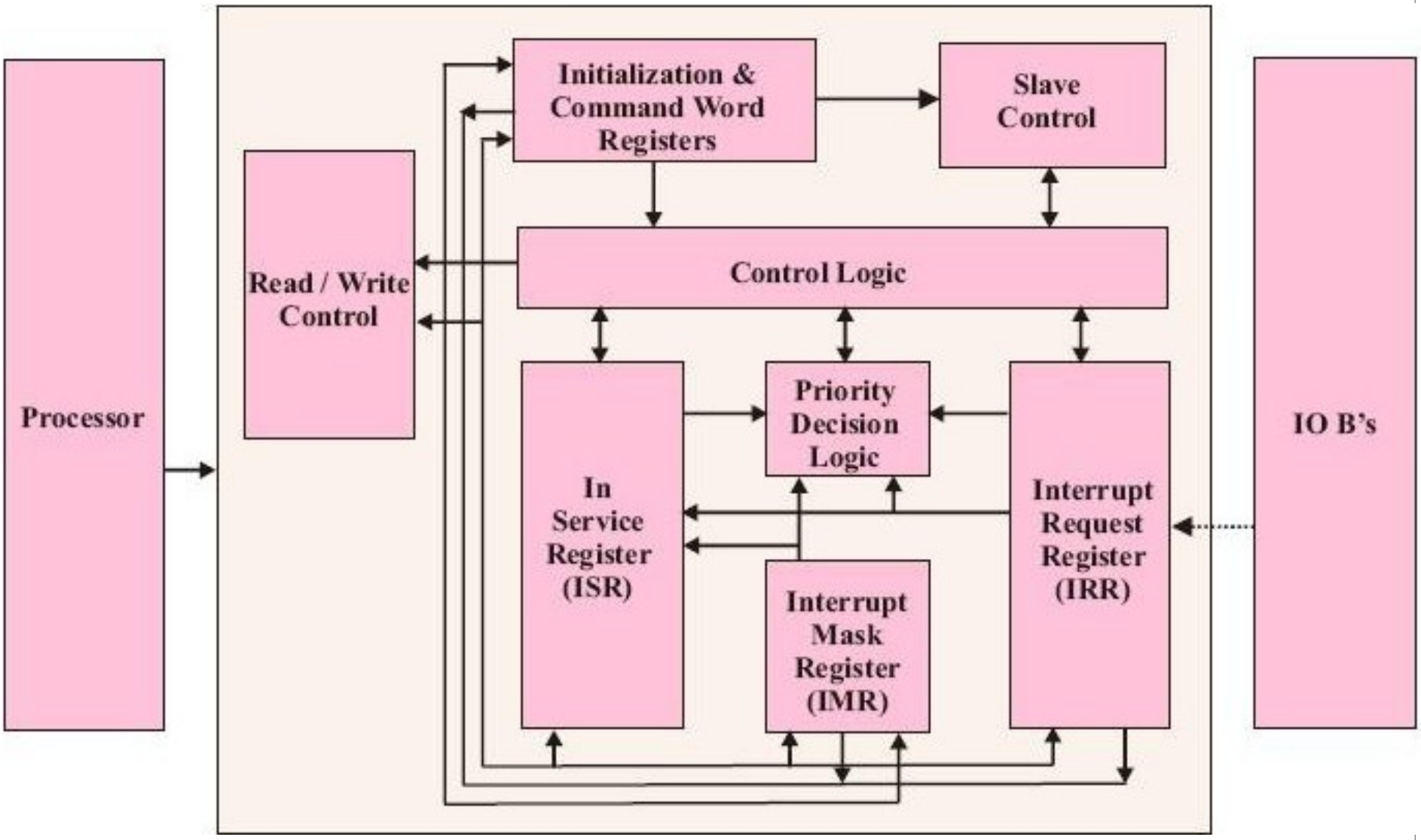
Interrupt-Controller: Modulaufbau



Interrupt-Controller: Modul

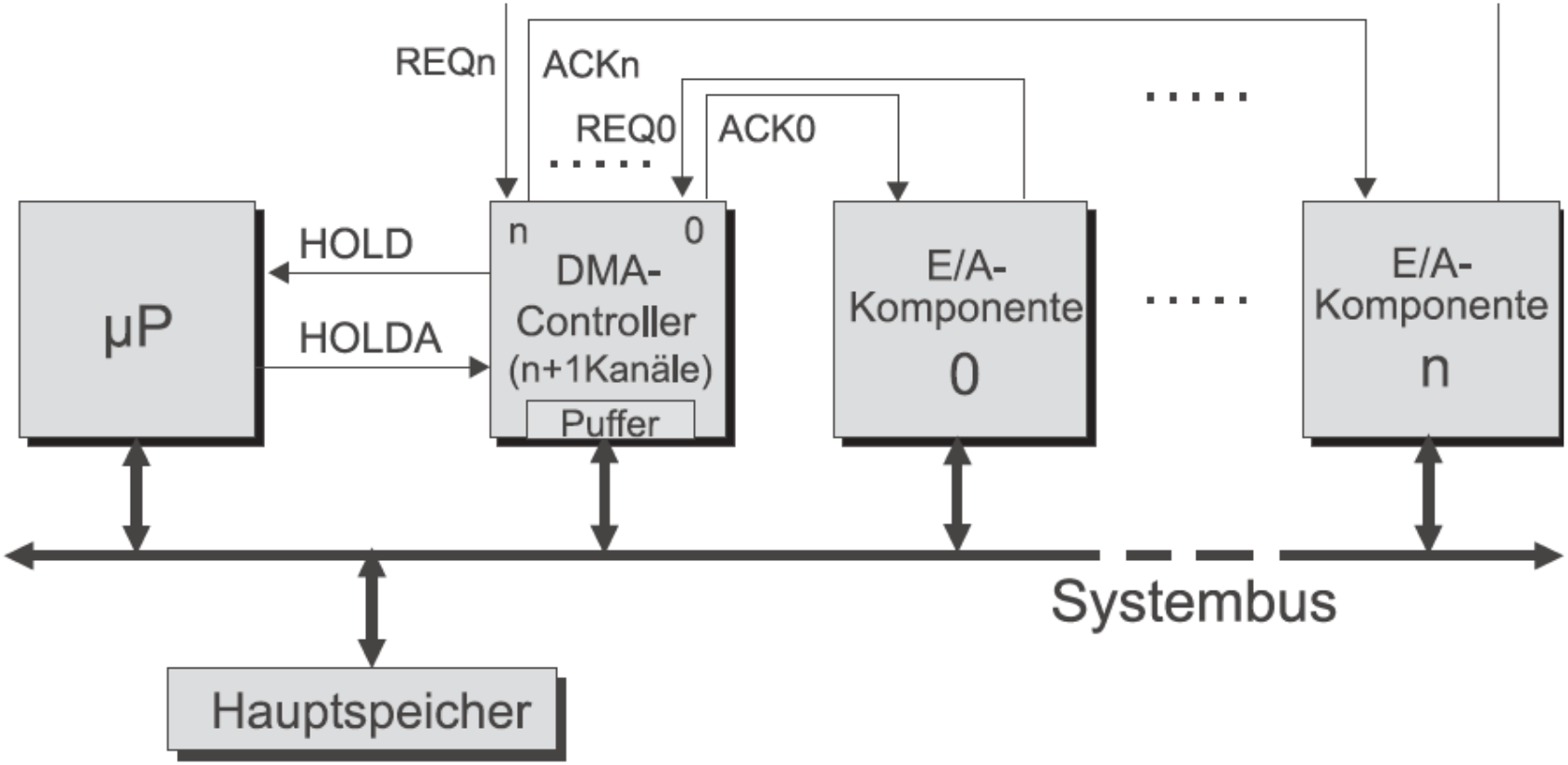
- es gibt mehrere IR_i-Eingänge (Interrupt-Kanäle), der Zustand wird im IRR-Register eingelatcht
- das Interrupt-Mask Register IMR kann vor oder während einer Programmaus führung mit beliebigem Bit-Muster geladen werden (Interrupt-Maskierung)
- der Prioritätendecoder wählt bei mehreren vorliegenden Interrupts denjenigen mit der höchsten Priorität aus
- tritt bei Durchführung einer aktuellen Interrupt-Routine ein Interrupt mit höherer Priorität auf, so leitet die Interrupt-Steuerung diesen dann weiter
- Das Interrupt-Service-Register ISR enthält alle Unterbrechungswünsche, die gerade ausgeführt werden oder unterbrochen worden sind
- Über die Interrupt-Vektor-Nummer kann der Prozessor die Startadresse der Interrupt-Routine ermitteln
- nach Abschluss einer Interrupt-Routine durch den Prozessor wird das Interrupt Service Bit der zugehörigen Quelle im ISR zurückgesetzt
- der Prioritätendecoder kann nun anhand des ISR feststellen, ob eine weitere Anforderung ansteht (Pending Interrupt) und ggf. über INT ausgeben.

