

□ Anmeldung zur TI-Klausur:

- Einwurf der Zulassungsbescheinigung in den Briefkasten im Untergeschoss des Informatikgebäudes am Fasanengarten **bis spätestens 24. August**. 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** (9.00 – 11.00 Uhr)
- **Informationswirtschaft: 60 Minuten** (9.00 – 10.00 Uhr)

□ Studentenausweise unbedingt in die Klausur mitbringen

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



Kapitel 9

- Zeitverhalten des Bussystems
- Systemsteuer- und Schnittstellenbausteine
- Ausnahmebehandlung



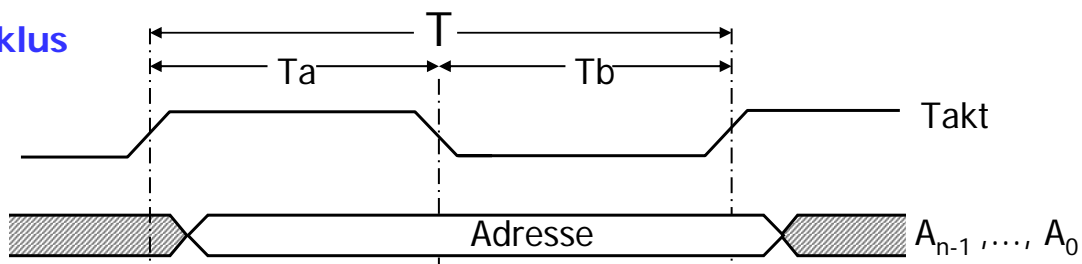
Zeitverhalten der Systembussignale

- ❑ Synchroner Systembus
- ❑ Semi-Synchroner Systembus
- ❑ Asynchroner Systembus



Zeitverhalten eines synchronen Systembus

T: Buszyklus



Timing

Adresse wird zu Beginn des Buszyklus (T_a) auf den Adressbus gelegt

Auswahl der Übertragungsrichtung durch R/\overline{W}

Lesen ($R/\overline{W} = 1$):

- Speicher (oder andere Systemkomponente) liefert ihre Daten gegen Ende des Buszyklus (T_b)
- Übernahme der Daten in den Prozessor mit der steigenden Flanke des Systemtakts

Schreiben ($R/\overline{W} = 0$):

- Prozessor legt die Daten zu Beginn der zweiten Takthälfte (T_b) auf den Systembus
- Übernahme der Daten in den Speicher durch die steigende Flanke von R/\overline{W} oder des Systemtakts

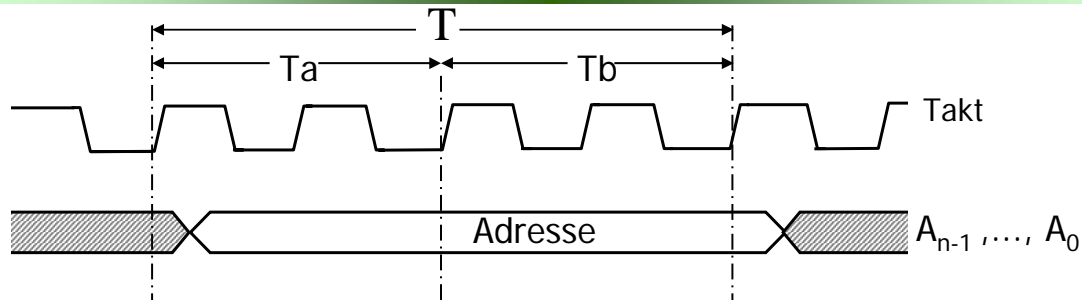


Synchroner Systembus

- Alle Vorgänge synchron zum Takt nach einem starren Muster ablaufen → **synchroner Systembus**
- Übergabe und Übernahme der Daten geschieht zu festgelegten Taktflanken
- Synchrone Busse in den Anfangsjahren der μ Pen
- **Nachteil:** Alle am Bus angeschlossenen Komponenten müssen strenge Zeitvorgaben erfüllen
 - ➔ Langsamste Komponente bestimmt die zulässige Geschwindigkeit des Busses, oder aber der Bus schließt den Einsatz von „schnellen“ Komponenten aus (z. B. keine schnelle Speicher ☹)



