

Klausur am 12. September 2005, 14:00 Uhr

□ Anmeldung zur TI-Klausur:

- Einwurf der Zulassungsbescheinigung in den Briefkasten im Untergeschoss des Informatikgebäudes am Fasanengarten **bis spätestens 02. September**. Es handelt sich um den gleichen Briefkasten, in dem die Übungsblätter eingeworfen werden **UND**
- **Online-Anmeldung auf der TI-Homepage**

□ Hilfsmittel sind nicht erlaubt

□ Dauer der Klausur:

- **Informatik: 120 Minuten** (14.00 – 16.00 Uhr)
- **Informationswirtschaft: 60 Minuten** (14.00 – 15.00 Uhr)

□ Studentenausweise unbedingt in die Klausur mitbringen

□ Hörsaal-Verteilung wird rechtzeitig bekannt gegeben (TI-Homepage)



Tag der Informatik 2005

Freitag, 15. Juli 2005, im Hörsaal am Forum

Programm:

- | | |
|-----------|--|
| 14:00 Uhr | Wissenschaftliches Kolloquium
Wissenschaftliche Präsentation des Instituts für Theoretische Informatik |
| 16:00 Uhr | Festvortrag (Prof. Dr. Wolfgang Paul)
Garantiert fehlerfreier Entwurf von Rechnersystemen |
| 17:30 Uhr | Akademische Feier
Grußworte
Semesterbericht der Dekanin Prof. Dr. Zitterbart
Auszeichnung der besten Vorlesungen
Verabschiedung der Absolventen und Promovierten
Auszeichnung der jahrgangsbesten Absolventen |

Fakultätsfest Informatik

Freitag, 15. Juli 2005, am Hörsaal am Forum

Biergarten, Grillstand, Cocktails

19:30 Uhr Fassanstich durch die Dekanin
Live-Musik: **Der Hausmeister**

20:45 Uhr Live-Band: **Willenlos**

ab 23 Uhr Disko im Foyer des Hörsaals

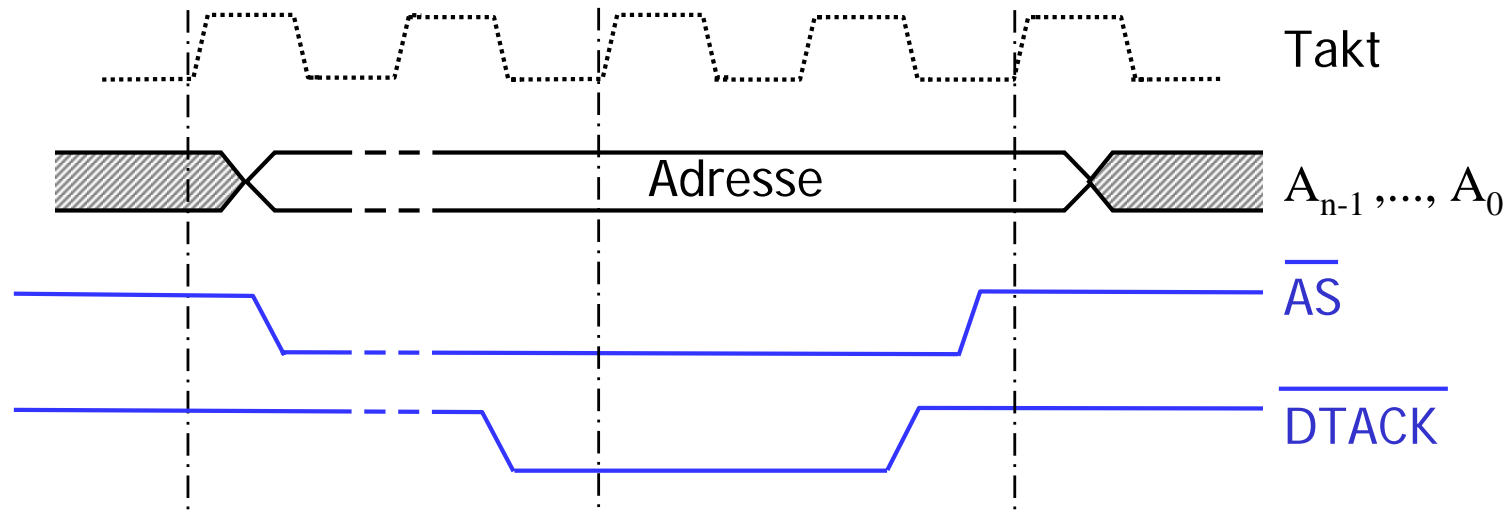
Helfer gesucht: <http://mathe-info.fachschaft.uni-karlsruhe.de>

Kapitel 9

- Zeitverhalten des Bussystems
- Systemsteuer- und Schnittstellenbausteine
- Ausnahmebehandlung



Asynchroner Systembus



Timing

- Auswahl der Übertragungsrichtung wieder durch R/\overline{W}
 - Mit $\overline{AS} = 0$ zeigt die CPU an, dass sie eine gültige Adresse auf den Adressbus gelegt hat
 - Mit $\overline{DTACK} = 0$ zeigt der Speicher (Komponente) an, dass er die Daten zur Verfügung gestellt (Lesen) oder übernommen (Schreiben) hat
 - Zwischen $\overline{AS} = 0$ und $\overline{DTACK} = 0$ kann eine beliebige Zeitspanne liegen
 - Wird $\overline{DTACK} = 0$, so nimmt die CPU die Adresse wieder vom Adressbus und setzt \overline{AS} wieder zu 1
 - Daraufhin nimmt der Speicher (Komponente) das Datum vom Datenbus und setzt \overline{DTACK} wieder zu 1
- ➔ vollständig asynchroner Übertragungsablauf, Anschluss von Komponenten mit fast beliebiger Zugriffszeit möglich



Asynchroner Systembus

- Zeitliche Abläufe werden durch Handshake-Signale gesteuert

Handshake-Signale:

- \overline{AS} (Address Strobe) von CPU
 - \overline{DTACK} (Data Transfer Acknowledge) von Speicher/Komponente
-
- Systemtakt spielt keine Rolle mehr für die Synchronisation der Übertragung (nur noch für die Synchronisation der Signale : synchrones Steuerwerk)



Beispiele

- Beispiele für asynchronen Systembus:
Motorola 68000 - 68030
- Beispiele für semi-synchronen Systembus:
Intel-Prozessoren, Motorola 68040



Multiplex-Bus

Nötig, falls bestimmte Gruppen von Signalen zeitlich hintereinander über dieselbe Busleitungen geschickt werden müssen (z. B. zur Einsparung von Busleitungen)

Beispiel:

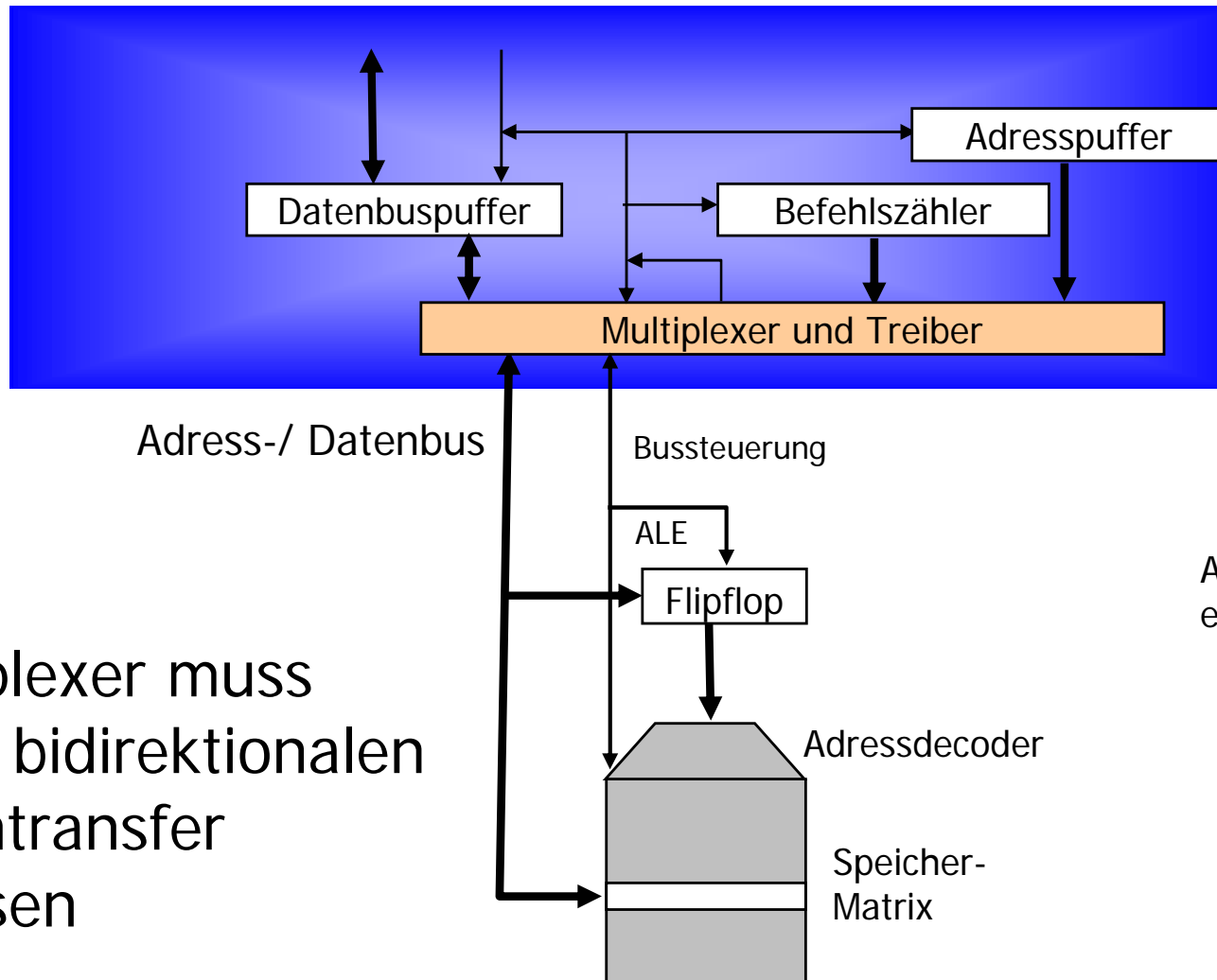
gemeinsamer Daten- und Adreßbus (Bsp.: PCI-Bus)

Die Adresse zur Übertragung eines Datums vom bzw. in den Speicher muss während der gesamten Zugriffszeit vorliegen

➔ Speichern der Adresse in Flipflops (Latches)



