UNIVERSITÄT MANNHEIM

Interrupts

technische informatik

Seminar Rechnerarchitektur

Christian Stöffler
Universität Mannheim
WS 2005/06

Gliederung

1. Übersicht, Beispiel und Motivation

2. Ablauf eines Interrupts
   - 2.1. Gerät – PIC: Prioritäten
   - 2.2. PIC – CPU: Maskierung
   - 2.3. CPU – Gerät: IRQ-Vektoren
   - 2.4. CPU – OS: Top Half, Bottom Half

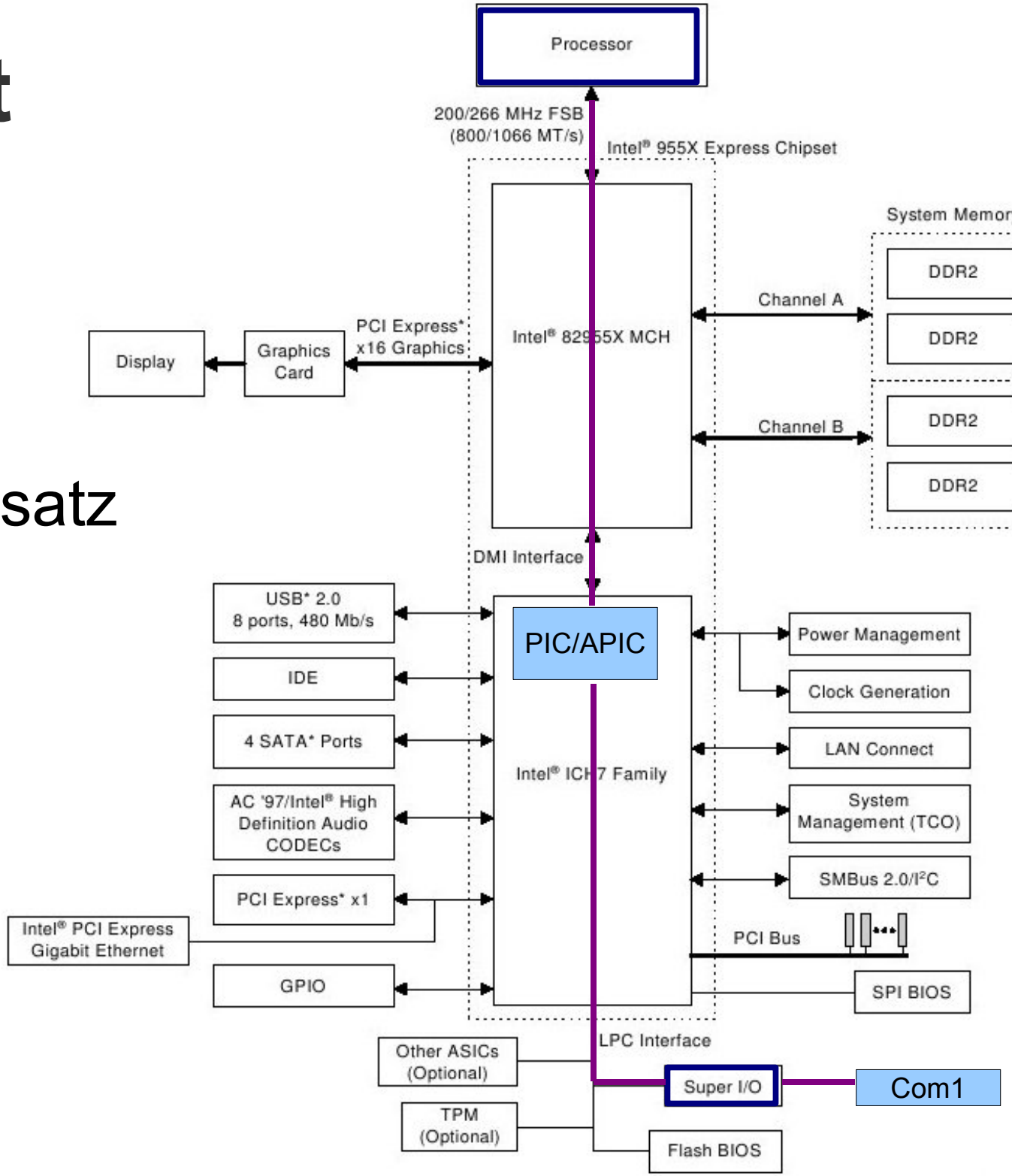
3. Intel 8259 PIC & APIC
   - 3.1. modes of Operation

