

❑ **Anmeldung zur TI-Klausur:**

Einwurf der Zulassungsbescheinigung in den Briefkasten im Untergeschoss des Informatikgebäudes am Fasanengarten **bis spätestens 27. August**. Es handelt sich um den gleichen Briefkasten, in dem die Übungsblätter eingeworfen werden.

❑ Hilfsmittel sind nicht erlaubt

❑ **Dauer der Klausur:**

○ **Informatik: 120 Minuten (9.00-11.00 Uhr)**

○ **Informationswirtschaft: 60 Minuten (9.00-10.00 Uhr)**

❑ Studentenausweise unbedingt in die Klausur mitbringen

❑ Hörsaalverteilung wird rechtzeitig bekannt gegeben (TI Homepage)

❑ **Online Anmeldung (Informationen am Donnerstag)**



Kapitel 4

- Systemsteuer- und Schnittstellenbausteine
- Ausnahmebehandlung:
 - Behandlung mehrerer Interruptquellen
 - Interrupt-Controller



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.

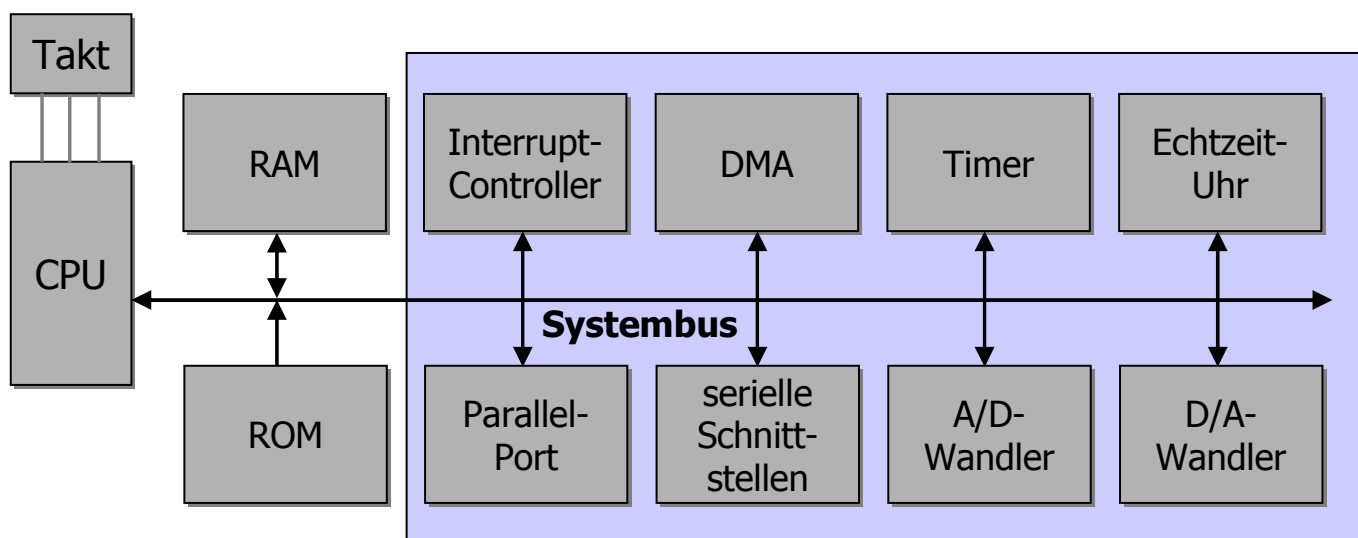
Aufgaben:

- 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

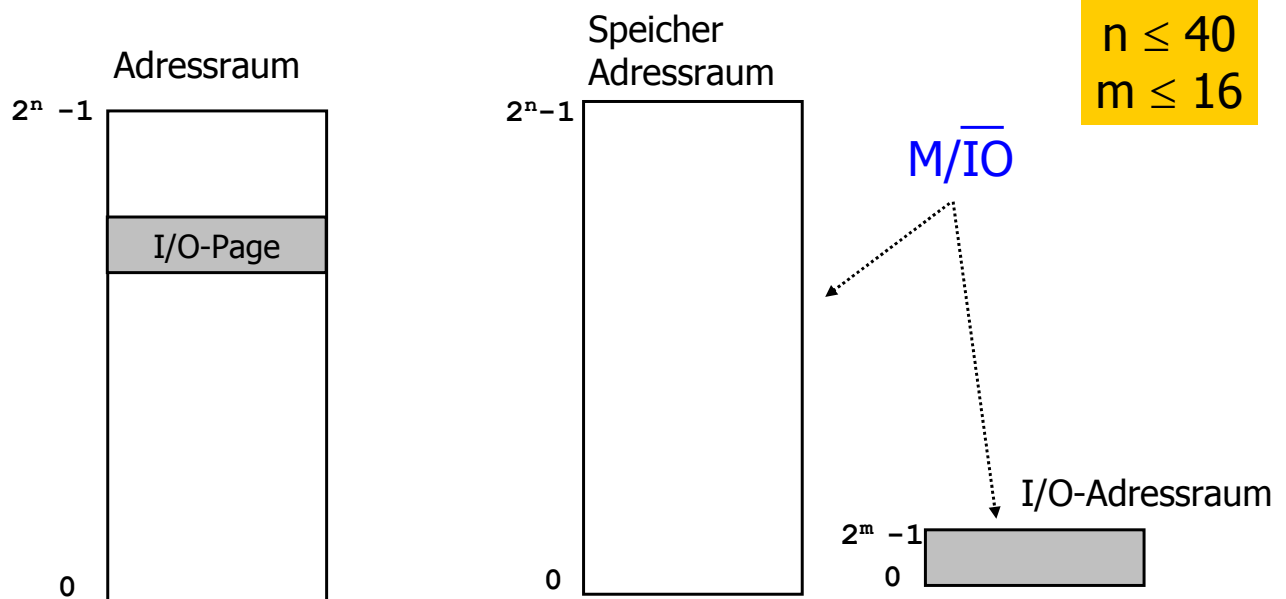
Jeder programmierbare Systembaustein erscheint für den Prozessor wie ein kleiner Satz von Registern, die unter einem zusammenhängendem Block von Adressen (Portadressen) angesprochen werden können.

