

■ 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

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



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 ↔ Peripherie

Speicher ↔ Speicher

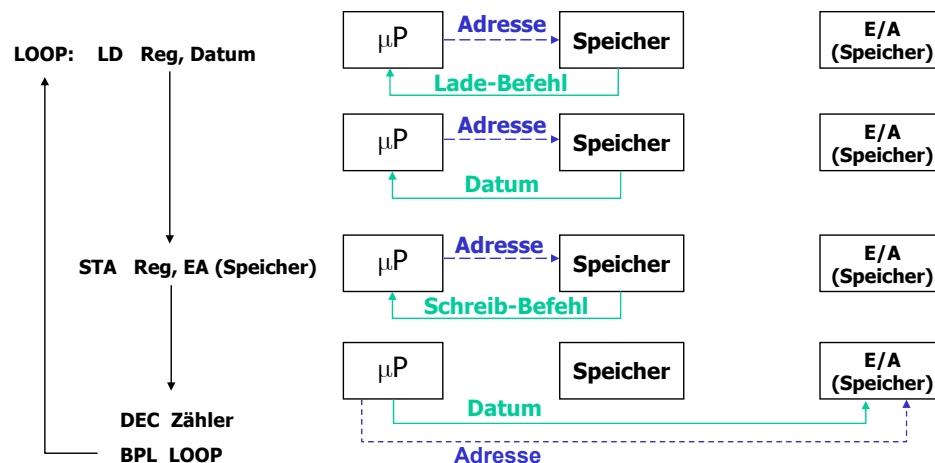
Peripherie ↔ Peripherie

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



Beispiel

Datenübetragung zwischen Speicher und Peripherie mittels Prozessor



Beispiel

Nachteil: mindestens 4 Speicherzugriffe / Datum nötig (ggf. sogar mind. 6 Speicherzugriffe durch Schleifenbefehle)

→ langsamer Datentransfer selbst bei kleinen Speicherzugriffszeiten

→ Prozessor belastet

Abhilfe:

Direkter Speicherzugriff (*direct memory access*, DMA)

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



Direkter Speicherzugriff

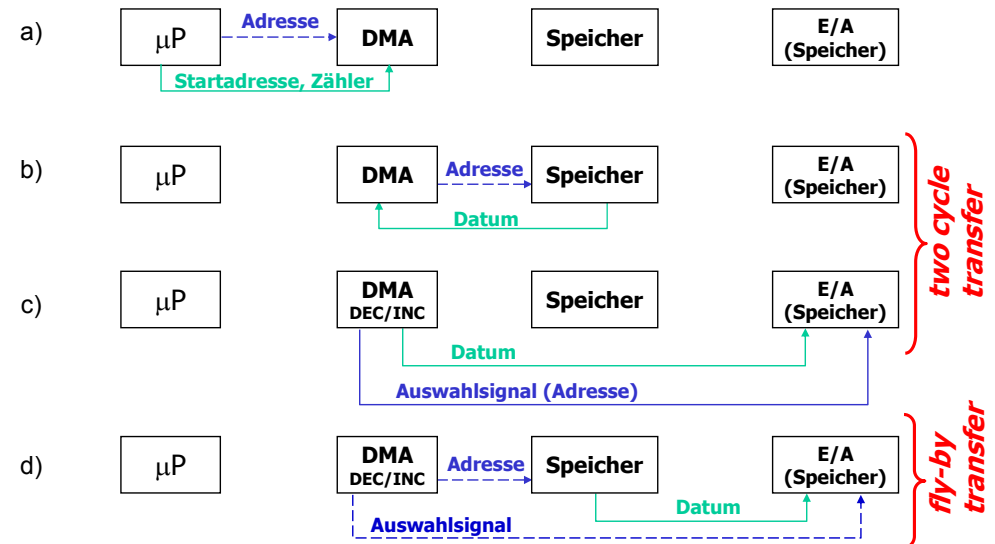
Ein spezieller Baustein, **DMA-Controller** genannt, koordiniert den Datentransfer

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

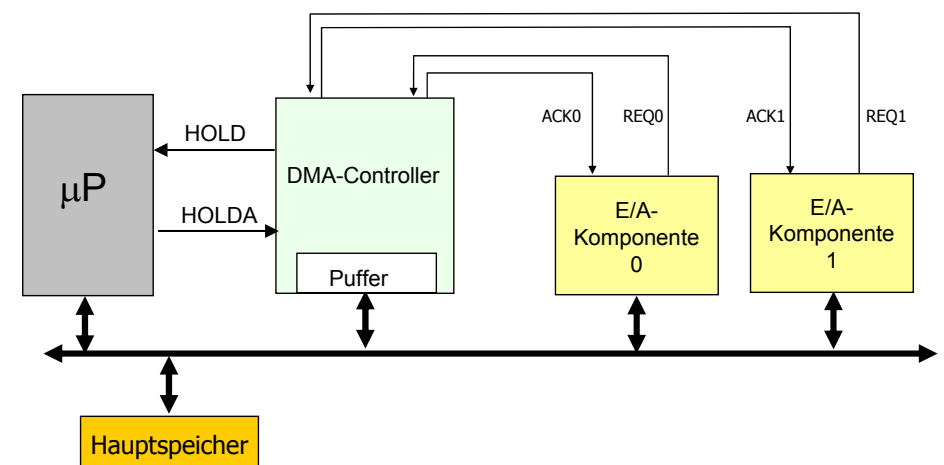
Mikroprozessor überträgt zu Beginn die Startadresse, Zieladresse, Anzahl der zu übertragenden Bytes, Übertragungsrichtung und Steuerinformationen

