TFH-Berlin

Mikrocomputer-Technik

MCT 49

Teil 7: PIC 8259

Studiengang Technische Informatik (BA)
Prof. Dr.-Ing. Alfred Rożek

Interrupt-Controller 8259A

TFH-Berlin

Der Interrupt-Controller verbindet in geordneter Weise Interrupt-Signale verschiedener Quellen (Geräte, Bausteine) mit dem Interrupt-Eingang des Prozessors

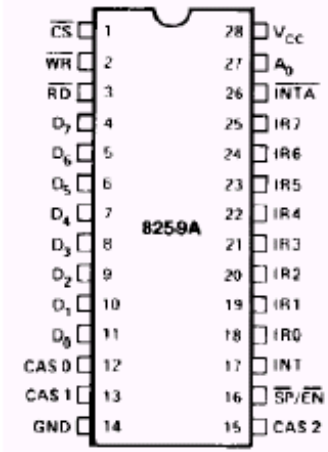
- Verwaltet bis zu acht Interrupts mit fest zugeordneten oder rotierenden Prioritäten
- Ermöglicht die Unterbrechung von Interrupts niederer Priorität (Interrupt-Service-Routinen) durch Interrupts höherer Priorität
- Durch Kaskadierung der Bausteine ausbaubar für bis zu 64 Interrupt-Quellen
- Alle Interrupt-Quellen sind einzeln maskierbar
- Die Interrupt-Signale können softwaretechnisch an die Anforderungen der angeschlossenen Geräte oder Bausteine angepasst werden (Pegel- oder Flankentriggerung)
- Übergibt einen der auslösenden Interrupt-Quelle zugeordneten Interruptvektor an den Prozessor

Anschluss-Pins des 8259A

TFH-Berlin

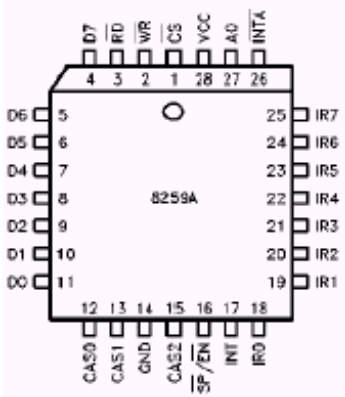
Symbol	Pin No.	Type	Name and Function
V_{CC}	28	I	**SUPPLY:** +5V Supply.
GND	14	I	**GROUND**
$\overline{CS}$	1	I	**CHIP SELECT:** A low on this pin enables $\overline{RD}$ and $\overline{WR}$ communication between the CPU and the 8259A. INTA functions are independent of CS.
$\overline{WR}$	2	I	**WRITE:** A low on this pin when CS is low enables the 8259A to accept command words from the CPU.
$\overline{RD}$	3	I	**READ:** A low on this pin when CS is low enables the 8259A to release status onto the data bus for the CPU.
D_7–D_0	4–11	I/O	**BIDIRECTIONAL DATA BUS:** Control, status and interrupt-vector information is transferred via this bus.
CAS_0–CAS_2	12, 13, 15	I/O	**CASCADE LINES:** The CAS lines form a private 8259A bus to control a multiple 8259A structure. These pins are outputs for a master 8259A and inputs for a slave 8259A.
$\overline{SP}/\overline{EN}$	16	I/O	**SLAVE PROGRAM/ENABLE BUFFER:** This is a dual function pin. When in the Buffered Mode it can be used as an output to control buffer transceivers (EN). When not in the buffered mode it is used as an input to designate a master (SP = 1) or slave (SP = 0).
INT	17	O	**INTERRUPT:** This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.
IR_0–IR_7	18–25	I	**INTERRUPT REQUESTS:** Asynchronous inputs. An interrupt request is executed by raising an IR input (low to high), and holding it high until it is acknowledged (Edge Triggered Mode), or just by a high level on an IR input (Level Triggered Mode).
$\overline{INTA}$	26	I	**INTERRUPT ACKNOWLEDGE:** This pin is used to enable 8259A interrupt-vector data onto the data bus by a sequence of interrupt acknowledge pulses issued by the CPU.
A_0	27	I	**A0 ADDRESS LINE:** This pin acts in conjunction with the $\overline{CS}$, $\overline{WR}$, and $\overline{RD}$ pins. It is used by the 8259A to decipher various Command Words the CPU writes and status the CPU wishes to read. It is typically connected to the CPU A0 address line (A1 for 8086, 8088).