Semi-synchroner Systembus



Timing

Lesen ($R/\overline{W} = 1$):

- hinreichend schneller Speicher liefert seine Daten im letzten Taktzyklus von (T_b)
- Übernahme der Daten in den Prozessor durch die steigende Taktflanke des nächsten Buszyklus

Schreiben ($R/\overline{W} = 0$):

- Prozessor legt die Daten zu Beginn der zweiten Takthälfte von (T_a) auf den Systembus
- Übernahme der Daten in den Speicher durch die steigende Flanke von R/\overline{W}



Semi-synchroner Systembus

Um auch langsamere Speicher benutzen zu können:

Steuereingang $\overline{\text{READY}}$

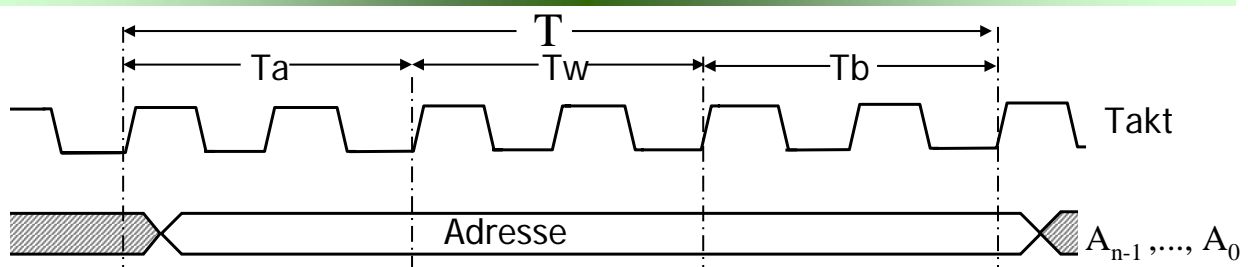
Buszyklus wird nur abgeschlossen, wenn rechtzeitig vor Ende von der Takthälfte (T_b) $\overline{\text{READY}} = 0$ ist

Ist $\overline{\text{READY}}$ am Ende des Buszyklus nicht 0

→ Es werden solange **Wartezyklen** (T_w , Dauer: z. B. halbe Buszykluslänge) eingefügt, bis $\overline{\text{READY}} = 0$ wird



Einfügen eines Wartezyklus



Semi-synchroner Systembus

- Moderne Prozessoren besitzen höhere Taktfrequenzen
- Schreibe- bzw. Lesezugriffe benötigt mehrere Taktzyklen
- **Steuereingänge (READY)** zur Synchronisation der Buszugriffe durch Einführung von Wartezyklen (*wait states*) → unterschiedlich schnelle Speicher und Geräte können individuell bedient werden
- Bezeichnung: Semi-synchrone Busse (Synchrone Busse mit Wartezyklen)
- **Timing:**
 - Adresse zu Beginn des Buszyklus (T_a) auf den Adressbus
 - Auswahl der Übertragungsrichtung wieder durch $\overline{R/W}$