Multiplex-Busschnittstelle

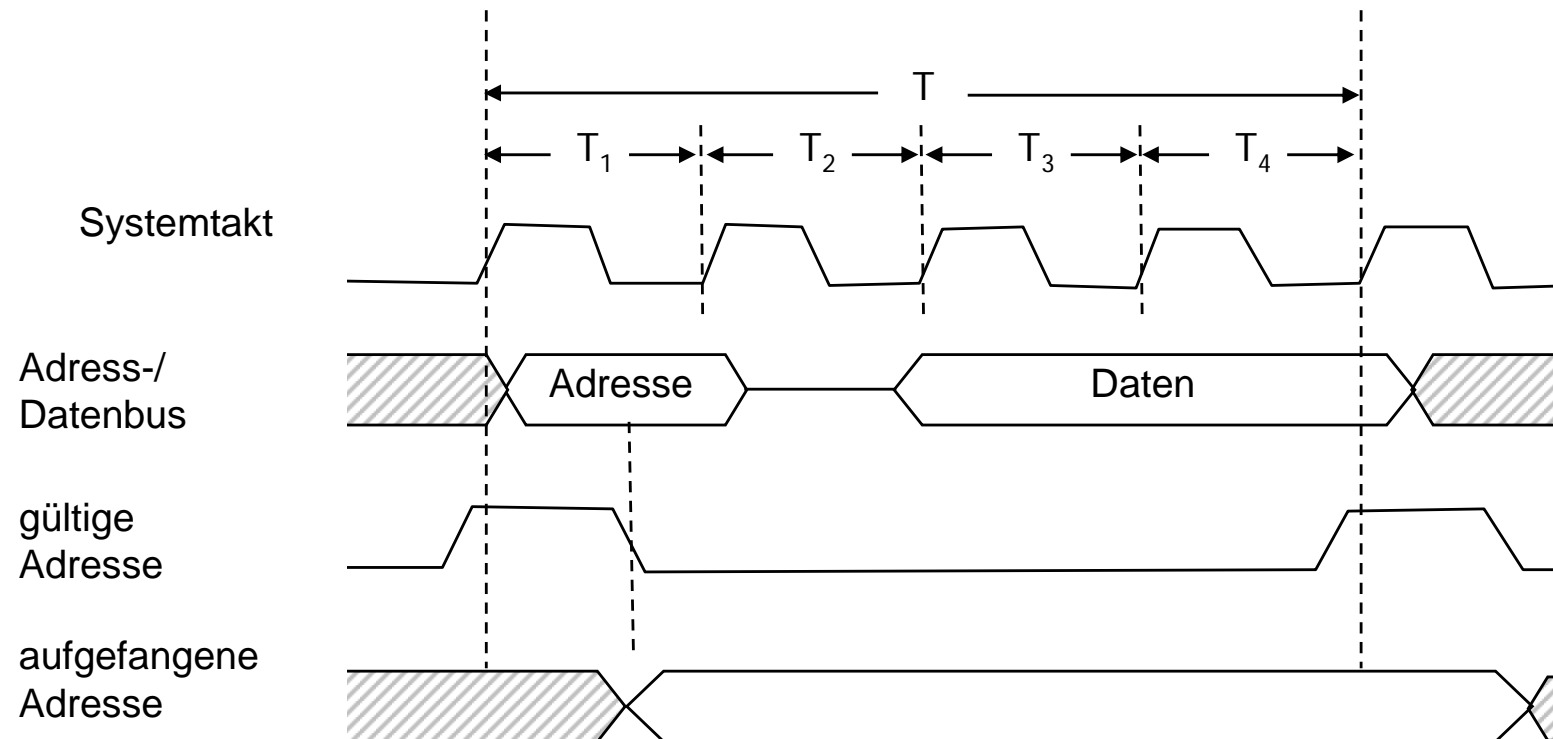


Multiplexer muss einen bidirektionalen Datentransfer zulassen

ALE zeigt das Vorliegen einer gültigen Adresse an

Zeitverhalten des Multiplexbusses

Daten/Adressen-Multiplex-Betrieb



Daten/Adress-Multiplex-Betrieb

- gültige Adresse auf dem gemeinsamen Bus wird durch ein Signal angezeigt, z. B. \overline{AS} (Address Strobe) oder \overline{VMA} (Valid Memory Address) oder \overline{ALE} (Address Latch Enable) oder ...
- mit der aktiven Flanke dieses Signals (im Beispiel fallende Flanke) wird die Adresse in ein vor den Speicher geschaltetes Adress Flipflop übernommen → stabile Adresse am Speicher
- Danach kann der Bus umgeschaltet werden und nun die Daten über die gleichen Leitungen geschickt werden

Weitere Multiplex-Möglichkeiten:

- höher und niederwertige Adressbits (z. B. bei dynamischen Speicherbausteinen)
- höher und niederwertige Datenbits
- gemischte Formen (z. B. niederwertige Adress und Datenbits, ...)



Systemsteuerbausteine

Übernehmen Funktionen, die aus technischen oder Kostengründen nicht im Prozessor integriert sind.

➤ **Nicht programmierbare Systemsteuerbausteine:** führen fest vorgegebene, vom Prozessor nicht beeinflussbare Funktion aus, z. B.

- Taktgeneratoren: Systemtakt und Synchronisationssignale
- Bus Steuerbausteine (Bus Controller)
- Steuerung des Systembuszugriffs (Bus Arbiter)
- Steuerbausteine für DRAM (Dynamic RAM Controller): Auffrischen



Systemsteuerbausteine

Übernehmen Funktionen, die aus technischen oder Kostengründen nicht im Prozessor integriert sind.

➤ **„Programmierbare“ Systemsteuerbausteine:** Funktionsweise kann durch dieser Bausteine kann durch Steuerworte vom Prozessor beeinflusst werden, z. B.

- DMA Controller
- Cache Controller
- Speicherverwaltungsbausteine (MMU)
- Zeitgeber / Zähler Bausteine
- Echtzeit Uhren (Real Time Clocks)
- Unterbrechungs Steuerbausteine (Interrupt Controller)



Schnittstellenbausteine (I/O-Controller)

Bindeglied zwischen dem Prozessor, dem Hauptspeicher und der Peripherie.

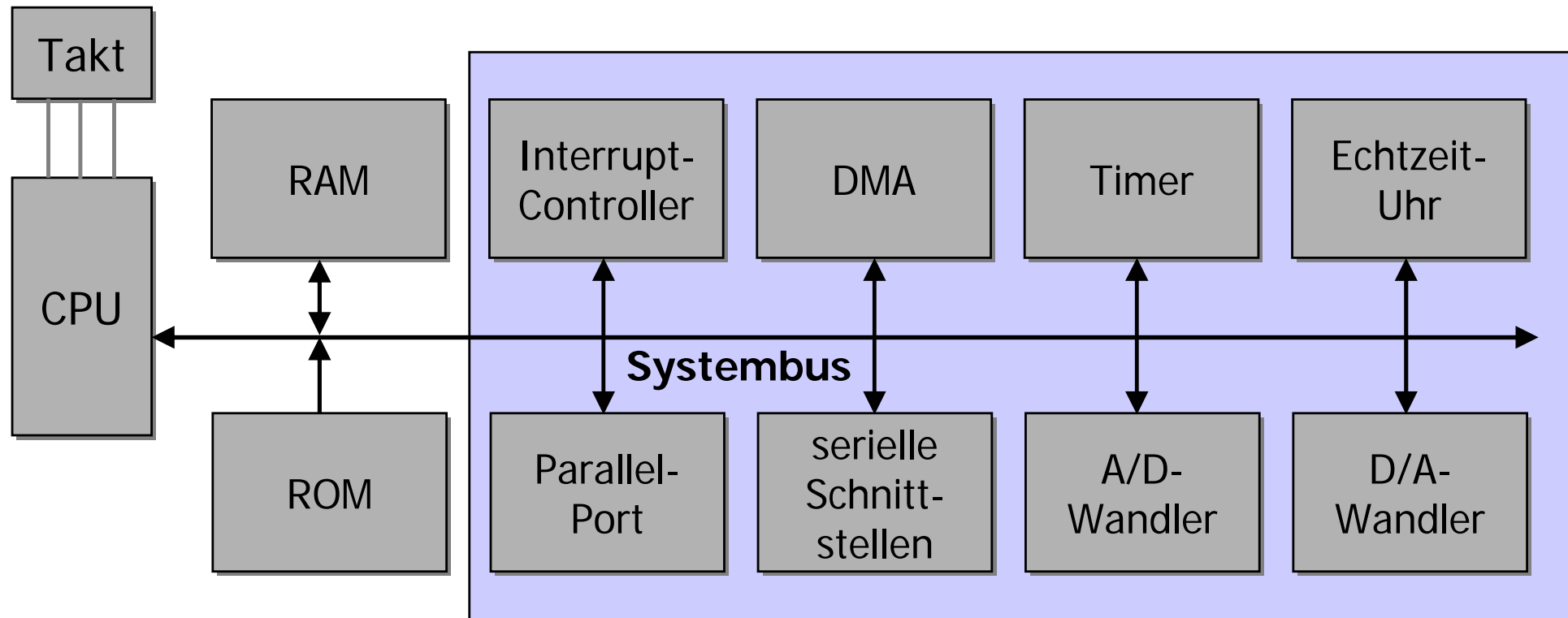
Aufgaben:

- Pufferung von Ein-/Ausgabe Daten, Anpassung unterschiedlicher Arbeitsgeschwindigkeiten im System
- Umsetzung von Daten: parallel/seriell, digital/analog, ...
- Erzeugung von Steuersignalen für Peripheriegeräte, z. B. zur Synchronisation
- Annahme und Erzeugung von Unterbrechungsanforderungen für Peripheriegeräte

Programmierbare Systemsteuerbausteine und Schnittstellenbausteine werden als **Systembausteine** bezeichnet.



