

Behandlung von Ausnahmesituationen

- ❑ Während des Betriebs eines Mikroprozessorsystems können Ausnahmesituationen (Exceptions) auftreten
- ❑ Eine 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
- ❑ Die Ausnahmebehandlung erfolgt durch eine Ausnahmeroutine (Interrupt Service Routine), deren Aktivierung durch eine Hardware-Komponente (Unterbrechungs-System, *Interrupt System*) im Steuerwerk unterstützt wird.
- ❑ Die Ausnahmeroutine hat Ähnlichkeit mit dem Aufbau eines Unterprogramms



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
- 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:** zum Programmablauf asynchrone Ereignisse. Sie werden durch die Hardware erzeugt
- ❑ **Prozessorinterne Ursachen:** zum Programmablauf synchrone Ereignisse. Sie treten fast nur durch Software (Programm) an fest vorgegebenen Stellen vor.



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.

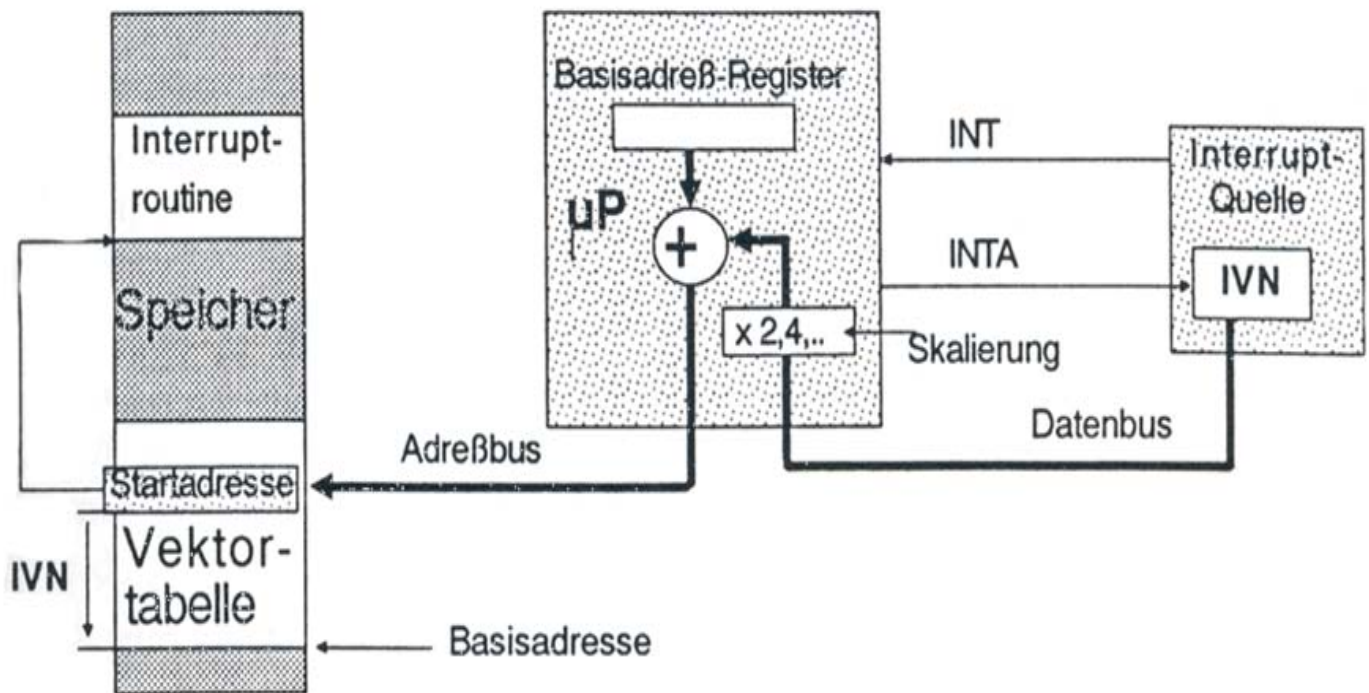


Prozessorinterne Ursachen

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-Unterbrechung (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	



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 zurücksetzen (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.

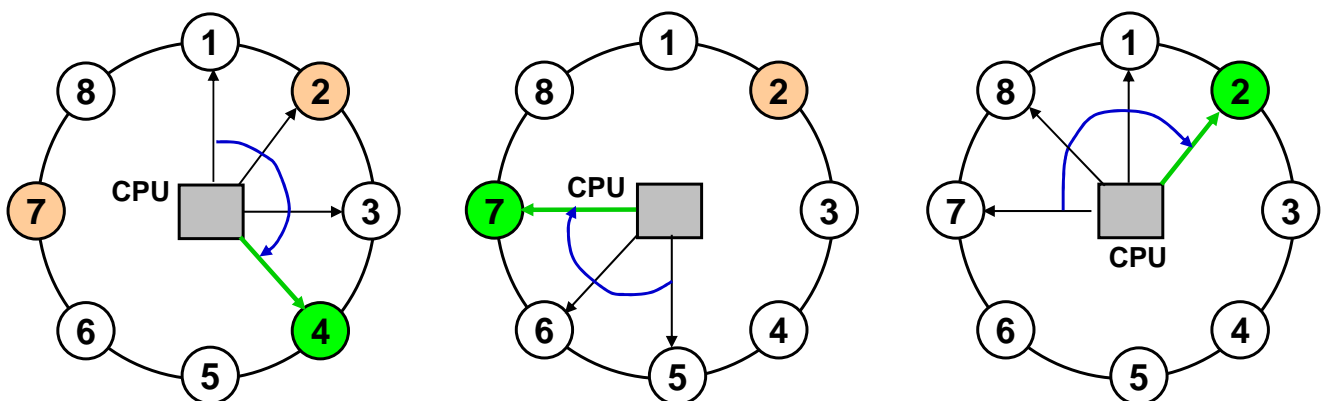
Nach (und auch während) der Abarbeitung eines Interrupts können weitere Anforderungen auftreten.

