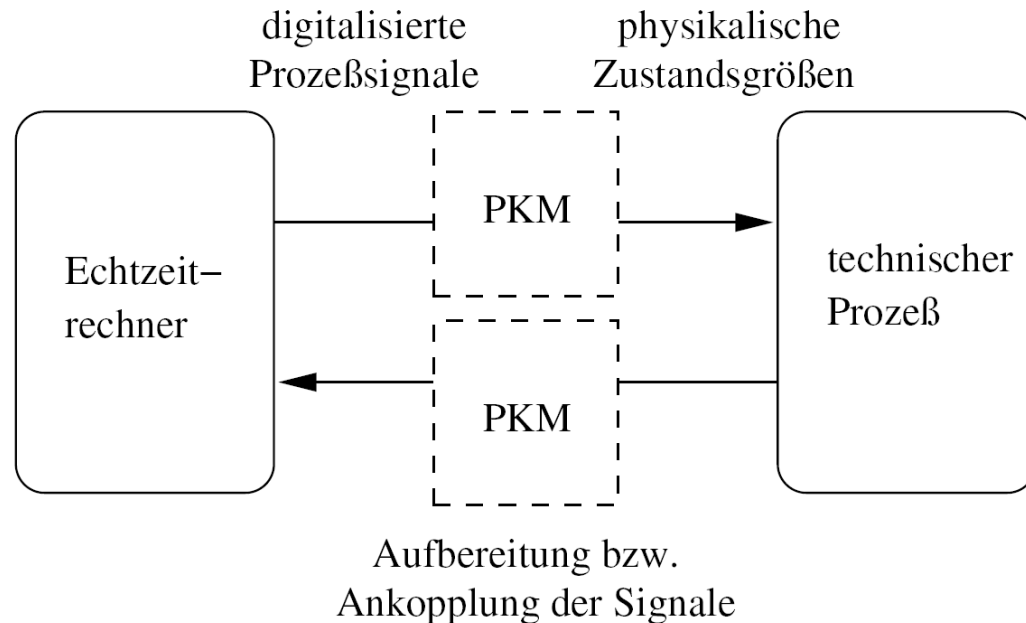


Prozessankopplung, HW

Übersicht

Problem: Einlesen und Beeinflussen physikalischer Zustandsgrößen eines technischen Prozesses

- **Anbindung eines Echtzeitrechners über Peripheriekoppelmodule (PKM's) bzw. Gerät (*device*) an technischen Prozess**



PKM: Peripheriekoppelmodul

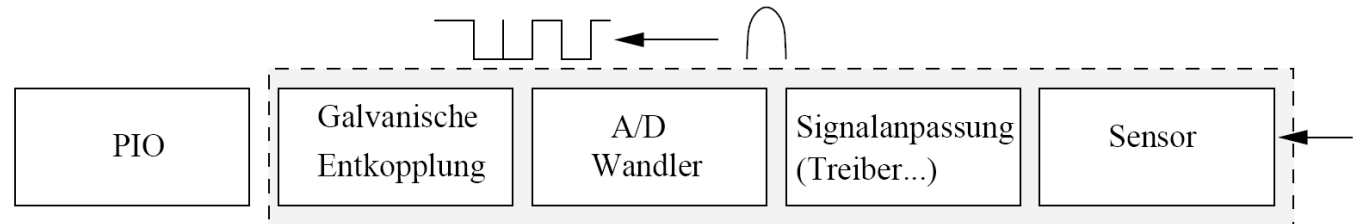


Prozessankopplung, HW

Übersicht

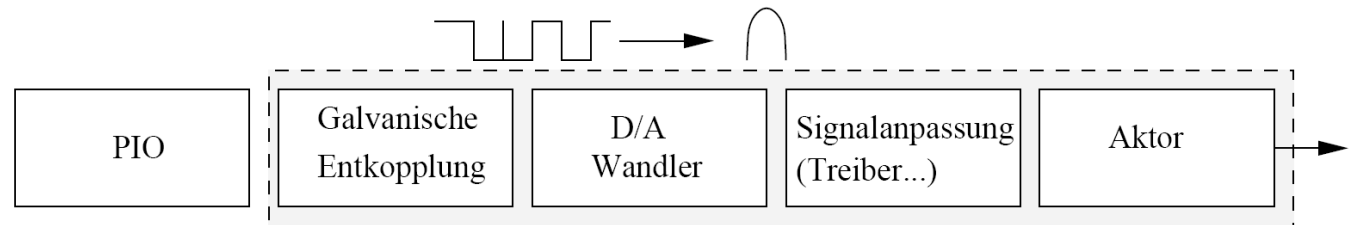
Erfassung der physikalischen Zustandgrößen

- Messwertaufnehmer, Sensoren (z.B. Temperatursensor)



Beeinflussung der Zustandgrößen

- Stellglieder, Aktoren (z.B. Motor, Ventil).



Anbindung der Peripheriekoppelmodule an Echtzeitrechner:

- Systembus



Prozessankopplung, HW

Übersicht

Peripheriekoppel-Modul (Gerät) belegt im Echtzeitrechner Hardware-Ressourcen

- **Speicheradressen**
- **I/O-Adressen**
- **Interrupts**

Logische Ebene

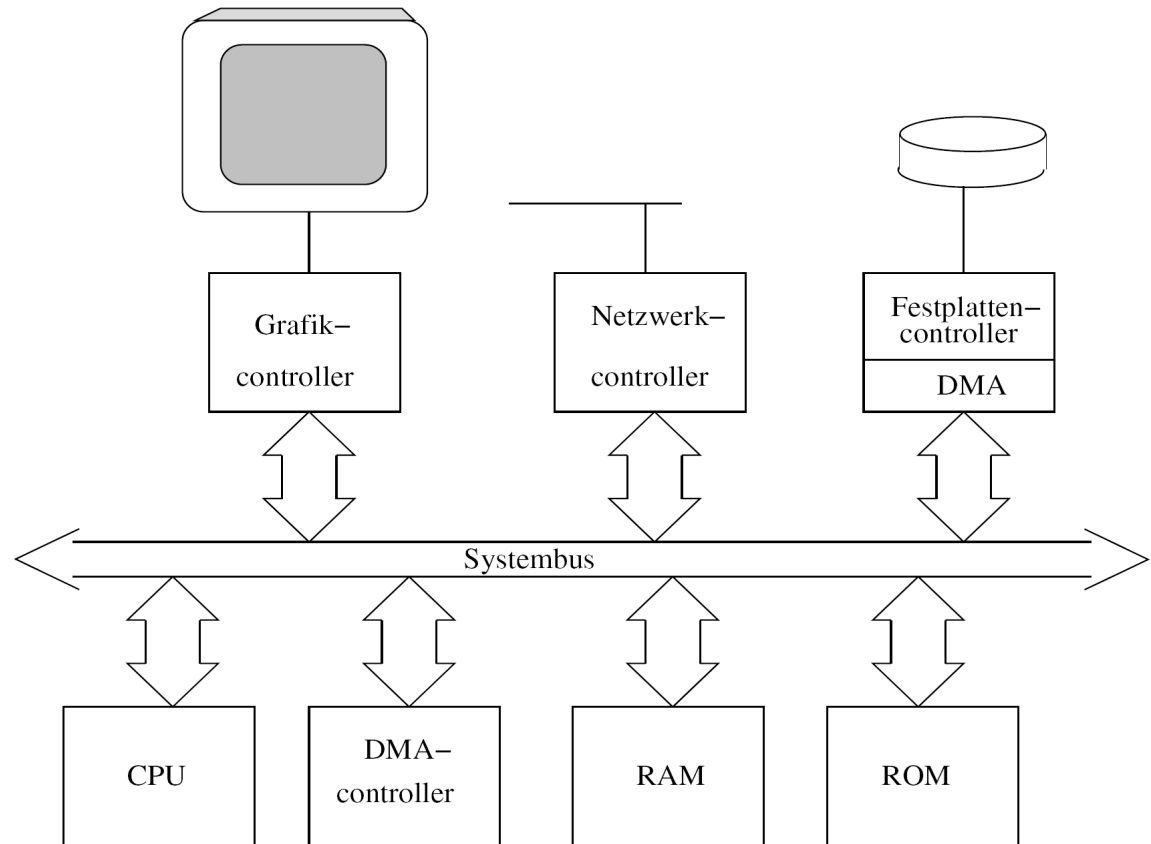
- **Einfache Geräte**
 - einzelner Sensor
 - einzelner Aktor
 - eine Adresse mit einem Wert
- **Intelligente Geräte**
 - mehrere Adressen (Datenregister, Kontrollregister, Statusregister), Vorverarbeitung