4. Interrupts über PCI / PCI-X
   - 4.1. Interrupt-Pins
   - 4.2. Message Signaled Interrupts (MSI)

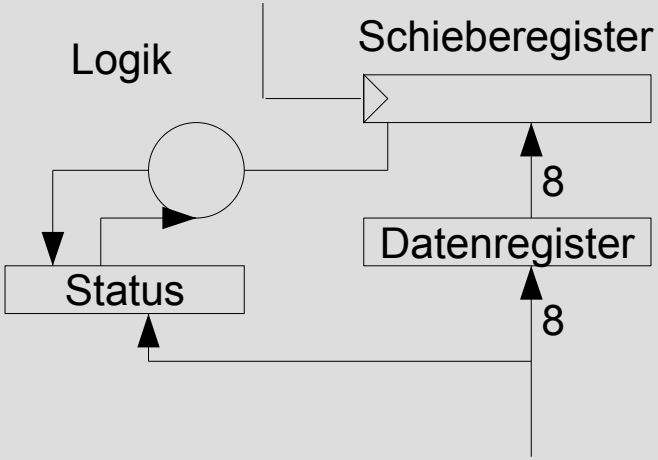
5. Interruptpakete über HyperTransport

Übersicht

intel 955X
Express Chipsatz

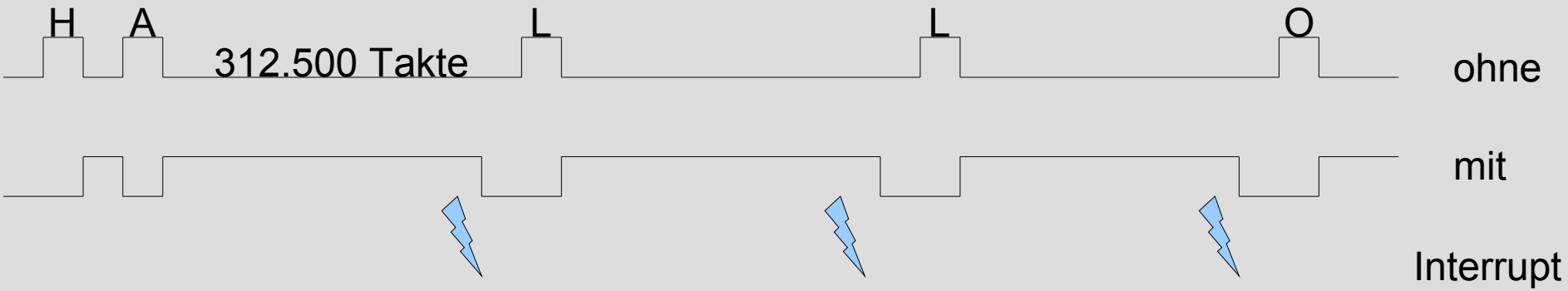


Beispiel: Serielle Schnittstelle



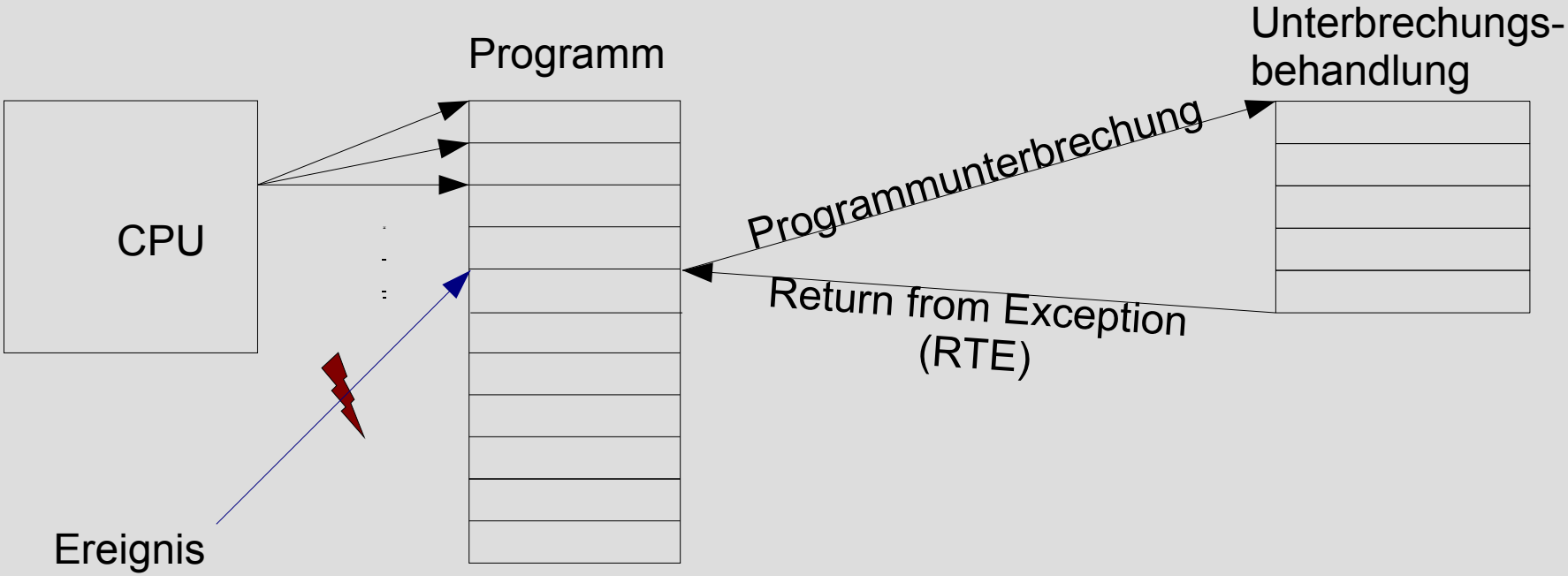
1 bit

- Übretragung von Characters im PIO-Mode über COM1
- 8 bit Character + Start-/Stop Bit + Parity Bit = 11 bit
- Übertragungsrate 9600 bit/s
- CPU: 3GHz, könnte 3.000.000.000 bit/s übertragen!
  → 312.500 Takte Leerlauf!



serielle Datenübertragung

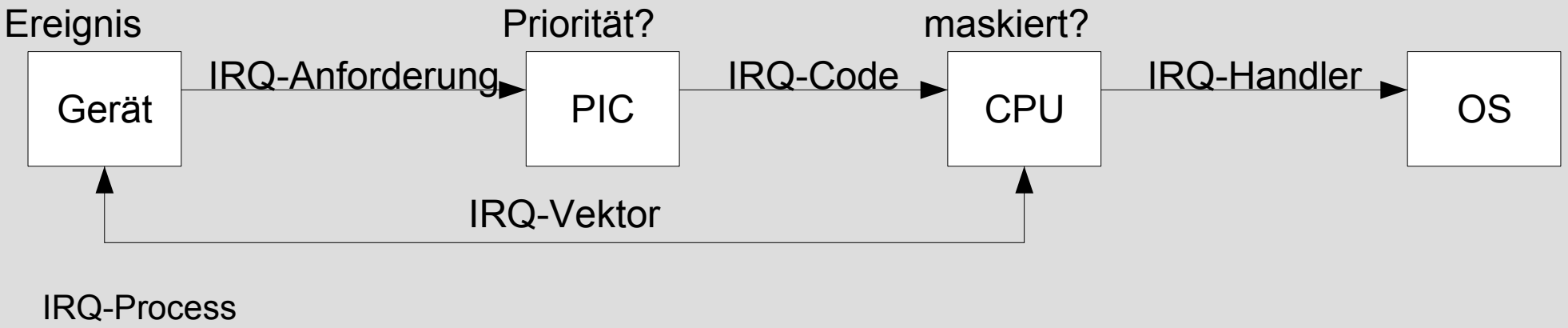
Motivation



Beispiele: Keyboard, Timer-IRQ, DMA

- **Interrupt (IRQ)** = Unterbrechung *asynchron* zur Prozessverarbeitung
- **≠Trap** = *synchrone* Unterbrechung, aus Programm heraus
- muss für laufendes Programm **transparent** sein!