➔ Zwei Alternativen zur Behandlung



Polling: 1. Variante

Zyklische Abfrage wird 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

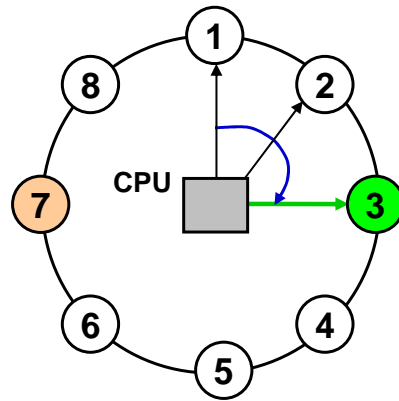
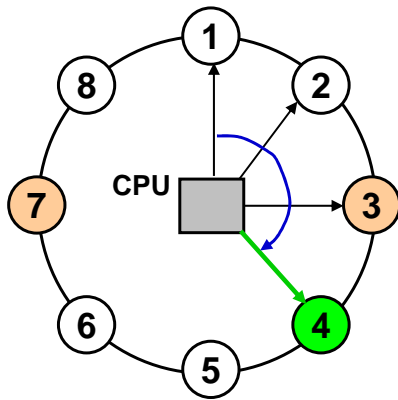


Interrupt-Anforderungen während der
Bedienung von Komponente 4



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.

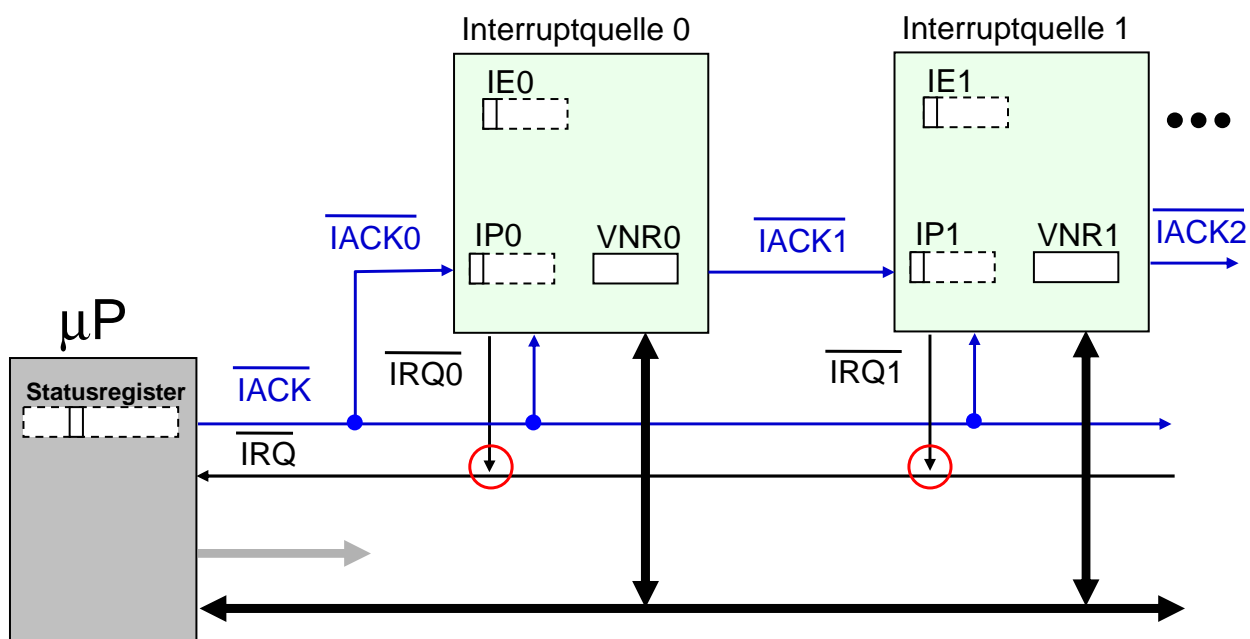


Daisy-Chain-Verfahren

- 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

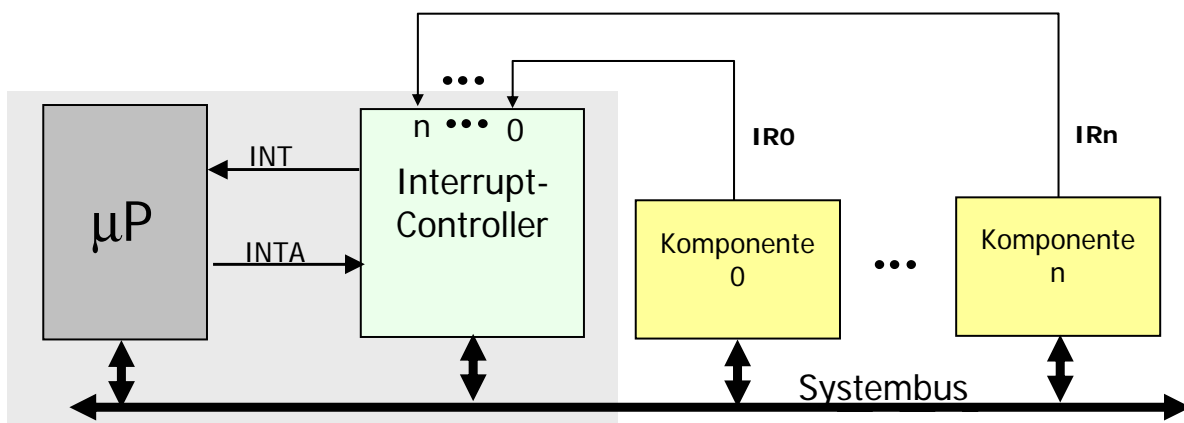


Daisy-Chain-Verfahren

- Anforderungen der einzelnen Quellen werden durch ODER zusammengeschaltet und über den Interrupteingang IRQ (Kreissymbol) zum Prozessor zugeführt.
- Prozessor leitet (bei gesetzten 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.