Interrupt-Controller: Behandlung



DMA-Controller

- **Direct-Memory-Access-Controller sind Spezialbausteine, die die CPU von der zeitraubenden und einfachen Aufgabe der Datenübertragung zwischen Speicher und den Peripheriebausteinen entlasten**
- **CPU gibt den Systembus i.d.R. zyklenweise an den DMA-Controller ab**



DMA-Controller

→zu Beginn eines DMA-Transfers wird der DMA-Controller von der CPU mit den notwendigen Informationen zur Datenübertragung versorgt (initialisiert).

- → Startadresse eines Datenbereichs im Speicher
- → Adresse einer Schnittstelle oder eines zweiten Datenbereichs im Speicher
- → Anzahl der zu übertragenden Daten
- → Richtung der Datenübertragung

→ Die Datenübertragung selbst wird dann vom DMA selbständig übernommen, dabei kann die CPU intern weiterarbeiten, aber nicht auf den Bus zugreifen

DMA-Ablauf

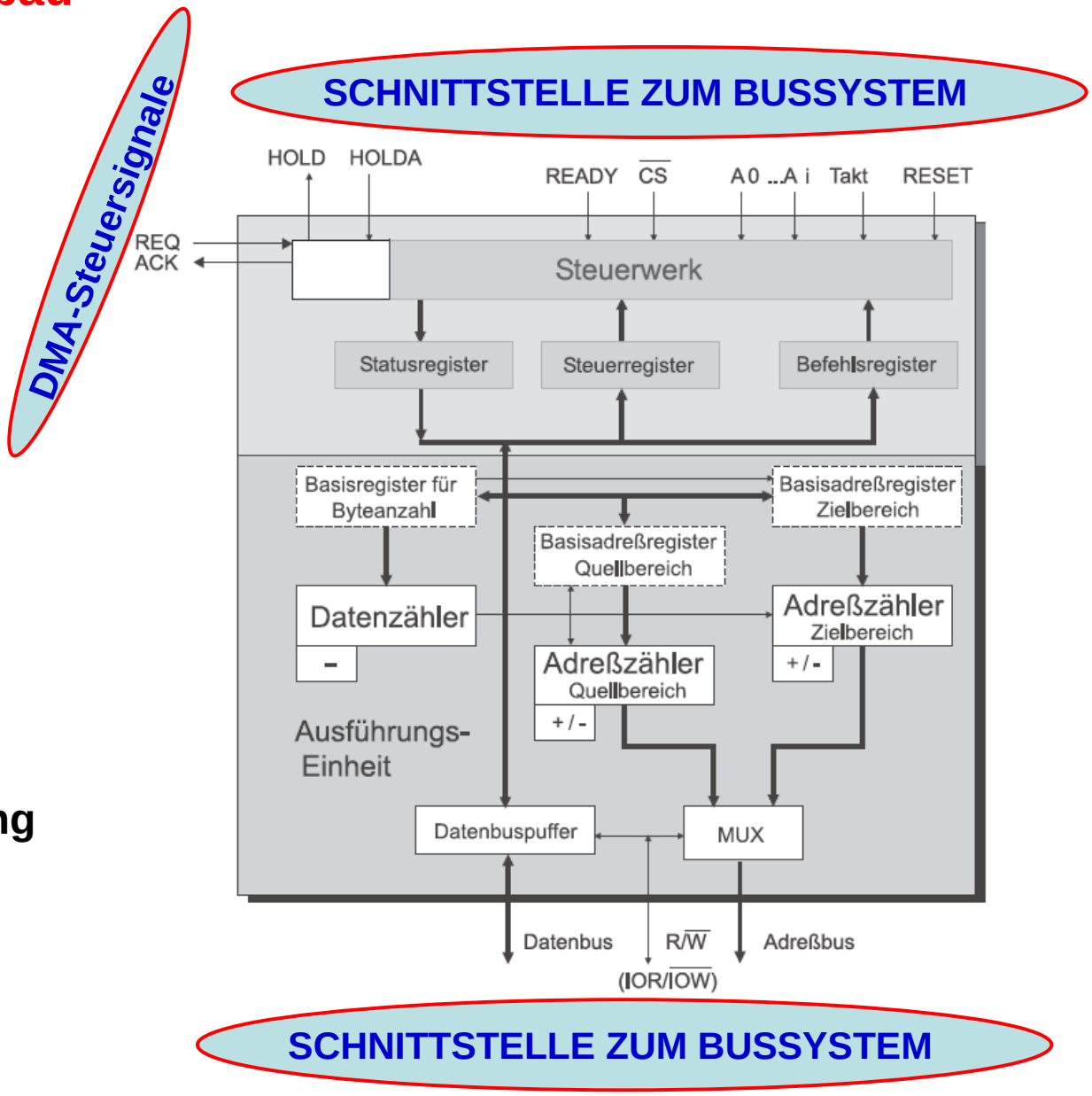
→Schnittstellenbausteine fordern Buszugriff über REQi-Leitungen
→DMA reicht dies über HOLD an die CPU weiter und gibt Bus über HOLDA frei, sobald sie selbst ihre letzte Bustransaktion beendet hat
→DMA-Controller informiert darüber die anfordernde Komponente über ACKi
→wenn Komponente REQi deaktiviert → DMA deaktiviert HOLD → CPU übernimmt Buszugriff und zeigt dies durch Rücksetzen von HOLDA an → DMA informiert Komponente mit ACKi

DMA-Controller: Aufbau

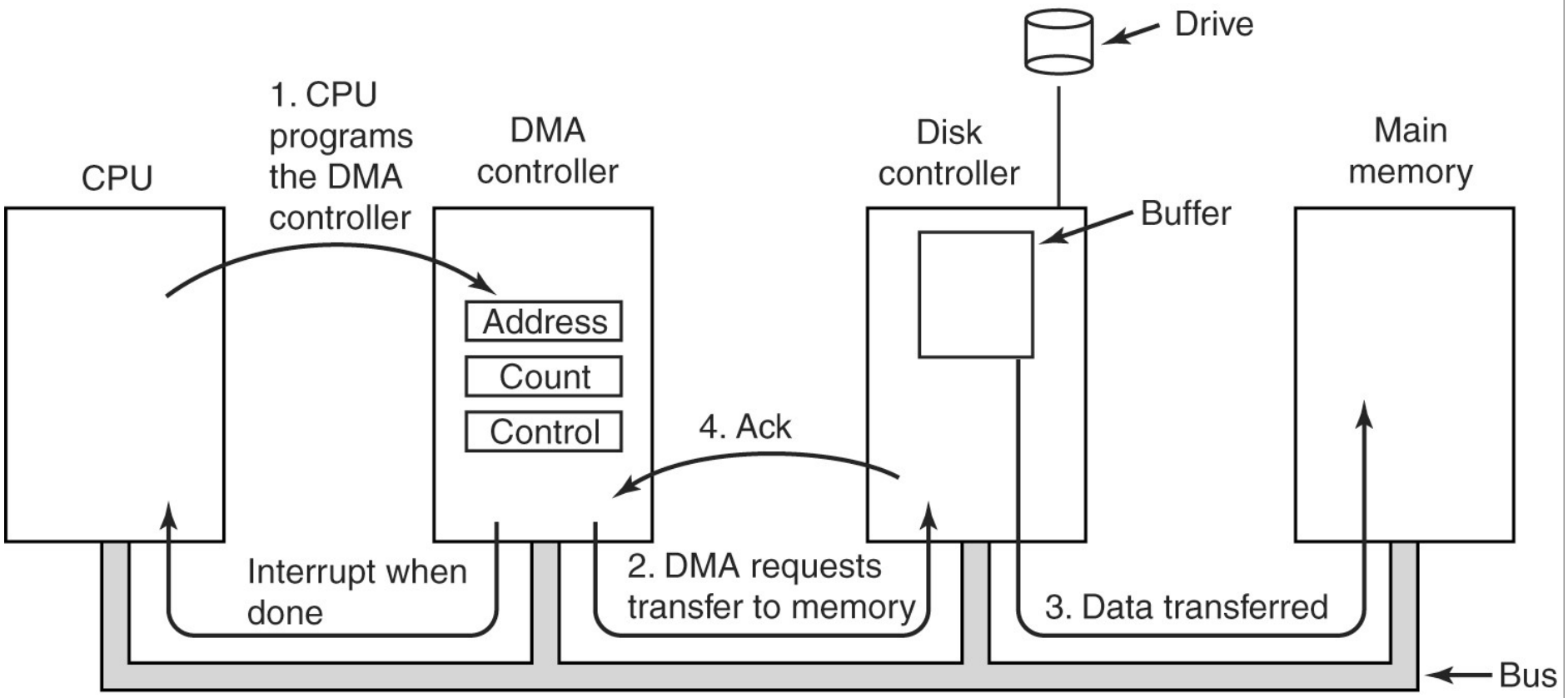
DMA-Modi

→Single Transfer Mode
Nur ein Datum wird übertragen

=> BURST-Mode
Blockweise Übertragung



DMA-Controller: Szenario



Timer

Timer (Zeitgeber- und Zählermodule)