Danach läuft der Transfer automatisch:

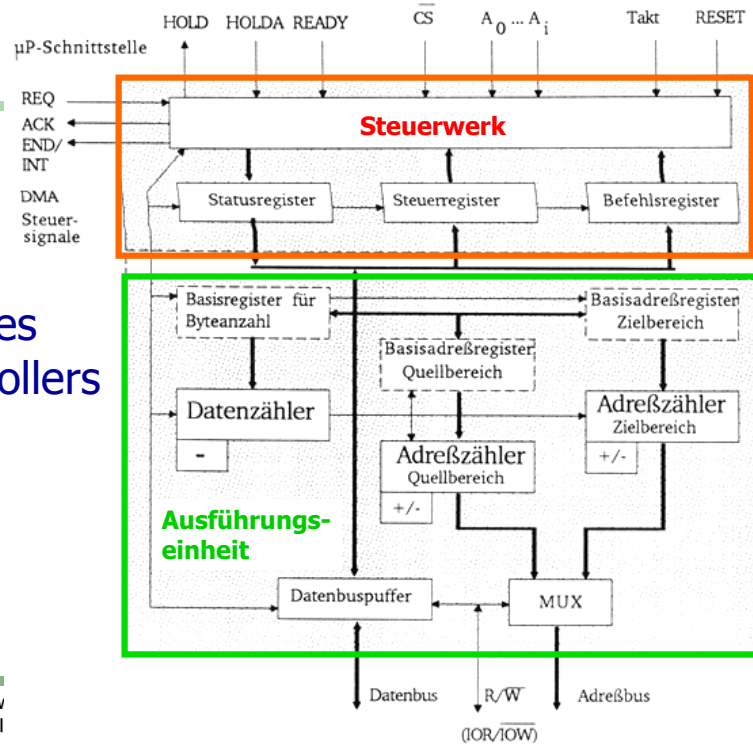
- "two cycle transfer"- Übertragungsverfahren:**
Datum wird in einem Register des DMA-Controllers zwischengespeichert (Fälle: a-c)
- "fly by transfer"- Übertragungsverfahren:**
Datum wird ohne Zwischenspeicherung direkt übertragen (Fall d). Geht jedoch nicht für Speicher-Speicher-Transfer



DMA-Controllers in μP-Systemen



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
- $A_0 \dots A_i$: 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 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

DMA-Übertragungsarten

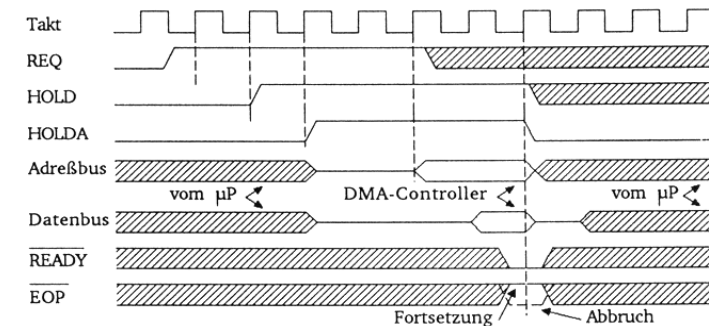
In der Regel kann man bei einem DMA-Baustein unter verschiedenen Übertragungsarten wählen, die sich durch die Belegung des Systembusses unterscheiden:

- 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)

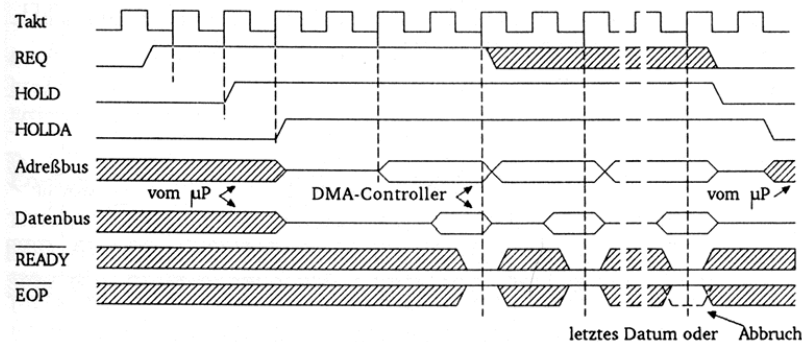


REQ: Anforderung eines Requesters zum Datentransfer
HOLD: Busanforderung durch den DMA-Controller
HOLDA: Busgewährung durch den μP
READY: beendet die Übertragung des Datums, wenn 0



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

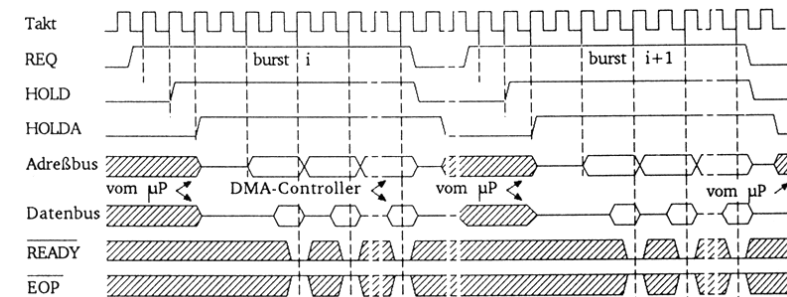
Der DMA-Controller überträgt hierbei alle Daten des Blocks ohne zwischenzeitliche Busfreigabe, sobald er den Systembus erhalten hat



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 :