Prozessankopplung, HW

Physikalische Ankopplung, Systembus

Parallele „Leitungen“ (Bus), die Peripherie (Speicher, Interrupt-Controller, Geräte-Controller usw.) an CPU anbinden



Prozessankopplung, HW

Physikalische Ankopplung, Systembus

Drei logische Einheiten

- **Adressbus (AB)**
 - Auswahl von einzelnen Speicherzellen über deren Adresse
- **Datenbus (DB)**
 - Datentransport zwischen CPU-Registern und Speicherzellen (bidirektional)
- **Kontrollbus (CB)**
 - Gültigkeit der Adressen auf dem Adressbus oder der Daten auf dem Datenbus (Strobe-Signale).
 - Schreib/Leserichtung
 - Interrupt
 - Busmastertransfer



Prozessankopplung, HW

Physikalische Ankopplung, Systembus

Vorab: Unterscheidung

- **Active High Signale**
 - Kein Strich über Signalnamen
 - Gültig/gesetzt, wenn Signalpegel 1
- **Active Low Signale**
 - Strich über Signalnamen
 - Gültig/gesetzt, wenn Signalpegel 0



Prozessankopplung, HW

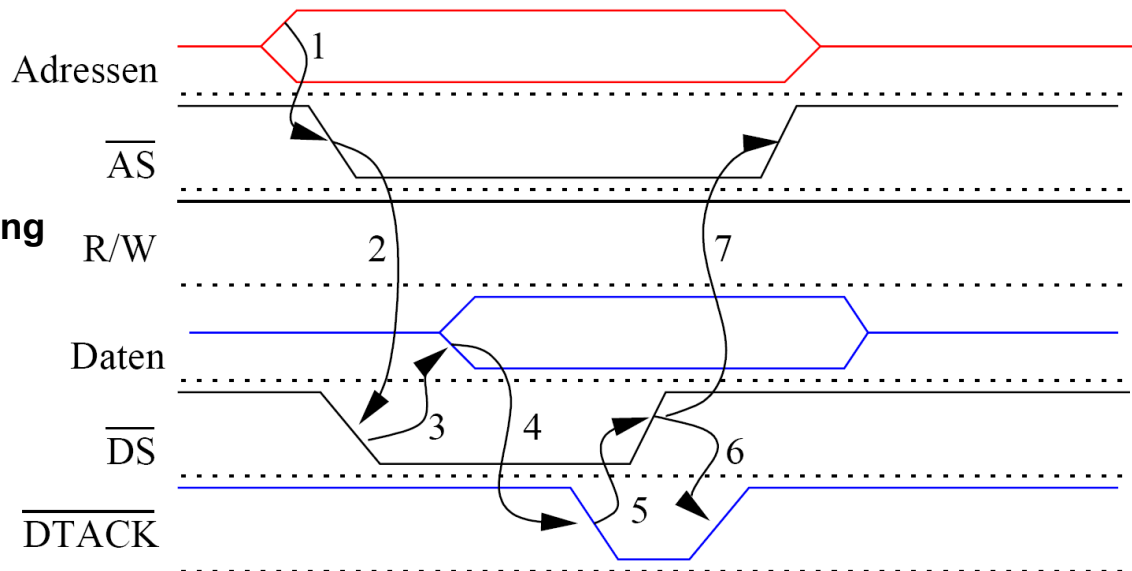
Physikalische Ankopplung, Systembus

Systembus ist ein Master/Slave Bus

Ein angeschlossenes Gerät

(meist CPU) ist der Busmaster

- kontrolliert den Adressbus
- Signal für die Schreib/Leserichtung
- Adress-Strobe-Signal
 - (Gültigkeit der Adresse)



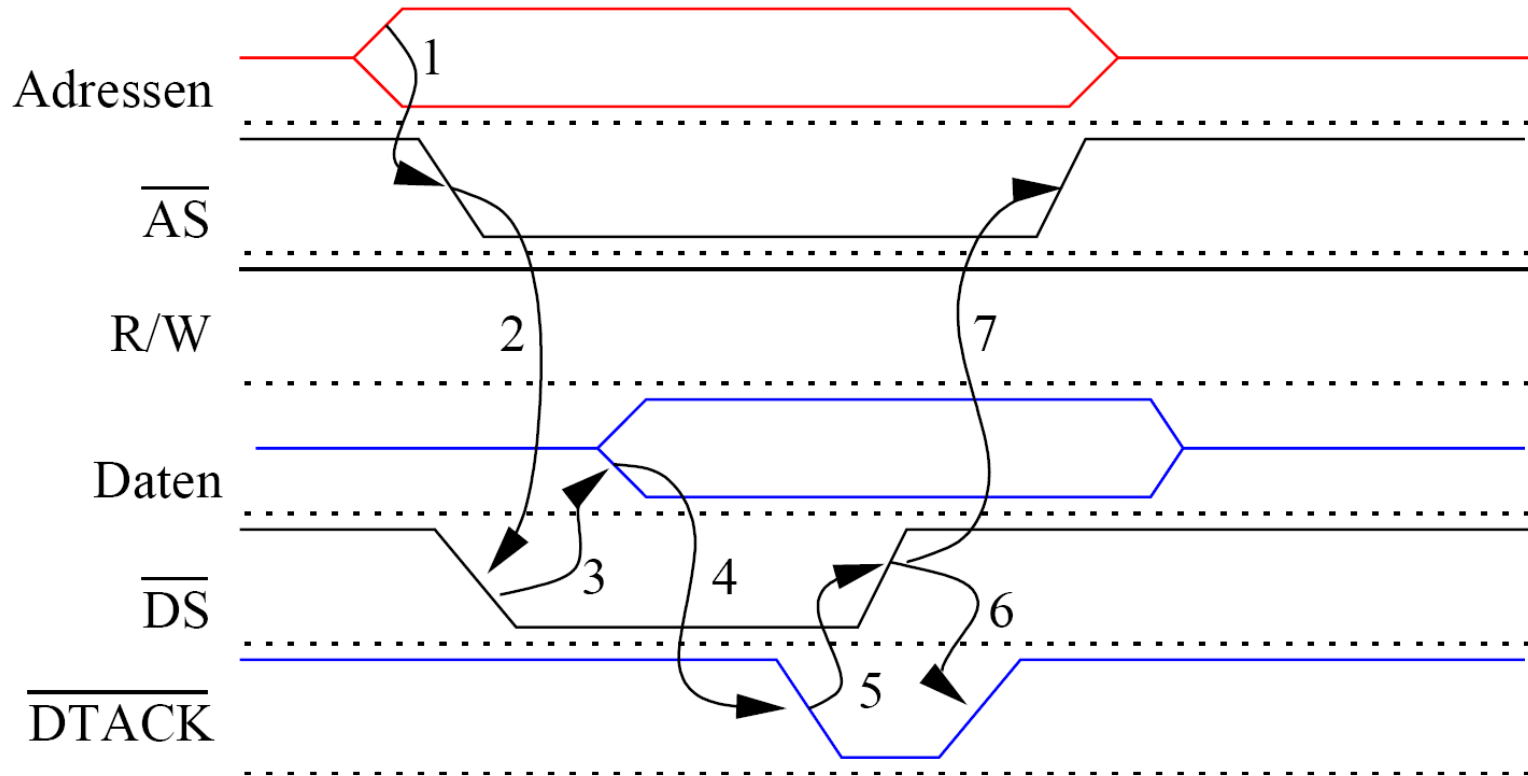
- Data-Strobe-Signal
 - Schreibzyklus: Busmaster hat gültige Daten auf den Datenbus angelegt
 - Lesezyklus: Master ist bereit, über den Datenbus Daten zu empfangen
- Slave (z.B. das Speichermodul) reagiert mit dem Data-Acknowledge-Signal
 - Schreibzyklus: wenn er die Daten übernommen hat
 - Lesezyklus: gültige Daten auf den Datenbus gelegt hat