Interrupt-Ablauf



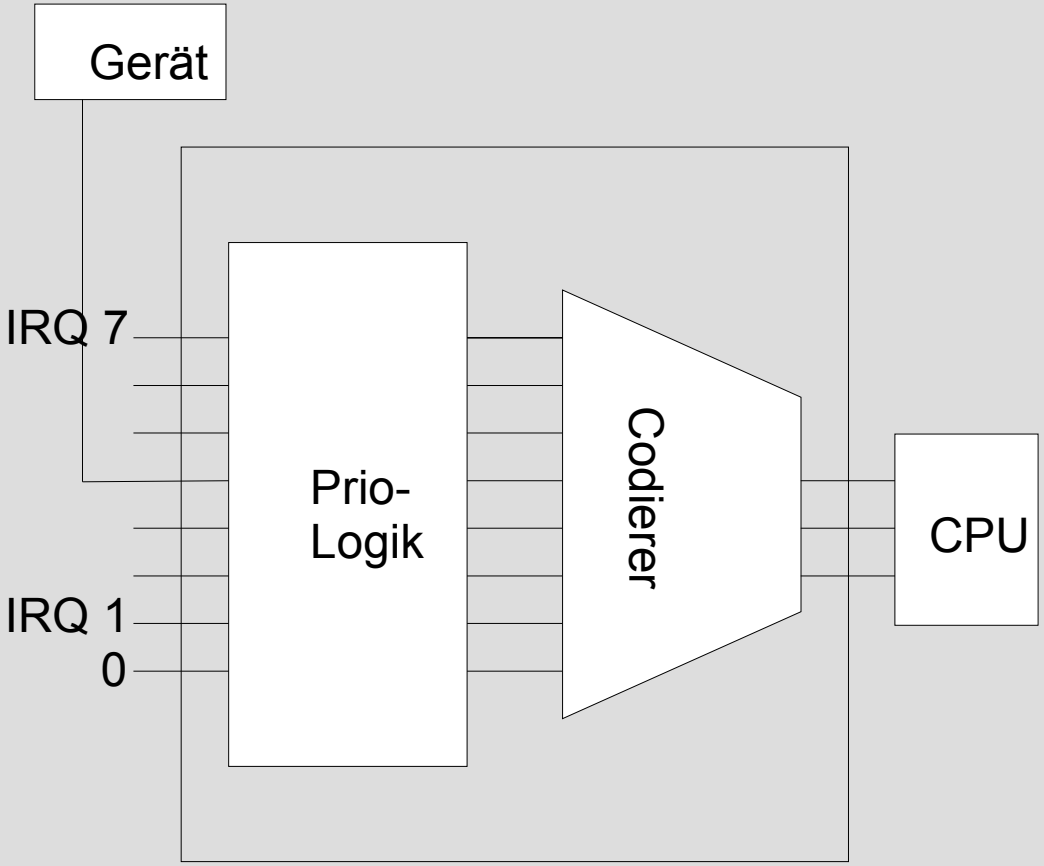
- IRQ-Leitung auf 0 setzen
- ggf. IRQ-Vektor ins Vektorregister schreiben

- Priorität bei gleichzeitigen IRQ
- IRQ-Code generieren

- nicht maskierte IRQ akzeptieren
- IRQ-Maske setzen
- Interrupt Acknowledge (IACK) -Zyklus →IRQ-Vektor

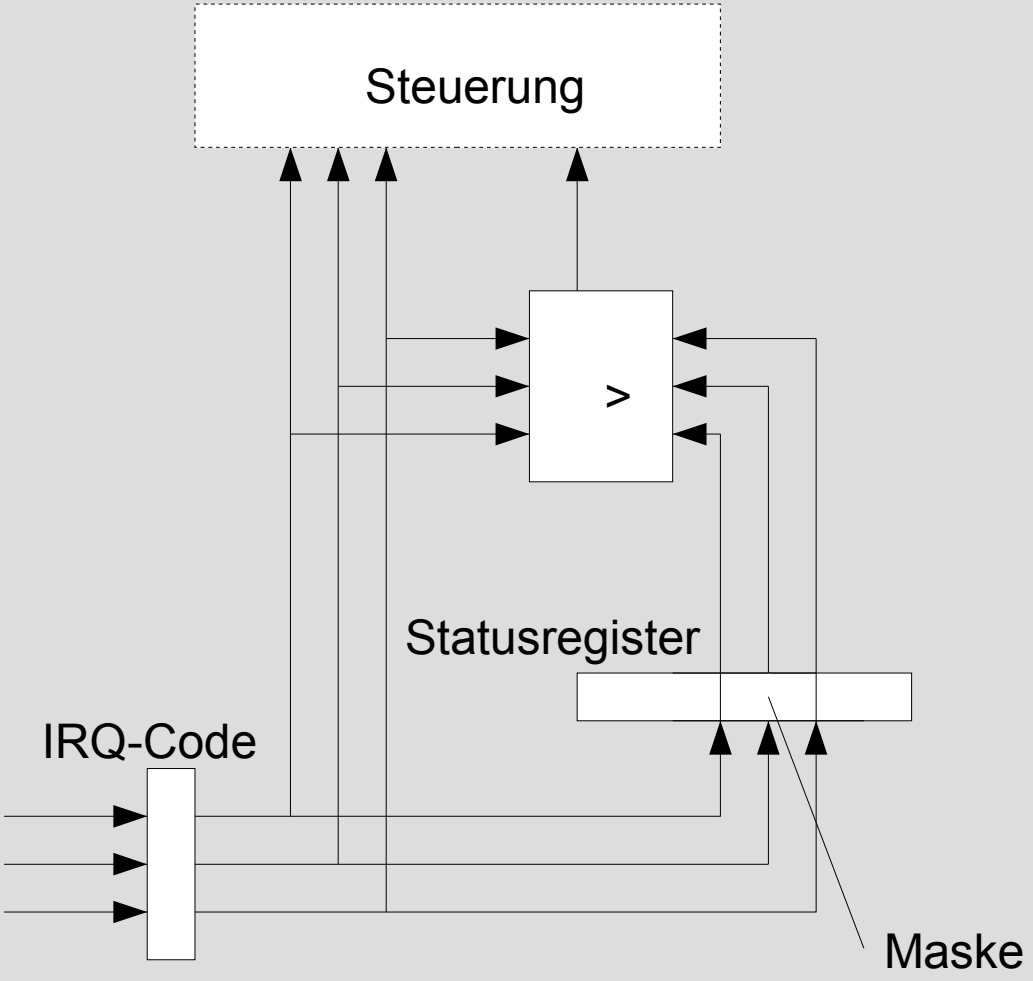
- CPU-Status retten
- IRQ zurücksetzen
- verarbeiten
- zurückspringen