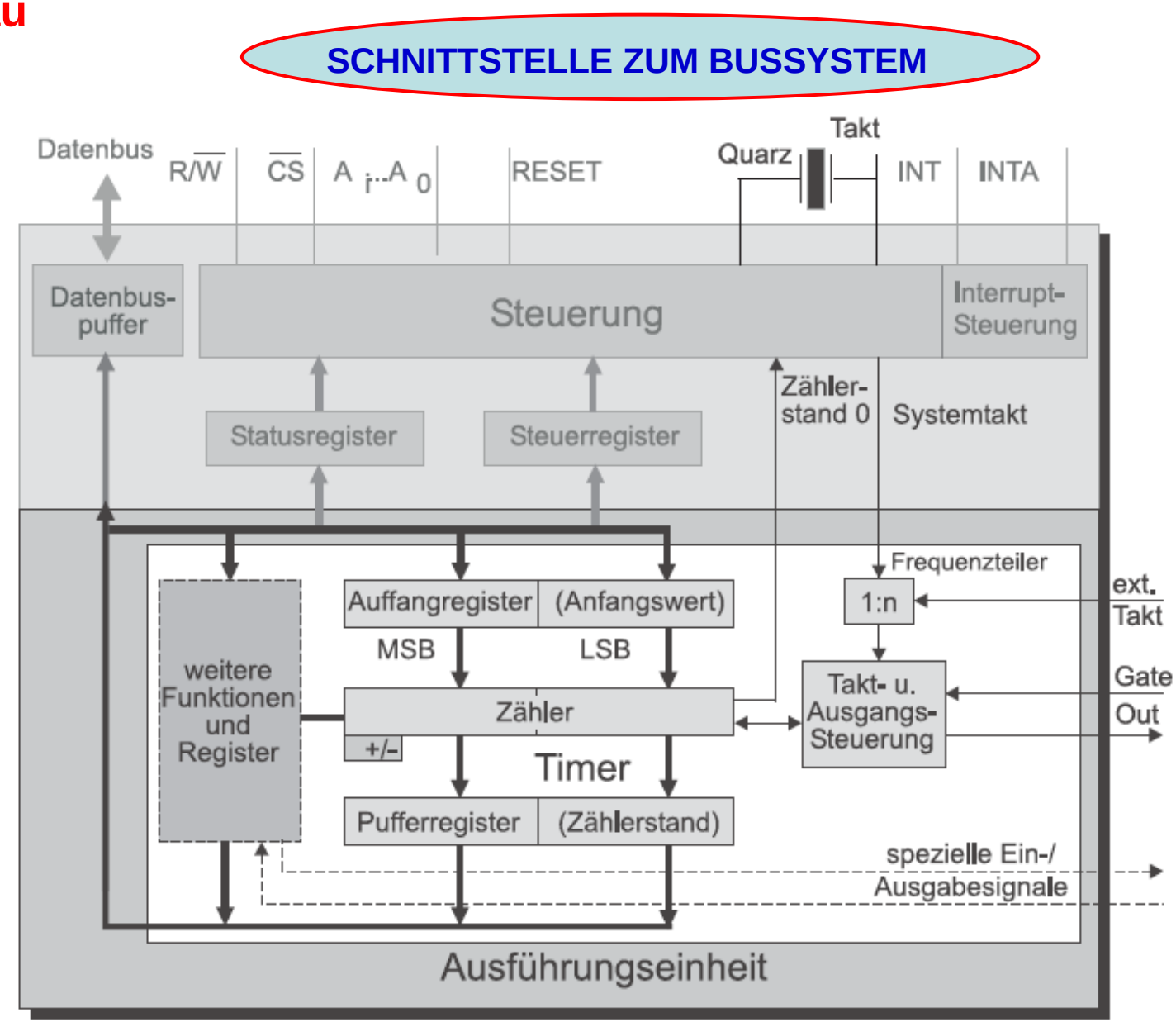
Einsatzgebiete von Timern

- **Erzeugung von internen oder externen Ereignissen**
  z.B . Impulsgenerator für Ereignisse oder als Interruptquelle
- **Signalgenerator, z.B. Rechtecksignale, Trigger-Impulse, PWM-Signale**
- **Erfassung externer Ereignisse, z.B. Ereigniszähler, Flankendetektor**
- **Zeitmessung**

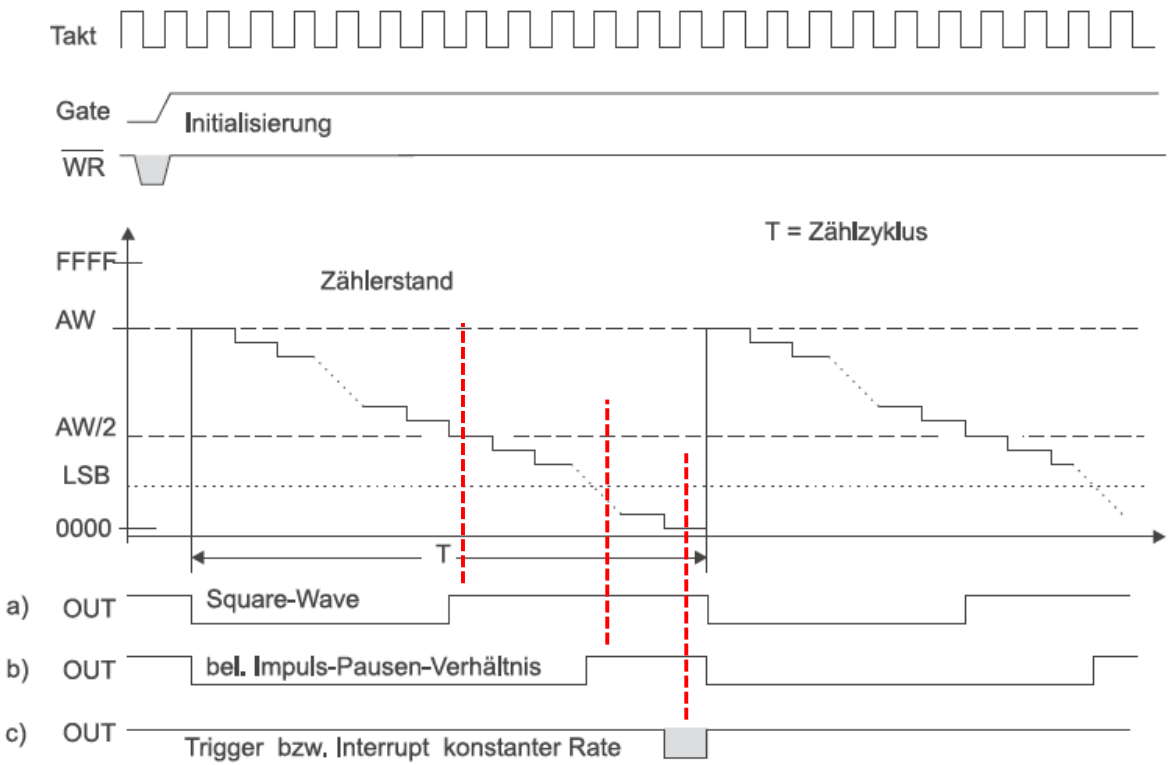
Aufbau eines Timer-Moduls

- **Anfangswertregister zur Initialisierungswert des Timers**
- **Synchroner Binärzähler, der Zählerstand wird mit jedem Takt dekrementiert**
- **Pufferregister, der Zählerstand wird je nach Betriebsmodus in das Puffer-register übertragen und kann anschließend von der CPU gelesen werden**
- **Frequenzteiler zum Herunterteilen des Taktes um einen Faktor 1:n**
- **Gate-Eingang dient als Eingangssignal zur Zeitmessung oder zur Aktivierung/Deaktivierung des Zählers bei Verwendung als Ereigniszähler**
- **Beim Erreichen von 0 wird ein Statusbit gesetzt und/oder Interrupt ausgelöst**

Timer: Aufbau



Timer: Als Taktgenerator



→betrachtet werden 3
3 Konfigurationen
a, b, c

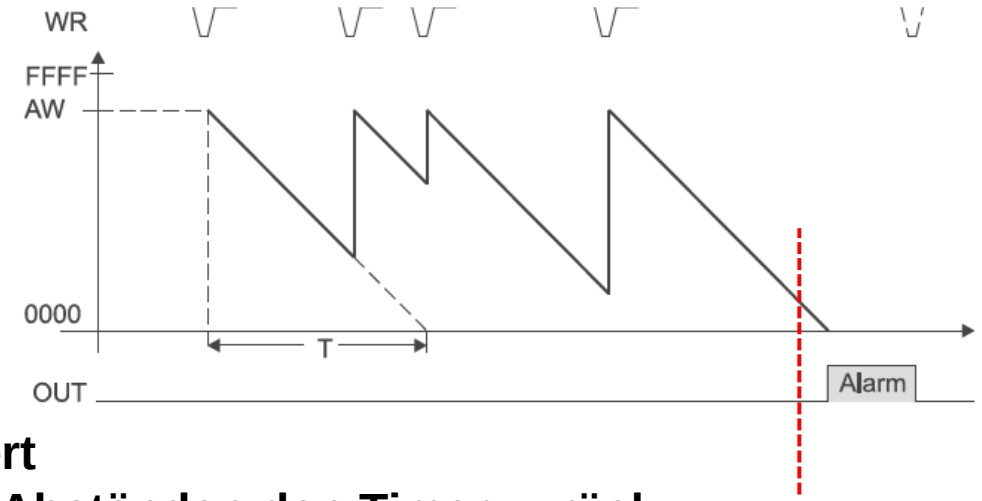
a)Taktschwingung besitzt gleich große Impuls- und Pausenlängen. Bei jeder Initialisierung geht OUT auf LOW und wechselt genau beim halben Anfangswert AW/2 nach HIGH

b)Triggerung des Ausgangs OUT durch ein variables Impuls/Pausen- Verhältnis z.B. durch Verwertung des LSB (Least-Significant-Bytes) des Zählers.

c)Ausgabe eines kurzen Pulses bei Zählerstand Null. Die Länge des Impulses stimmt entspricht dann der Schwingungsdauer des Zähltaktes.