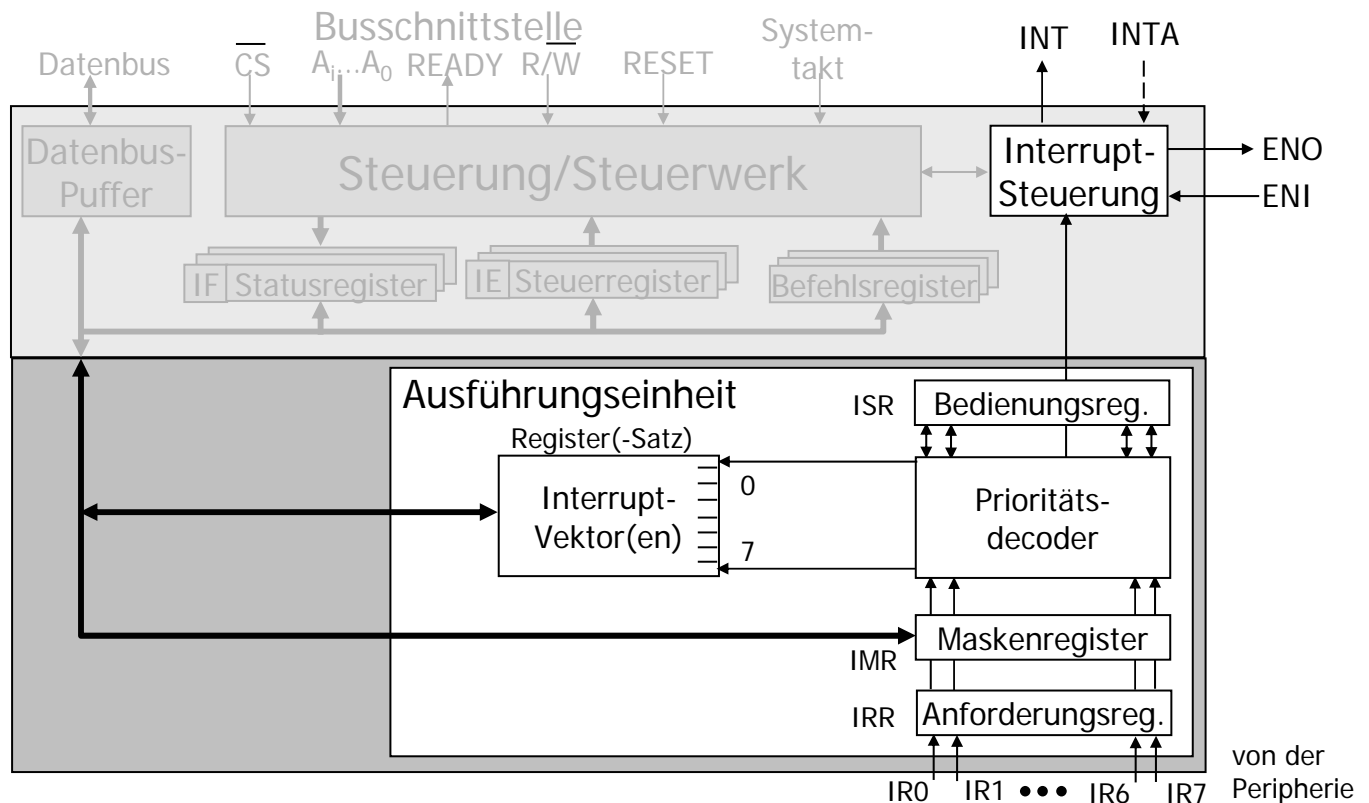
Mikroprozessor-System mit Interrupt-Controller



- 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



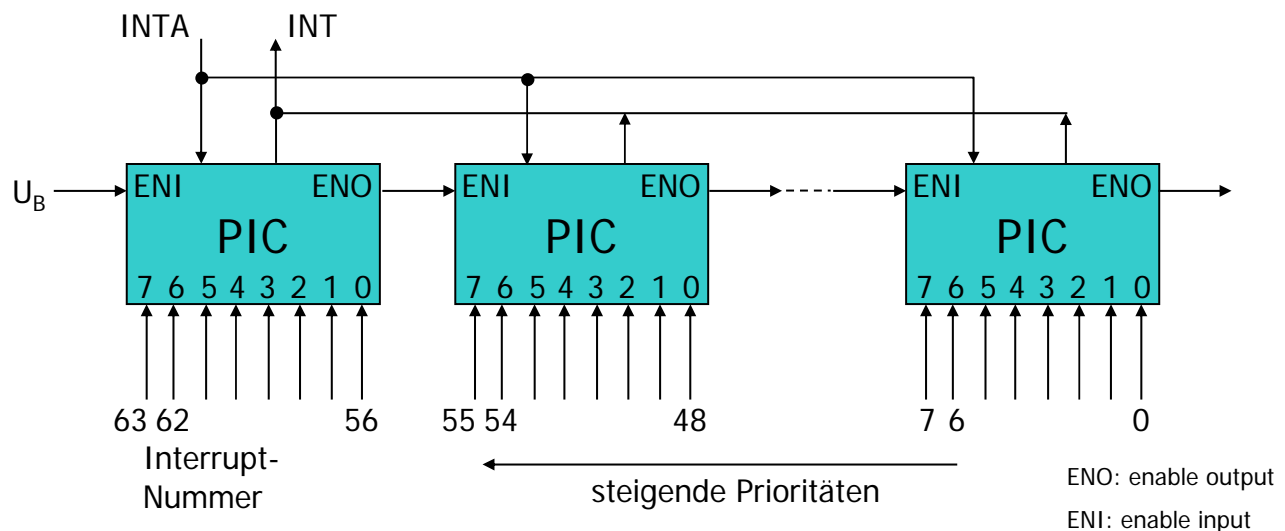
Aufbau eines Interrupt-Controllers

- ❑ Systemkomponenten melden ihre Unterbrechungswünsche über die Interrupt Request Eingänge (IR_0, \dots, IR_7). 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 Prioritätsdekor weitergereicht und letztlich ausgeführt, die im IMR nicht maskiert sind.
- ❑ Der Prioritätsdekor 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