Prioritäten



- Logik entscheidet über Priorität bei mehreren Signalen
- IRQ1 – IRQ7 active low
- 3-bit Code an CPU
- kein Interrupt = alles auf 1

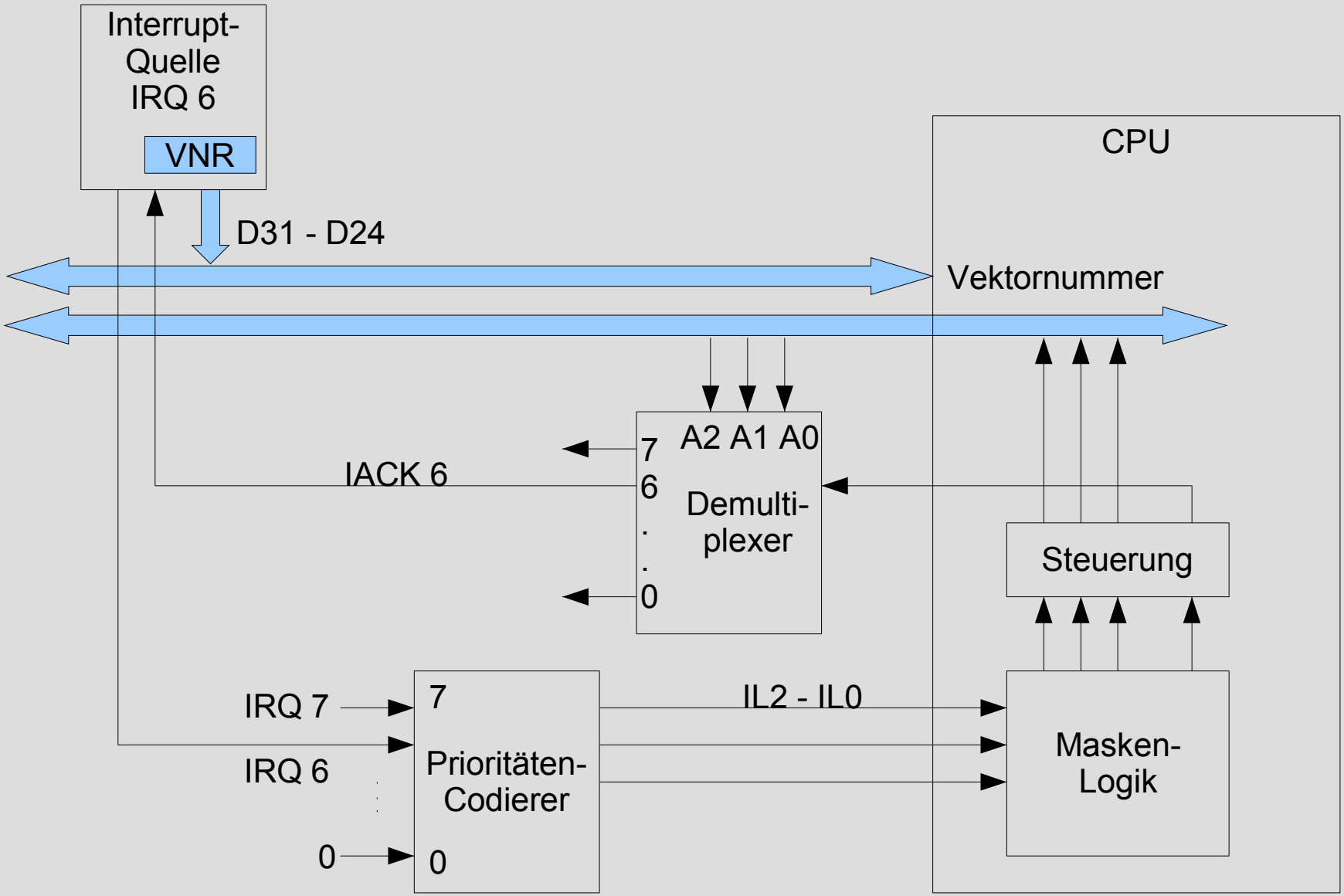
Ablauf in der CPU – IRQ Maske



IRQ-Mask Compare

- Statusregisterinhalt in Pufferregister retten
- Statuswechsel in Supervisor Mode
- IRQ-Maske setzen
  - verhindern von „geschachtelten Interrupts“
  - NMI kann nicht maskiert werden
- IACK-Zyklus
  - Interrupt-Vektor holen
  - alternativ: Autovektor-IRQ

IACK-Zyklus



IACK-Zyklus

Vektornummer → Betriebssystem (OS)