DIP



231468–2

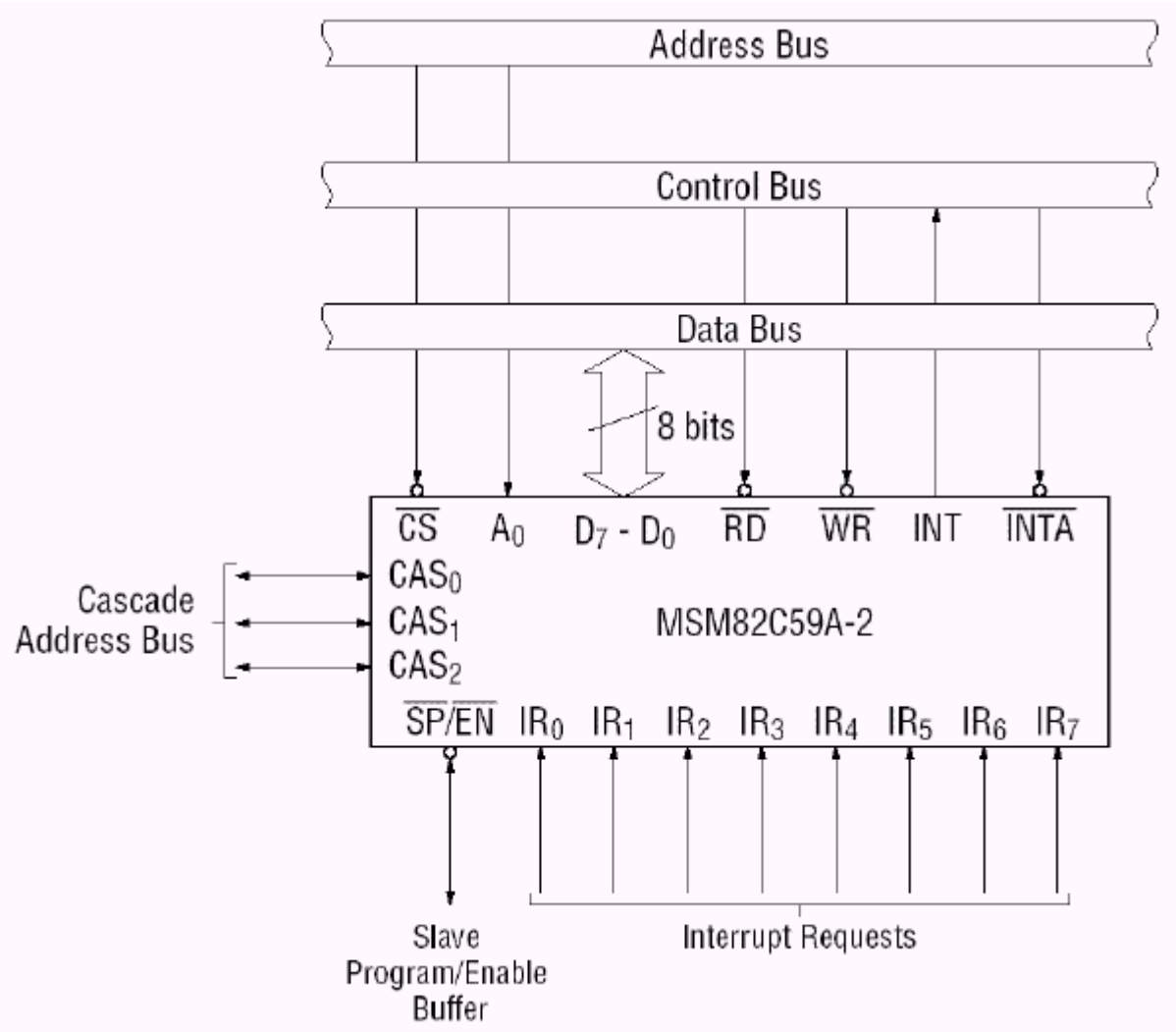
PLCC



Quelle: Intel

Integration des 8259A in ein Mikrocomputersystem