Prozessankopplung, HW

Physikalische Ankopplung, Systembus

Buszyklus (lesend)



Prozessankopplung, HW

Physikalische Ankopplung

Buszyklus: Zeitüberwachung

Problem:

- Kein Fortschritt im Buszyklus („Busstillstand“), wenn einer Adresse kein Teilnehmer zugeordnet ist

Lösung:

- Überwachung der Zeit zwischen Setzen Adress-Strobe-Signal und Data-Acknowledge- Signal
- Bus-Error (Interrupt) bei Zeitüberschreitung



Prozessankopplung, HW

Physikalische Ankopplung

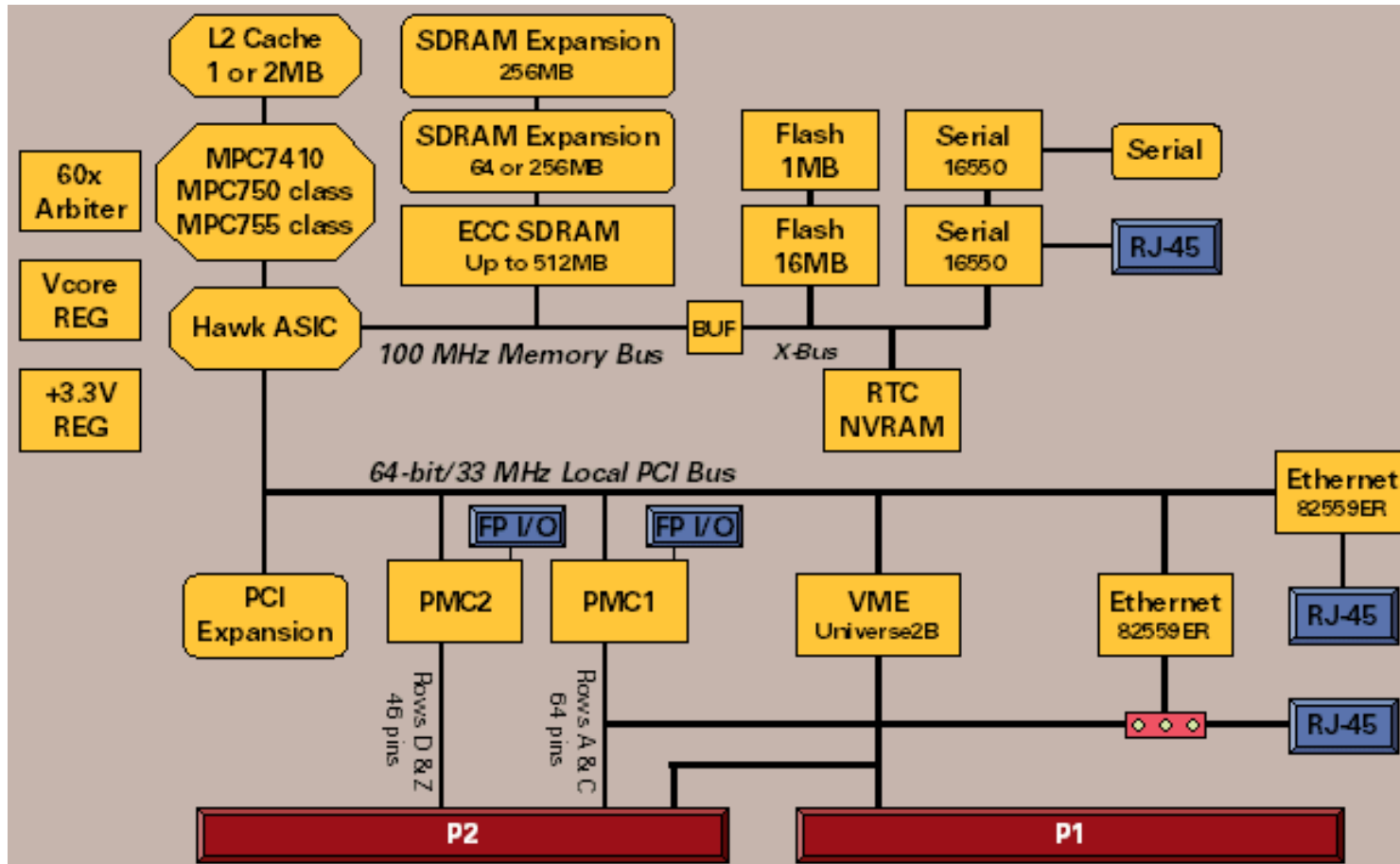
Bushierarchien

- **Proprietärer Systembus (Prozessorbus, direkt CPU-Signale)**
- **Standardisierte, verbreitete Systembusse**
 - ISA Bus (Intel-Systembus im PC)
 - VMEbus (abgeleitet von Motorola Prozessoren)
 - PCI(e)-Bus
- **Kopplung zweier Busse über Bus-“Bridge“**
 - Übersetzung zwischen Busprotokollen in Hardware
 - z.B. PC-Mainboards:
 - North-Bridge: AGP, RAM, PCIe, Verbindung zur CPU: FSB
 - South-Bridge: langsame Devices: PCI, ISA, USB,



Physikalische Ankopplung, Bushierarchien

- **Üblich: mehrere Busse (z.B. heutige VME-CPU's: CPU-Bus – PCI-Bus – VME-Bus)**



Peripherieanbindung über den Systembus

Ankopplung von I/O-Geräten über Systembus prinzipiell

- **Spezielle Ein/Ausgabebefehle (z.B. 80x86)**
 - **Eigener „Speicherbereich“ für Peripherie**
 - **Grund: Beschränkter Adressbereich (früher)**
 - **Geringerer Hardwareaufwand auf Geräteseite zur Adressdekodierung (wg. eingeschränktem Adressbereich)**
 - **Nicht mehr zeitgemäß wg. großer (>32 Bit) Adressbereiche**
- **Memory-Mapped-I/O**
 - **I/O-Module im Adressraum des Prozessors**
 - **Ansprechen der Geräte-Register mit normalen Move-, Load- oder Store-Befehlen des Prozessors**