Systembausteine in einem Mikrorechner



Speicherbezogene und isolierte Adressierung

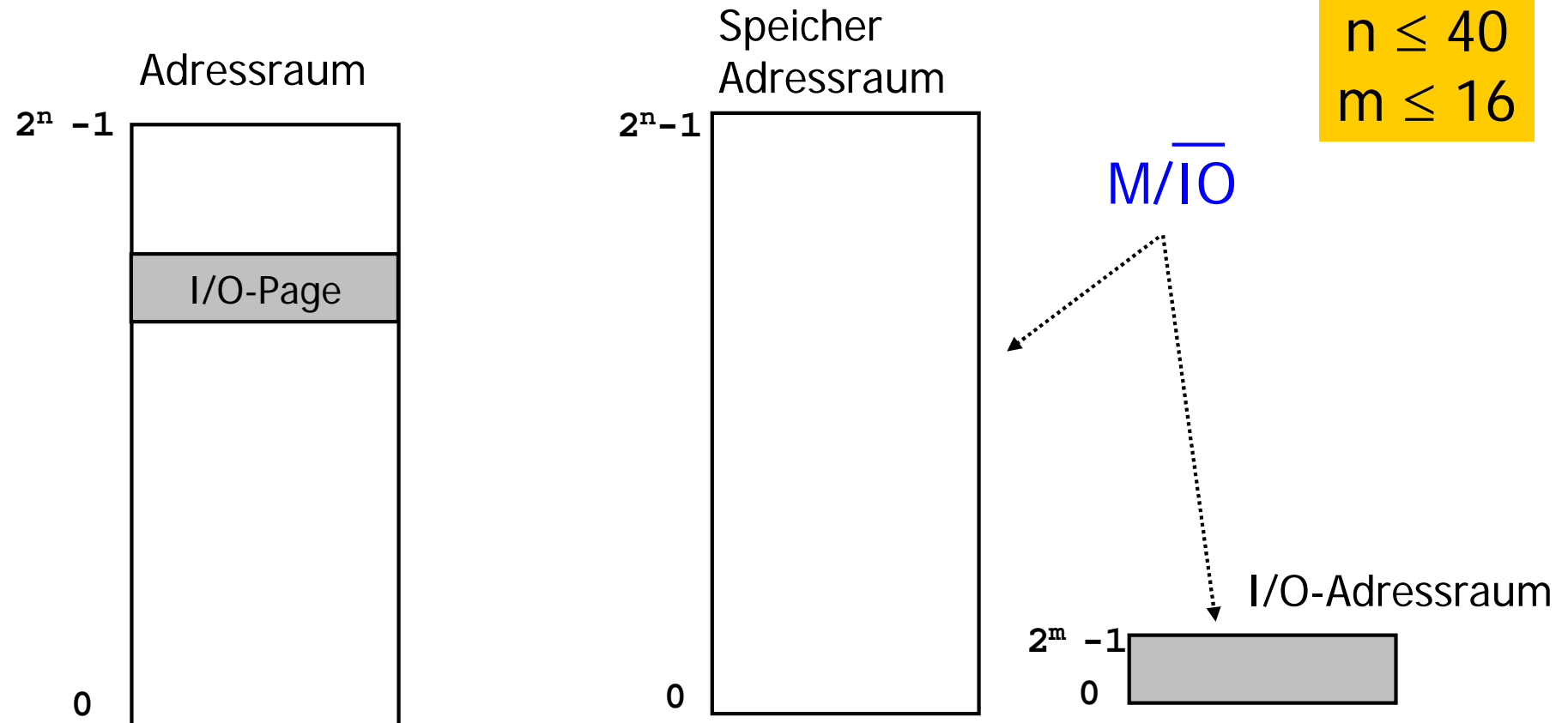
Jeder programmierbare Systembaustein erscheint für den Prozessor wie ein kleiner Satz von Registern, die unter einem zusammenhängendem Block von Adressen (Portadressen) angesprochen werden können.

Zwei Adressierungstechniken:

- **Speicherbezogene Adressierung (Memory Mapped I/O):**
Der Adressblock wird in einem gemeinsamen Adressraum mit allen anderen Speicheradressen untergebracht (680XX & RISC)
- **Isolierte Adressierung (Isolated IO):** zwei getrennte Adressräume für Speicher und Ein-/Ausgabe. Auswahl des Adressraumes durch ein zusätzliches Signal (memory input/output: $\overline{M/\overline{IO}}$). Intel Prozessoren



Adressierung von Peripherie-Bausteinen



**Speicherbezogene
Adressierung**

**Isolierte
Adressierung**



Adressierung von Peripherie-Bausteinen

➤ Speicherbezogene Adressierung:

Kein Unterschied zwischen Speicheradresse und Adresse eines Registers eines Peripherie-Bausteins,

Häufig wird ein zusammenhängender Speicherbereich für Peripherie-Bausteine verwendet: I/O-Page

➤ Isolierte Adressierung:

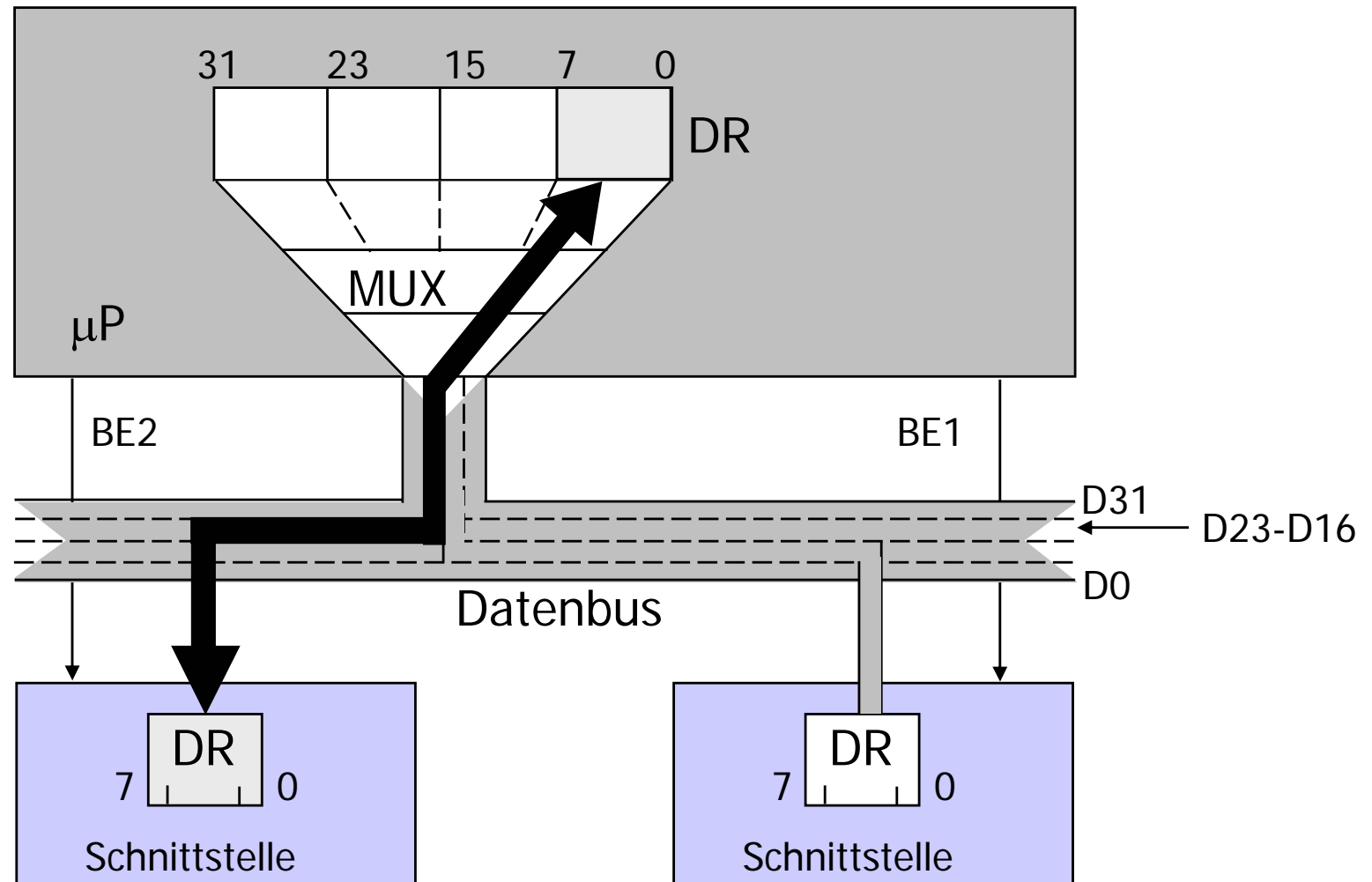
getrennte Adressräume für Speicher und Peripherie (eigener I/O-Adressraum)

Auswahl des Adressraums durch M/\overline{IO} -Signal



Anschluss der Schnittstellenbausteine an dem μP

8-bit-Schnittstelle am 32-bit-Prozessor

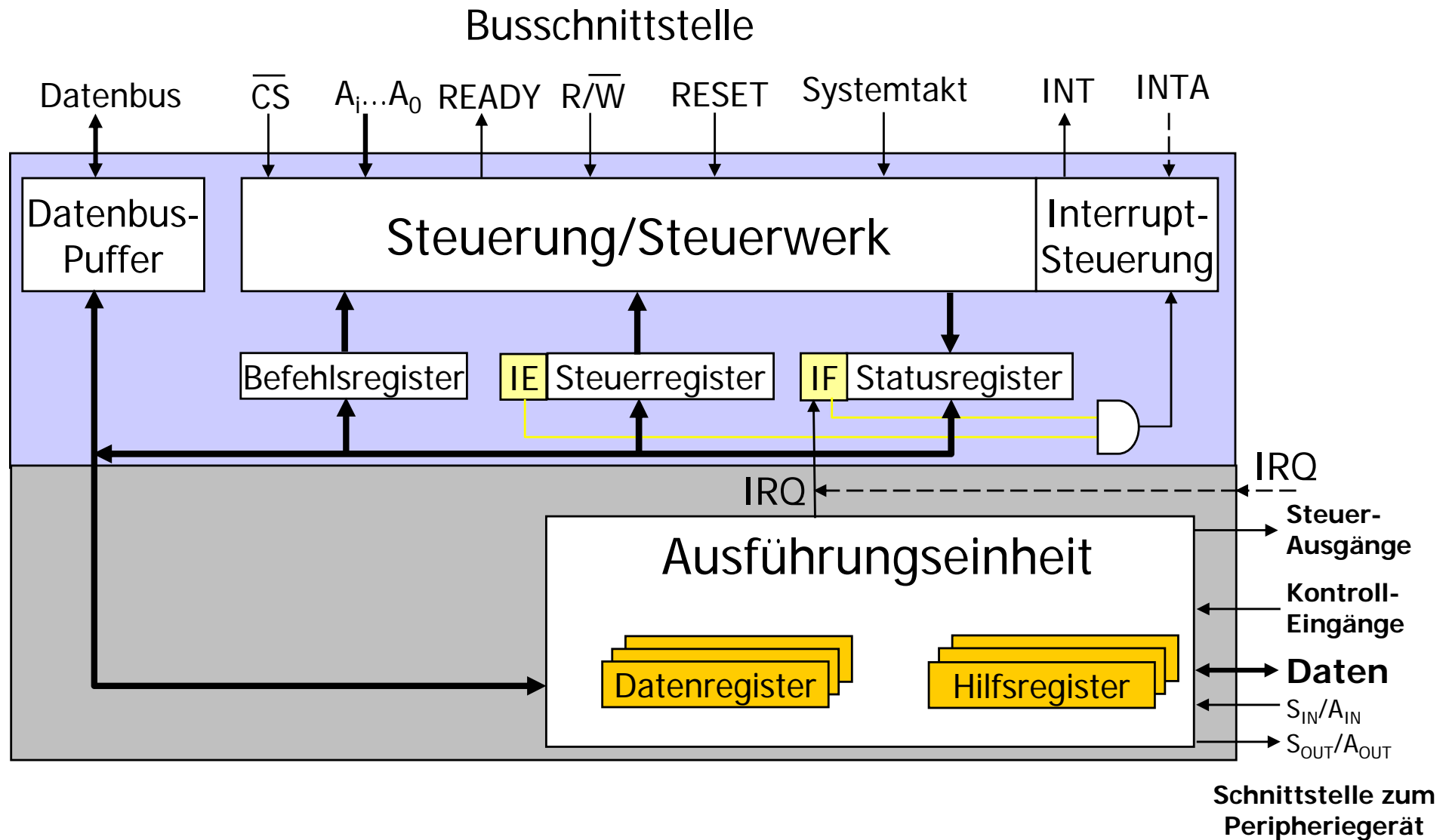


Anschluss der Schnittstellenbausteine an dem μP

- ❑ Noch sehr viele Schnittstellenbausteine haben eine Datenbusbreite von 8 Bit.
- ❑ Grund:
 - Viele Eingabegeräte sind zeichenorientiert
 - Für moderne 16, 32, und 64 Bit Prozessoren werden noch die für ältere Prozessoren entwickelten Schnittstellenbausteine eingesetzt.
- ❑ Moderne Prozessoren unterstützen verschiedene Operandenlängen durch zusätzliche Steuersignale ($BE_0, \dots, BE_{7/3}$) → Verbindung einzelne Bytes des μP -Datenbusses mit Schnittstellenbausteine möglich.