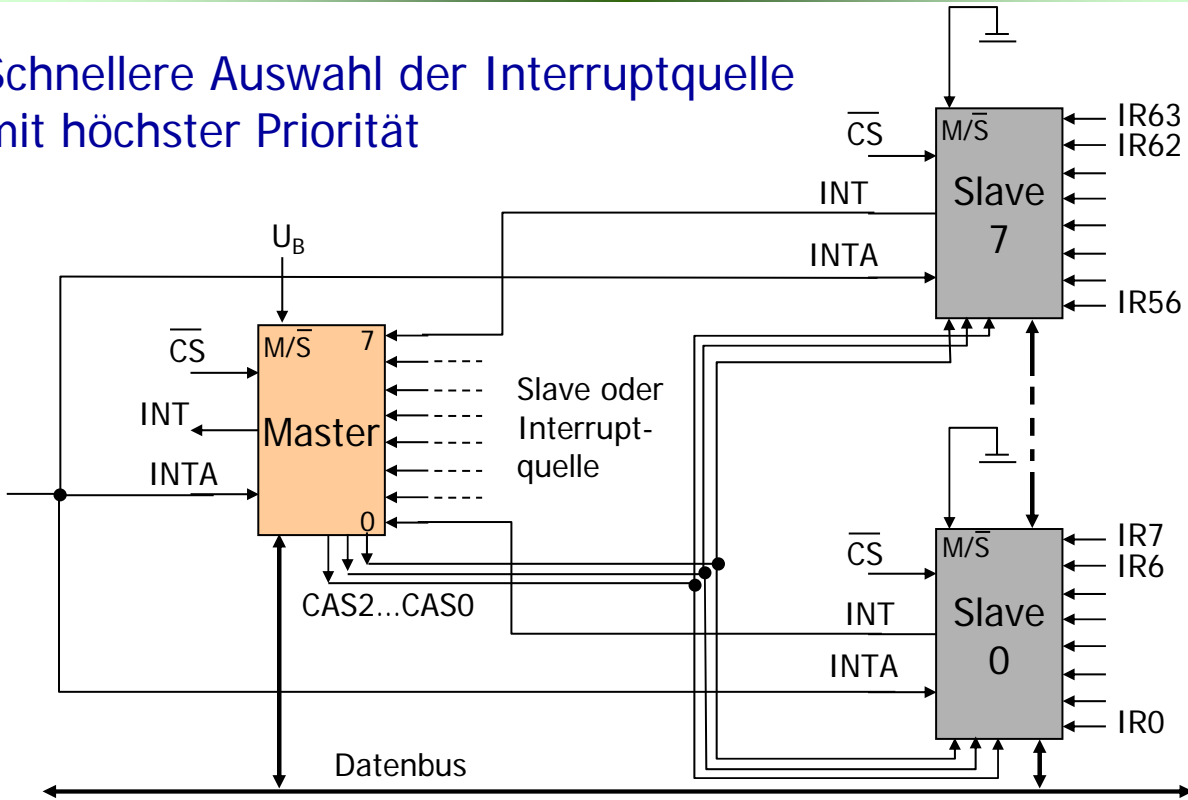
Einsatz mehrerer Interrupt-Controller

- ❑ 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

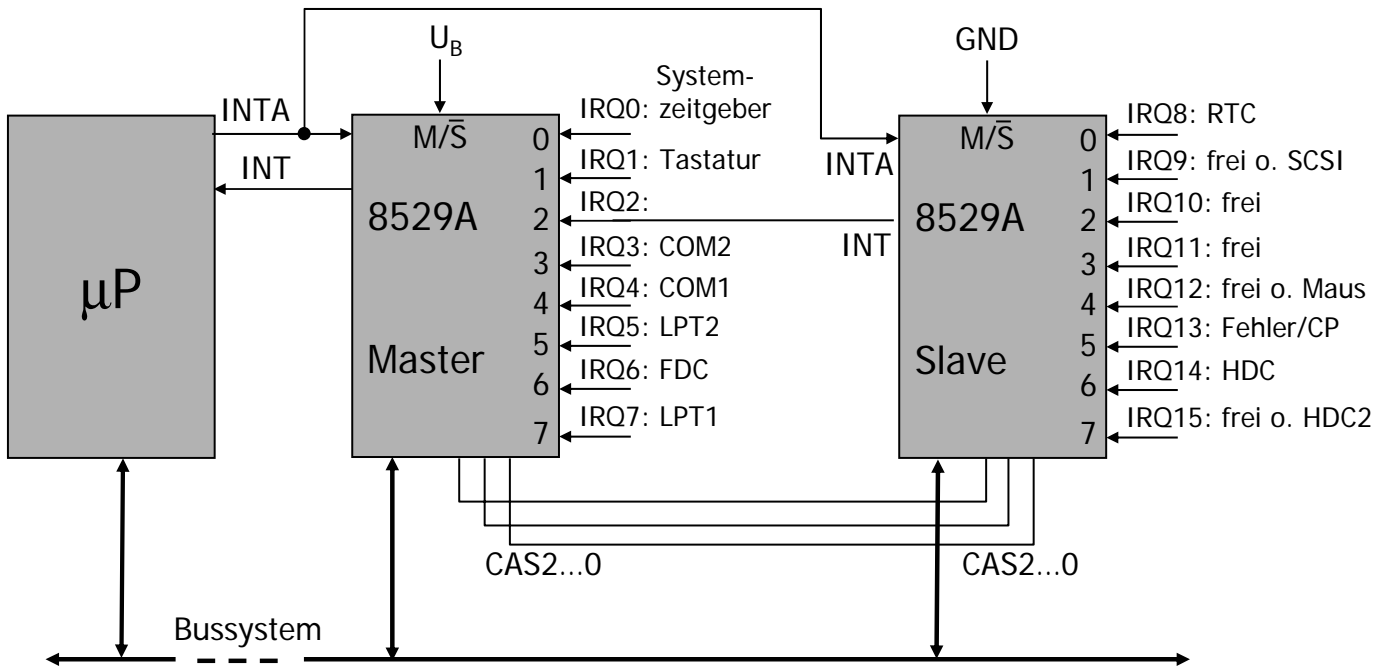
Schnellere Auswahl der Interruptquelle mit höchster Priorität