Zusammenfassung

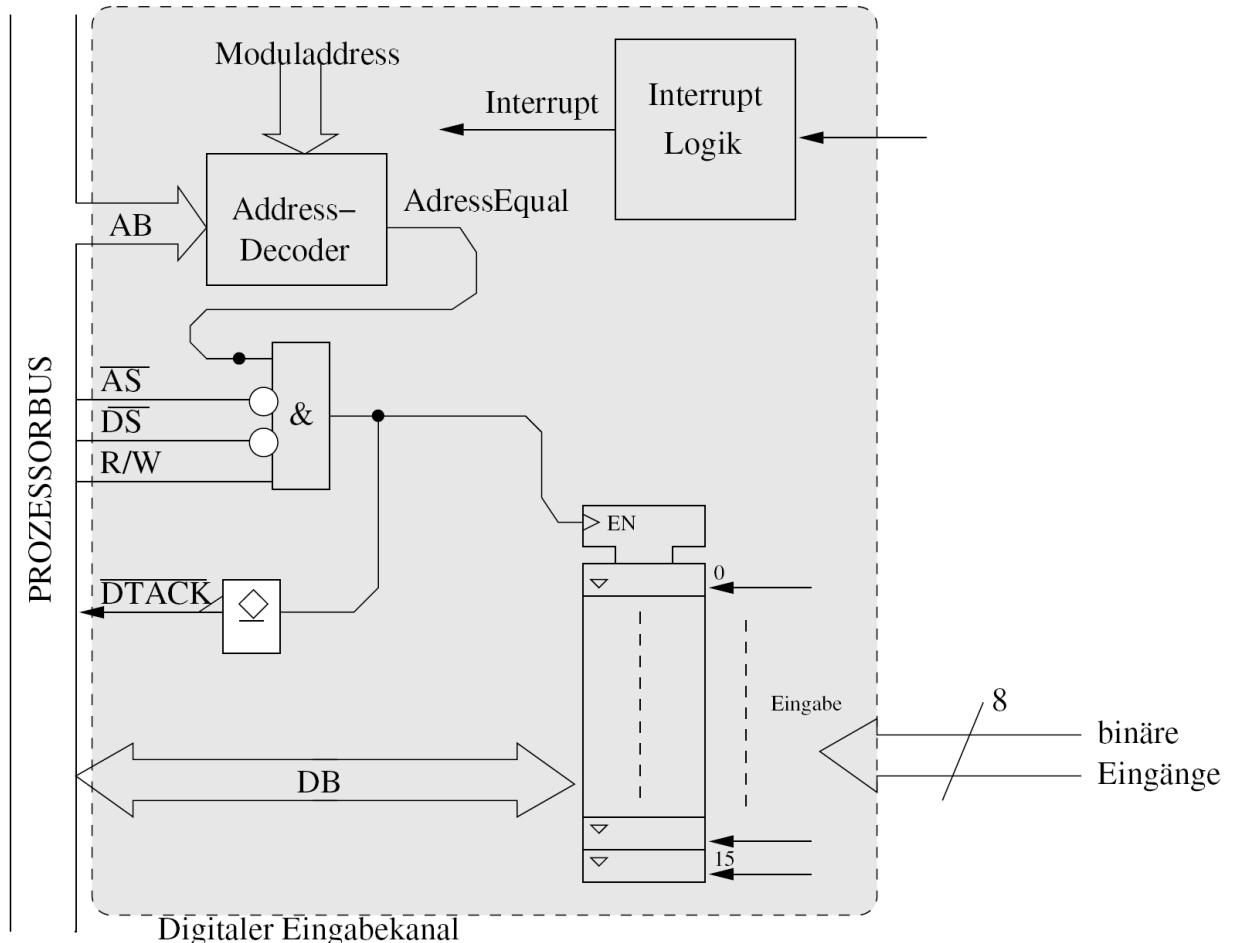
- **Peripheriekoppel-Modul (Gerät)**
- **Systembus**
- **Buszyklus**
- **Bushierarchien**



Prozessankopplung, HW

Peripherieanbindung über den Systembus

Memory-Mapped-I/O (z.B. parallele Schnittstelle)



Prozessankopplung, HW

Peripherieanbindung über den Systembus

Einteilung von Peripherie gemäß Zugriffsgeschwindigkeit

Geschwindigkeit	Zeitkonstante	Technologie
schnell	ns	direkter Speicherzugriff, DMA
mittel	μ s	programmgesteuerter Kanal
langsam	ms	Interrupt



Peripherieanbindung über den Systembus

Im Folgenden:

- Anbindung serieller Prozess-Signale
- Anbindung digitaler Prozess-Signale
- Anbindung analoger Prozess-Signale



Prozessankopplung, HW

Ankopplung serieller Prozess-Signale

Häufig: Serielle Ankopplung von Signalen eines technischen Prozesses

Einzelne Zeichen (Bytes, 8 Bit) werden seriell übertragen

- Weniger Verkabelungsaufwand als parallel
- Elektrisch robuster (z.B. differentiell; Schirmung, kein Übersprechen) als parallel
- (RS232, RS485, RS422, USB, ...)

Meist: Vorgegebene Baudrate (Bits/s, z.B. 9600, 19200, 38400, 112000), Startbit zum Synchronisieren, 8 Datenbits, ggf. Parity, 1-2 Stopbits

Module am Systembus: “serielle Schnittstelle”, UART (*universal asynchronous receiver transmitter*)

(Geräte-)Register:

- „Virtuelle Speicherzelle“ eines Geräts
- Über Adresse ansprechbar
- Einzelne Bits lesbar/schreibbar, haben jeweils spezielle Funktion/Bedeutung

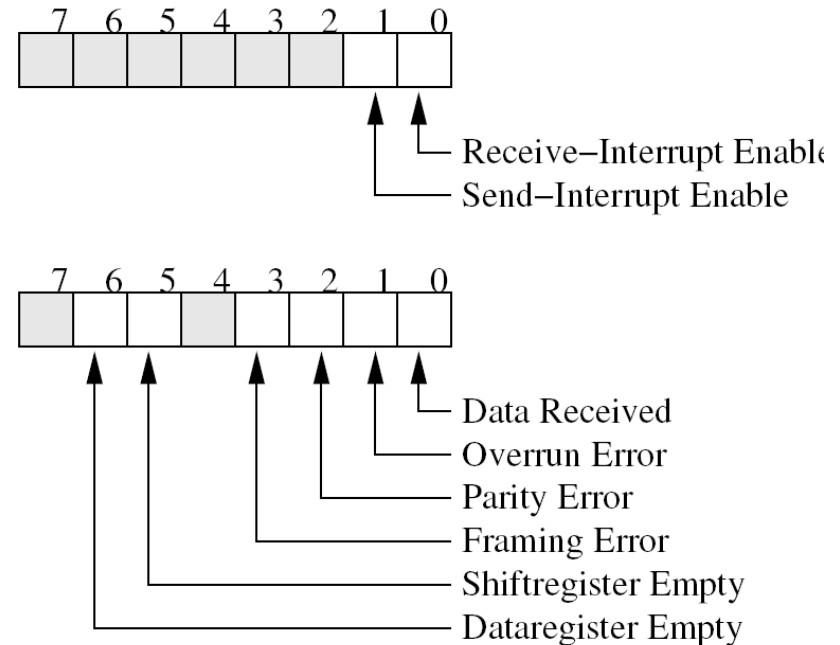


Prozessankopplung, HW

Ankopplung serieller Prozess-Signale

Typische Registerausstattung

- **Senderegister zur Parallel-Seriellwandlung**
 - Datenübertragung startet, wenn Register beschrieben
 - Vor Beschreiben prüfen, ob letzte Übertragung fertig (Statusregister)
- **Empfangsregister zur Seriell-Parallelwandlung**
 - „Aufbau“ und Auslesen empfangener Zeichen
- **Kontrollregister (Betriebsart)**
 - Sende-Interrupt (Zeichen gesendet)
 - Empfangs-Interrupt (Zeichen empfangen)
- **Statusregister (Betriebszustand)**
- **Bitratenregister (Übertragungsrate)**