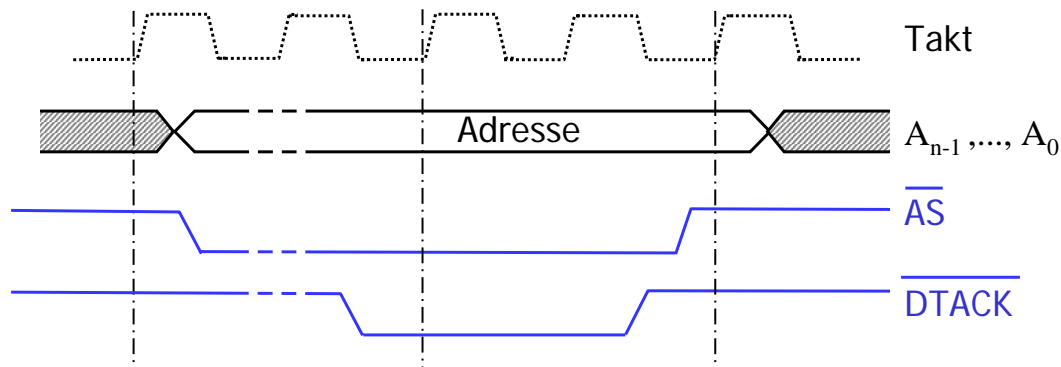


Semi-synchroner Systembus

- Bus ist immer noch synchron (streng am Takt orientiert), die Dauer eines Buszyklus ist jedoch nicht mehr fest, sondern in Vielfachen von Taktzyklen variierbar
 - ➔ **semi-synchroner Systembus**
- höherer Steueraufwand als synchroner Systembus
- Zeitverhalten ist auf verschieden schnelle Bausteine anpassbar
- Sind nur ausreichend schnelle Bausteine im System, kann **READY** z. B. fest auf 0 gelegt werden. Anderenfalls kann dieses Signal durch eine geeignete Verzögerungsschaltung (z. B. Monoflop) erzeugt werden



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:

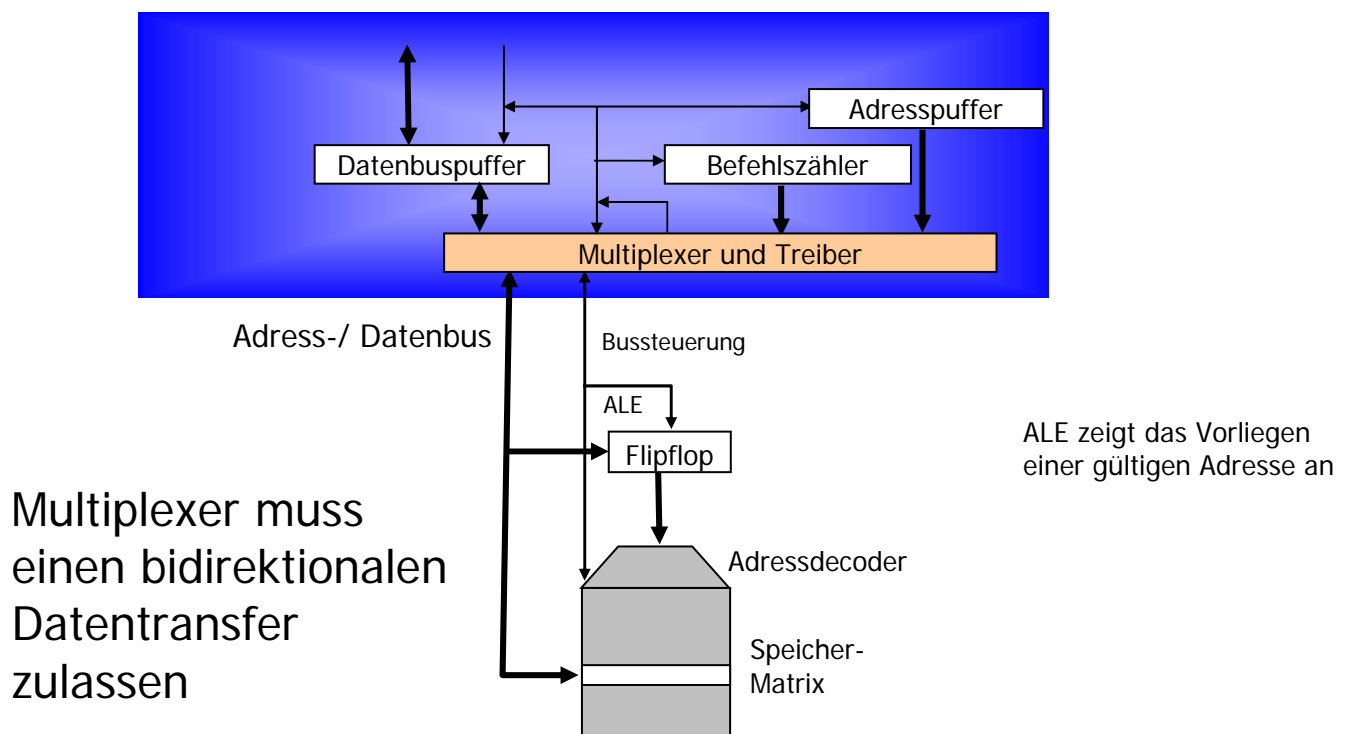
gemeinsamer Daten- und Adreßbus (Bsp.: PCI-Bus)