- Einsprungadresse berechnen
  - adr = Vektornummer*4 + Vektorbaseregister
- Prozessorstatus retten (auf Supervisor Stack)
  - Statusregister, insb. Condition Codes, Statusbit, etc.
- IRQ-Programms auf CPU laden
  - PC = adr
- OS: IRQ-Handler ausführen
  - Top Half aufrufen, Bottom Half dann später
- Rückkehr bei Return from Exception (RTE)

IRQ-Handler

- Top Half
  - nimmt Anforderung entgegen
  - markiert BH zur späteren Ausführung
  - setzt IRQ-Request zurück
  - RTE, insb. alten Status wieder laden
- Bottom Half
  - implementiert durch BH oder Tasklet
  - enthält den eigentlichen Handler
  - wird erst später ausgeführt
  - alle IRQ an -> Top Half kann weitere bedienen
  - führt langwierige Operationen durch, z.b. E/A
- Vorteil: System bleibt reaktionsfähiger

Bottom Halfs aufrufen

- ältere Variante
- alle BHs vordefiniert -> nur 32 Stück
- interessante BH mit zugeh. Taskqueues:
  - IMMEDIATE_BH: führt tq_immediate aus, so schnell wie möglich (Scheduler oder bei Systemcall-Rücksprung)
  - TQUEUE_BH: tq_timer nach allen Timer-Ticks
- Bsp.: `linux/drivers/acorn/block/mfmhd.c`

```
mfm_tq.routine = (void (*)(void *)) mfm_initialise;
queue_task(&mfm_tq, &tq_immediate);
mark_bh(IMMEDIATE_BH);
```

Tasklets

- modernere Variante
- mehrere Tasklets parallel ausführen (SMP), aber: nie parallel zu sich selbst
- dynamisches erstellen von Tasklets
- Bsp.: `linux/drivers/char/keyboard.c`

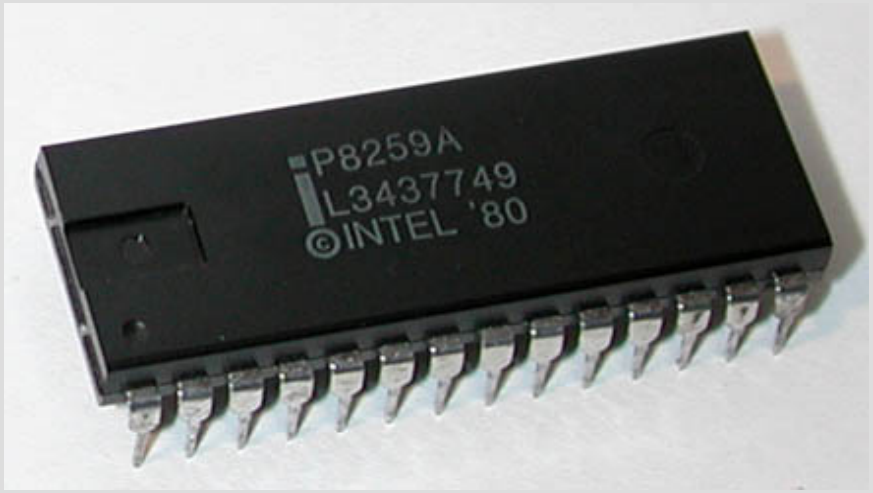
```
DECLARE_TASKLET_DISABLED(keyboard_tasklet, kbd_bh, 0);
```

und

```
tasklet_schedule(&keyboard_tasklet);
```

intel 8259 Programmable Interrupt Controller (PIC)

- zuständig für
  - system timer
  - keyboard controller
  - serial ports
  - parallel ports
  - floppy disk
  - IDE
  - mouse
  - DMA channels



- nicht multiprozessor-fähig
- i955X: 2 Stück als Master / Slave → 15 IRQs
- Statusregister:
  - IRR: Interrupt Request Register → wenn sich ein Gerät meldet
  - ISR: Interrupt Service Register → während IRQ-Vektor geliefert wird
  - IMR: Interrupt Mask Register → nicht mehr in CPU
- „Programmable“ → modes of Operation

modes of operation

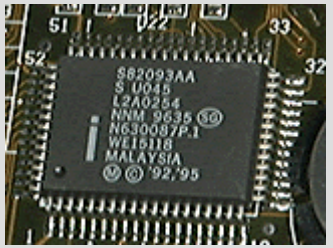
- Fully nested mode
  - höchste Priorität: IRQ 0, niedrigste: IRQ 7
  - Prioritäten können rotiert werden
- Special Fully nested mode
  - weitere IRQ des gleichen Geräts während der IRQ-Verarbeitung
  - Prüfen und Zurücksetzen durch Software → evtl weitere IRQ?
- Automatic Rotation Mode (Equal Priority Devices)
  - rotiert Geräte mit gleicher Priorität
  - nach IRQ: niedrigste
  - alle anderen sind einmal vor nächstem IRQ dran!
- Specific Rotation mode (Specific Priority)
  - Auswahl der niedrigsten Priorität per Software
  - z.B. IRQ 5 → höchste Priorität: IRQ 6

modes of operation

- Poll Mode
  - spart Einträge in der interrupt vector table
  - eine pollende ISR → ruft andere Handler auf
- Cascade Mode
  - 15 Prioritäten
  - 3-bit Bus als Verbindung Master – Slave
  - EOI Befehl muss zweimal benutzt werden (Master + Slave)
- Normal End of Interrupt (EOI)
  - EOI Befehl per Software schreiben
- Automatic End of Interrupt mode
  - EOI wird nach IACK-Zyklus automatisch generiert
  - vorsicht mit Special Fully Nested!