Prozessankopplung, HW

Ankopplung digitaler Prozess-Signale

Parallele Ein-/Ausgabeeinheit (PIO)

- Erfassung digitaler Zustandsgrößen (z.B. Schalter, binäre Sensoren, ...)
- Ausgabe digitaler Informationen (z.B. zum Schalten von Relais)

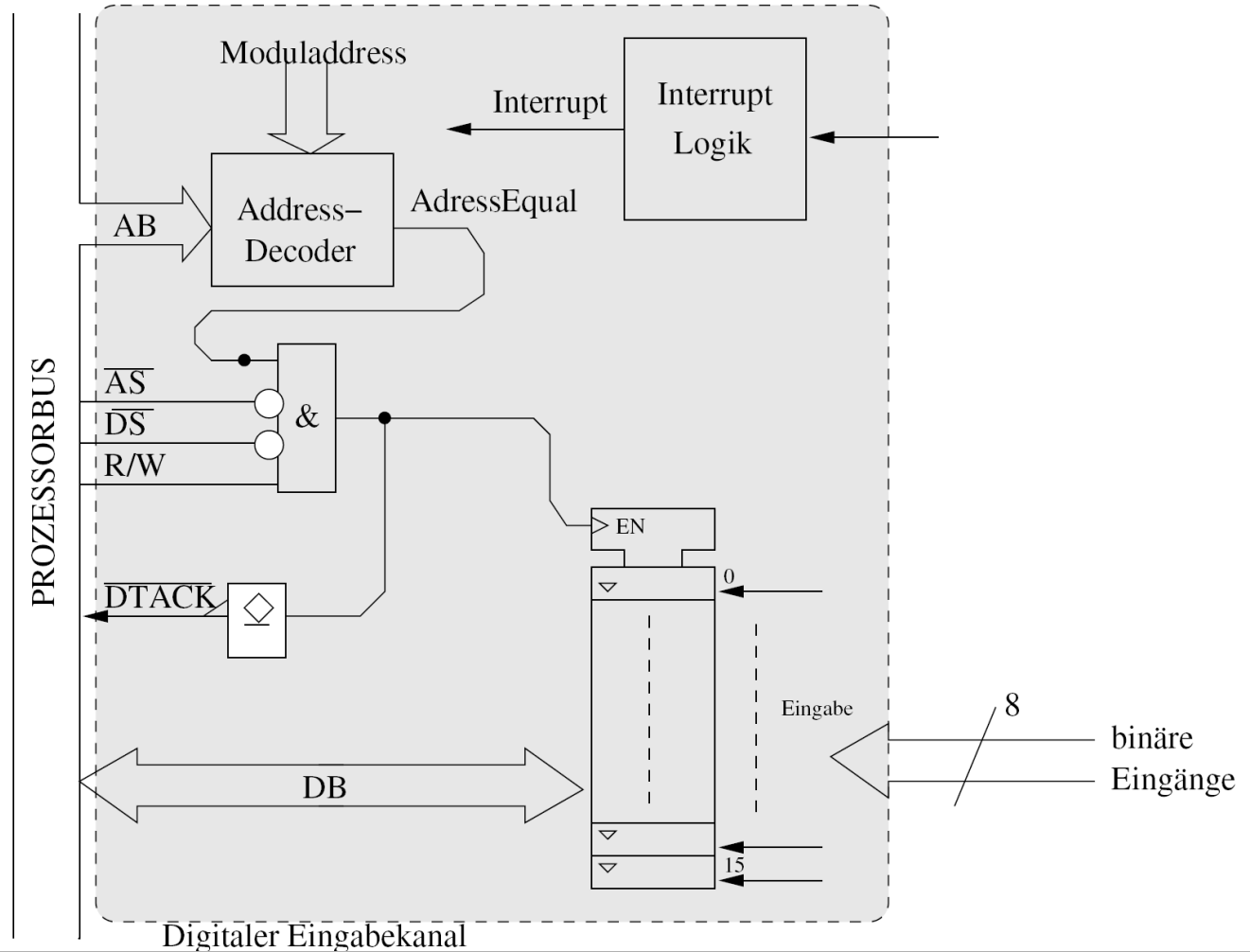
Registerausstattung

- Mehrere Schreib/Lese-Register, Richtung (meist bit- bzw. bitgruppenweise konfigurierbar)
- Control-Register konfiguriert IO-Richtung (Eingabe/Ausgabe) einzelner Bits oder Bitgruppen



Prozessankopplung, HW

Ankopplung digitaler Prozess-Signale



Prozessankopplung, HW

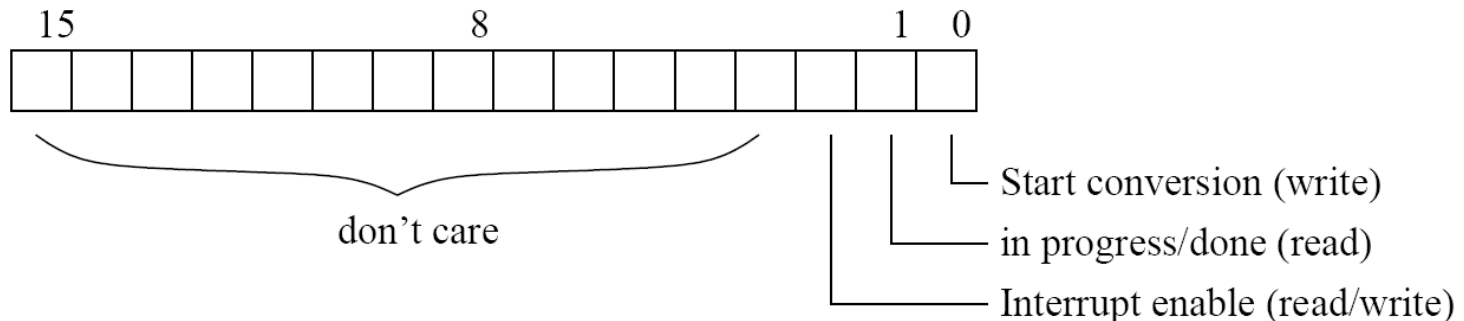
Ankopplung analoger Prozess-Signale

Ausgabe analoger Größen:

- Digital Analog Wandler (DA Wandler, Digital Analog Converter, DAC)
- Schreib-Register, dessen Wert von ADC in analogen Wert gewandelt wird

Erfassung analoger Größen:

- Analog Digital Wandler (AD Wandler, Analog Digital Converter, ADC)
- Wandlungszeiten im Nano- bis hin in den Millisekundenbereich
- Lese-Register, enthält nach Wandlung dem Analogwert
- Control-Register