Kaskadierung von Interrupt-Controllern

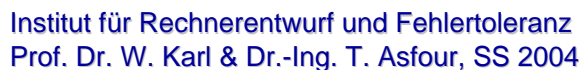
- ❑ Ein Controller übernimmt die Rolle des *Masters*.
- ❑ An den IRI-Eingängen des Masters werden wahlweise eine Interruptquelle oder ein weiterer Controller als *Slave* angeschlossen.
- ❑ Die Konfiguration des Interruptsystems wird in einem Register des Masters festgehalten.
- ❑ Bei Controllern mit 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

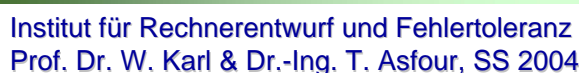
Slave am Eingang IRQ2



19-27

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



19-28



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

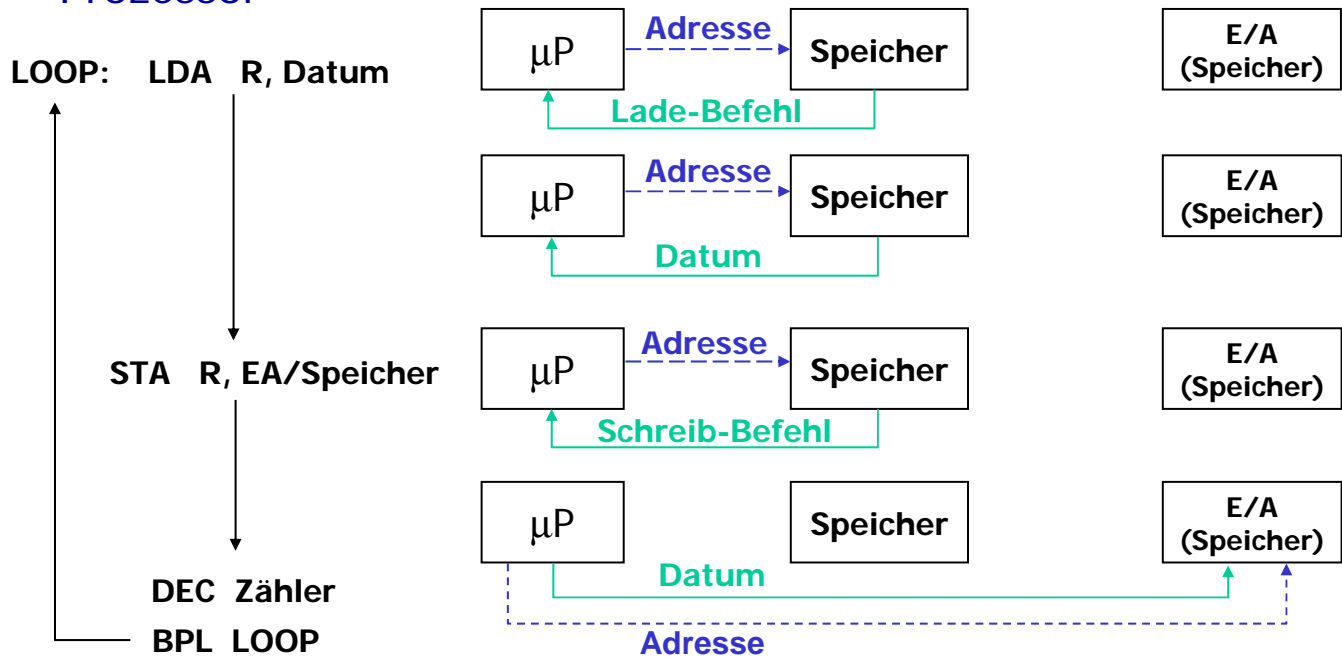
Peripherie \Leftrightarrow Peripherie

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

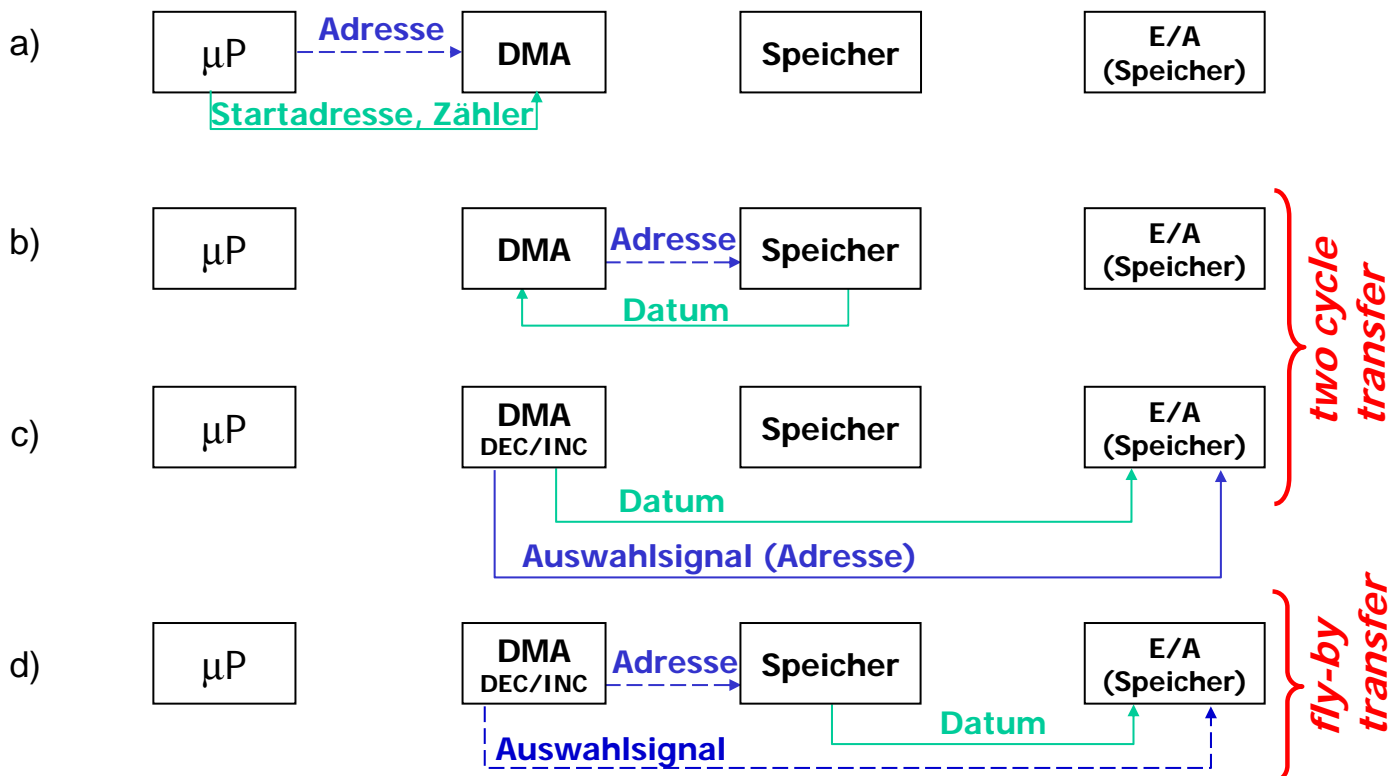
Ein spezieller Baustein, **DMA-Controller** genannt, koordiniert den Datentransfer

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

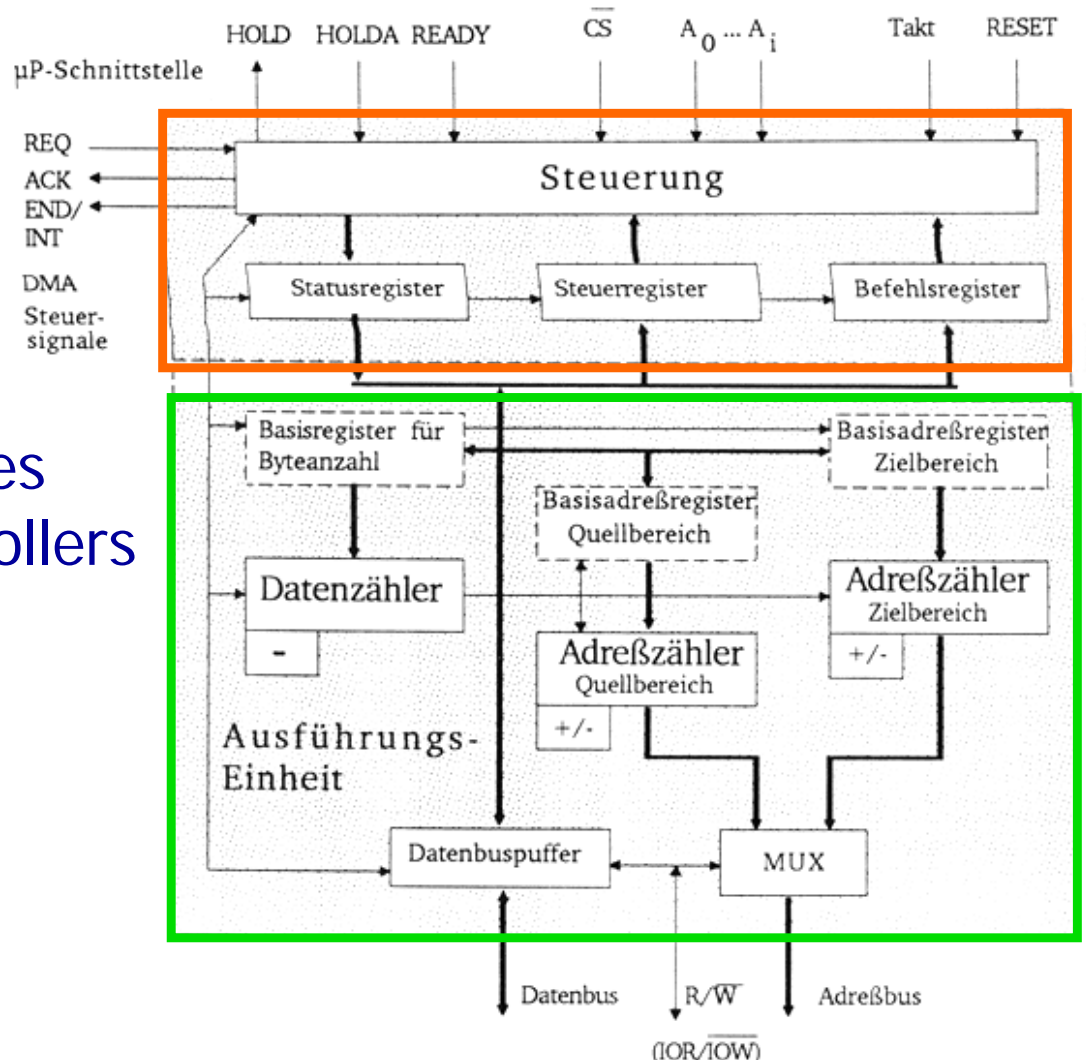
Mikroprozessor überträgt zu Beginn die Startadresse, Zieladresse, Anzahl der zu übertragenden Bytes, Übertragungsrichtung und Steuerinformationen