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

➔ Speichern der Adresse in Flipflops (Latches)

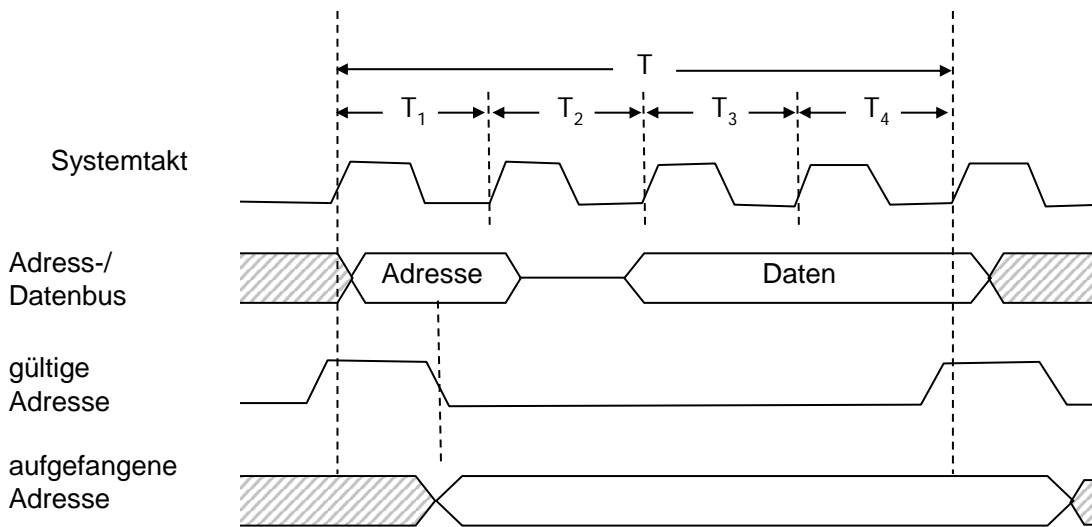


Multiplex-Busschnittstelle



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 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.

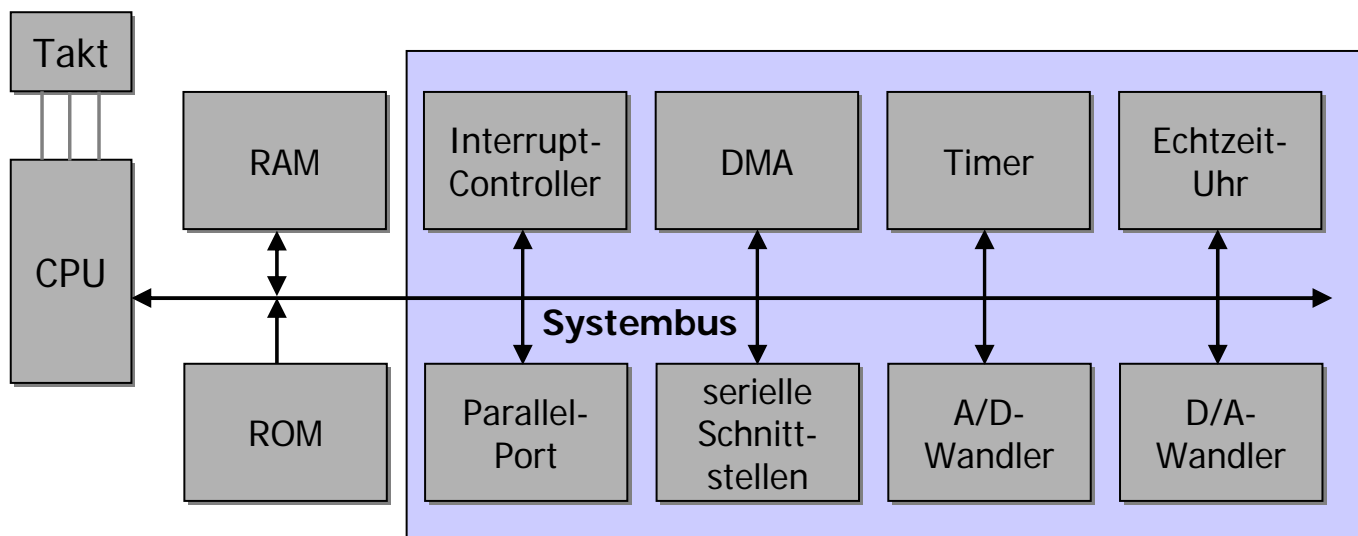
Aufgaben:

- 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

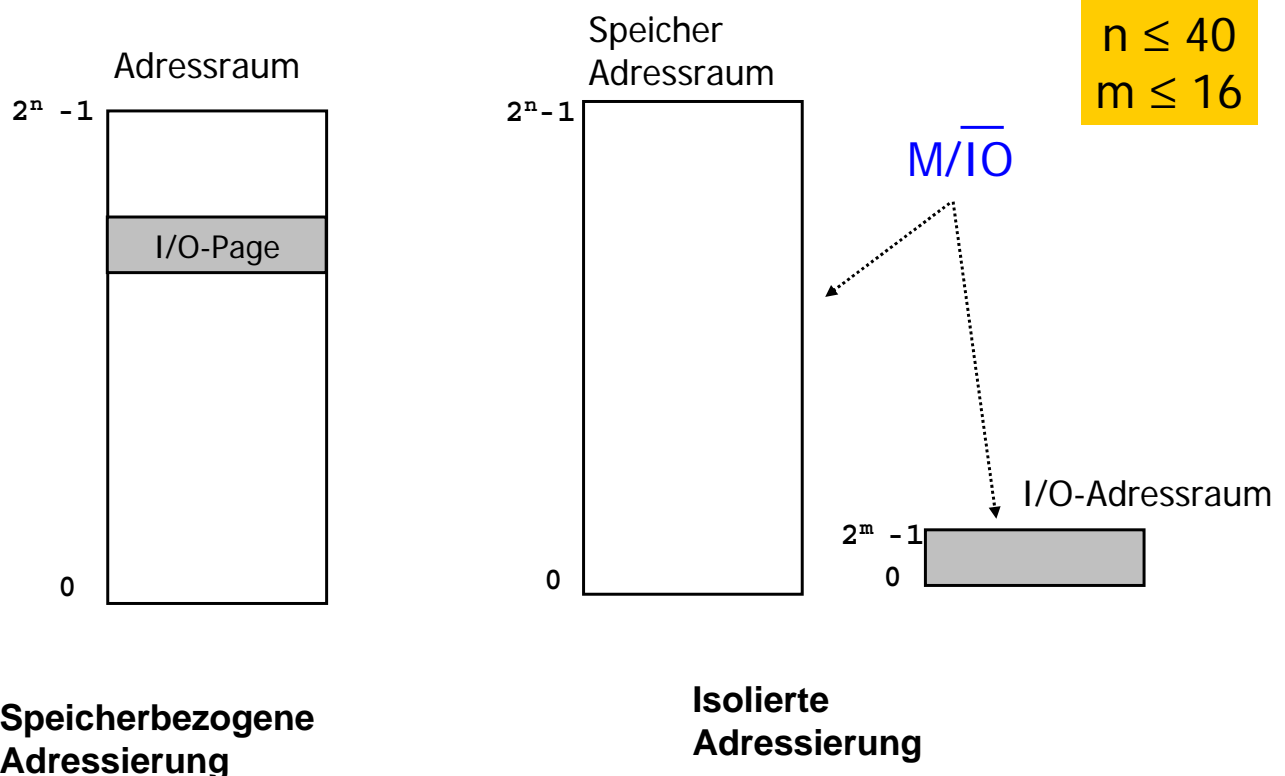
Jeder programmierbare Systembaustein erscheint für den Prozessor wie ein kleiner Satz von Registern, die unter einem zusammenhängendem Block von Adressen (Portadressen) angesprochen werden können.