Timer: Als Watchdog

→Ein sog. Watchdog-Timer ist ein Timer, der zur Kontrolle der Abarbeitung eines Programms eingesetzt wird



→Timer wird ständig dekrementiert
→Software setzt in regelmäßigen Abständen den Timer zurück
→Schafft die Software dies nicht, ist sie wahrscheinlich „hängengeblieben“, d.h. der Zähler erreicht dann den Nulldurchgang und es wird ein Impuls erzeugt
→Dieser Impuls kann dann einen Interrupt oder Reset auslösen

→Verhinderung von System-Komplettausfällen durch Softwareversagen

→Watchdogs gibt es nicht nur als Module innerhalb eines Mikrocontrollers, sondern als auch separate ICs

Parallele Schnittstellen (PIO)

➔wie alle anderen Systembausteine auch erscheint die parallele Schnittstelle für die CPU wie ein Register, das gelesen und beschrieben werden kann

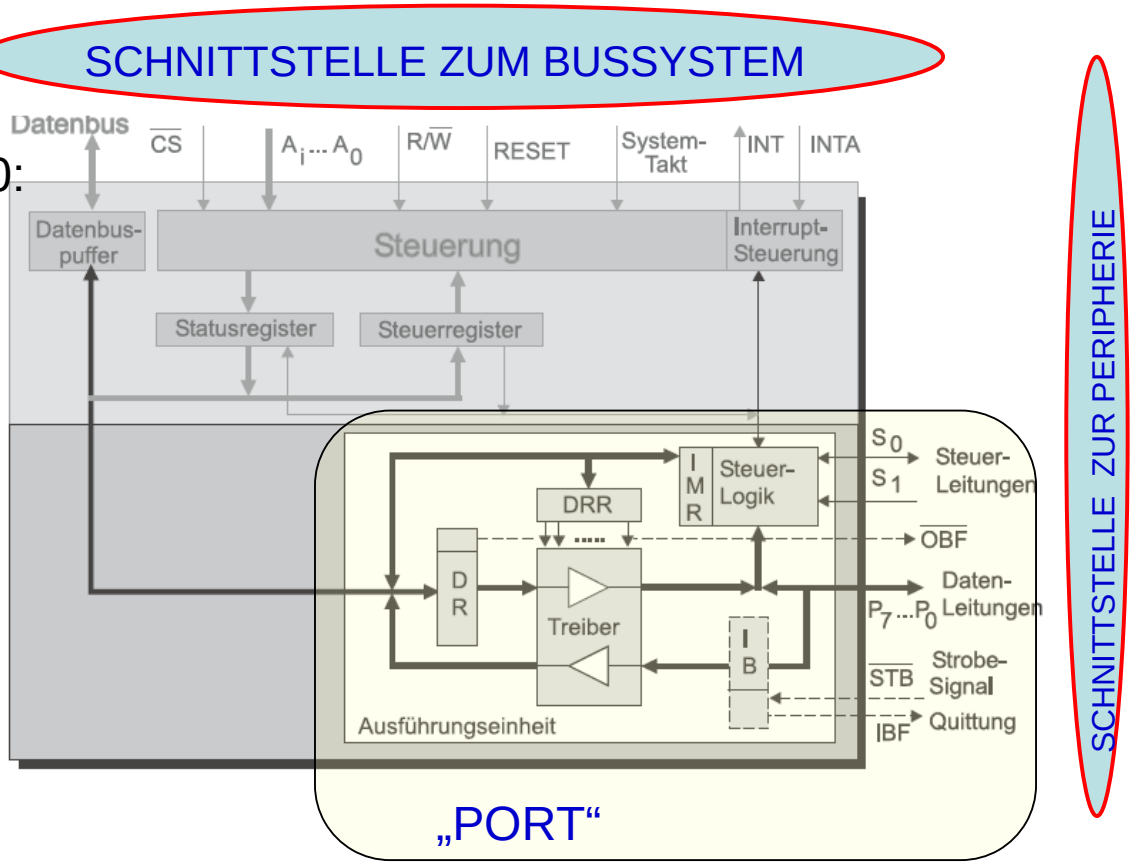
Bestandteile der PIO

Data Direction Register DRR=0:
P ist Eingang
DRR=1: P ist Ausgang

Data Register (DR)
Zwischenspeicher

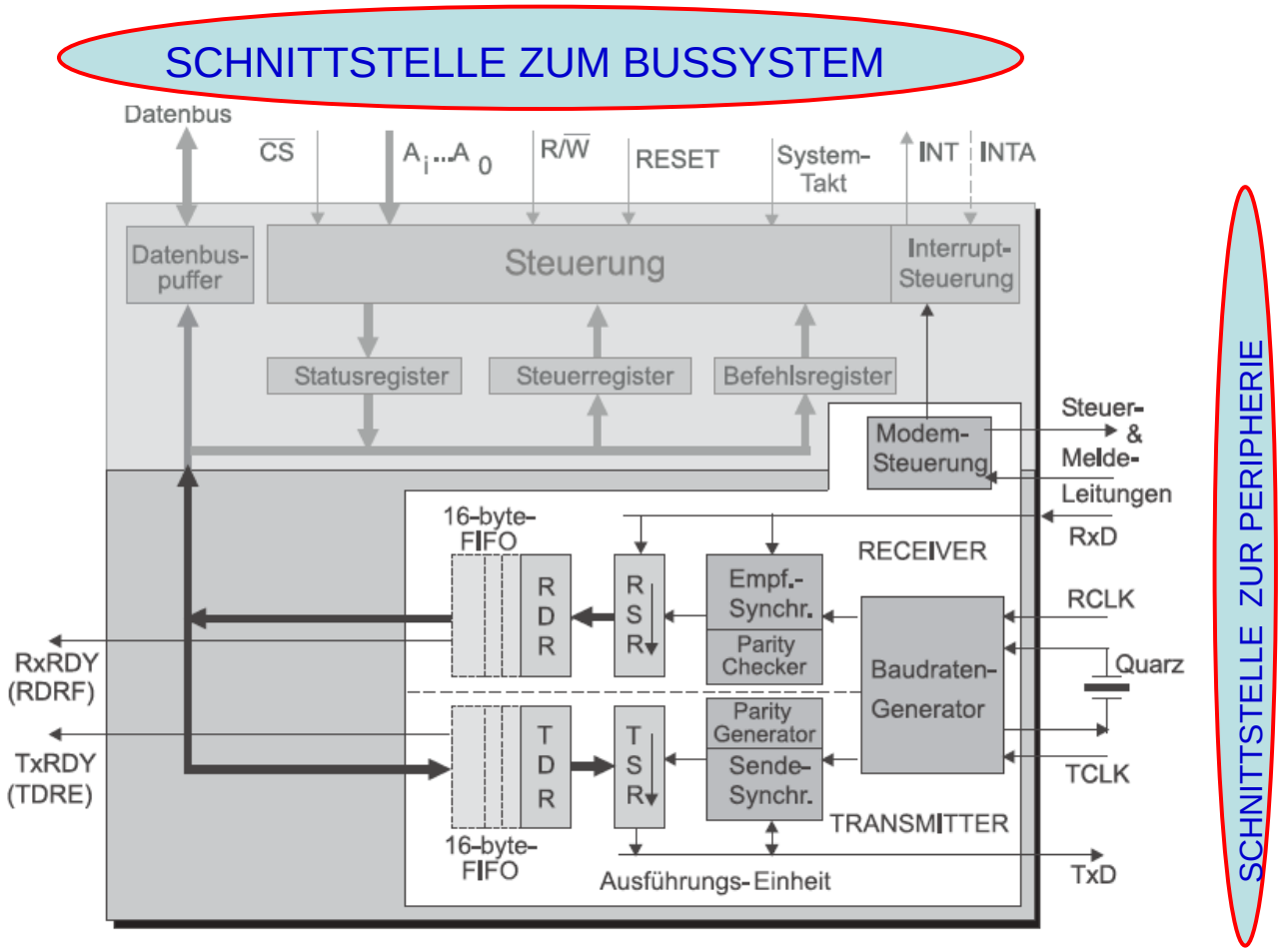
Interrupt Mask Register IMR

Steuerlogik
Treiber

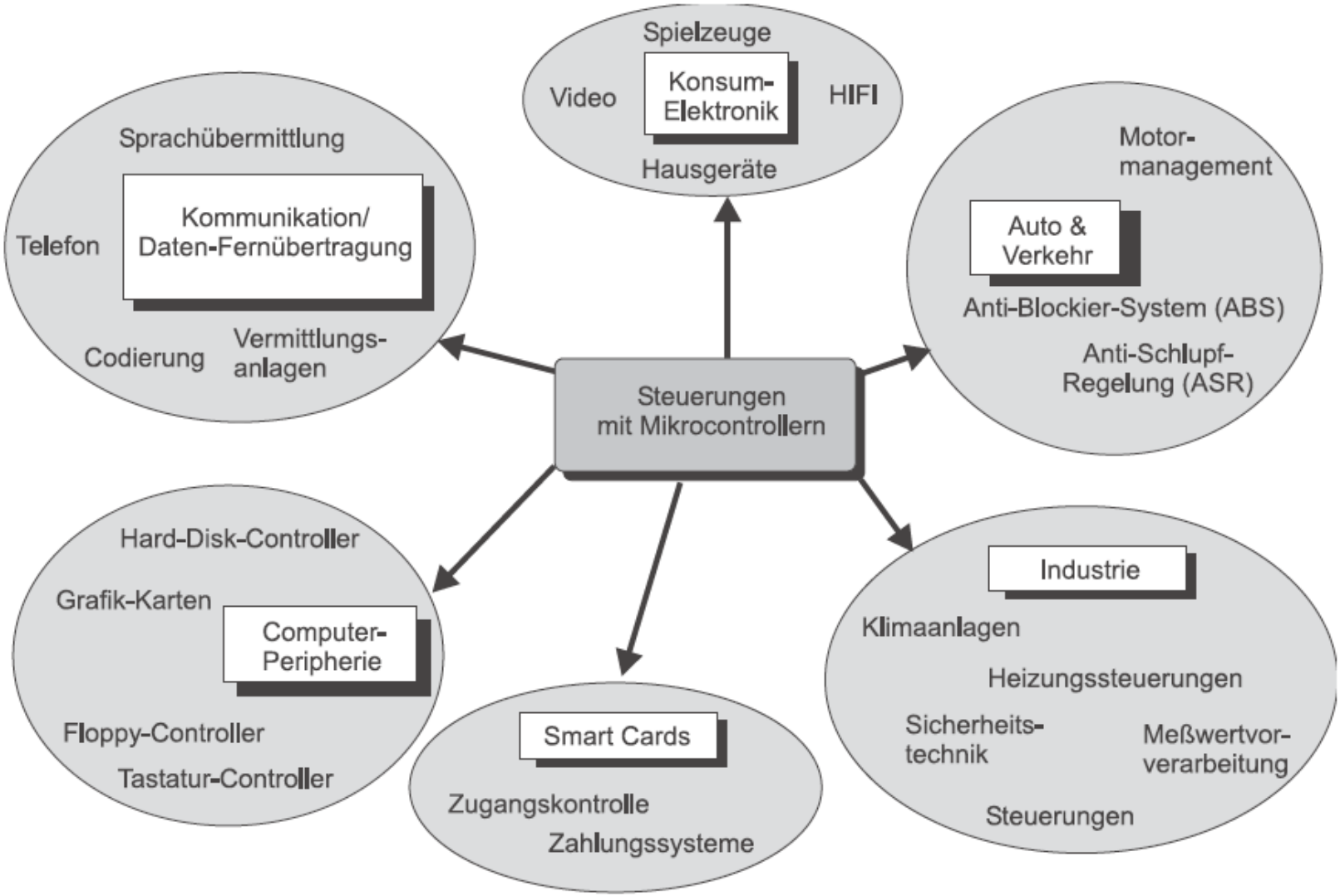


Serielle Schnittstellen

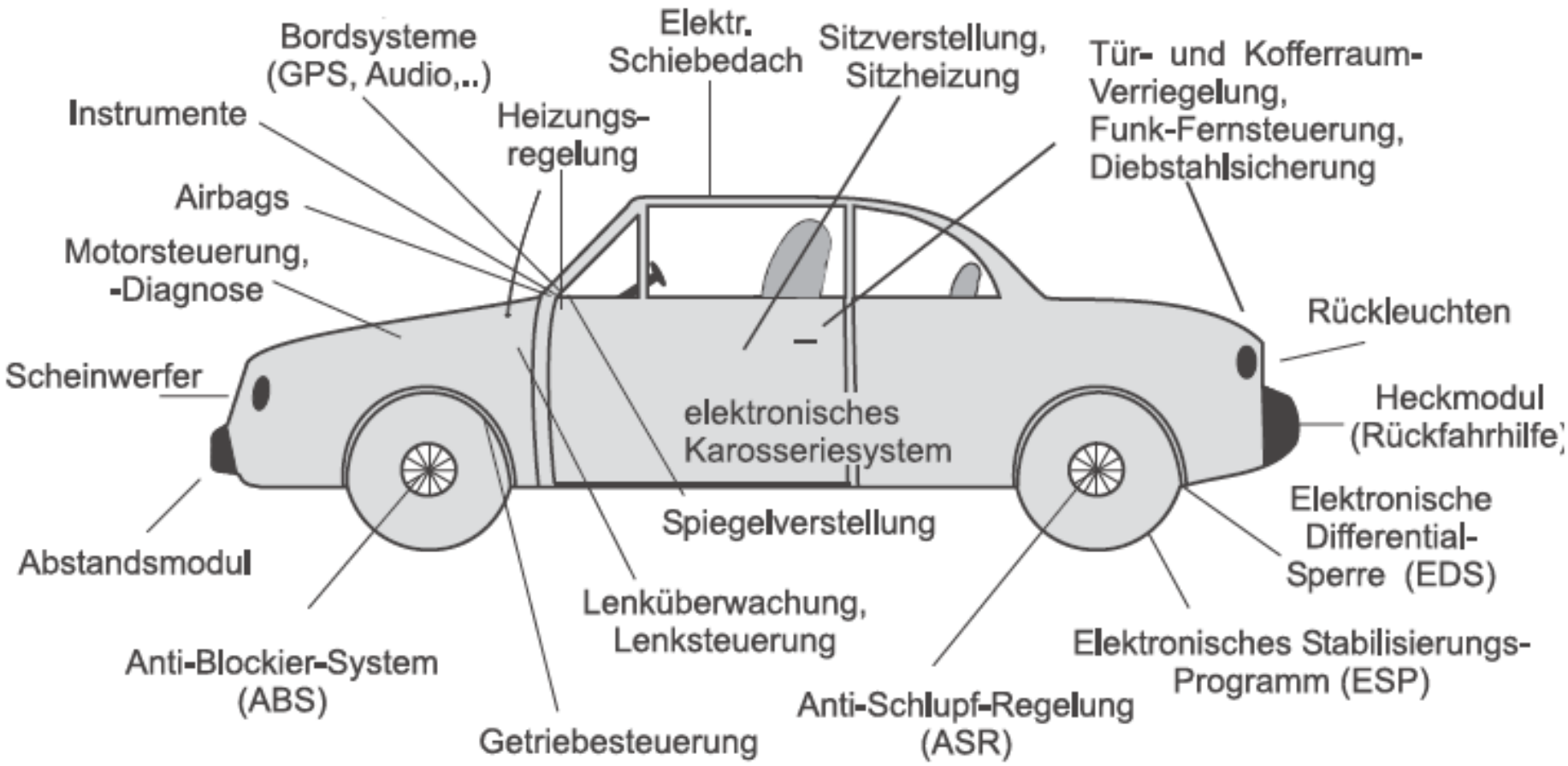
- erscheint der CPU wie ein Register, das gelesen und beschrieben werden kann