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{CS} : Selektion des Bausteins zur Programmierung durch μP
- $A_0 - A_i$: 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 μP einen Datentransfer abbrechen kann

3 Steuerregister zur Kontrolle und Programmierung durch den μP



Aufbau eines DMA-Controllers

Operationswerk: führt den Datentransfer aus

Besteht im wesentlichen aus 3 Zählern, die vom μ P 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



DMA-Übertragungsarten

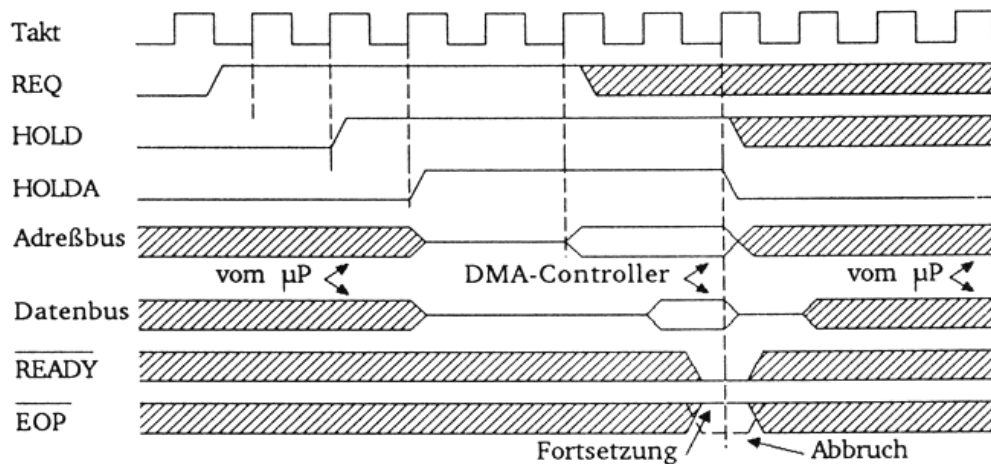
In der Regel kann man bei einem DMA-Baustein unter verschiedenen Übertragungsarten wählen, die sich durch die Belegung des Systembusses unterscheiden:

- **Einzeltransfer** (*single transfer mode*)
- **Block-Transfer** (*block transfer mode, burst mode*)
- **Transfer auf Anforderung** (*demand transfer mode*)



Einzeltransfer (*single transfer mode*)

Hierbei wird jeweils genau ein Datum übertragen, danach der Systembus wieder für den μP freigegeben (*cycle stealing*, dem μP werden einzelne Buszyklen entzogen)

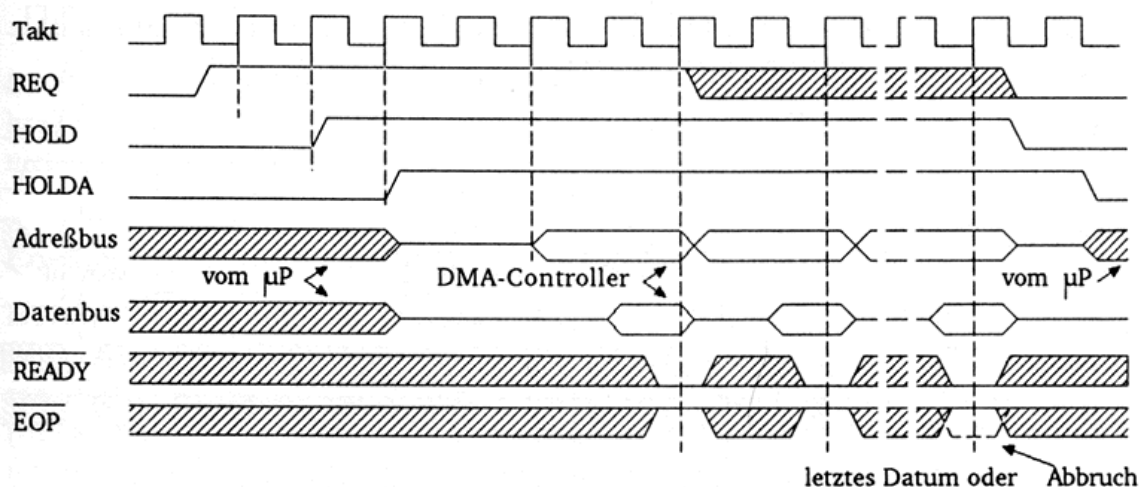


- REQ: Anforderung eines Requesters zum Datentransfer
- HOLD: Busanforderung durch den DMA-Controller
- HOLDA: Busgewährung durch den μP
- READY: beendet die Übertragung des Datums, wenn 0