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:

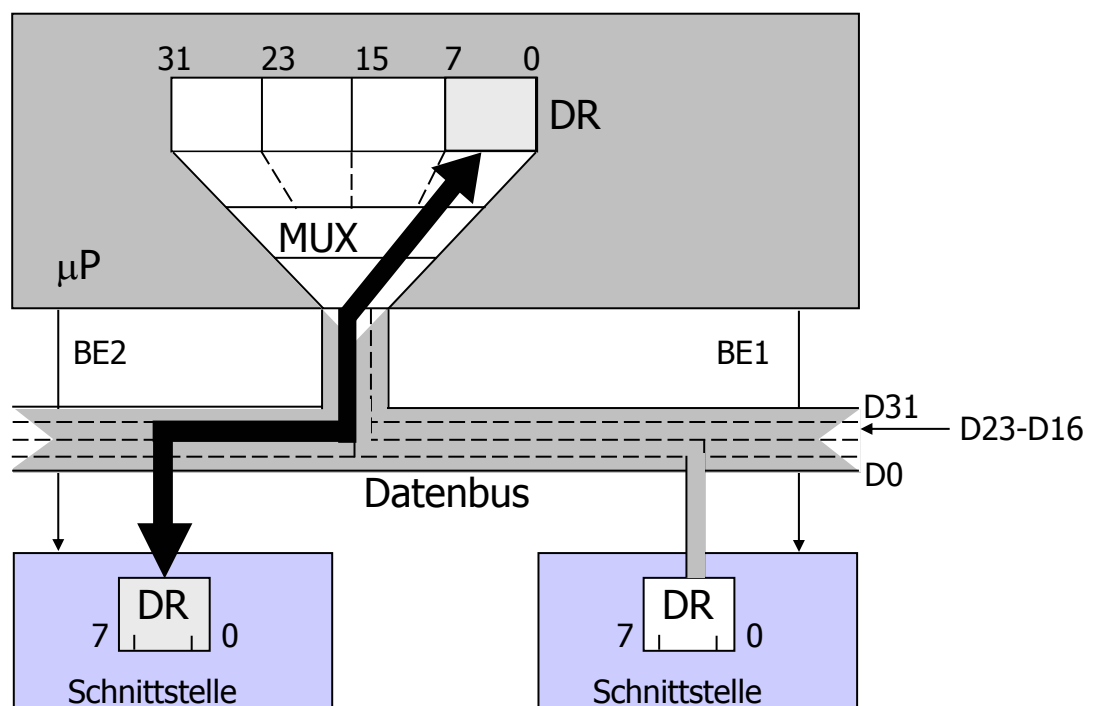
getrennte Adressräume für Speicher und Peripherie (eigener I/O-Adressraum)

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



Anschluss der Schnittstellenbausteine an dem μP

8-bit-Schnittstelle am 32-bit-Prozessor

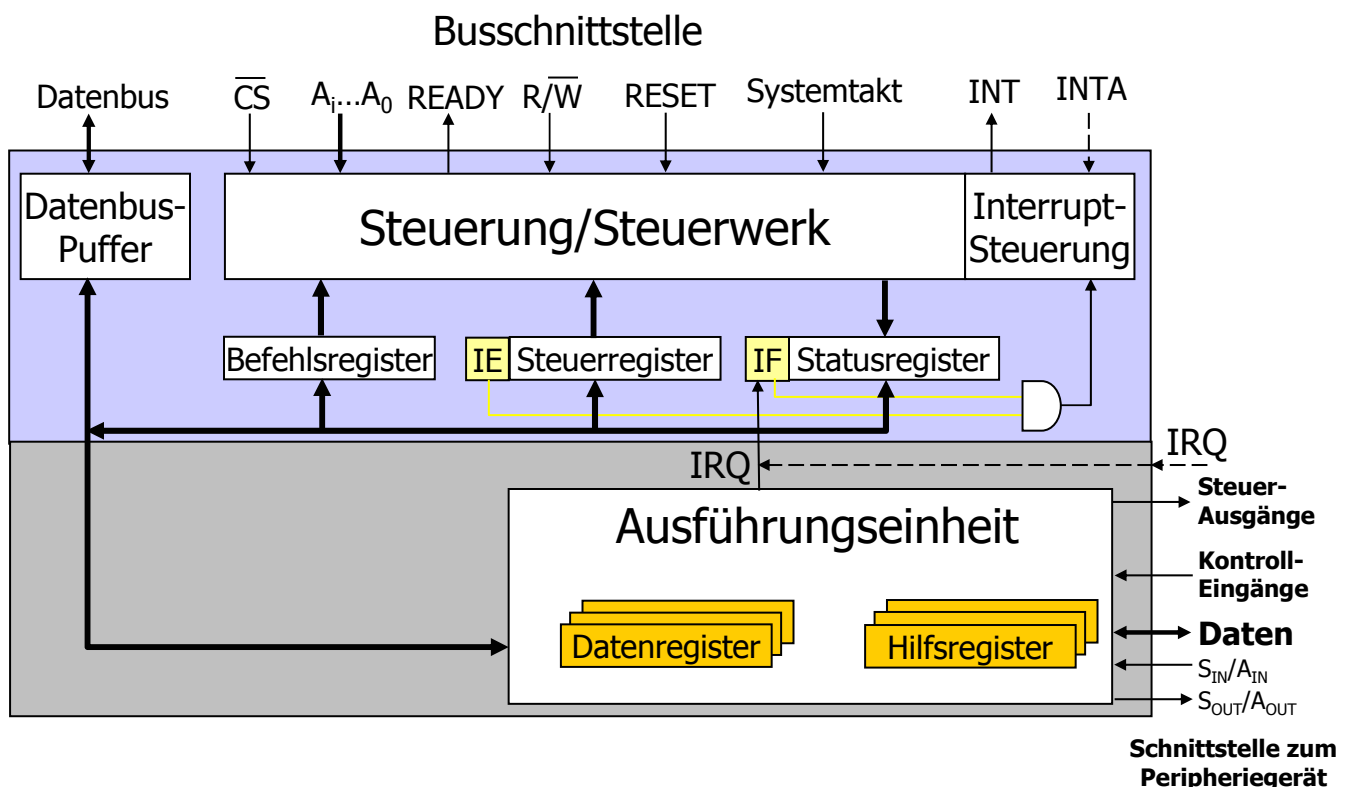


Anschluss der Schnittstellenbausteine an dem μP

- Noch sehr viele Schnittstellenbausteine haben eine Datenbusbreite von 8 Bit.
- Grund:
 - Viele Ein-Ausgabegeräte 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:

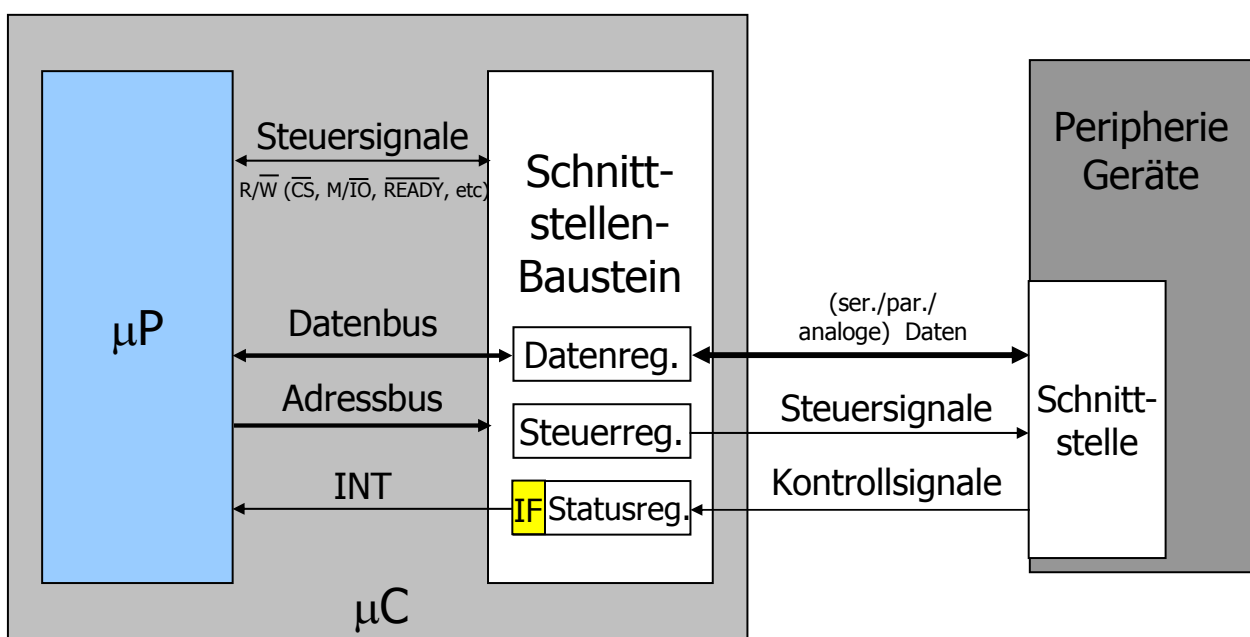
- Steuerung der internen Komponenten (Register, Multiplexer, ...) und Datenpfade
- Schnittstelle zum Prozessor
- Register zur Bausteinprogrammierung: Statusregister (Bausteinstatus), Steuerregister (Betriebsart), Befehlsregister (aktuelle Operation)

Ausführungseinheit:

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



Schnittstellenbaustein zwischen μP und Peripheriegerät



- **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

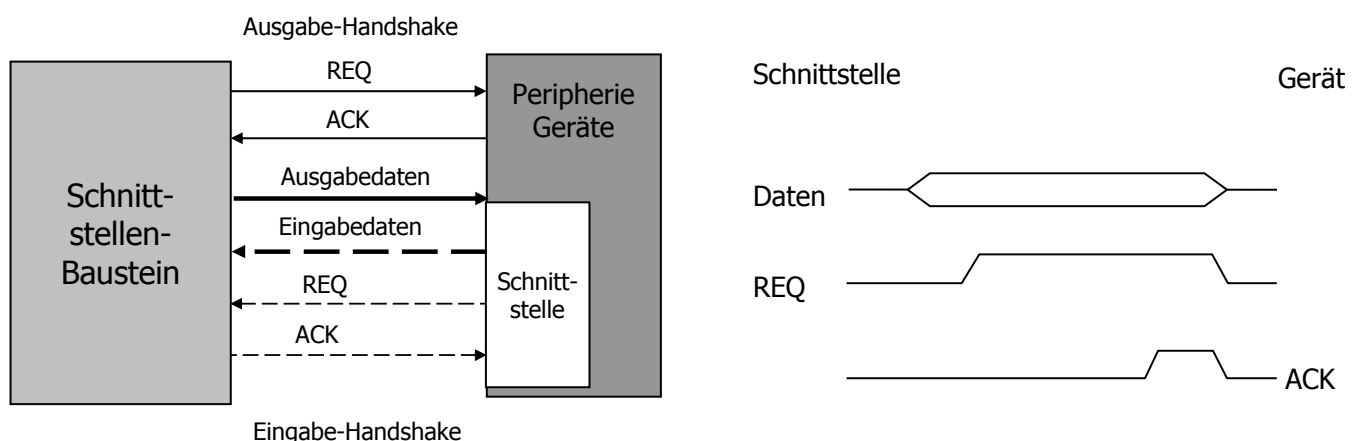
- **Programmierte Ein-/Ausgabe:** Prozessor fragt ständig das Statusregister des Peripheriebausteins ab, ob er bereit ist ein Datum zu übertragen. Zwischen den Übertragungsanforderungen kann der Prozessor andere Aufgaben erledigen oder in einer Schleife ausschließlich das Statusregister abfragt (*busy 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-Controlelr

Vorher: Behandlung von Ausnahme-Situationen



Behandlung von Ausnahmesituationen

Während des Betriebs eines Mikroprozessorsystems können Ausnahmesituationen (Exceptions) auftreten