Mikrocontroller: Einsatzgebiete

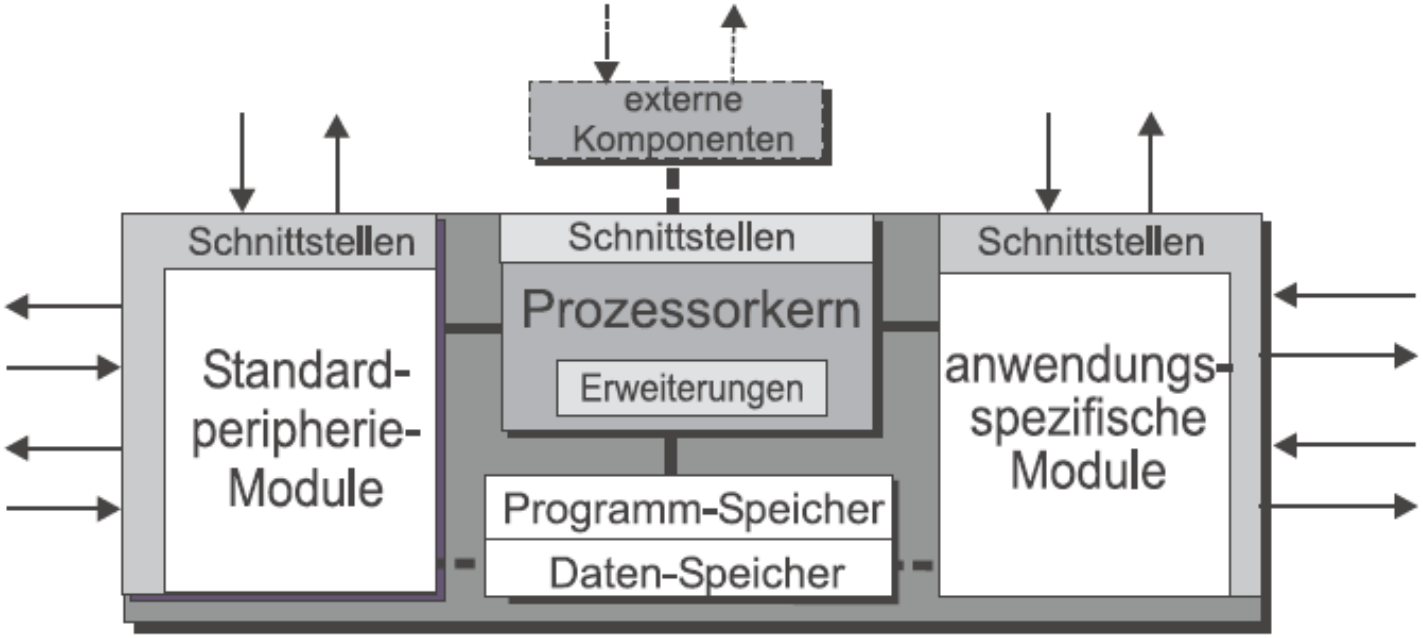


Mikrocontroller: Sensorik Im Fahrzeug



Mikrocontroller: Eigenschaften

- Anwendungssoftware oft mit OS integriert
- Festwertspeicher (PROM, ROM, EPROM, EEPROM, FLASH) als Programmspeicher
- Integration vieler Komponenten/Module auf einem Chip (ASIC)
- Power Modi: Down / Sleep
- Eignen sich gut für Batteriebetrieb
- Üblicherweise werden Watchdog-Timer zur Betriebssicherheit eingesetzt



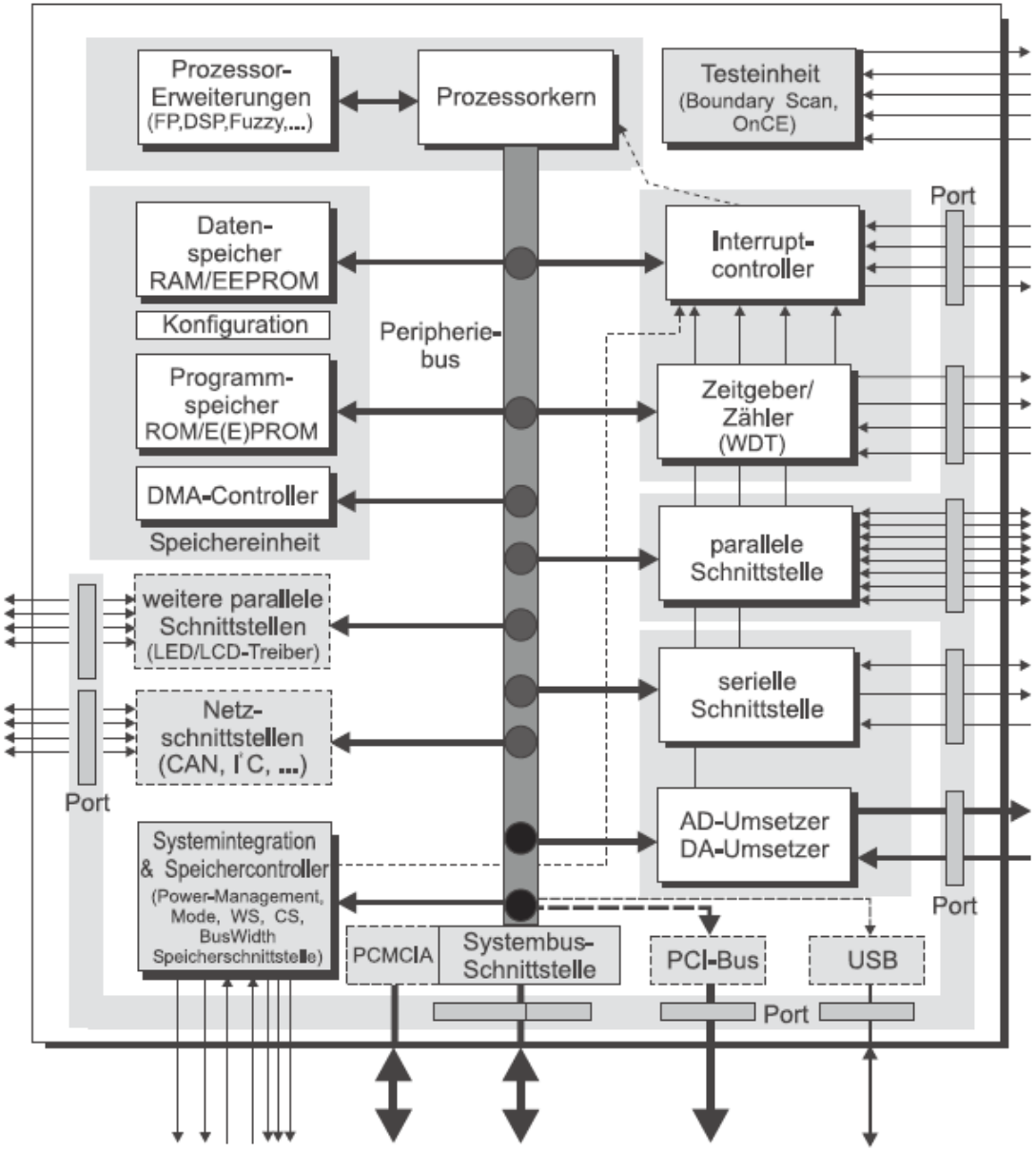
Mikrocontroller: Typischer Aufbau

- Core (RISC oder CISC) mit Integer-Rechenwerk
- Festkomma: Häufig Hardware-Multiplikations/Divisionswerk
- FPU: Eher mit CPU-Erweiterungen
- Speicher : i.d.R. Festwertspeicher mit bis zu 128 KByte
- Adressbereich I/O-Modul-Adressierung oft Memory-Mapped
- Zeitgeber/Zähler-Modul
- DMA-Controller
- Interrupt-Controller (für IRQ von integrierten und externen Komponenten)
- I/O-Ports
- Parallele Schnittstellen: Parallele I/O-Leitungen oft für Steuer-/Kontrollsignale benutzt statt für Daten

Mikrocontroller: Typischer Aufbau

- Asynchrone und synchrone serielle Schnittstellen
- A/D-Umsetzer: oft mehrere Eingänge, über Analog-Multiplexer auf den A/D-Wandler geschaltet
- Seltener: Integrierte D/A-Umsetzer
- Speichercontroller (zur Generation der Steuersignale für unterschiedliche externe Speicher (DRAM, SRAM, ROM, FLASH,…)
- Module zur Leistungskontrolle und unterschiedliche Systemsteueraufgaben
  - ➔ Clockfrequenzkontrolle, Kontrolle zur Spannungsabschaltung, RESET
- Schnittstellen und Busse für spezielle Anwendungen
  - ➔ USB, I^2C, CAN (Fahrzeug), PCI-Bus