Prinzipieller Aufbau eines Systembausteins



Prinzipieller Aufbau eines Systembausteins

Steuerwerk:

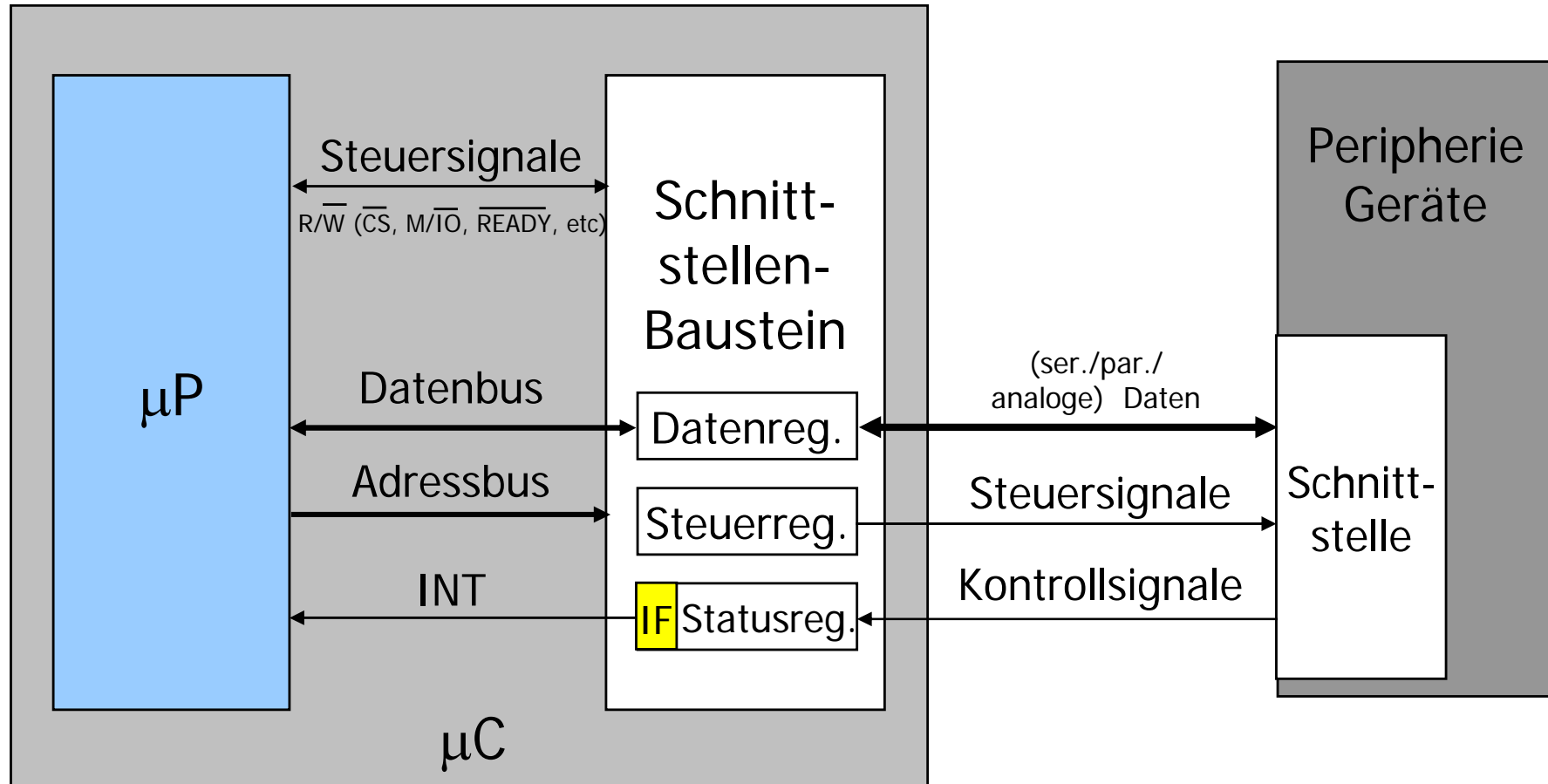
- Steuerung der internen Komponenten (Register, Multiplexer, ...) und Datenpfade
- Schnittstelle zum Prozessor
- Register zur Bausteinprogrammierung: Statusregister (Bausteinstatus), Steuerregister (Betriebsart), Befehlsregister (aktuelle Operation)

Ausführungseinheit:

- Stellt die spezifischen Bausteinfunktionen zur Verfügung
- Verschieden Daten- und Hilfsregister (je nach Funktion)
- Schnittstelle zum Peripheriegerät



Schnittstellenbaustein zwischen μP und Peripheriegerät



Schnittstellenbaustein zwischen μ P und Peripheriegerät

- **Datenleitungen:** unidirektional/bidirektional, parallel/seriell
- **Steuerleitungen:** zum Steuern des Peripheriegeräts vom Prozessor, z. B. Ein-/Ausschalten des Peripheriegeräts, Synchronisation der Übertragung
- **Meldeleitungen:** um dem Prozessor Informationen über den Zustand des Peripheriegeräts zu übermitteln, z. B. Peripheriegerät bereit, Störung, ...



Ein-/Ausgabe Verfahren

Für Datenaustausch existieren drei Varianten:

- **DMA-Übertragung:** große Übertragungsraten. Prozessor wird auch entlastet.
- **Interruptgesteuerte Ein-/Ausgabe:** jeder Datentransfer vom oder zum Peripheriegerät wird über die INT-Leitung angefordert.
 - **Vorteil:** Prozessor kann zwischen den Datentransfers andere Aufgaben erledigen (wichtig bei langsamen Peripheriegeräten)
 - **Nachteil:** erhöhter Zeitaufwand durch die Programmumschaltung zur Interrupt-Routine.



Ein-/Ausgabe Verfahren

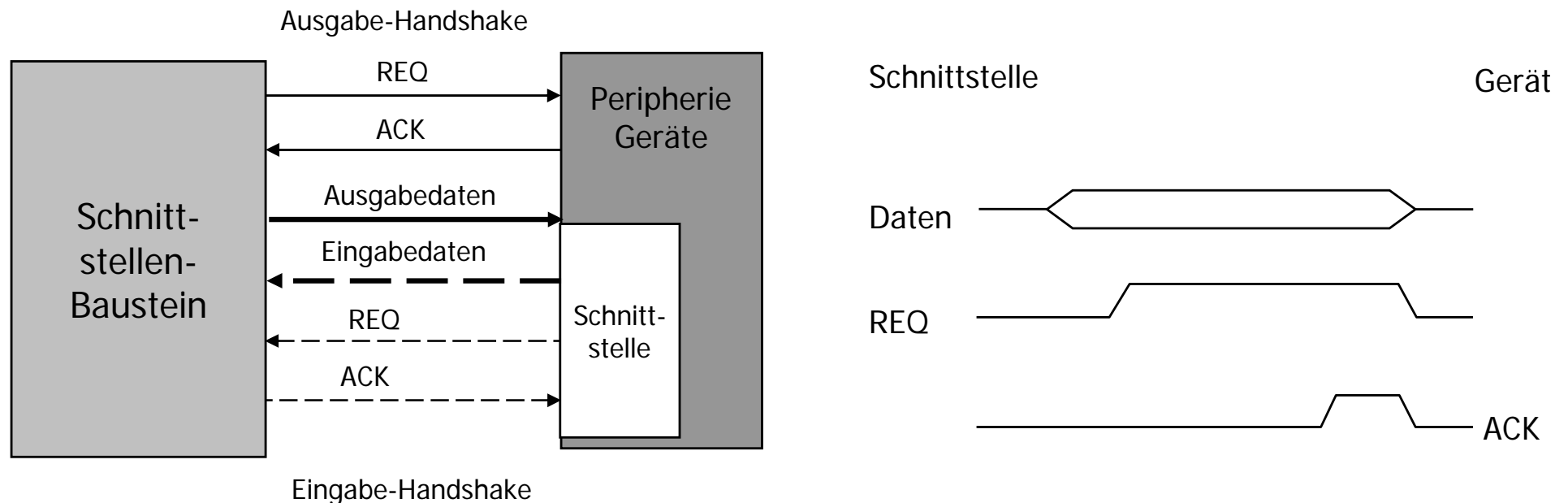
- **Programmierte Ein-/Ausgabe:** Prozessor fragt ständig das Statusregister des Peripheriebausteins ab, ob er bereit ist ein Datum zu übertragen. Zwischen den Übertragungsanforderungen kann der Prozessor andere Aufgaben erledigen oder in einer Schleife ausschließlich das Statusregister abfragt (*busy waiting*: Aktives Warten)
 - **Vorteil:** hohe Reaktionsgeschwindigkeit, insbesondere bei nur 1 Peripheriegerät und kein *busy waiting*
 - **Nachteil:** Reaktionsverzögerung, wenn gleichzeitig mehrerer Peripheriegeräte abgefragt werden müssen.



Synchronisation der Datenübertragung zwischen Schnittstelle und Peripheriegerät

Hardware- oder Softwaremäßig

Beispiel für eine hardwaremäßige Synchronisation:



Synchronisation der Datenübertragung zwischen Schnittstelle und Peripheriegerät

- ❑ Getrennte Datenwege für beide Übertragungsrichtungen (**Volduplex-Betrieb**).
- ❑ **Halbduplex-Betrieb:** Daten werden bidirektional über dieselben Leitungen ausgetauscht.
- ❑ Für jede Richtung eine *Request*-Leitung (*REQ*) und ein *Acknowledge*-Leitung (*ACK*)
- ❑ Schnittstellenbaustein legt das Datum auf seinen 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



Interrupt-Controller



Interrupt Controller

- ❑ Behandlung von AUSnahmen wurde in der Übung behandelt
- ❑ Hier: Behandlung mehrerer Interrupt-Quellen



Behandlung mehrerer Interrupt-Quellen

Interrupt-Quellen werden durch Interrupt Controller zyklisch abgefragt (Interrupt-Flag im Statusregister des Controllers).

Komponenten mit gesetztem Interrupt-Flag

→ Abfrage abbrechen und die entsprechende Interruptroutine abrufen.