Advanced Programmable Interrupt Controller (APIC)

- Multiprozessor-Unterstützung
- 24 APIC-Interrupts
- unterstützt PCI-Express Message Based Interrupts



Message Adress Format	
31 – 20	immer FEEh
19 – 12	Destination ID
11 – 4	Extended Destination ID
3	Redirection Hint
2	Destination Mode
1 – 0	immer 00

PCI-Express Message Format

Message Data Format	
31 – 16	immer 0000h
15	Trigger Mode
14	Delivery Status
13 – 12	immer 00
11	Destination Mode
10 – 8	Delivery Mode
7 – 0	IRQ-Vektor

PCI / PCI-X Bus

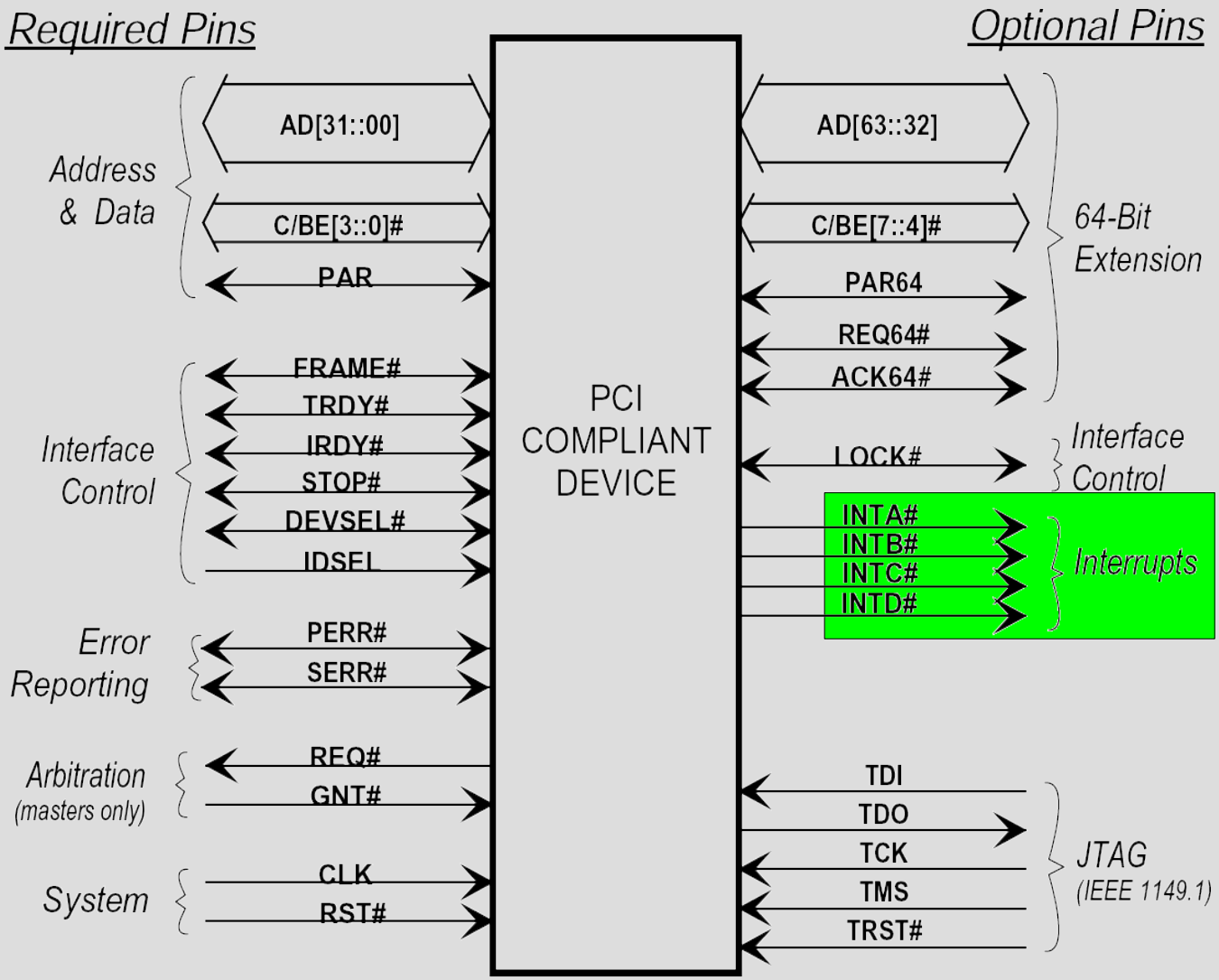
Spezifikation	Bus-Breite	Taktfrequenz	Datentransferrate	Signalspannung	Geräte pro Bus
PCI 2.3	32	33	0,133 GByte/s	5V	6
PCI 2.3	64	33	0,266 GByte/s	5V	6
PCI 2.3	32	66	0,266 GByte/s	3,3V	3
PCI 2.3	64	66	0,533 GByte/s	3,3V	3
PCI-X 1.0	64	66	0,533 GByte/s	3,3V	4
PCI-X 1.0	64	100	0,800 GByte/s	3,3V	2
PCI-X 1.0	64	133	1,066 GByte/s	3,3V	1
PCI-X 266 (2.0)	64	133 DDR	2,133 GByte/s	1,5V	1
PCI-X 533 (2.0)	64	133 QDR	4,266 GByte/s	1,5V	1

PCI-Overview

- Interrupt Sharing
- Burst Modus
- 4 IRQ pro Gerät (INTA bis INTD)
- Adressen-Daten-Multiplexing

- bei 1 Gerät pro Bus: mehrere parallele Busse
- PCI-X abwärtskompatibel zu PCI
- ECC Fehlerkorrektur