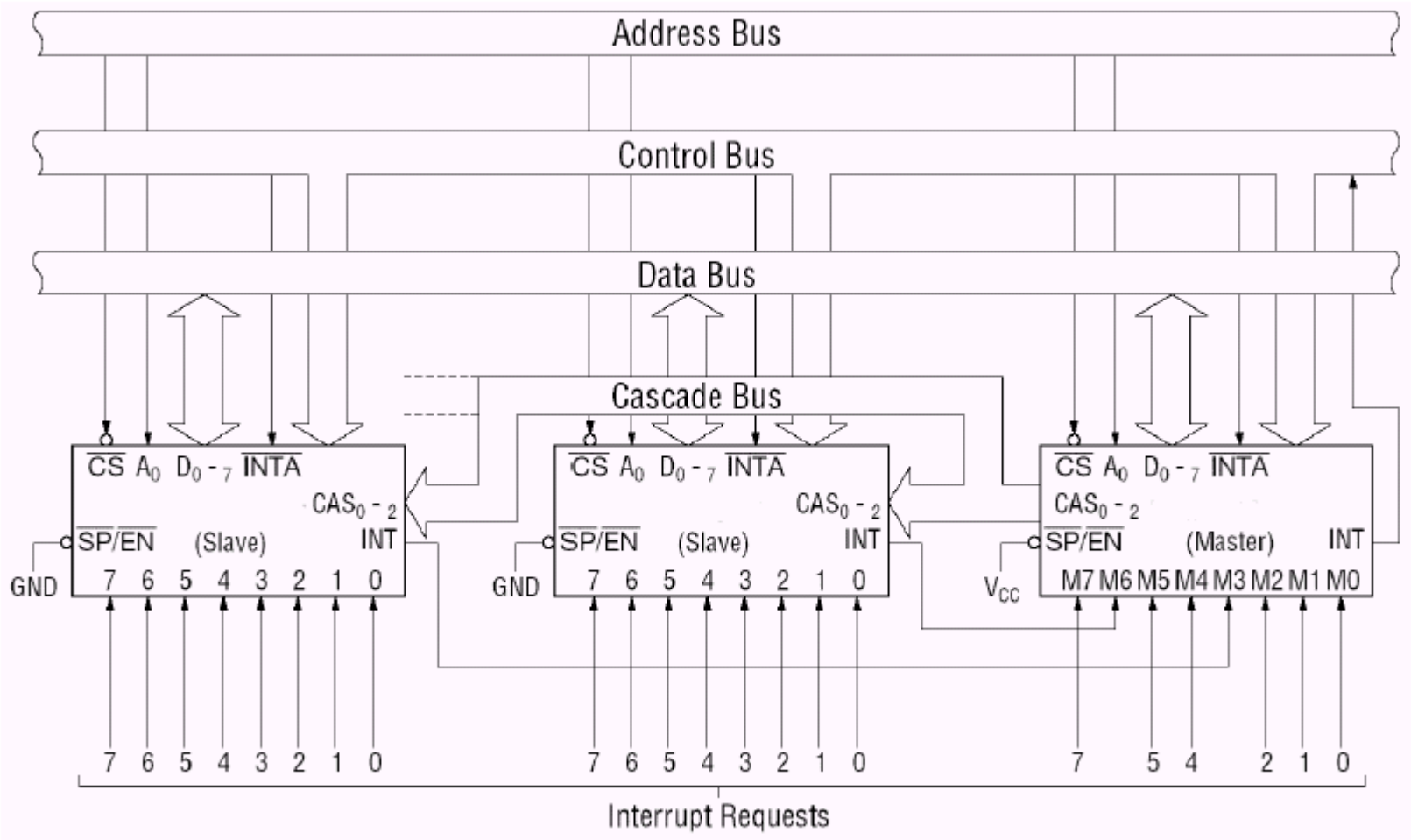
TFH-Berlin



Quelle: OKI Semiconductor

Kaskadierung von PICs