Eine solche Ausnahmesituation erfordert eine **vorübergehende Unterbrechung** oder gar den **Abbruch** des laufenden Programms

Ursachen:

- Fehler im System bei der Ausführung des Anwenderprogramms oder Fehler der Hardware
- Wunsch externer Systemkomponenten, die Aufmerksamkeit des Prozessors zu erhalten



Behandlung von Ausnahme-Situationen

Die Ausnahme-Behandlung erfolgt durch eine Ausnahmeroutine (Interrupt Service Routine)

Die Auswahl und Aktivierung der Ausnahmeroutine wird durch eine Hardware-Komponente im Steuerwerk unterstützt: Unterbrechungs-System (Interrupt System)

Die Ausnahmeroutine hat Ähnlichkeit mit dem Aufbau eines Unterprogramms. Es gibt jedoch wesentliche **Unterschiede:**



Ausnahmeroutine/Unterprogramm

- Aktivierung:
 - *call subroutine* bei Unterprogramm
 - Hardware Aktivierung durch *Externes Signal* bei Ausnahmeroutine
- Beendigung:
 - *ret*-Befehl bei Unterprogrammen (*return from subroutine*)
 - *reti* Befehl bei Ausnahmebehandlung (*return from interrupt*)
- Einsprungsadresse ins Unterprogramm direkt im Programm, bei Ausnahmebehandlung über Interrupttabelle
- Unterprogrammaufruf sichert meist nur den PC auf den Stack, Ausnahmebehandlungs-Aufruf meist auch das PSW



Ausnahmeroutine/Unterprogramm

- Unterprogrammaufrufe werden immer durchgeführt, die meisten Ausnahmebehandlungen werden nur aktiviert, falls das Interrupt Enable Bit im Steuerregister (PSW) gesetzt ist

Ursachen für Ausnahmebehandlungen

- Prozessorexterne Ursachen: asynchrone Ereignisse (durch die Hardware)
- Prozessorinterne Ursachen: synchrone Ereignisse (fast nur durch Software)



Prozessorexterne Ursachen

➤ **RESET:**

Rücksetzen des Mikrorechnersystems, z. B. ausgelöst durch Taste, Schwankungen der Betriebsspannung, Watch-Dog, ...

➤ **HALT:**

Anhalten des Prozessors, z. B. zur Vermeidung von Zugriffskonflikten auf dem Systembus bei DMA-Zugriffen, Paritätsfehler

➤ **ERROR:**

Aufruf einer Fehlerbehandlungsroutine, z. B. bei Bus-Fehlern



Prozessorexterne Ursachen

- **(Hardware)-Interrupt:** Unterbrechungsanforderung von einem peripheren Gerät, z. B. um verfügbare Daten eines Eingabegerätes anzukündigen

2 Arten: maskierbar/nicht maskierbar (NMI)

- **Nicht maskierbare Interrupts (NMI):** werden nach Abschluss des gerade ausgeführten Befehls unbedingt durchgeführt (wichtige Ausnahmesituationen, z. B. Zusammenbruch der Betriebsspannung)
- **Maskierbare Interrupts:** werden nur dann ausgeführt, wenn im Steuerregister des Prozessors das *Interrupt-Enable* Flag (IE-Bit) gesetzt ist.



Prozessorexterne Ursachen

Prozessorinterne Ursachen: Bei Prozessoren, die auf dem Chip interruptfähige Komponenten besitzen (z. B. serielle oder parallele Schnittstellen, Zeitgeber, ..). Treten synchron zum Programmablauf auf.

- **Software Interrupts:** durch SWI- (*software interrupt*) oder INT-Befehl im Programm ausgelöste Unterbrechungen
- **Traps:** Ausnahmesituationen durch prozessorinterne Ereignisse, z. B. Overflow, Divide by Zero, Stack Overflow, ungültiger OpCode, Seitenfehler (*Page Trap*), Einzelschritt-Unterbrachung (Trace)



Interrupt-Vektortabelle

Interrupt-Vektortabelle (Ausnahme-Vektortabelle):

- Festwertspeicher an spezieller Speicheradresse (Oft in einer der untersten Speicherseiten)
- Enthält die Startadressen der Behandlungsroutinen
- Interrupt-Quelle liefert bei Interrupt eine Interrupt-Vektor-Nummer (IVN), welche den Eintrag in der Interruptvektor-Tabelle charakterisiert
- Basis-Adressregister enthält die Basisadresse der Tabelle



Beispiel einer Vektortabelle

Index	Ausnahmesituation	
0	divide by 0	Division durch 0
1	overflow	Zahlenbereichüberschreitung
2	array bound check	Indexbereichsüberschreitung
3	invalid opcode	illegaler Befehlscode
4	SVC (supervisor call)	Betriebssystem-Aufruf
5	privilege violation	unerlaubter Aufruf einer privilegierten Operation
6	trace	Einzelschritt-Modus
7	
-	(weiter Traps)
i	
i+1	RESET	Rücksetzen
i+2	BERR	Busfehler
i+3	NMI	nicht maskierbarer Interrupt
-	
k	