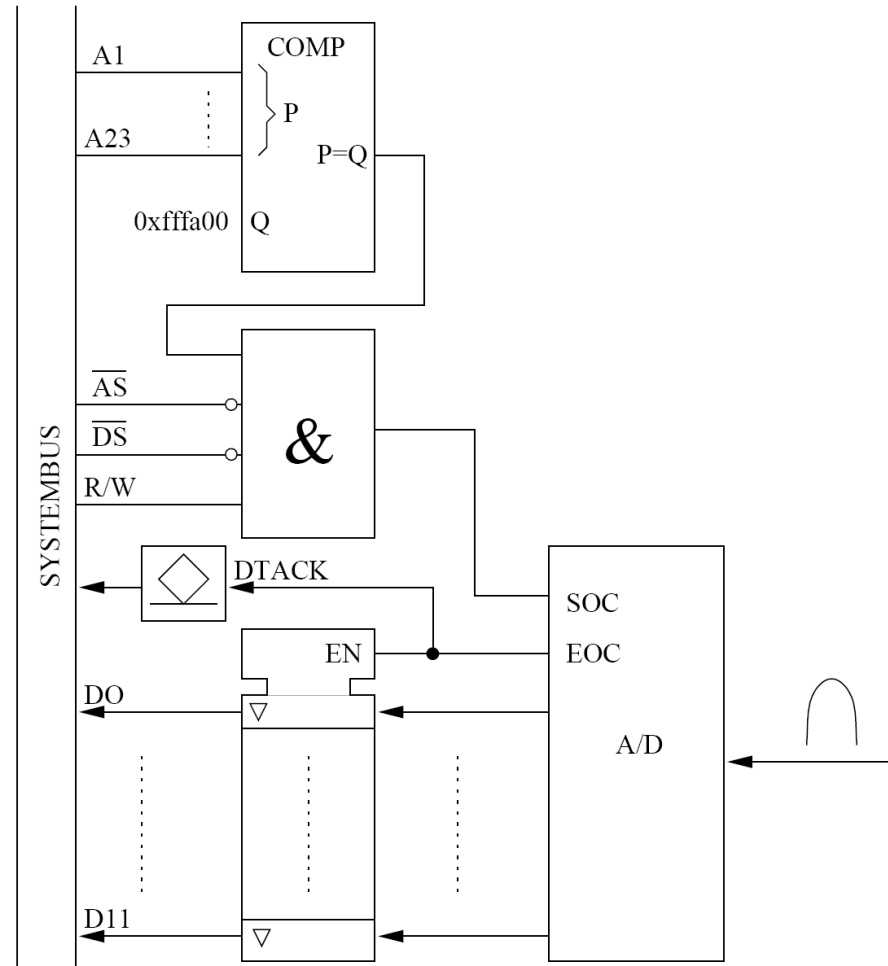


Prozessankopplung, HW

Ankopplung analoger Prozess-Signale

Ankopplung eines schnellen A/D-Wandlers

- Wandlung in
1 Buszyklus möglich



Prozessankopplung, HW

Peripherieanbindung über den Systembus

Einteilung von Peripherie gemäß Zugriffsgeschwindigkeit

Geschwindigkeit	Zeitkonstante	Technologie
schnell	ns	direkter Speicherzugriff, DMA
mittel	μ s	programmgesteuerter Kanal
langsam	ms	Interrupt



Prozessankopplung, SW

Direkter Speicherzugriff, Treiberebene

- Schnelle Peripherie wird wie ein normaler Speicherbaustein angeschlossen
- Daten können vom technischen Prozess in einem Buszyklus gelesen bzw. ausgegeben werden (z.B. Parallelschnittstelle)
- Softwaretechnischer Zugriff (im Treiber) wie auf eine sonstige Speicherzelle im Adreßraum des Prozessors

```
int DigitalWert;  
volatile int *Wandler; // Der Wandler liefert 12 Bit zurueck,  
...  
Wandler=0xFFFA00;  
...  
DigitalWert=*Wandler;  
DigitalWert=DigitalWert&0x0FFF; // nur die unteren 12 Bit sind  
    gueltig  
...
```



Prozessankopplung, SW

Programmgesteuerter Kanal, Treiberebene

- Zugriff auf mittelschnelle Geräte (mehrere 10 Nano - Mikrosekundenbereich)
- Voraussetzungen
 - Direkter Speicherzugriff würde zu langen Buszyklen führen
 - Interruptbehandlung zu ineffizient (zu viele Befehle)
- Programmgesteuerter Kanal:
- mehrere Zugriffe zur Bedienung eines Gerätes notwendig
- z.B. ADC:
 - Wandlungsprozess starten
 - zu einem späteren Zeitpunkt abfragen, ob Wandlungsende erreicht
 - Falls ja, Daten einlesen



Prozessankopplung, SW

Programmgesteuerter Kanal, Treiberebene

```
volatile int *ControlWort, *StatusWort, *DatenWort;  
int DigitalWert;  
...  
// Initialisierung  
ControlWort=0xFFA00;  
StatusWort =0xFFA02;  
DatenWort =0xFFA04;  
...  
// Starten einer Wandlung  
*ControlWort=0x0001; // Setzen von Bit 0 startet die Wandlung  
// Warten auf das Wandlungsende  
while( *StatusWort!=0x0002 ) // EOC ist gleich Bit 1  
;  
// Auslesen der Daten  
DigitalWert = *DatenWort;  
...
```



Prozessankopplung, SW

Programmgesteuerter Kanal, Treiberebene

```
volatile int *ControlWort, *StatusWort, *DatenWort;  
int DigitalWert;  
...
```

// Schlüsselwort **volatile** im C-Code: Inhalt kann sich ändern (ist flüchtig)

Ansonsten: Compiler optimiert den Schleifenzugriff
while(*StatusWort != 0x0002) :

// Inhalt des Statuswortes einmal in ein Prozessorregister kopieren,
// immer wieder auf Ungleichheit mit 0x0002 überprüfen (oder gleich
// wegoptimieren)

while(*StatusWort != 0x0002) { ... } // ist gleich Bit 1

```
;  
// Auslesen der Daten  
DigitalWert = *DatenWort;  
...
```



Prozessankopplung, SW

Interruptgesteuerter Kanal

- Zugriff auf langsame Peripherie (Mikro-Millisekunden)
- Ähnlich programmgesteuerter Kanal (mehrere Register)
- Zugriffszeiten zu lange für aktives Warten (Busy-Waiting)
- Ein-/Ausgabemodul informiert die Steuerung per Interrupt über Zustandsänderungen, ersetzt zyklisches Abfragen des Zustands (*polling*)
- Interruptgesteuerter Kanal: HW-Interface ist um Interruptlogik erweitert



Prozessankopplung, SW

Direct Memory Access (DMA)

Beispiel: Transfer von Datenblöcken vom Speicher zur Peripherie (z.B. Platte, Ethernet) durch CPU

- Zugriff über den Systembus $t_B=100\text{ns}$
- Zugriffszeit, die der Speicher benötigt (vernachlässigbar) $t_M=0$
- Zugriffszeit, die die Peripherie benötigt (vernachlässigbar) $t_{IO}=0$
- Programmlaufzeit $t_V=1\mu\text{s}$

Transferrate:

$$r = 1 / (t_V + 2 t_B + t_M + t_{IO}) = 0.833 \text{MWorte/s}$$

Wortbreite 32 Bit: maximale Transferrate ca. 3 MByte/s.