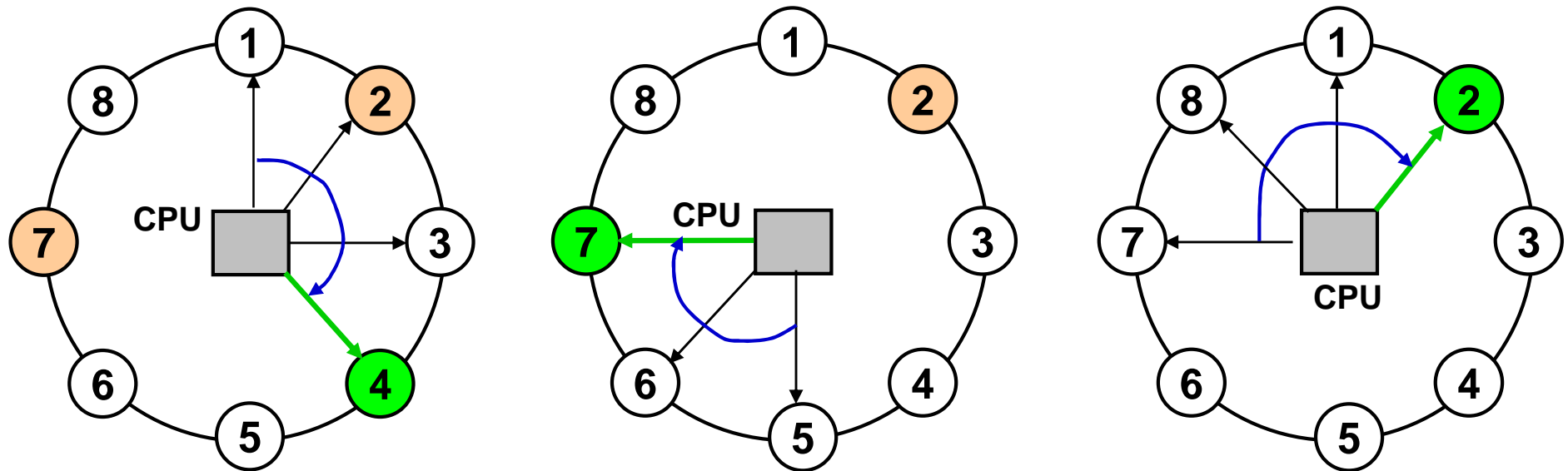
Nach (und auch während) der Abarbeitung eines Interrupts können weitere Anforderungen auftreten.

→ Zwei Alternativen zur Behandlung



Polling: 1. Variante

Zyklische Abfrage wird mit der Komponente fortgesetzt, die in der vorgegebenen Reihenfolge der zuletzt bedienten Komponente folgt → Alle Komponenten haben die gleiche Chance, bedient zu werden („faire“ Prozessor-Zuteilung)

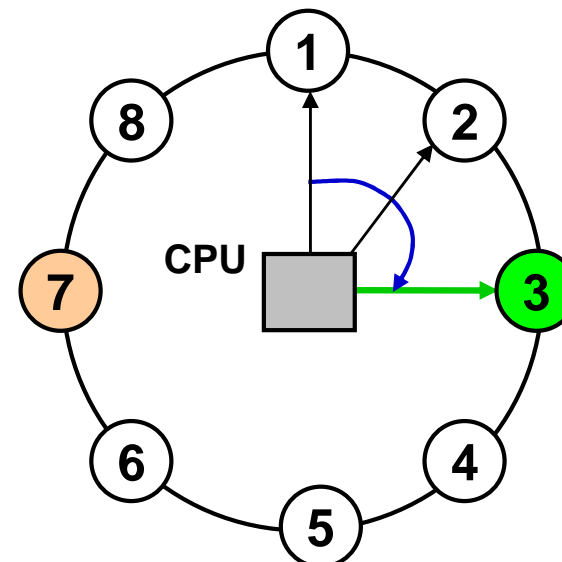
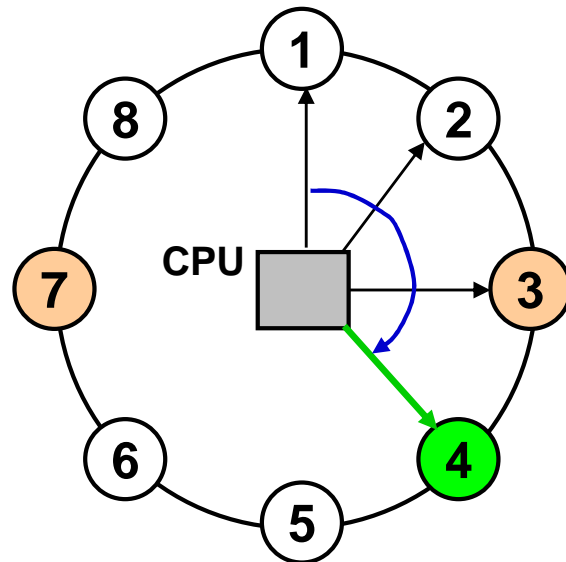


 Aktuell bearbeitete Interrupt-Anforderung

 Interrupt-Anforderungen während der Bedienung von Komponente 4

Polling: 2. Variante

Zyklische Abfrage beginnt immer mit der eindeutig festgelegten ersten Komponente → Verschiedenen Komponenten werden verschiedene Prioritäten zugeordnet. Komponenten mit hoher Priorität werden schneller bedient.



 Aktuell bearbeitete Interrupt-Anforderung

 Interrupt-Anforderungen während der Bedienung von Komponente 4

Polling

Nachteil des Polling-Verfahrens:

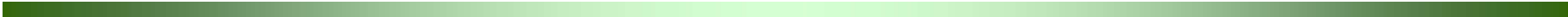
Die Priorisierung und Identifizierung von Interrupts durch die zyklische Abfrage (Software) ist sehr zeitaufwendig.



Daisy Chain Verfahren

- Priorisierung und Identifizierung von Interrupts wird durch eine **Zusatzhardware** durchgeführt.
- Zusammenschalten von Interruptquellen zu einer Prioritätskette (Interrupt-Daisy-Chain).
- Jede Interruptquelle hat eine Priorisierungsschaltung (dezentrale Priorisierung), die mit der des Vorgängers und Nachfolgers mit Signalleitungen verbunden ist.
- Erste Quelle der Kette hat die höchste Priorität. Die Priorität der andern Quellen nimmt mit jedem Glied in der Kette um eine Stufe ab.



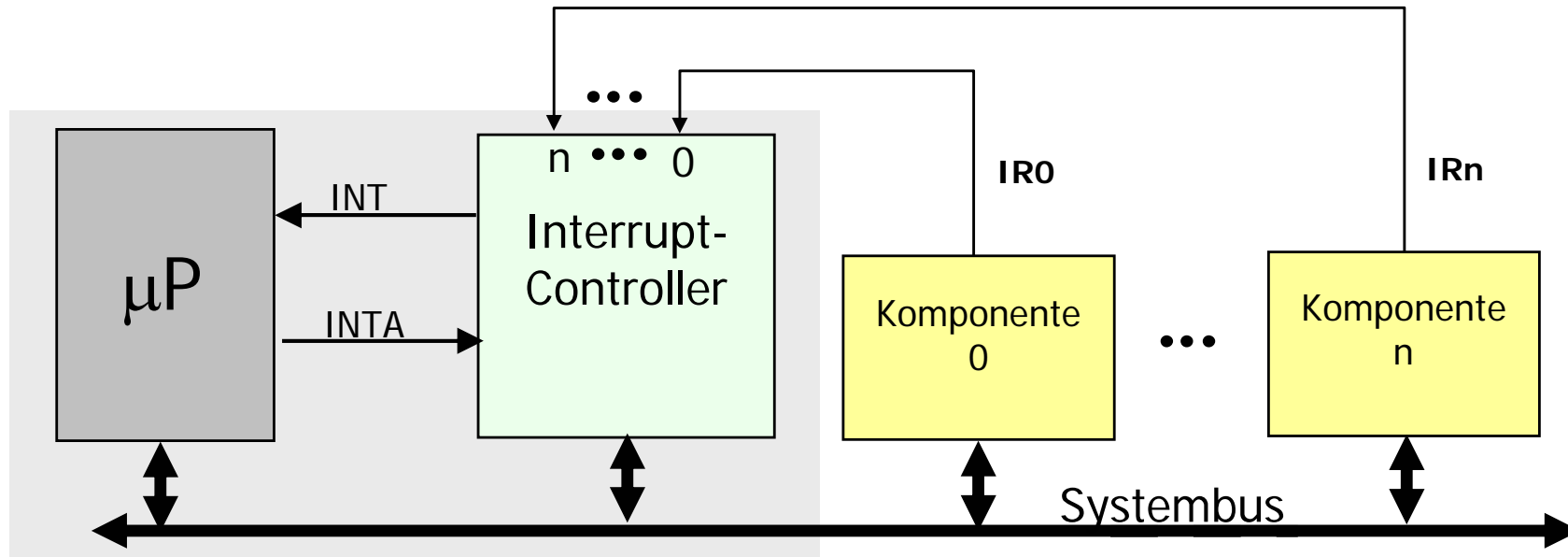


Daisy Chain Verfahren

- Anforderungen der einzelnen Quellen werden durch ODER zusammengeschaltet und über den Interrupteingang IRQ (Kreissymbol) zum Prozessor zugeführt.
- Prozessor leitet (bei gesetztem Interrupt-Enable-Flag) einen Interruptzyklus durch das Aktivieren von IACK ein.
- IACK wirkt direkt auf die Interruptquellen und auf den IACK-Eingang der ersten Quelle in der Kette.
- Eine Quelle, die während IACK = 1 eine Anforderung anmeldet, verhindert die Weitergabe des aktiven Pegels von IACK an die folgenden Kettenglieder.