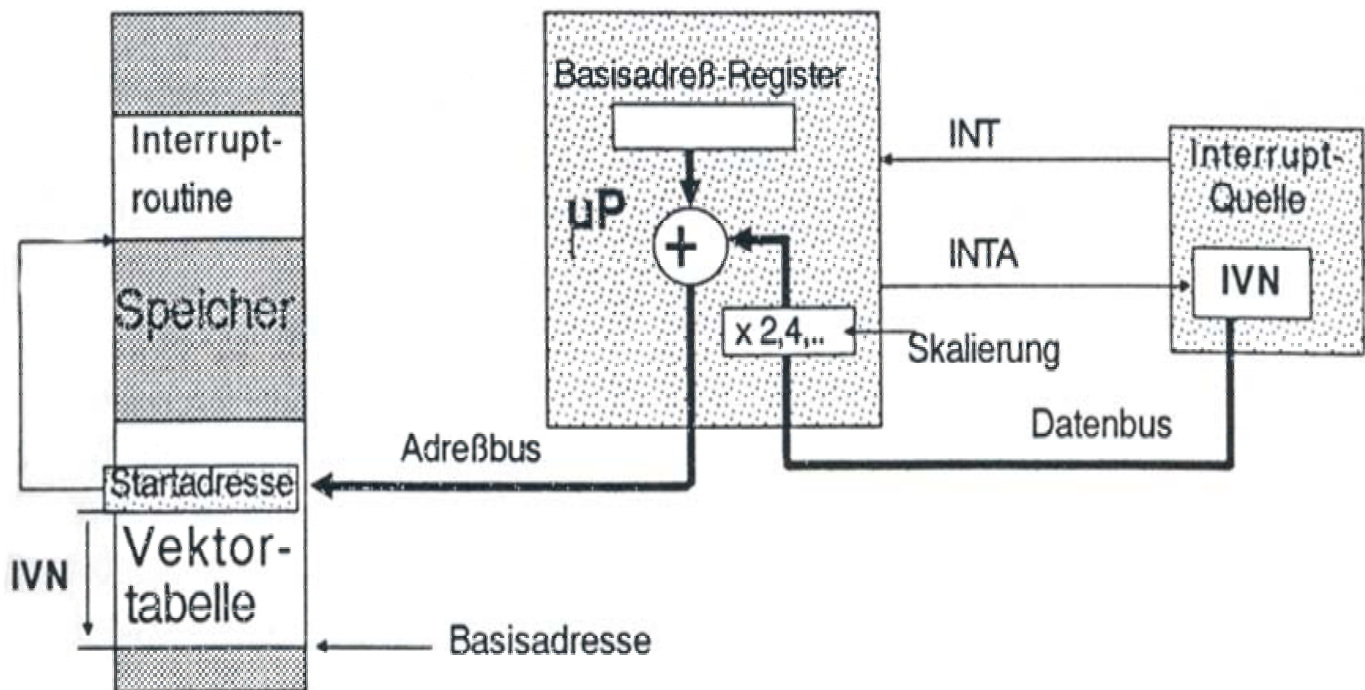


Beispiel einer Vektortabelle

Index	Ausnahmesituation	
k+1	error	Coprozessor-Fehler
-	Weitere Coprozessor-Meldungen
l		
l+1	page fault	Seitenfehler
-	Weitere Fehler der Speicherverwaltung
m	
m+1	Reserviert für zukünftige Erweiterungen und für Testzwecke des Herstellers
-	
n	
n+1	user vectors	Durch Systementwickler frei zuzuordnende, maskierbare Interrupts
-	
255	



Berechnung der Startadresse der Interrupt Service Routine



Ablauf einer Interrupt Service Routine

- Interrupt-Aktivierung
- Beenden des gerade in Ausführung befindlichen Befehls
- Feststellen, ob Software-Interrupt oder interner/externer Hardware-Interrupt vorliegt
- Feststellen, ob das Interrupt Enable Bit gesetzt ist
➔ Interrupt zugelassen
- Falls HW-Interrupt: Quelle des Interrupts finden, INTA-Leitung (Interrupt Acknowledge) aktivieren
- PSW und PC auf den Stack sichern
- Interrupt Enable Bit rücksetzen (und damit weitere Unterbrechungen verhindern)



Ablauf einer Interrupt Service Routine

- Startadresse der Interrupt-Service-Routine ermitteln und in den PC laden
- Interrupt Service Routine ausführen:
 - meist werden zuerst die benutzte Register auf den Stack gesichert
 - Interrupt Enable Bit wieder setzen
 - Am Ende der Interrupt Service Routine: IRET ~~Befehl~~
- PSW und PC werden wiederhergestellt und mit dem unterbrochenen Programm wird fortgefahren.



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.

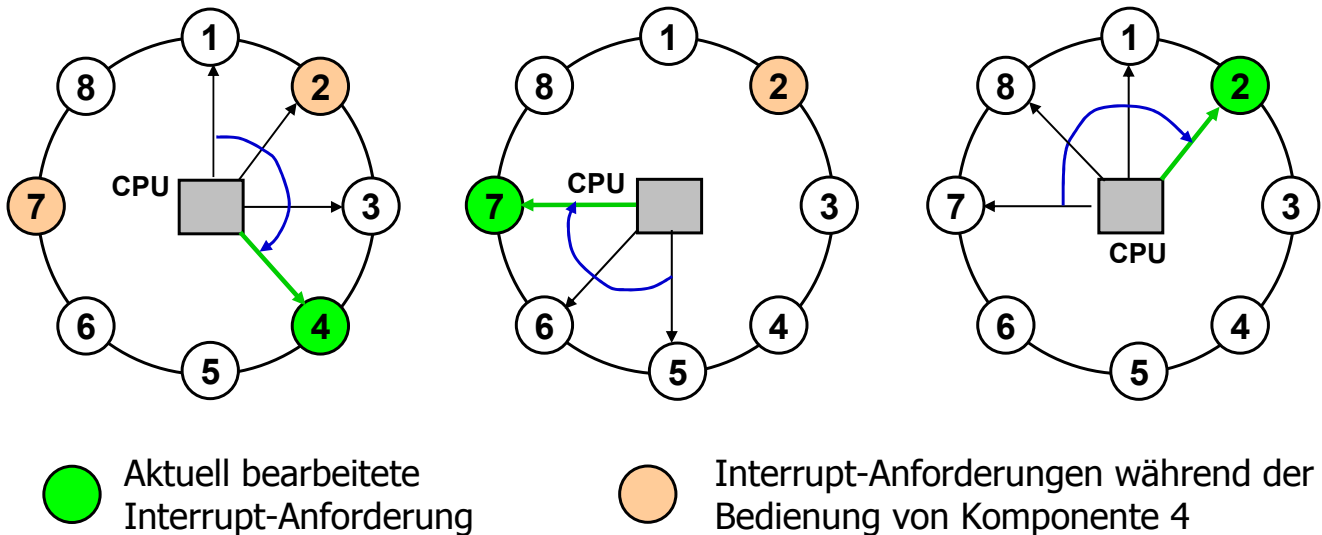
Nach (und auch während) der Abarbeitung eines Interrupts können weitere Anforderungen auftreten.