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



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:

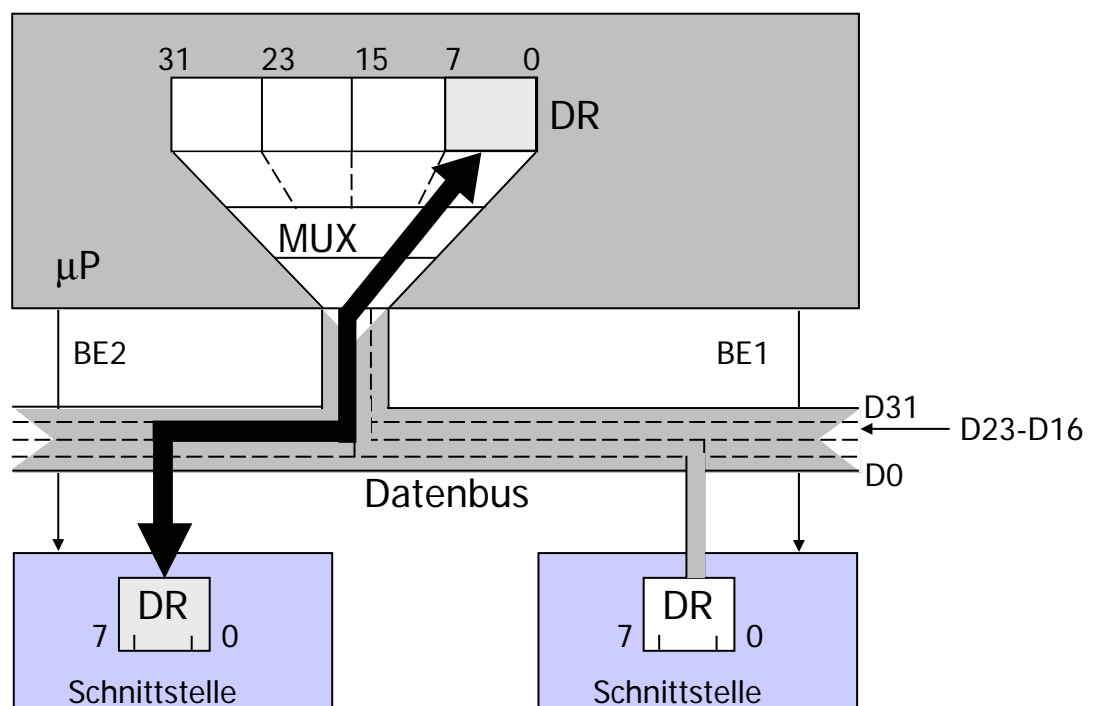
getrennte Adressräume für Speicher und Peripherie (eigener I/O-Adressraum)