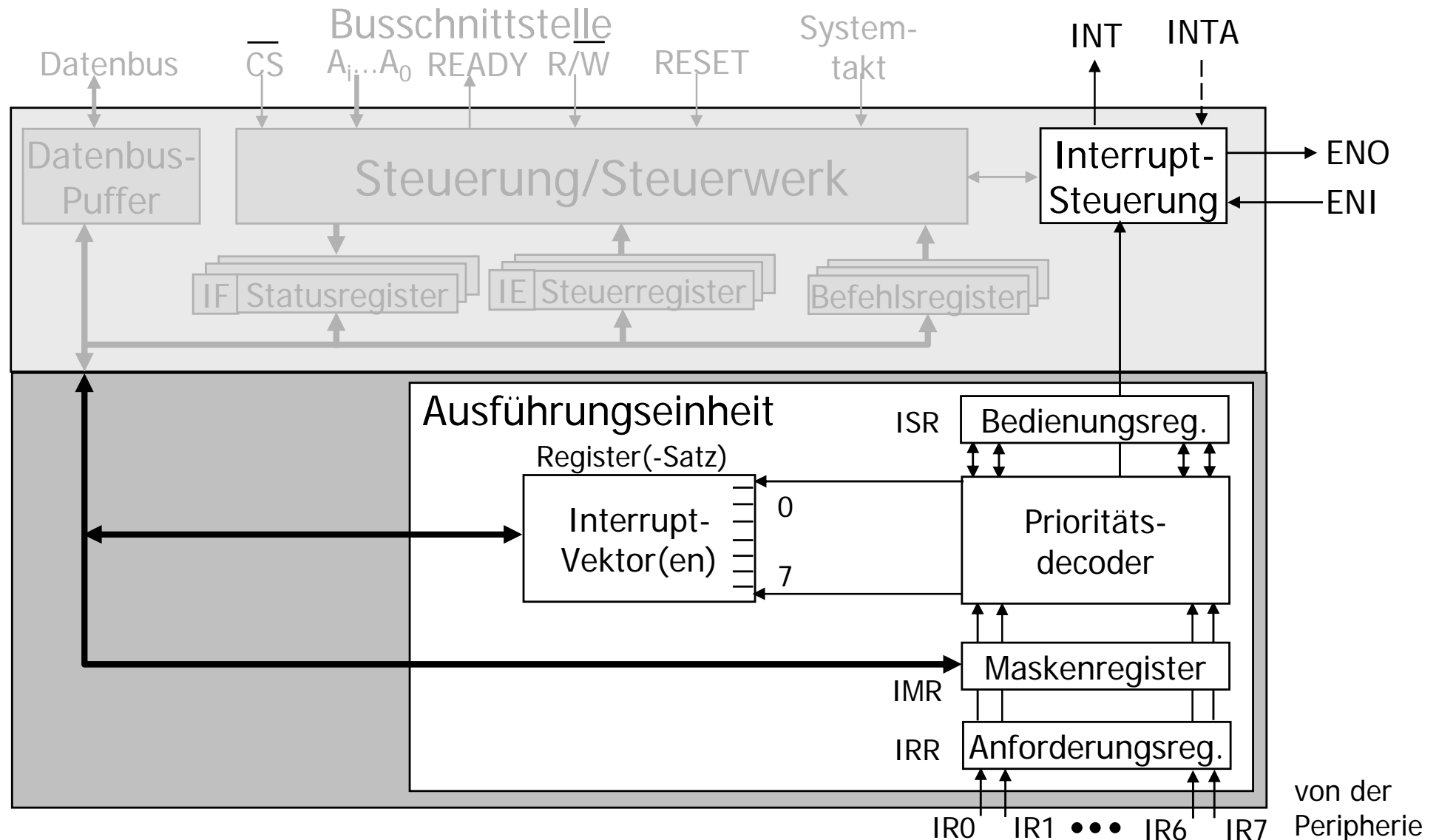


Mikroprozessor-System mit Interrupt-Controller



- Systemkomponenten teilen dem Interrupt-Controller über ihre Anforderungsleitungen IR_i ihre Unterbrechungswünsche mit.
- Der Controller ermittelt die Interruptquelle höchster Priorität und gibt deren Anforderung über das INT-Signal an den Prozessor weiter
- Prozessor prüft anhand des IE-Bit im seinem Steuerregister, ob zu diesem Zeitpunkt Unterbrechungen zugelassen sind. Wenn ja, dann unterrichtet er, so bald wie möglich, den Controller von der Annahme der Unterbrechungsanforderung (über das INTA-Signal)

Aufbau eines Interrupt Controllers



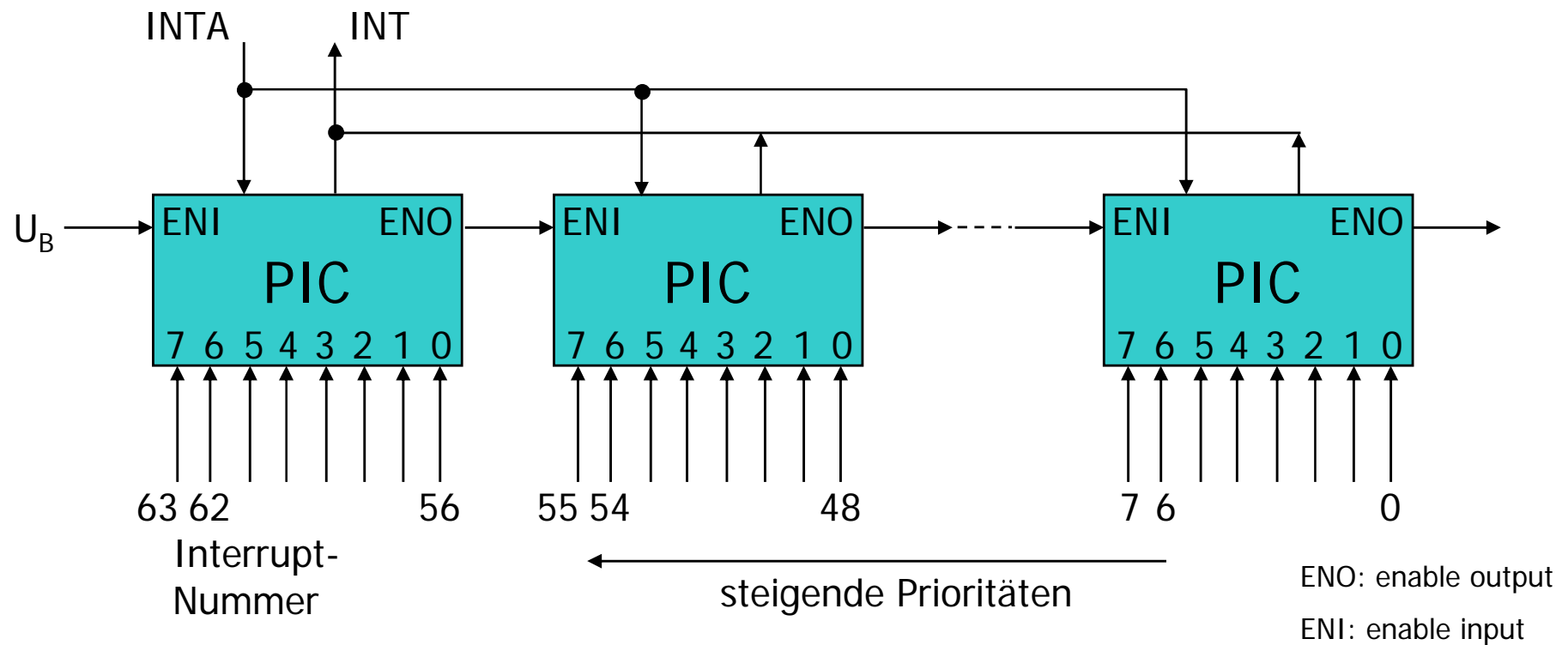
Aufbau eines Interrupt Controllers

- ❑ Systemkomponenten melden ihre Unterbrechungswünsche über die Interrupt Request Eingänge (IR0, ..., IR7). Diese werden im Anforderungsregister IRR (*Interrupt Request Register*) gespeichert.
- ❑ Jedes Bit des Maskenregister IMR (*Interrupt Mask Register*) haben die gleiche Funktion wie das IE Bit im Prozessor.
- ❑ Die im IRR angezeigten Unterbrechungswünsche werden nur diejenigen an den Prioritätsdekoder weitergereicht und letztlich ausgeführt, die im IMR nicht maskiert sind.
- ❑ Der Prioritätsdekoder ermittelt diejenige Unterbrechungsanforderung mit der höchsten Priorität aus und veranlasst die Interruptsteuerung ein Anforderungssignal über den INT Ausgang an den Prozessor auszugeben.
- ❑ Eine Quittierung der Anforderung meldet der Prozessor über den INTA Leitung.



Einsatz mehrerer Interrupt-Controller

Meist sind weit mehr als acht Interruptquellen vorhanden, deshalb werden mehrer Interrupt-Controller eingesetzt. Hier *Daisy Chaining* aus Interrupt-Controllern



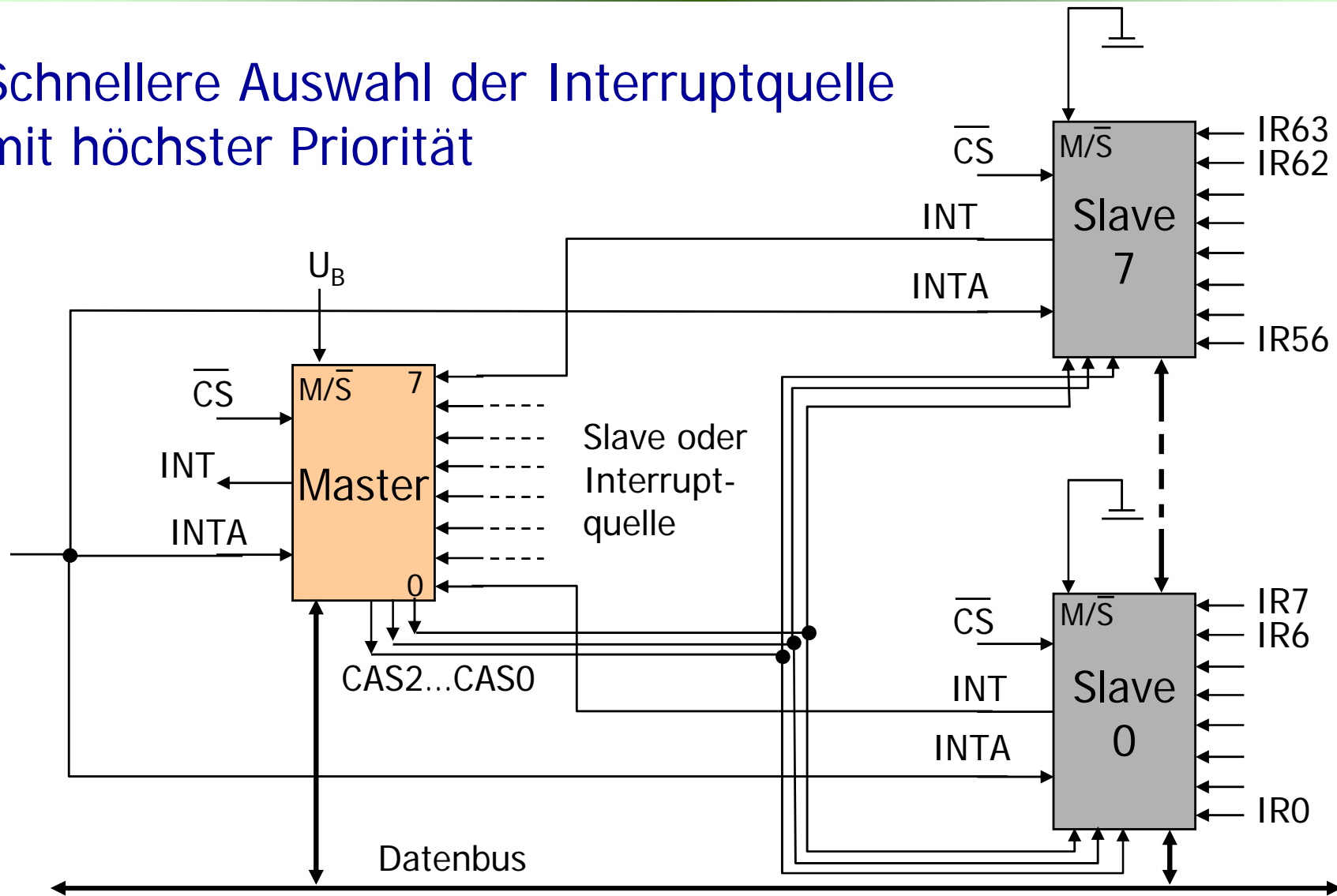
Einsatz mehrerer Interrupt-Controller

- ❑ Die Interruptsteuerung wird durch das ENI-Signal aktiviert
- ❑ Jeder Controller gibt über das Ausgangssignal ENO das Recht zur Interrupt-Erzeugung an seinen „rechten“ Nachbar genau dann, wenn er selbst keine Anforderung vorliegen hat.
- ❑ Die INT bzw. INTA aller Controller werden zusammengeschaltet, aber das INTA-Signal wird von dem Controller ausgewertet, der über seinen ENI-Eingang aktiviert ist und an einem seiner Interrupteingänge eine Anforderung vorliegen hat.