- ➔ Zwei Alternativen zur Behandlung:



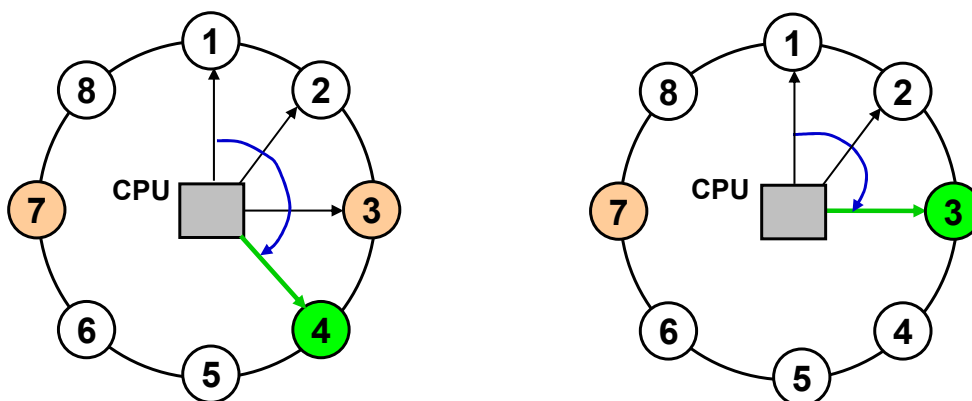
Polling: 1. Variante

Zyklische Abfrage wird 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)



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.