Festplatten: 40-50MByte/s.

Höhere Transferraten durch Minimierung von t_V bzw. t_B

- t_B (133MHz): 7.5 ns
- t_V : Direktübertragung von Peripherie zu Speicher ohne CPU: DMA



Prozessankopplung, SW

Direct Memory Access (DMA)

- Verfahren zum Austausch/Transfer von zusammenhängenden Datenbereichen (Blöcken) zwischen Speicher (RAM) und Peripherie (z.B. Festplatte).
- Hardwarerealisierung des Datentransports unter Programmkontrolle (programmgesteuerter Kanal)

„Intelligente“ Hardware, für den Datentransport zuständig: DMA-Controller

CPU gibt die Kontrolle über den Bus an DMA-Controller ab

Prinzip: Bus-Arbitrierungssignale als Teil des Control-Bus

- Slave (DMA-Controller), der Busmaster werden möchte, teilt dies der CPU über ein Signal mit (z.B. Bus Request oder Hold)
- CPU signalisiert, dass die Kontrolle über den Bus übernommen werden kann (Bus Grant)
- Slave übernimmt den Bus und ist damit neuer Busmaster (Bus Grant Acknowledge)
- Nach Beendigung der Übertragung: DMA-Controller gibt den Bus wieder an die CPU zurück (Zurücknahme der Signale Bus Grant Acknowledge bzw. Hold).



Prozessankopplung, SW

Direct Memory Access (DMA)

DMA dient zur

- Erhöhung der Übertragungsrate
- Entlastung der CPU

Drei Phasen des DMA-Betriebs

1. Initialisierungsphase/Aufsetzen des Kanals

- Die CPU ist Bus-Master und beschreibt die DMA-Register
- DMA-Controller ist Slave

2. DMA-Betrieb (zyklisch bis Übertragungsende, “*cycle stealing*”-Phase)

- Der DMA-Controller wird Busmaster
- Überträgt ein Datum zwischen Peripherie und Speicher
- DMA-Controller gibt den Bus wieder frei, CPU wird Busmaster

3. Ende des DMA-Betriebs

- Der DMA-Controller informiert die CPU über das DMA-Ende per Interrupt



Prozessankopplung, SW

Direct Memory Access (DMA)

Modi:

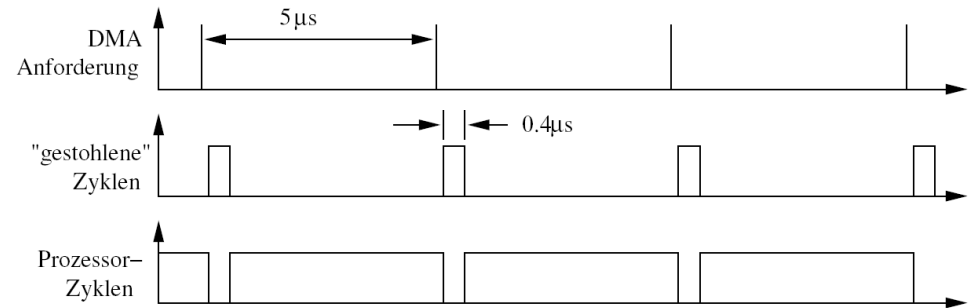
1. Normalbetrieb

- Der CPU wird pro zu übertragendem Datum ein Speicherzyklus gestohlen

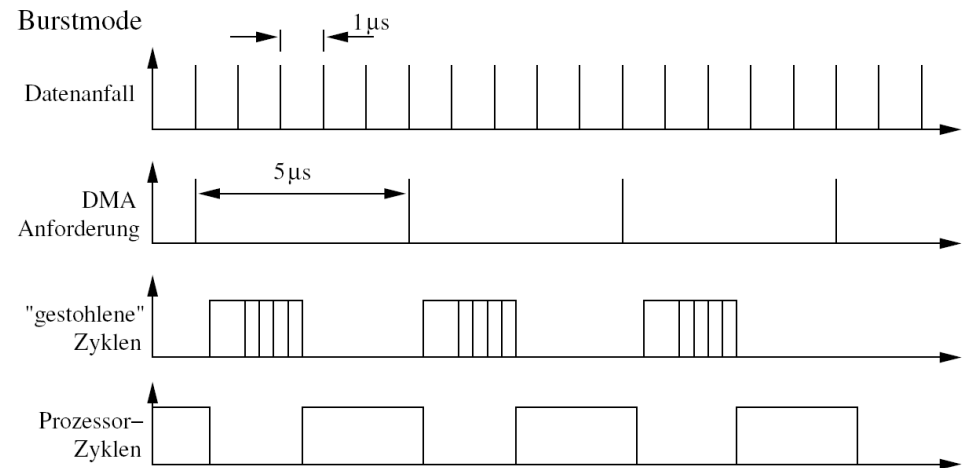
2. Burstbetrieb (*burst mode*)

- Zugriff auf aufeinander folgende Speicherzellen bei dynamischen Speicherbausteinen (DRAMs) erfolgt schneller
- DMA für einen ganzen Block aufeinanderfolgender Daten

Normalbetrieb



Burstmode

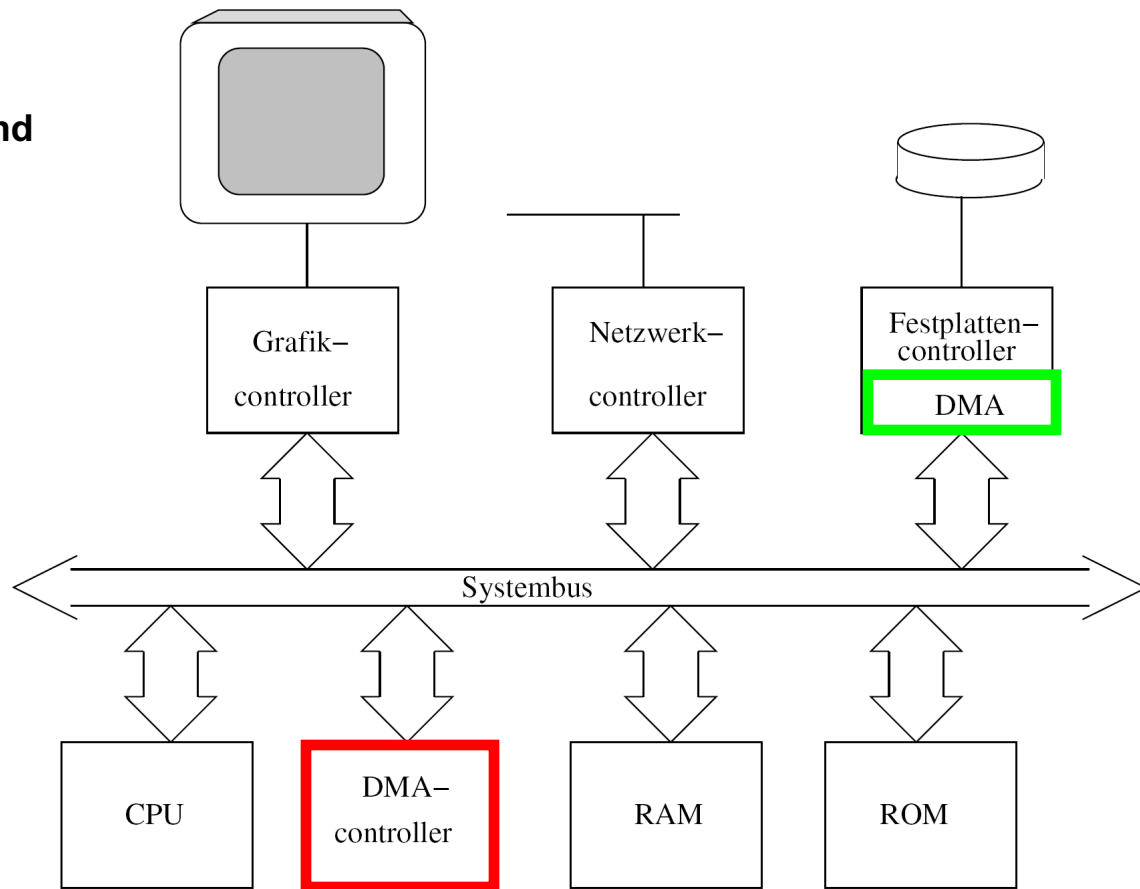


Prozessankopplung, SW

Direct Memory Access (DMA)