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



Anschluss der Schnittstellenbausteine an dem μP

8-bit-Schnittstelle am 32-bit-Prozessor

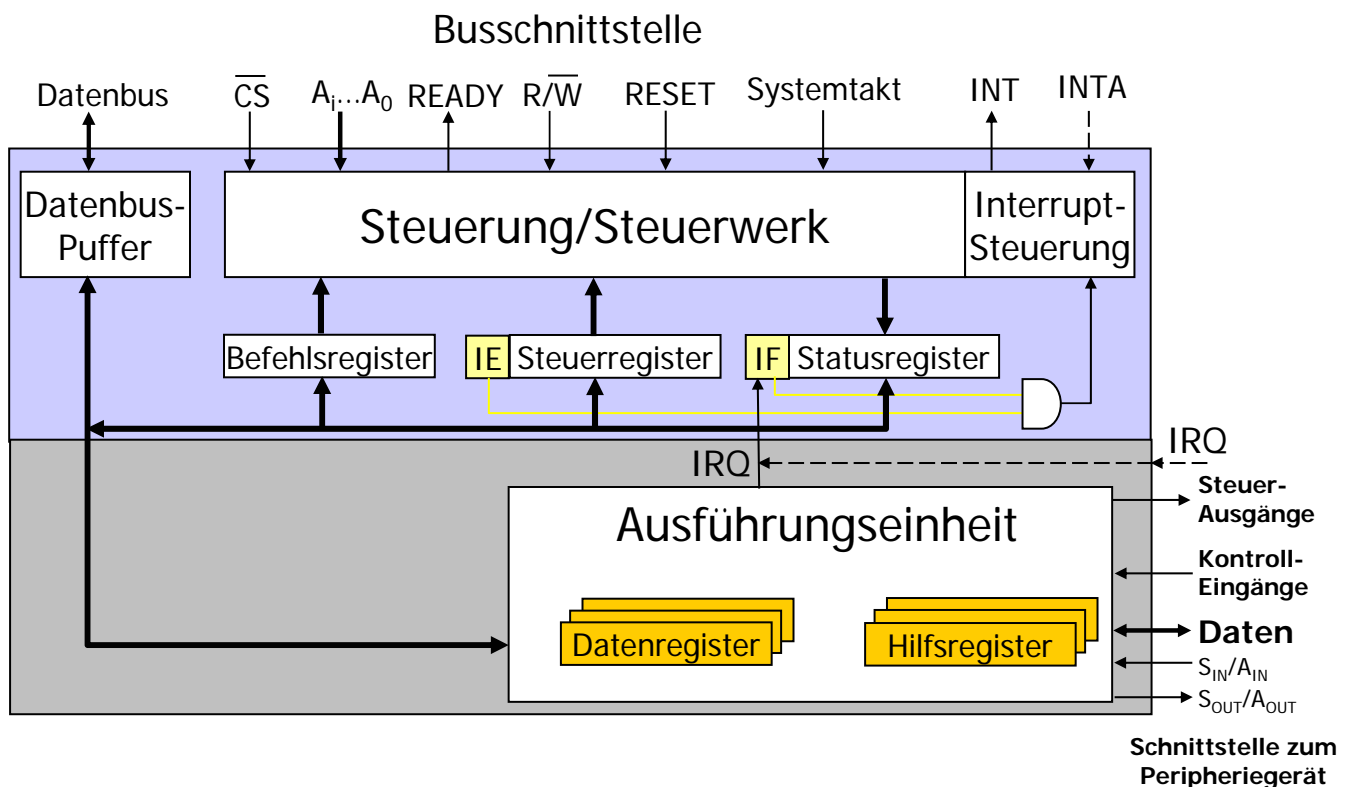


Anschluss der Schnittstellenbausteine an dem μP

- ❑ 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:

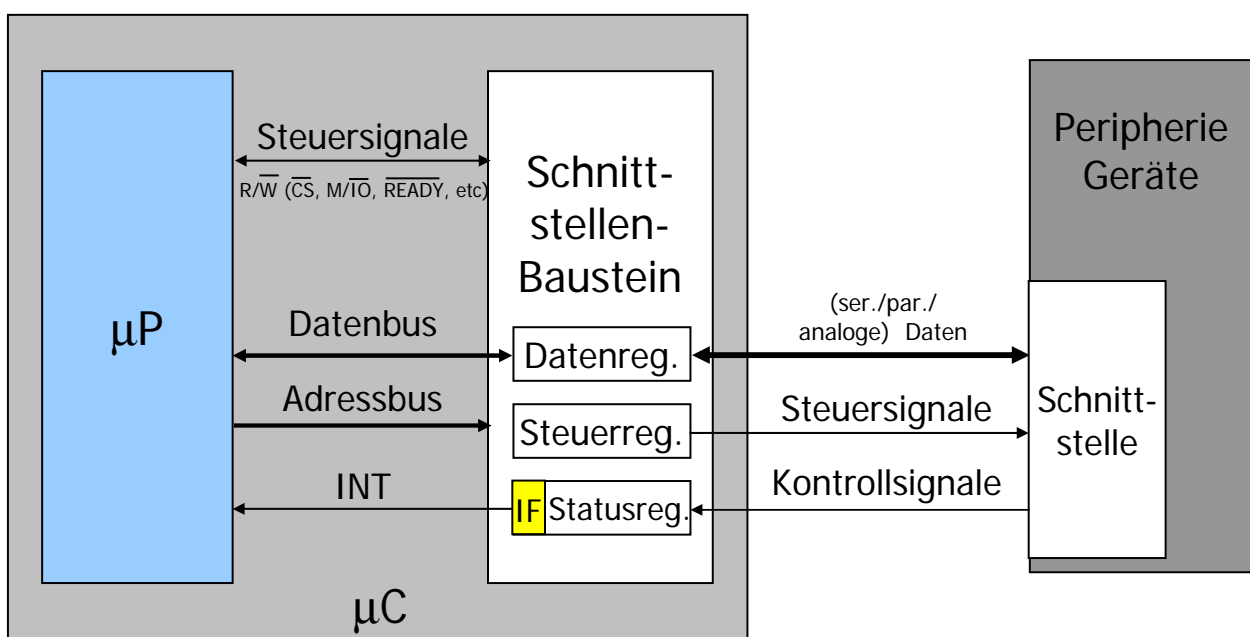
- Steuerung der internen Komponenten (Register, Multiplexer, ...) und Datenpfade
- Schnittstelle zum Prozessor
- Register zur Bausteinprogrammierung: Statusregister (Bausteinstatus), Steuerregister (Betriebsart), Befehlsregister (aktuelle Operation)

Ausführungseinheit:

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



Schnittstellenbaustein zwischen μP und Peripheriegerät



- **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

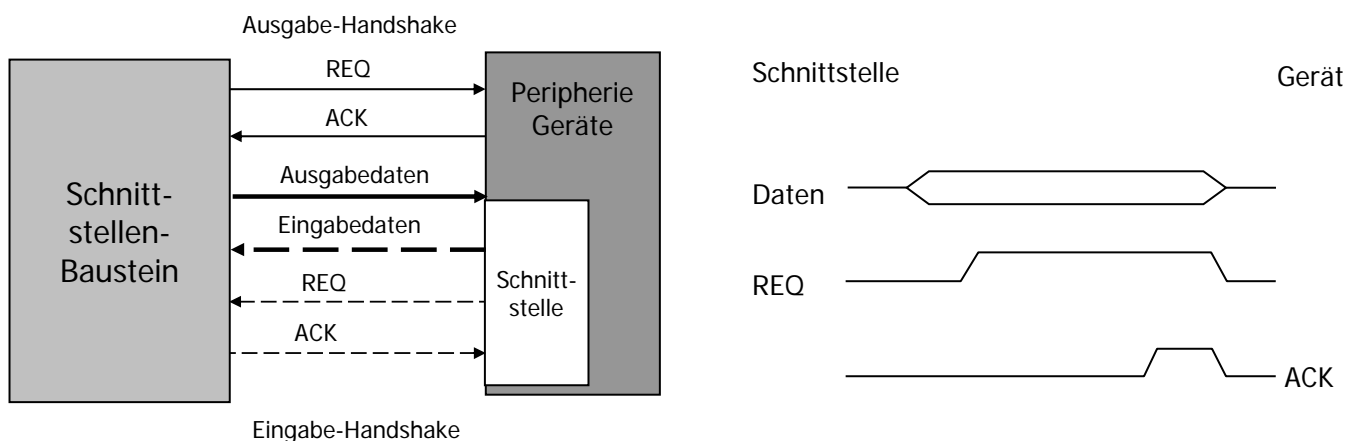
- **Programmierte Ein-/Ausgabe:** Prozessor fragt ständig das Statusregister des Peripheriebausteins ab, ob er bereit ist ein Datum zu übertragen. Zwischen den Übertragungsanforderungen kann der Prozessor andere Aufgaben erledigen oder in einer Schleife ausschließlich das Statusregister abfragt (*busy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

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)