Block-Transfer (*block transfer mode, burst mode*)

Der DMA-Controller überträgt hierbei alle Daten des Blocks ohne zwischenzeitliche Busfreigabe, sobald er den Systembus erhalten hat

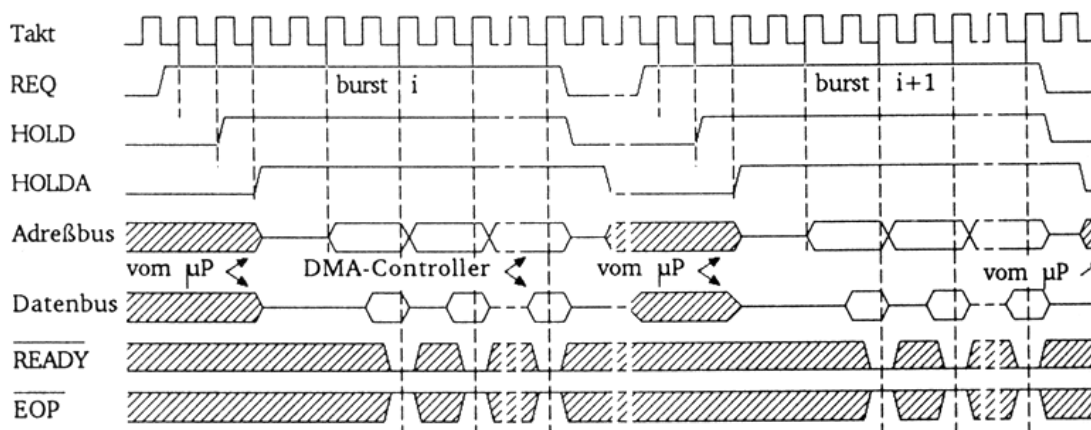


Transfer auf Anforderung (*demand transfer mode*)

Mittelstellung zwischen Einzeltransfer und Blocktransfer

Datenübertragung wird solange ohne Unterbrechung durchgeführt, solange das REQ-Signal aktiv ist

➔ Datentransfer in Schüben, häufig von Peripheriegeräten benutzt (echter *Burst-Mode*)



DMA-Übertragungsarten

Besonders zu beachten bei DMA-Transfers sind:

- ❑ unterschiedliche Datenbreite von Quelle und Ziel, z. B. Übertragungen von einem 8-Bit breiten Peripherie-Gerät (Drucker, Terminal, ...) in den 32-Bit breiten Hauptspeicher
- ❑ Übertragung von non-aligned Daten, d. h. mehrere Speicherzugriffe zum Lesen oder Schreiben eines Datums sind notwendig

➔ DMA-Controller sind durch verschiedene Maßnahmen an diese Anforderungen angepaßt, wie z. B:



DMA-Übertragungsarten

- ❑ Wahlweise Inkrementierung der Adresszähler um 1, 2 oder 4
- ❑ Zerlegung bzw. Sammeln von Daten vor dem Weitertransport in internen Registern bei unterschiedlicher Breite von Quelle und Ziel
- ❑ Automatische Erkennung von non-aligned Zugriffen und Zerlegung dieser Zugriffe in mehrere Speicherzyklen

Moderne DMA-Controller enthalten 2 - 8 DMA-Kanäle, jeder Kanal besitzt eigene Leitungen REQ_i , ACK_i , END_i und kann so unabhängig von den anderen einem Requester zugeordnet werden



Register des Steuerwerks eines DMA-Controllers

Orientiert am Intel 82380 DMA-Controller
4 unabhängige DMA-Kanäle auf dem Baustein

Register des Steuerwerks:

Statusregister

R3	R2	R1	R0	TC3	TC2	TC1	TC0
----	----	----	----	-----	-----	-----	-----

Befehlsregister

E3	E2	E1	E0	P3	P2	P1	P0	SR3	SR2	SR1	SR0	IE3	IE2	IE1	IE0
----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----

Steuerregister (individuell für jeden Kanal)

T1	T0	AI	M1	M0	C	RT	RA1	RA0	RB1	RB0	TT	TA1	TA0	TB1	TB0
----	----	----	----	----	---	----	-----	-----	-----	-----	----	-----	-----	-----	-----



Register des Steuerwerks eines DMA-Controllers

Statusregister:

Bitpaare pro Kanal zur Anzeige folgender Informationen:

R_i : Übertragungsanforderung (*request*) für Kanal i liegt vor

TC_i : Datenübertragung auf Kanal i beendet (*terminal count*)

Befehlsregister:

Bits zur Bestimmung der Arbeitsweise der einzelnen Kanäle

E_i : Freigabe des Kanals i (*enable*)

P_i : Prioritätensteuerung zwischen den Kanälen

SR_i : Software-Request (als Alternative zur REQ-Leitung) möglich

IE_i : Interrupt Enable für Kanal i



Steuerregister (eines pro Kanal)

Steuerregister:

bestimmt die Betriebsart des Kanals

T_1, T_0 : Typ der Übertragung (*send, receive, verify*)

AI : automatische Initialisierung (automatisches Nachladen der Zähler aus den Basisregistern nach Übertragungsende)

M_1, M_0 : Modus der Übertragung (Einzeltransfer, Blocktransfer, Transfer auf Anforderung)

C : Transferzyklen (*two-cycle transfer, fly-by transfer*)



Steuerregister (eines pro Kanal)

TT	Typ der Komponente für Target und Requester
RT:	(Arbeitsspeicher / Peripherie)
TA ₁ , TA ₀	Aktion des Adresszählers für Target und Requester
RA ₁ , RA ₀ :	(Inkrement, Dekrement, Hold)
TB ₁ , TB ₀	Busbreite für Target und Requester (8, 16, 32 Bit)
RB ₁ , RB ₀ :	