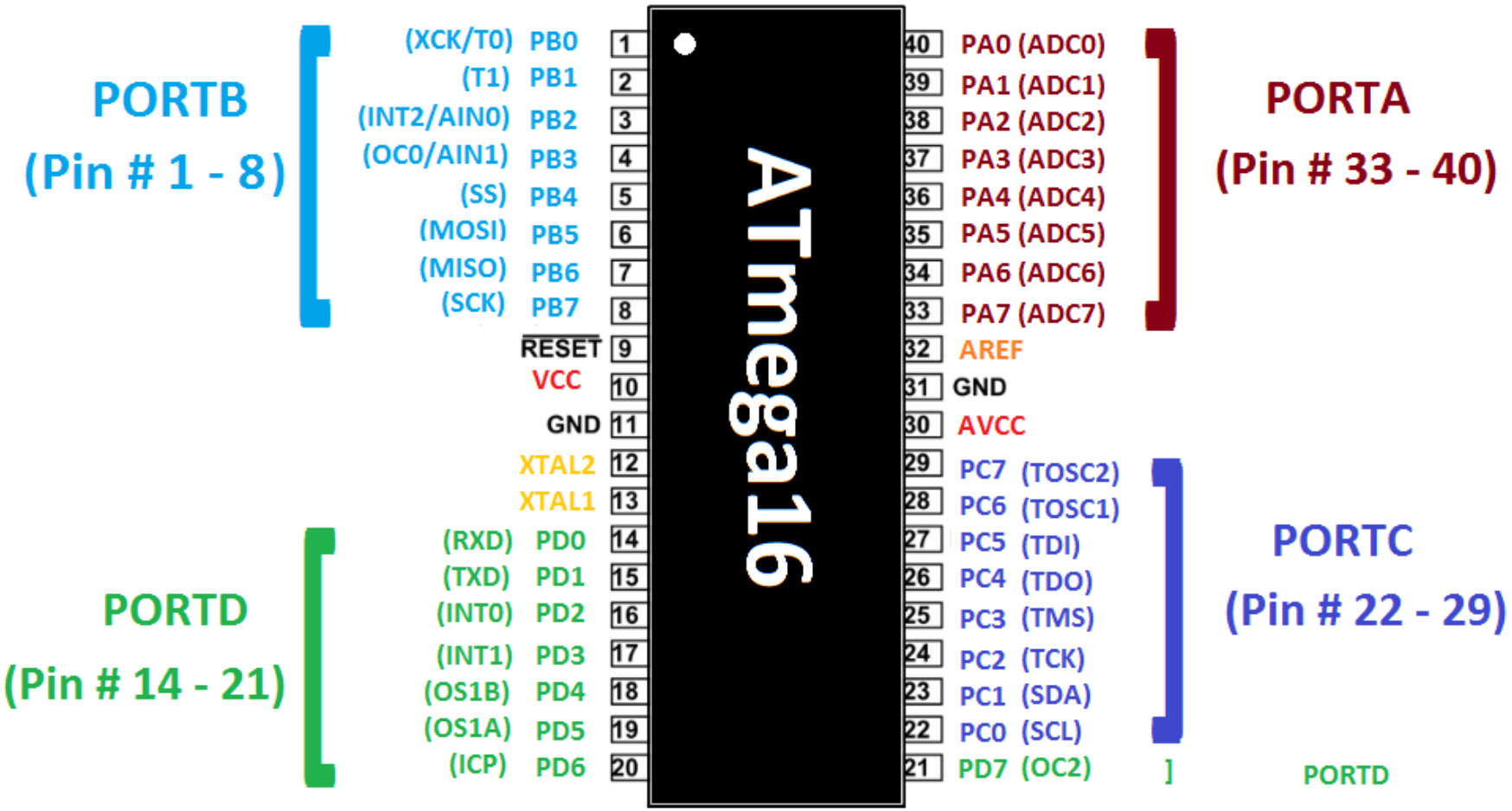
Mikrocontroller: Typische Feinstruktur



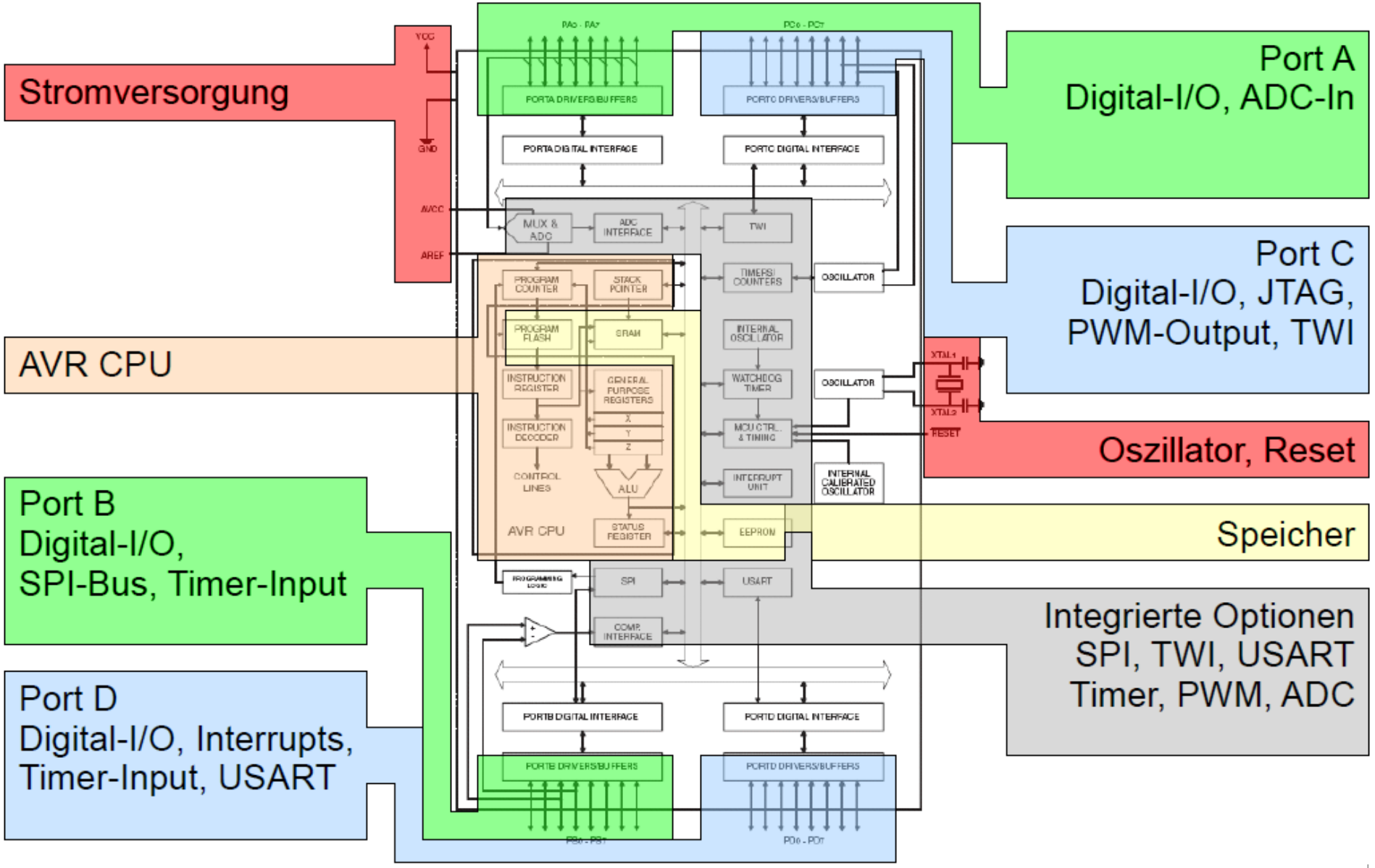
Mikrocontroller: ATMEGA 16 pinout

- bis zu 16 MHz, integrierte Oszillatorschaltung
- 16/8/8 Bit Timer / Counter, acht 10 Bit ADC
- PWM, SPI, TWI, USART, Brown-Out, Watchdog

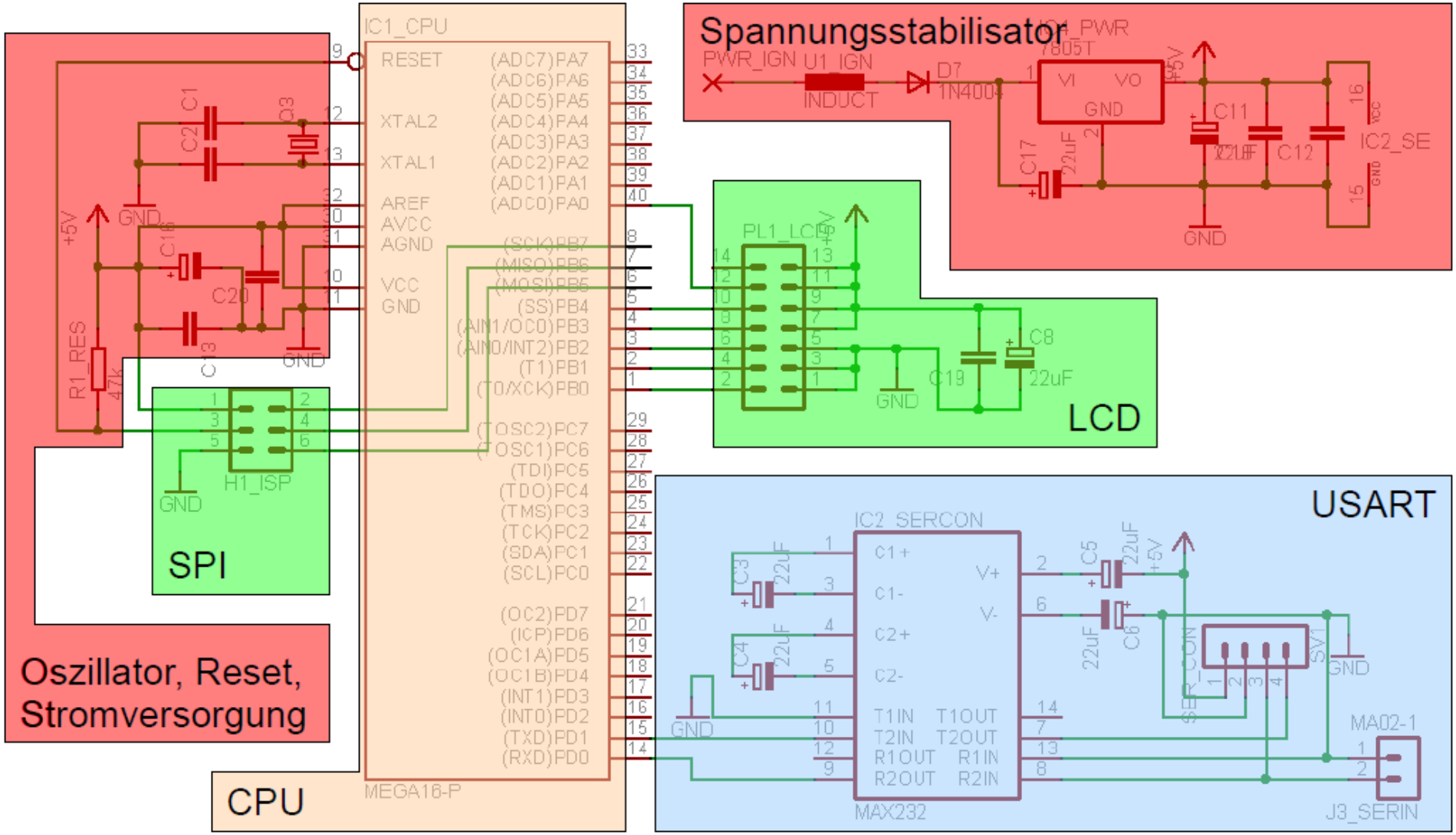
- 40-Pin Gehäuse
- 1 Kilobyte SRAM
- 16 Kilobytes Flash
- 512 Bytes EEPROM