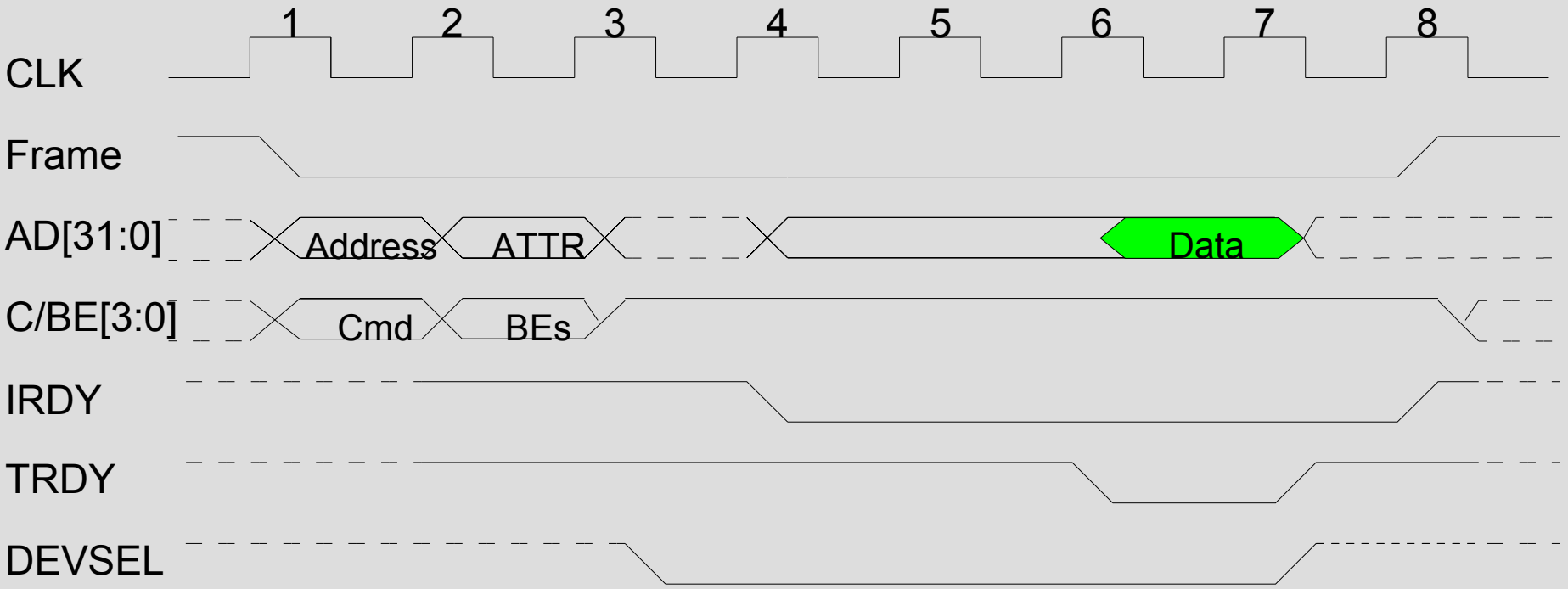
Interrupts über PCI / PCI-X



Pin-Belegung:

AD → Adress & Data
C/BE → Bus Command & Byte enable
Frame → Cycle Frame, bei Transaktion 0
IRDY → Initiator ready
TRDY → Target ready
DEVSEL → Device Select
INTA – INTD → PIC / APIC, normal nur INTA benutzt

Interrupt Acknowledge Command

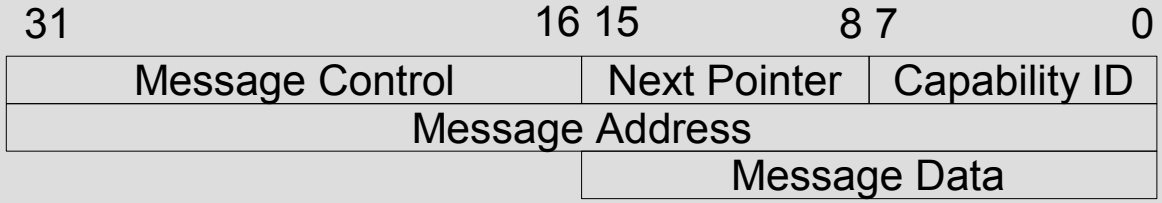


IACK-Command Signals

- entspricht im wesentlichen normaler read transaction
- C/BE = 0000 (PCI-X Command Encoding)
- CLK 1-8 = 1 Frame
- AD[31:0] = beliebiger zulässiger Wert
- Data = IRQ-Vektor

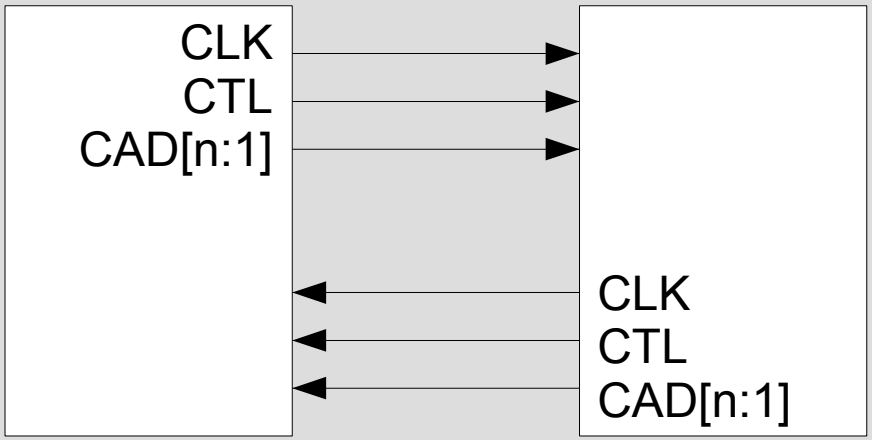
Message Signaled Interrupts (MSI)

- optional ab PCI > v2.1
- zusätzliche „Capability“-Register



- Capability ID == 05h → MSI-fähiges Gerät!
- OS muss PCI-Device für MSI konfigurieren
  → Message Adress Register setzen
  → Message Data Register setzen
  → bit 0 vom Message Control register setzen
- mehrere verschiedene Nachrichten möglich → restliche bits
- IRQ über INTA – INTD ist dann deaktiviert!
- PCI-X: IRQ-Devices müssen beides unterstützen!