DMA-Controller

- **Eigener Bus-Teilnehmer**
 - Geringer Hardwareaufwand
 - Doppelte Anzahl an Buszugriffen
 - Treiber-Abhängigkeit
 - Gerät
 - DMA-Controller
- **Teil eines Geräts**



Prozessankopplung, SW

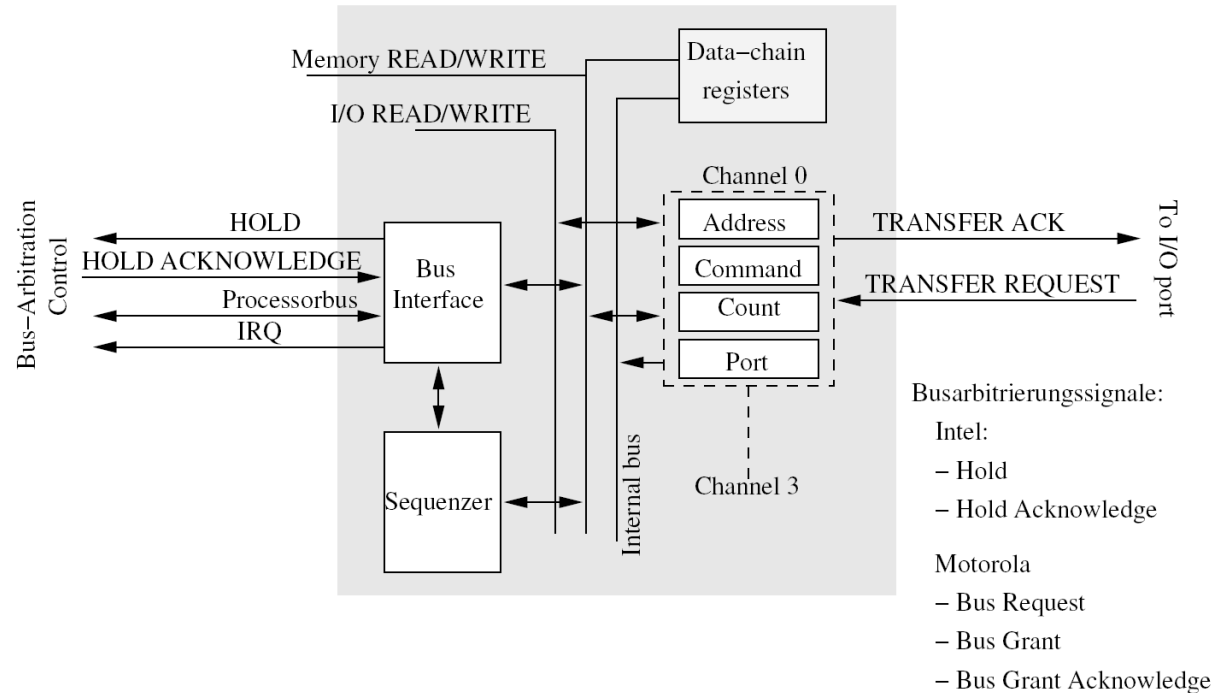
Direct Memory Access (DMA)

DMA-Controller

Mehrere DMA-Kanäle (typischerweise 4)

Pro Kanal 4 (wesentliche) Register:

- Hauptspeicheradressregister
- I/O Port (Geräte-Adresse)
- Byte-Count-Register
- Commandregister
- Übertragungsrichtung
- Start
- ...



Prozessankopplung, SW

Softwaretechnische Ankopplung

Einfache Peripherie

Zusammenfassung:

Anbindung auf Treiberebene über Registersätze („memory mapped“) und eine der 4 Arten

- Direkter Speicherzugriff
- Programmgesteuerter Kanal
- Interruptgesteuerter Kanal
- DMA

Typisch: Kontroll-, Status-, Datenregister



Prozessankopplung, SW

Softwaretechnische Ankopplung

Intelligente Peripherie

Intelligenter Sensor oder Aktor: Peripheriegerät, das mit eigener Rechenleistung ausgestattet ist und eine Vorverarbeitung des Mess- oder Stellwertes durchführt („Embedded System“)

Vorverarbeitung:

- Skalierung
- Messbereichsumschaltung
- Konvertierung
- Filterung
- ...

Parametrierung

Kommandierung (Auswahl von Funktionen, Angabe von Parametern,...)

Programmdownload („Flashen“)



Prozessankopplung, SW

Softwaretechnische Ankopplung

Intelligente Peripherie

Datenfluss

Satz von Registern (Speicherbereich): Service Description Block (SDB)

Typisch: drei Register (-bereiche)

- **Kommandoregister (in Analogie zum Funktionsnamen, die Funktion ist durch Software auf dem Peripherieboard bestimmt)**
- **Argumentenregister (der bzw. die Parameter)**
- **Status- bzw. Ergebnisregister (Ergebnis des Funktionsaufrufes)**

Kontrollfluss

- **Flags(z. B. Statusregister), über die der Datenaustausch synchronisiert wird**
- **Interrupts**



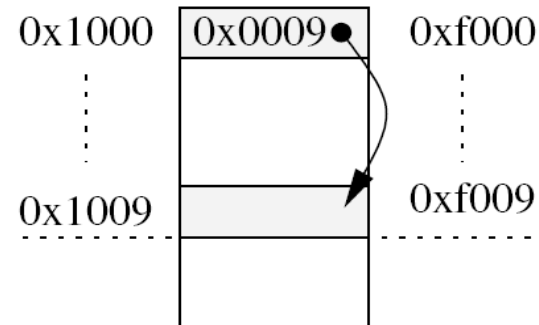
Prozessankopplung, SW

Softwaretechnische Ankopplung

Intelligente Peripherie

Dual Ported Ram, DP-RAM:

- Gemeinsamer Speicherbereich zum Austausch größerer Datenmengen zwischen Peripherie und Steuerrechner (Hostsystem)
- Speicher mit zwei (bis auf exklusiven Zugriff) unabhängigen Busanbindungen
- Strukturierung des gemeinsamen Speichers z.B. durch Zeiger aus Relativ-Adressen
- Strenge Synchronisation des Zugriffs auf das DP-Ram auf Protokollebene:
z.B. pollender Zugriff durch schnelle CPU kann Zugriff durch die andere Seite blockieren



Mem-Basis aus Sicht des Prozessors: 0x1000

Mem-Basis aus Sicht des Moduls: 0xf000

Adresse = Basis + Offset

Prozessor $0x1000 + 0x0009 = 0x1009$

Modul $0xf000 + 0x0009 = 0xf009$