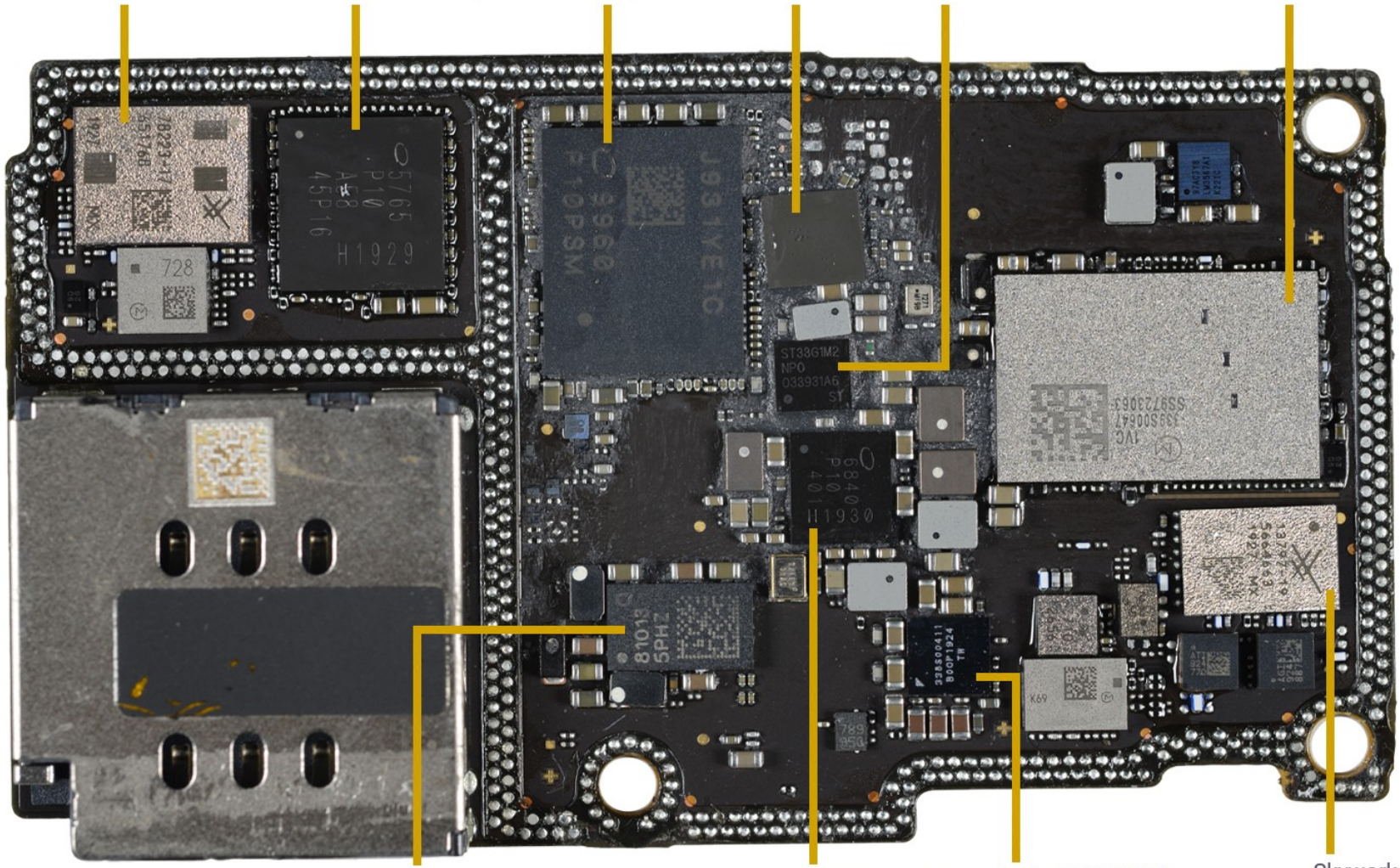
Mikrocontroller: ATMEGA 16, Blockdiagramm



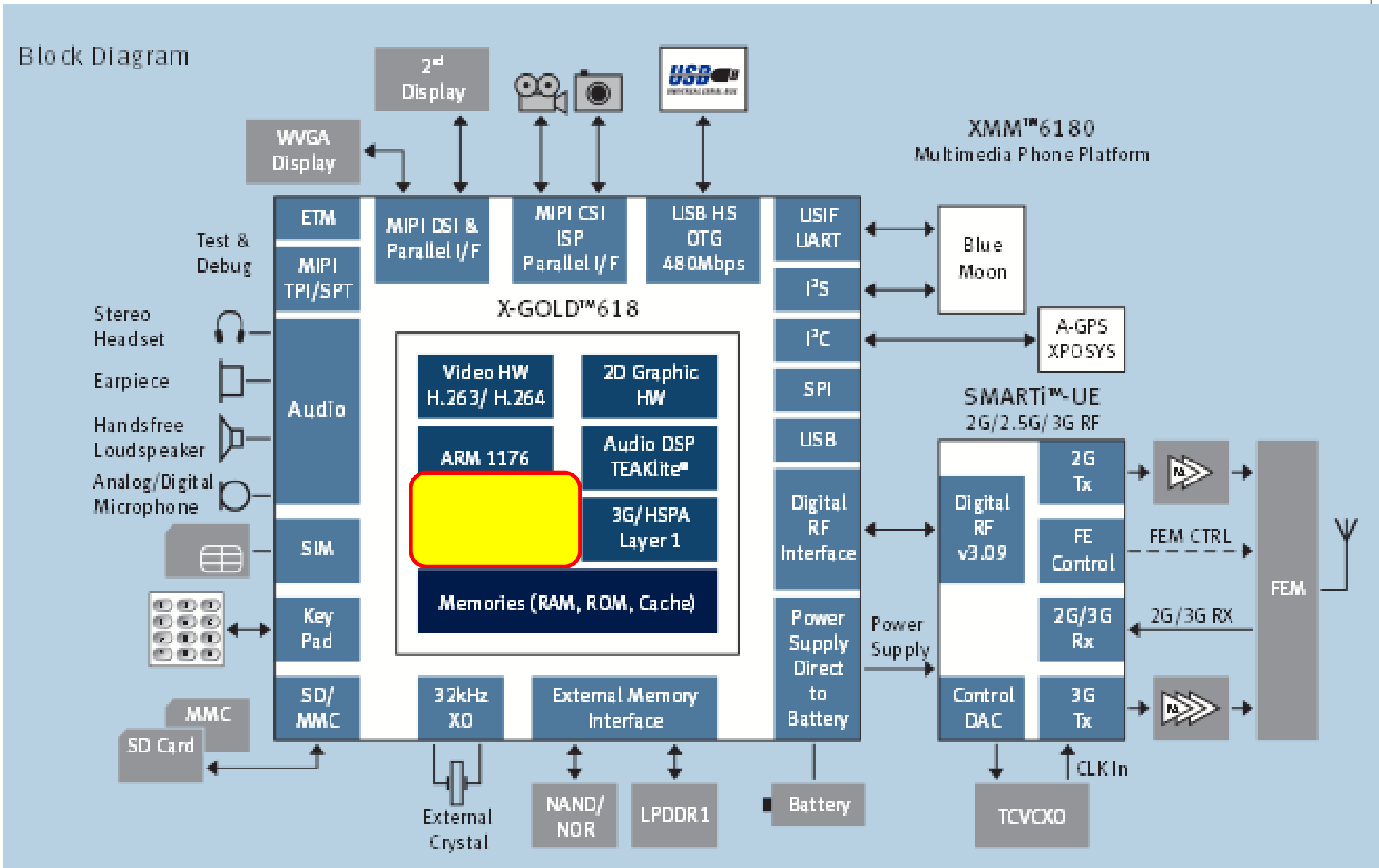
Mikrocontroller: ATMEGA 16, Beispielschaltung



Systemarchitektur: iPhone 11 Pro Max



Systemarchitektur: Infineon Gold X 618



Systemarchitektur: ARM 1176