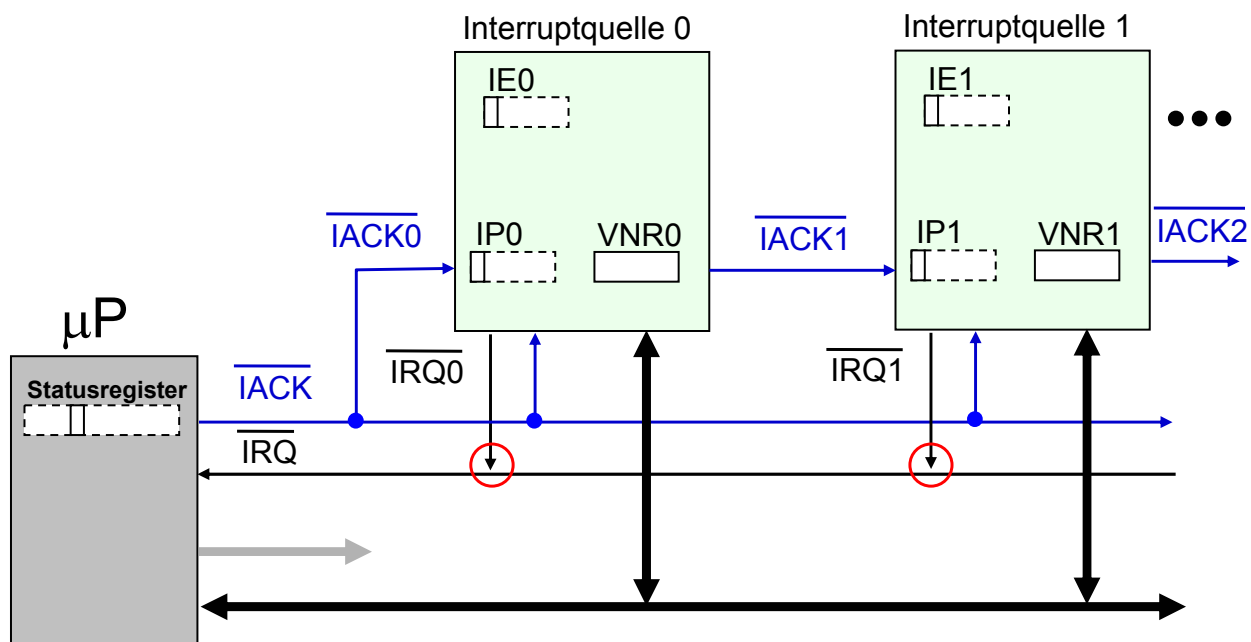


Daisy-Chain-Verfahren

- 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

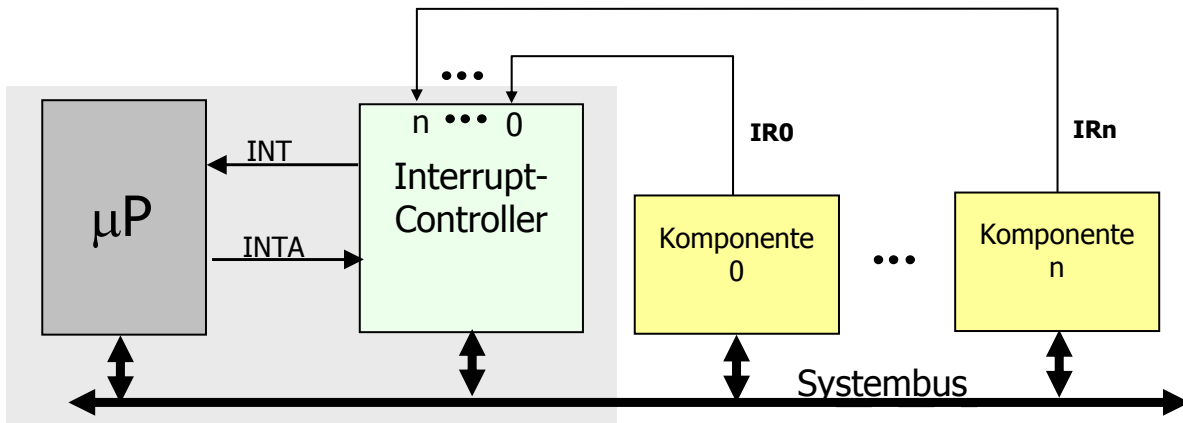


Daisy-Chain-Verfahren

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



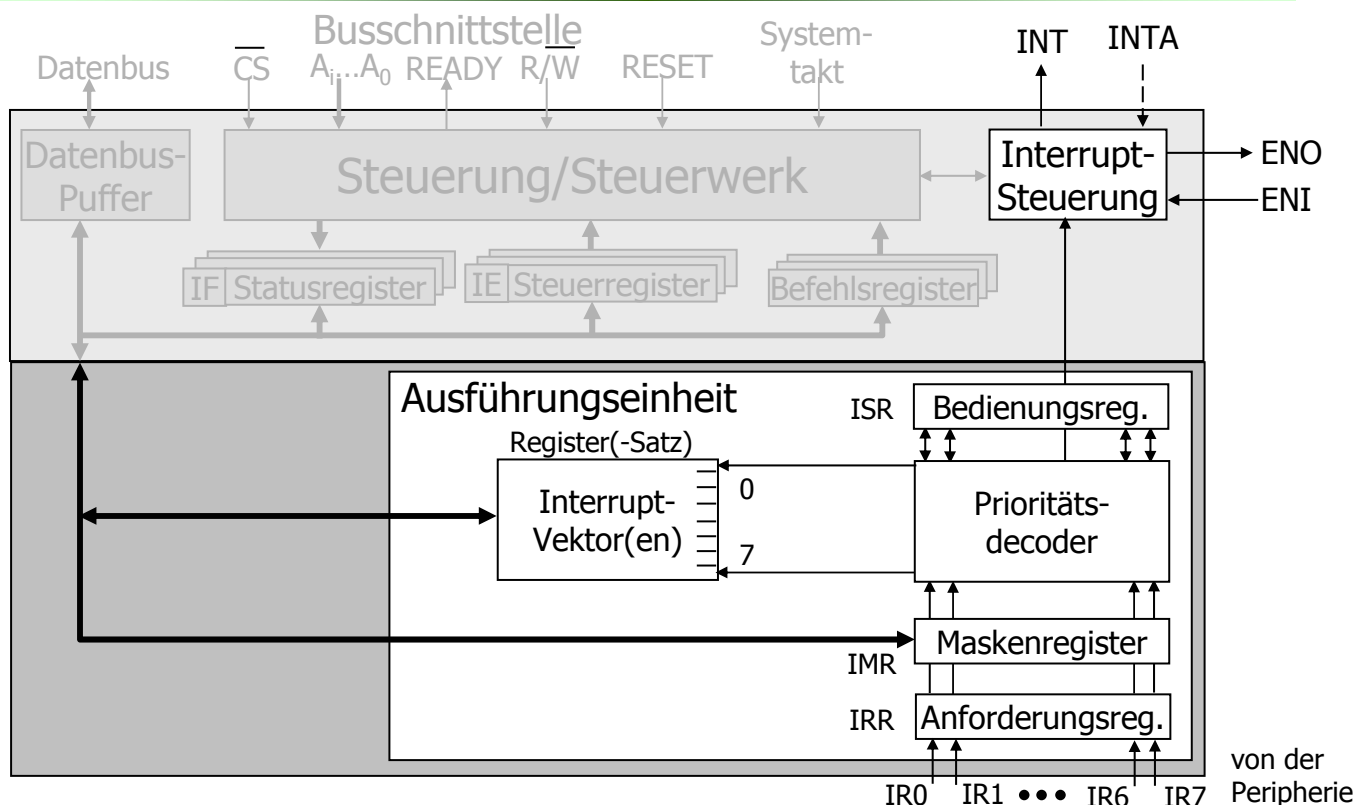
Mikroprozessor-System Interrupt-Controller



- 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



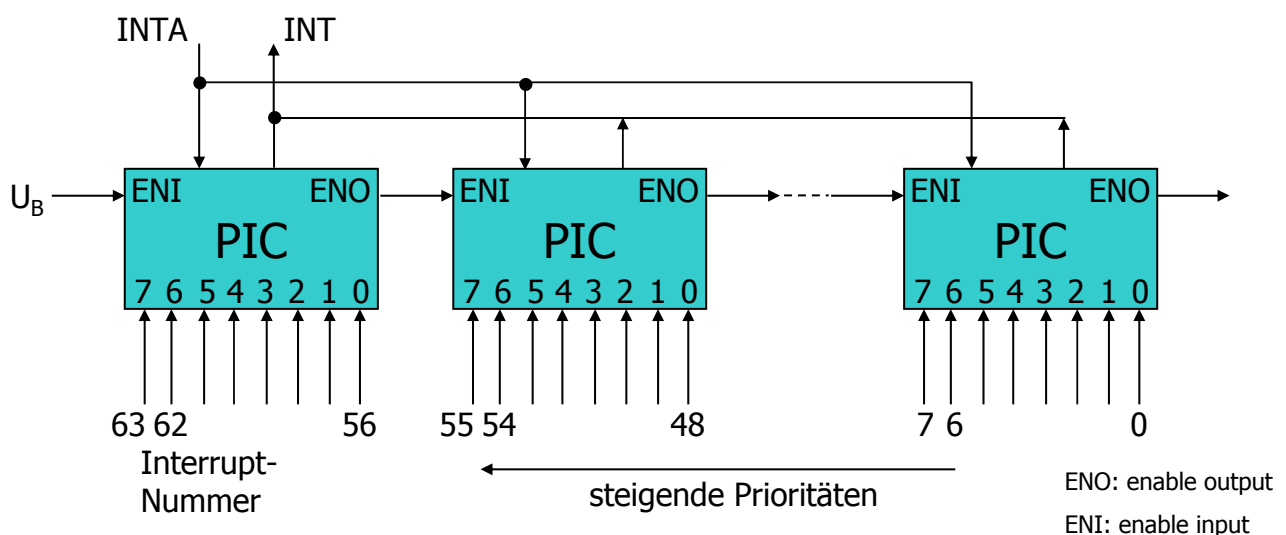
Aufbau eines Interrupt-Controllers

- 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 Peioritätdekoder weitergereicht und letztlich ausgeführt, die nicht im IMR 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 üner 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