Kaskadierung von Interrupt-Controllern

Schnellere Auswahl der Interruptquelle
mit höchster Priorität

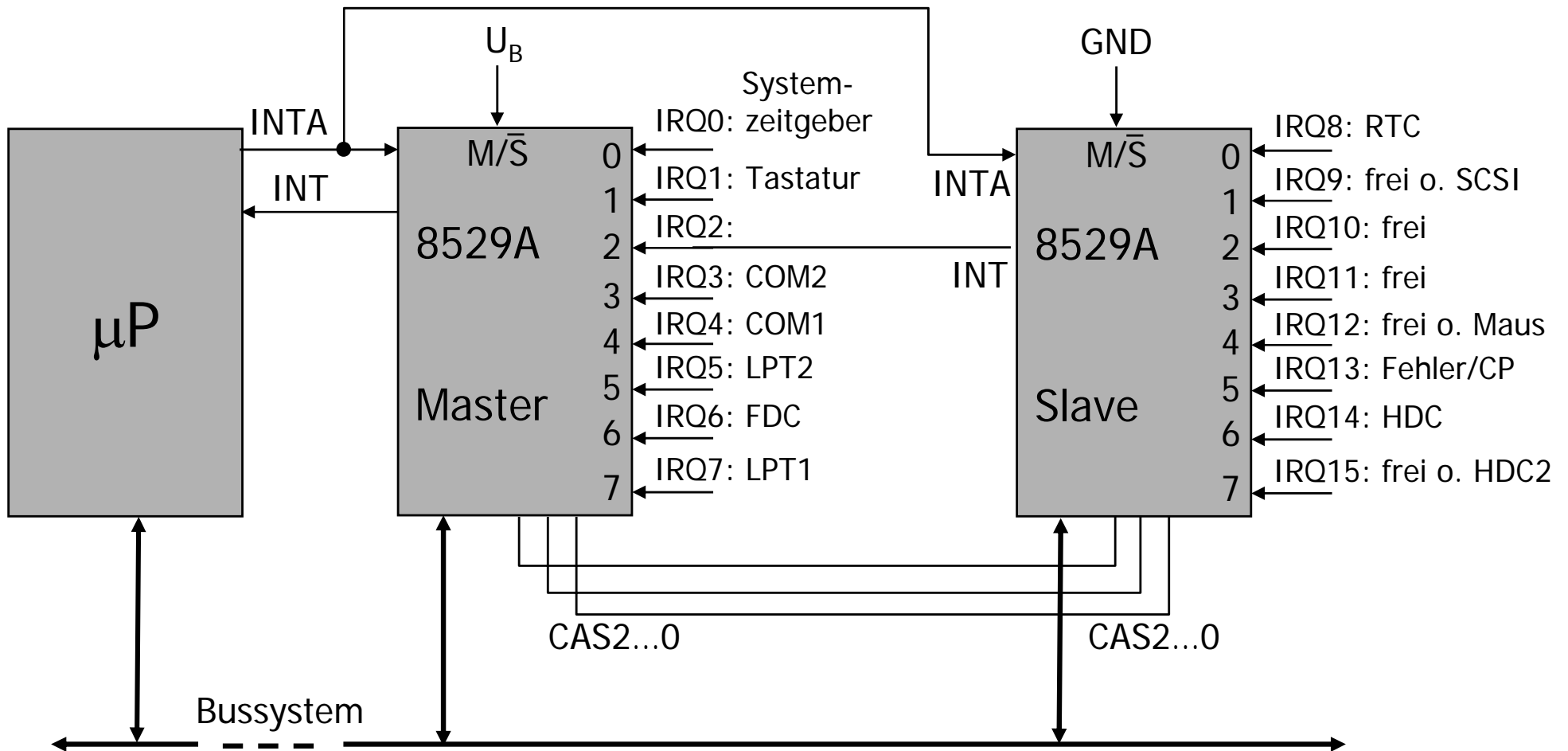


Kaskadierung von Interrupt-Controllern

- ❑ Ein Controller übernimmt die Rolle des *Masters*.
- ❑ An den IRI-Eingängen des Masters werden wahlweise eine Interruptquelle oder ein weiterer Controller als *Slave* angeschlossen.
- ❑ Die Konfiguration des Interruptsystems wird in einem Register des Masters festgehalten.
- ❑ Bei Controllern mit jeweils acht Eingängen können maximal 9 Controller (ein Master und ein Slave) mit insgesamt 64 Interrupteingängen.
- ❑ Nur der Master ist über INT und INTA mit dem Prozessor verbunden
- ❑ Über ein M/S-Signal wird jedem Controller mitgeteilt, in welchem Modus er arbeiten muss.



Kaskadierter Interrupt-Controller im PC



Priorität: 0,1,8-15, 3-7

Slave am Eingang IRQ2



Kaskadierter Interrupt-Controller im PC

- ❑ Slave am IRQ2-Eingang des Masters
- ❑ Insgesamt 15 Interrupteingänge IRQ7-IRQ0 (außer IRQ2) am Master, IRQ15-IRQ8 am Slave
- ❑ Gleichzeitig auftretende Anforderungen nach fest zugeordneten Prioritäten
- ❑ Abkürzungen:
 - COM = serielle Schnittstelle (Communication Port)
 - LPT = parallele Schnittstelle (Line Printer)
 - RTC = Echtzeit-Uhr (Real-Time Clock)
 - FDC = Floppy Disk Controller
 - HDC = Festplatten-Controller (Hard Disk Controller)
 - CP = Gleitkomma-Coprozessor





Direkter Speicherzugriff

Bisher wurde der Datentransfer zwischen Prozessor und Speicher bzw. Prozessor und Peripherie betrachtet.

Oft ist es jedoch auch nötig, Daten zu transportieren zwischen:

Speicher \Leftrightarrow Peripherie

Speicher \Leftrightarrow Speicher

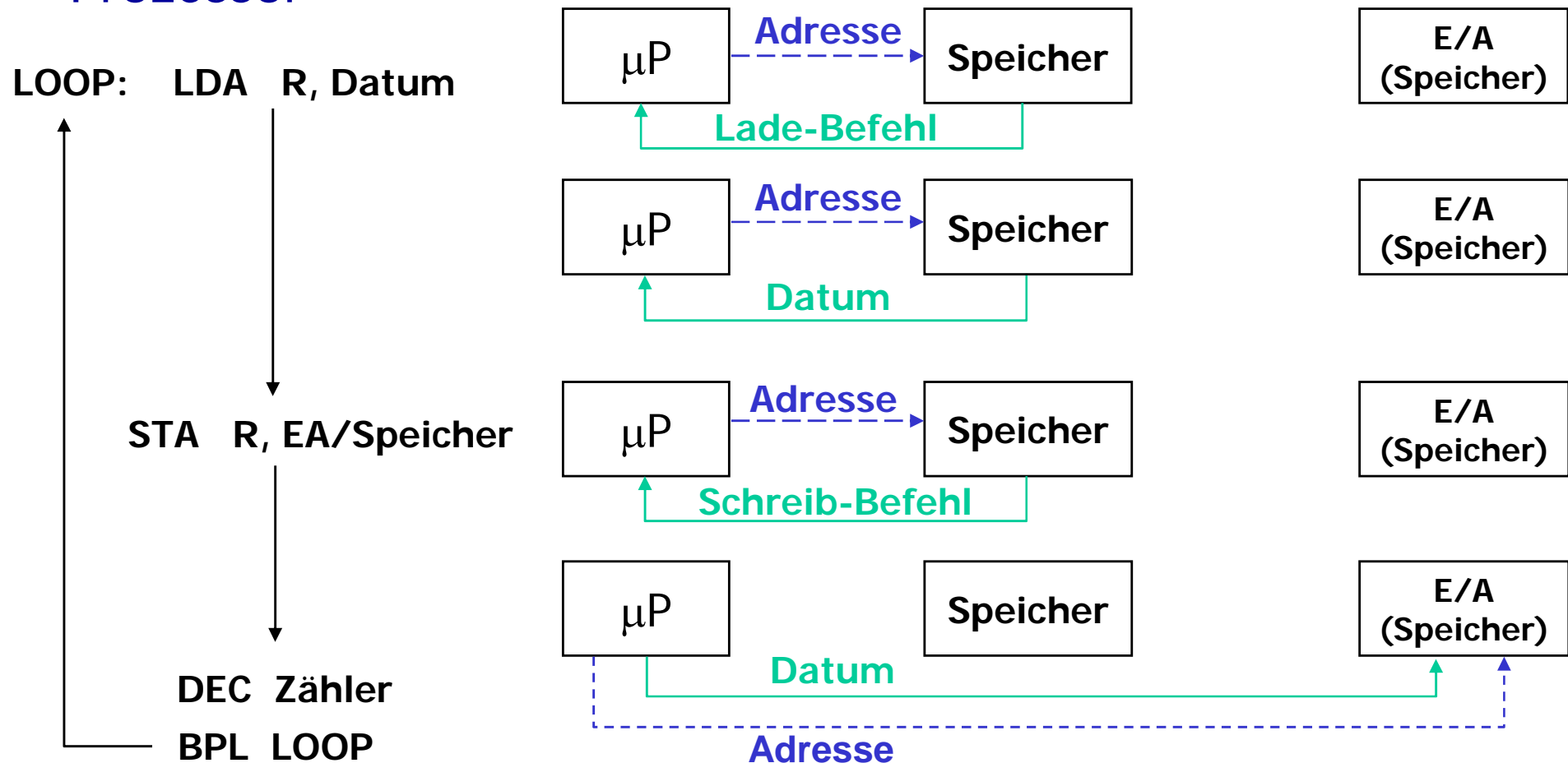
Peripherie \Leftrightarrow Peripherie

In älteren und kleineren Systemen wird dieser Datentransfer ebenfalls über den Prozessor abgewickelt.