HyperTransport Link



HyperTransport I/O Link

- CAD = Command, Addresses, and Data
  → HyperTransport requests, responses, addresses und data
  → 2, 4, 8, 16 oder 32 bit
- CTL=1 → Kontrollpaket
  CTL=0 → Daten
  → 1 bit
- CLK = Takt für CAD und CTL Signale
  → Eigener CLK für jedes byte CAD
  → CTL gleicher Takt wie CAD[0]
  → 1, 2 oder 4 bit
- (außerdem PWROK und RESET)

Paketstruktur

Code	Command
00000	NOP
00001	Reserved-HOST
00010	Flush
00011	Reserved-HOST
0001xx	
0001xx	Wr (sized)
101xxx	
01xxxx	Rd (sized)
100xxx	Reserved-I/O
110000	RdResponse
110001	Reserved-HOST
110010	
110011	TgtDone
11010x	Reserved-HOST
110110	Reserved-I/O
110111	Extended FC
11100x	Reserved-HOST
111010	Broadcast
111011	Reserved-HOST
111100	Fence
111101	Atomic-RMW
111110	AddrExt
111111	Sync/Error

Command Field Encoding

- Pakete = Vielfache von 4 bytes (doublewords)
- Kontrollpakete: 4 oder 8 bytes
- relevante Felder im Kontrollpaket:
  - SeqID[3:0] – Sequenznummer
    → Reihenfolge
  - Cmd[5:0] – Befehlsfeld
    → Pakettyp
  - PassPW – Paket darf Vorgänger „überholen“
    → Performance
  - UnitID[4:0] – HT-ID des Geräts
  - Addr[63:2] – Zieladresse
    → nicht immer alle bits benutzt
  - Reserved – muss immer 0 sein
- Datenpakete:
  - 4 - 64 bytes
    → Count[1:0]-Field

IRQ-Request Paket

Bit-Time	CTL	7	6	5	4	3	2	1	0
0	1	SeqID[3:2]		Cmd[5:0]: 1010X1					
1	1	PassPW	SeqID[1:0]		UnitID[4:0]				
2	1	Count[1:0]		Rsv	Reserved				
3	1	IntrInfo[7:6]		IntInfo[5]	IntrInfo[4:2]			Count[3:2]	
4	1	IntrInfo[15:8]							
5	1	IntrInfo[23:16]							
6	1	IntrInfo[31:24]							
7	1	Addr[39:32]							
8	0	IntrInfo[39:32]							
9	0	IntrInfo[47:40]							
10	0	IntrInfo[55:48]							
11	0	Reserved							

IRQ-Request Packet

- Cmd 1010X1 → Wr(sized), posted, cache-coherent
- Count immer 0 → nur ein doubleword
- IntrInfo[4:2] → IRQ-Typ, 111 reserviert für End of Interrupt (EOI)
- IntrInfo[5] → RQEOI, warten auf EOI
- IntrInfo[31:8] → Rückgabe in EOI

→ Rest ist systemabhängig

End of Interrupt (EOI) Pakete

Bit-Time	7	6	5	4	3	2	1	0
0	SeqID[3:2]			Cmd[5:0]: 111010				
1	PassPW	SeqID[1:0]		UnitID[4:0]				
2	Reserved							
3	Reserved				MT[2:0]=111b		Rsv	
4	IntrInfo[15:8]							
5	IntrInfo[23:16]							
6	IntrInfo[31:24]							
7	Addr[39:32]							

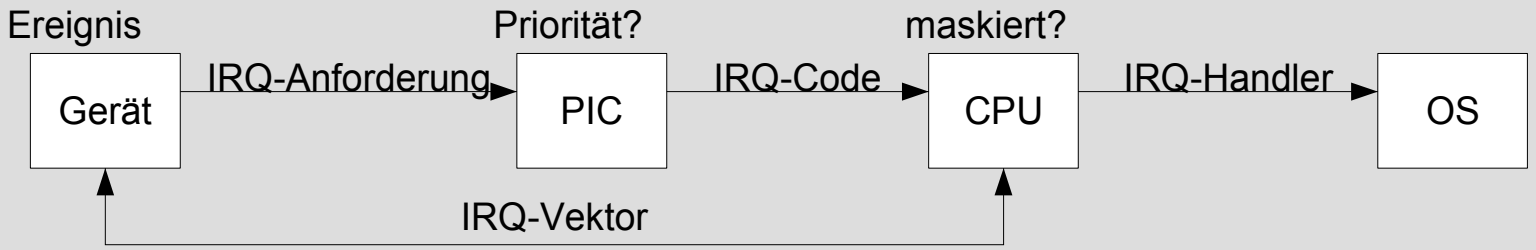
EOI-Packet

- Broadcast Message Packets → Cmd = 111010
- Devices vergleichen IntrInfo[31:8] und setzen IRQ zurück
- IntrInfo[4:2] → Message Type MT = 111b

- Alternativ → write ins *Interrupt Definition Register* des Devices

Zusammenfassung

- Ablauf eines Interrupts



- Intel 8259 PIC & APIC → modes of Operation
- Interrupts über PCI / PCI-X → Interrupt Signale
- Interrupts über HyperTransport → Interrupt Pakete

Quellen

- Flik – Mikroprozessortechnik, 6. Auflage
- Prentice-Hall – Motorola MC68020 User's Manual
- Addison-Wesley Longman – PCI-X System Architecture
- PCI Local Bus Specification
- PCI-X Addendum to the PCI Local Bus
- HyperTransport I/O Link Specification
  http://www.hypertransport.org/docucontrol/HTC20031217-0036-0009.pdf
- O'Reilly – Linux Gerätetreiber, 2. Auflage
  http://www.oreilly.de/german/freebooks/linuxdrive2ger/book1.html
- intel I/O Controller Hub 7 (ICH7) Family Datasheet
  http://download.intel.com/design/chipsets/datashts/30701301.pdf
- intel 82955X Memory Controller Hub (MCH) Datasheet
  http://download.intel.com/design/chipsets/datashts/30682801.pdf
- intel 82093 Advanced Programmable Interrupt Controller (APIC) Datasheet
  http://www.intel.com/design/chipsets/datashts/29056601.pdf
- AMD 8131 HyperTransport PCI-X Tunnel Datasheet
  http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24637.pdf
- http://www.hypertransport.org/tech/tech_latency.cfm
- http://www.elektronik-kompendium.de/sites/com/0310091.htm
- http://www.watch.impress.co.jp/akiba/hotline/980131/image/atx5863.gif
- http://www.cpu-world.com/Support/82/Intel-P8259A.jpg
- http://www.pcisig.com/specifications/