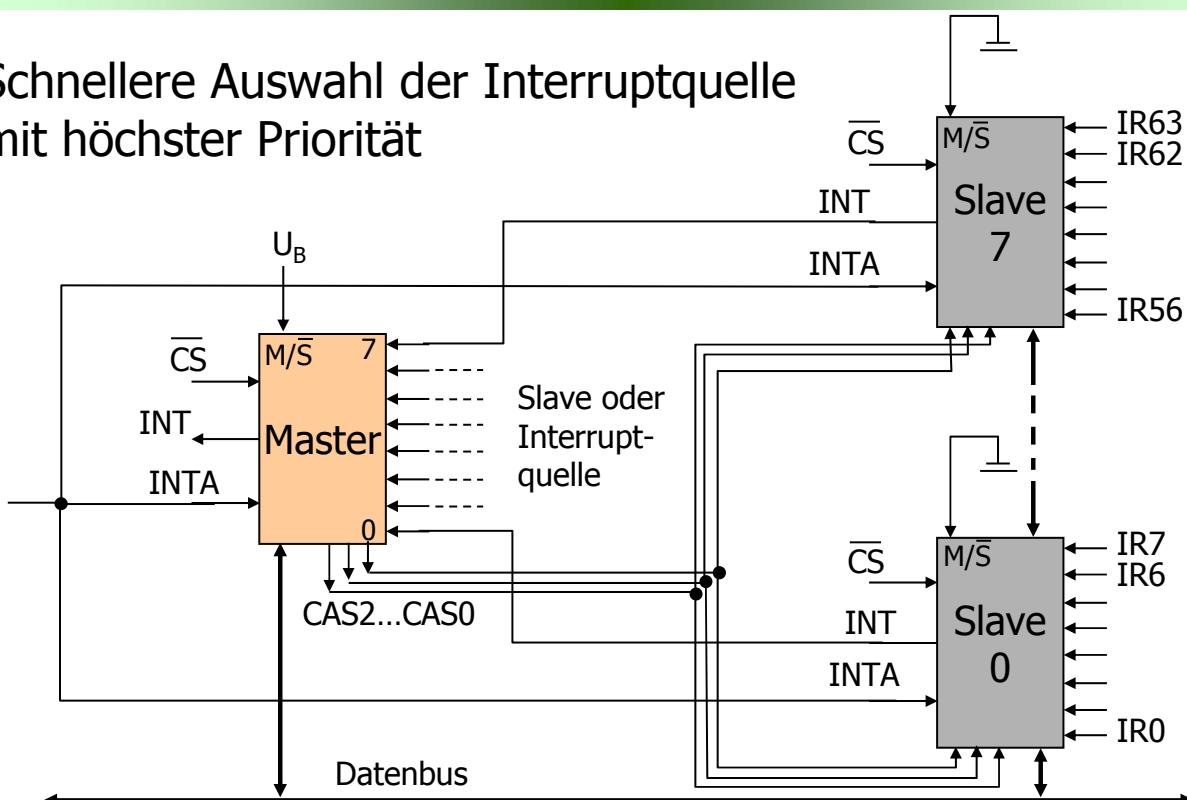
Einsatz mehrerer Interrupt-Controller

- 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 zusammenschaltet, 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

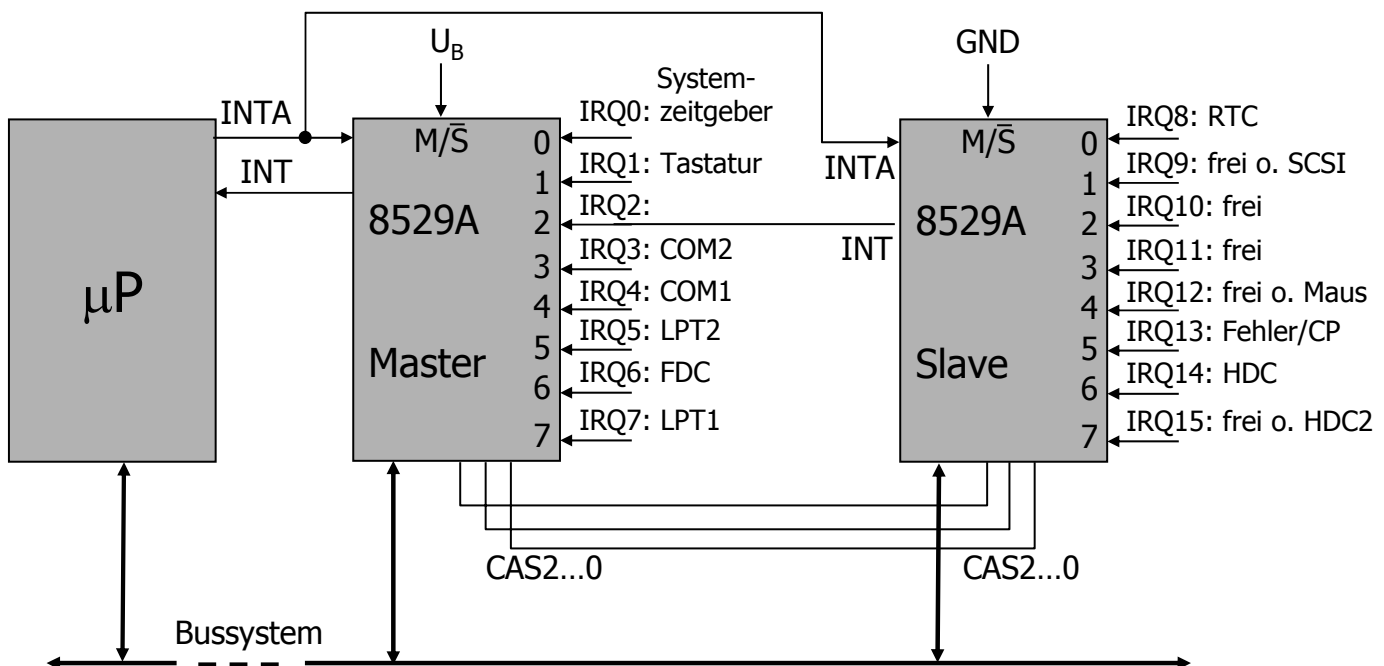


Kaskadierung von Interrupt-Controllern

- 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