Beispiel

Datenübertragung zwischen Speicher und Peripherie mittels Prozessor



Beispiel

Nachteil

mindestens 4 Speicherzugriffe / Datum nötig (ggf. sogar mind. 6 Speicherzugriffe durch Schleifenbefehle)
→ langsamer Datentransfer, Prozessor belastet

Abhilfe:

Direkter Speicherzugriff (*direct memory access, DMA*)

Hierbei erfolgt der Datentransfer ohne Beteiligung des Mikroprozessors direkt zwischen den beteiligten Komponenten



Direkter Speicherzugriff

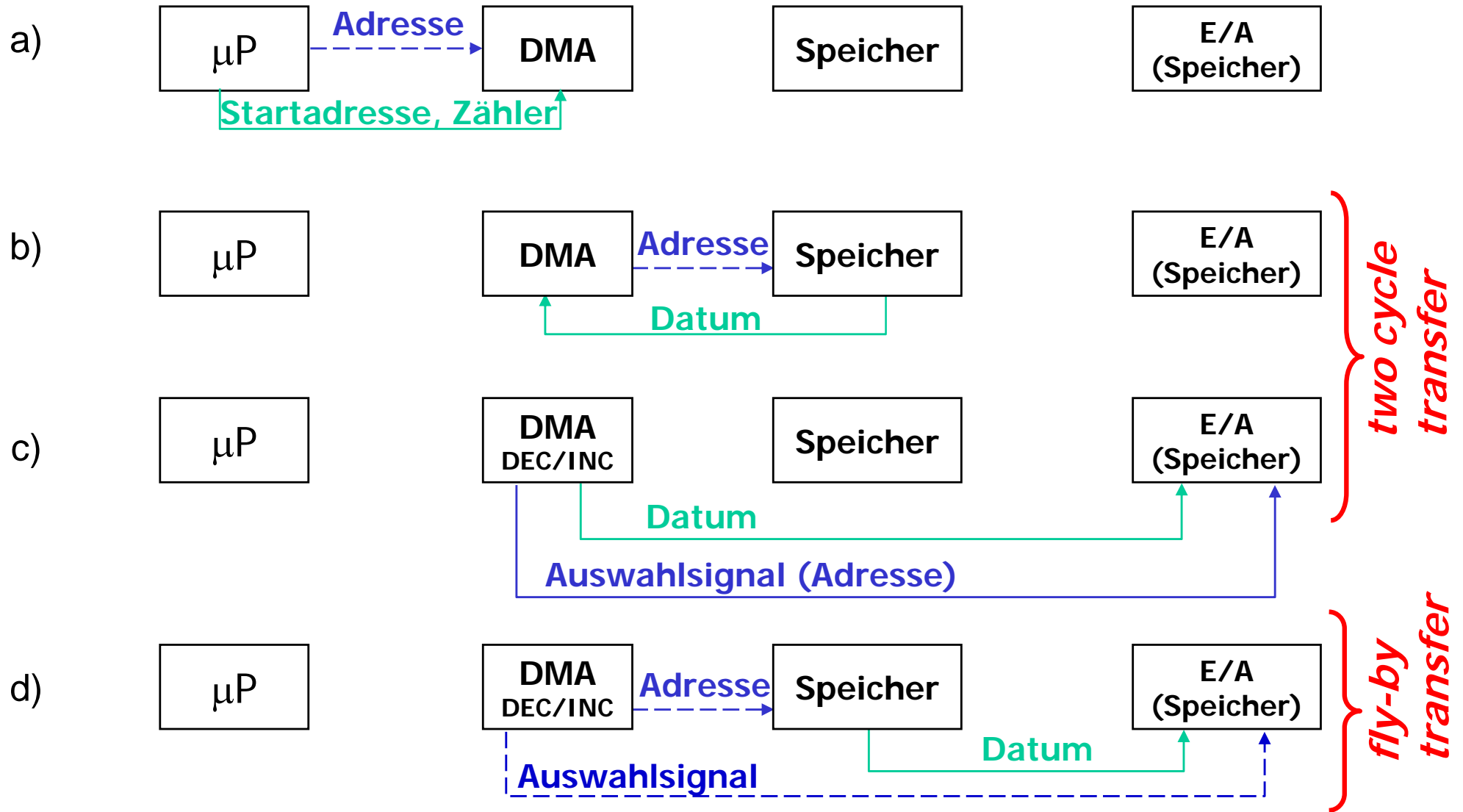
Ein spezieller Baustein, **DMA-Controller** genannt, koordiniert den Datentransfer

Vorteile:

- ❑ Speicherzugriffe für das Holen der Lade-, Speicher- und Schleifenbefehle entfallen, da die Datenübertragung hardwaremäßig (ohne Programm) ausgeführt wird
→ nur 2 (ggf. sogar nur 1) Speicherzugriff / Datum nötig
- ❑ Der Prozessor wird entlastet und kann derweil andere Dinge tun (sofern diese nicht den Systembus benötigen)



Prinzip des direkten Speicherzugriffs



Prinzip des direkten Speicherzugriffs

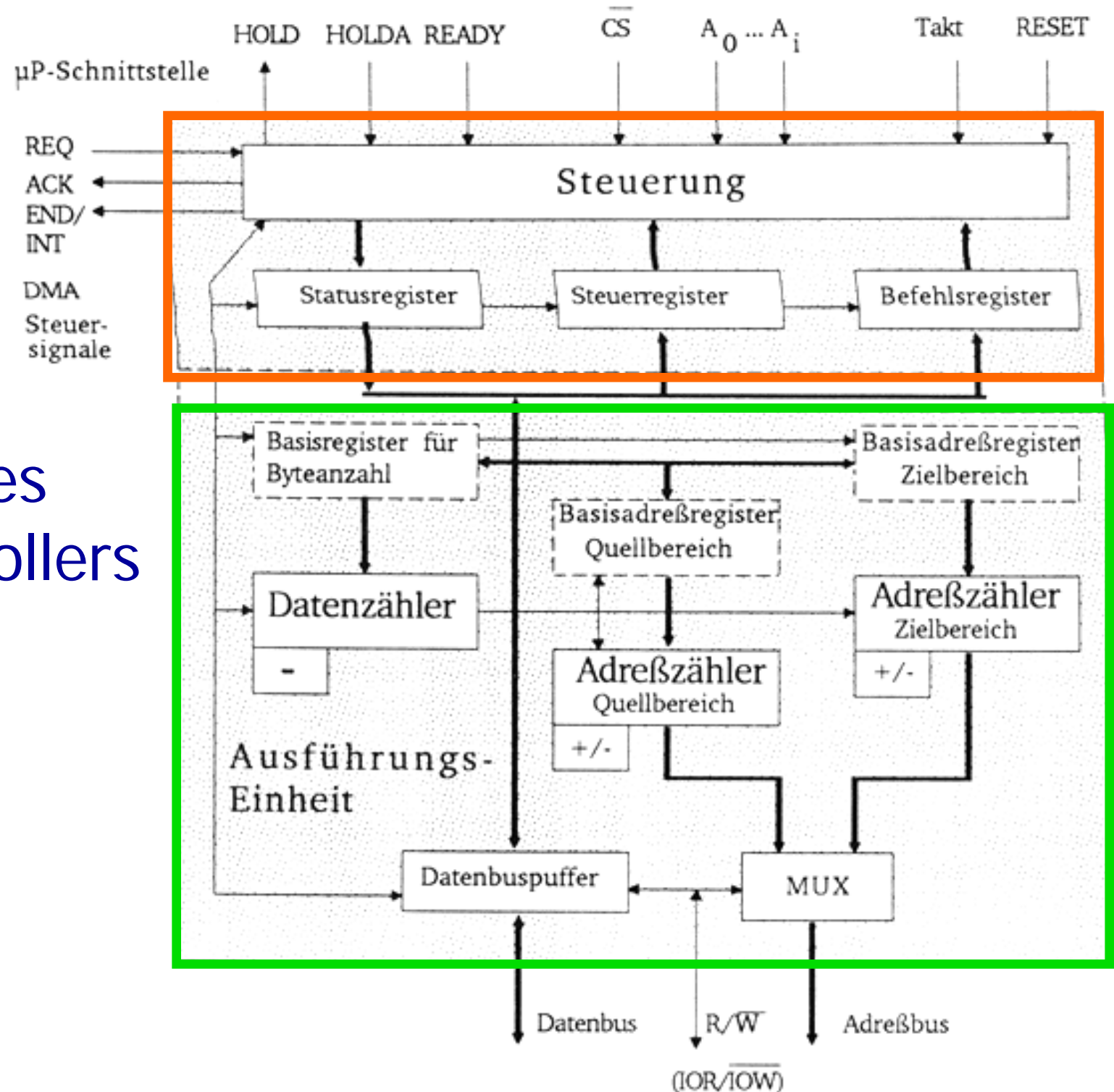
Mikroprozessor überträgt zu Beginn die Startadresse, Zieladresse, Anzahl der zu übertragenden Bytes, Übertragungsrichtung und Steuerinformationen

Danach läuft der Transfer automatisch:

- ***“two cycle transfer”* - Übertragungsverfahren:**
Datum wird in einem Register des DMA Controllers zwischengespeichert (Fälle: a c)
- ***“fly by transfer”* - Übertragungsverfahren:**
Datum wird ohne Zwischenspeicherung direkt übertragen (geht jedoch nicht für Speicher-Speicher Transfer) (Fall d)



Aufbau eines DMA-Controllers



Aufbau eines DMA- Controllers

Steuerwerk: steuert den Datentransfer

Steuersignale DMA-Controller \Leftrightarrow Mikroprozessor

- ❑ Takteingang: Systemtakt
- ❑ RESET: Baustein in definierten Anfangszustand rücksetzen
- ❑ $\overline{\text{CS}}$: Selektion des Bausteins zur Programmierung durch μP
- ❑ A_0 - A_1 : niederwertige Adressleitungen zur Auswahl eines der internen Register des Bausteins
- ❑ READY: Aufforderung an den Baustein zum Einfügen von Wartezyklen (von der Peripherie)
- ❑ HOLD: Anforderung des Systembusses vom μP durch den Baustein
- ❑ HOLDA: Gewährung des Systembusses durch den μP



Aufbau eines DMA- Controllers

Steuersignale DMA-Controller \Leftrightarrow Peripherie

- ❑ REQ: *DMA request* Peripherie stellt Anforderung zum Datentransfer
- ❑ ACK: *DMA acknowledge* Baustein akzeptiert Anforderung zum Datentransfer
- ❑ END, EOP: *end of process* Peripherie oder Prozessor wird vom Baustein über das Ende des Datentransfers informiert. Manchmal bidirektionale Leitung, durch die der μ P einen Datentransfer abbrechen kann

3 Steuerregister zur Kontrolle und Programmierung durch den μ P



Aufbau eines DMA- Controllers

Operationswerk: führt den Datentransfer aus

Besteht im wesentlichen aus 3 Zählern, die vom μP zu Beginn eines Datentransfers geladen werden:

- **Datenzähler:**
Anzahl der zu übertragenden Daten, wird bis 0 dekrementiert, informiert dann das Steuerwerk über das Ende des Datentransfers
- **Adresszähler1:**
Quelladresse des Datentransfers. Wird bei Transfers aus dem Speicher wahlweise inkrementiert oder dekrementiert. Bei Transfers von der Peripherie bleibt die Adresse unverändert
- **Adresszähler2:**
Zieladresse des Datentransfers, Verhalten wie Adresszähler 1

Die Basisregister für die drei Zähler speichern jeweils die Anfangswerte