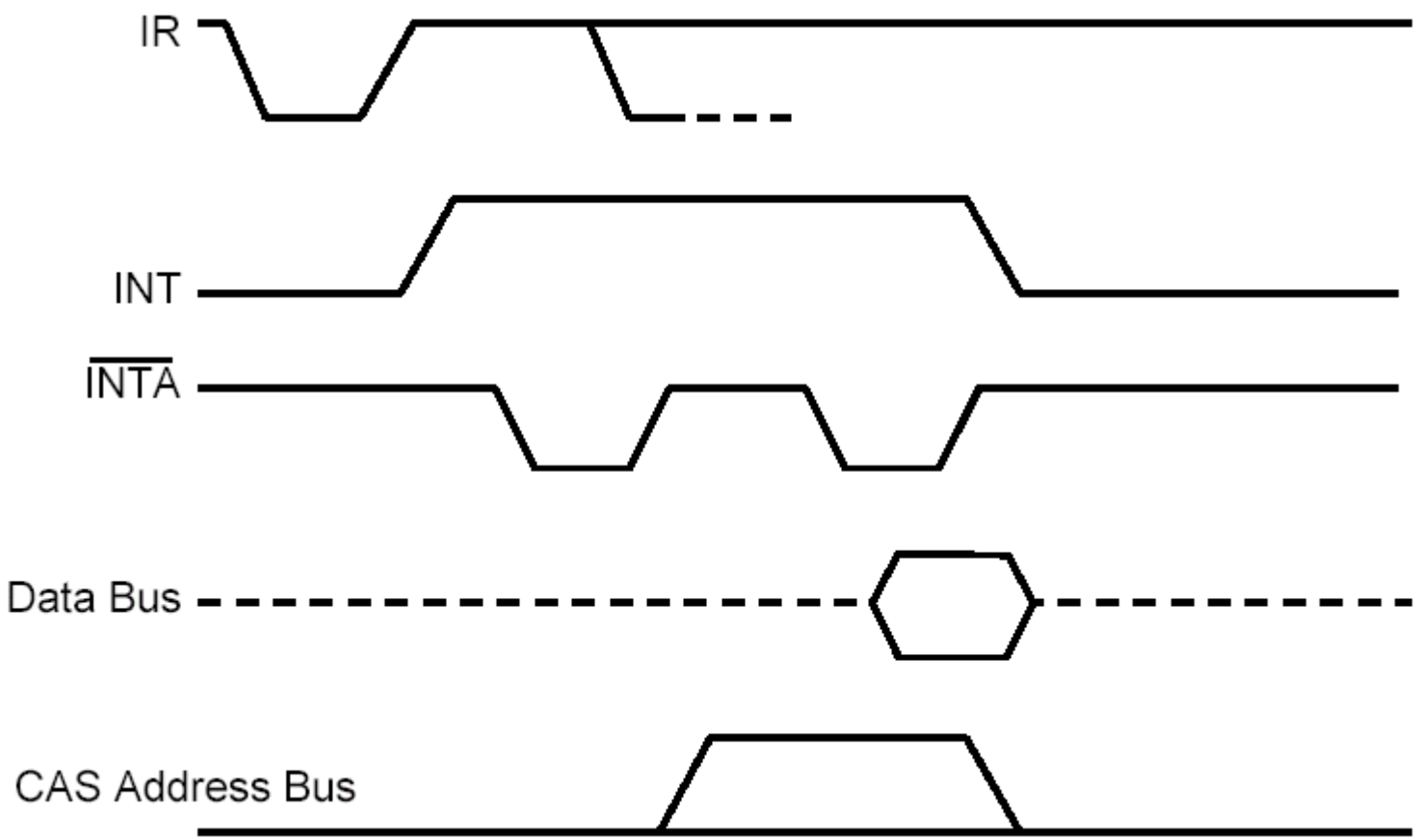
TFH-Berlin



Quelle: OKI Semiconductor

Singnalverlauf zwischen CPU und PIC
(INTA-Sequenz)

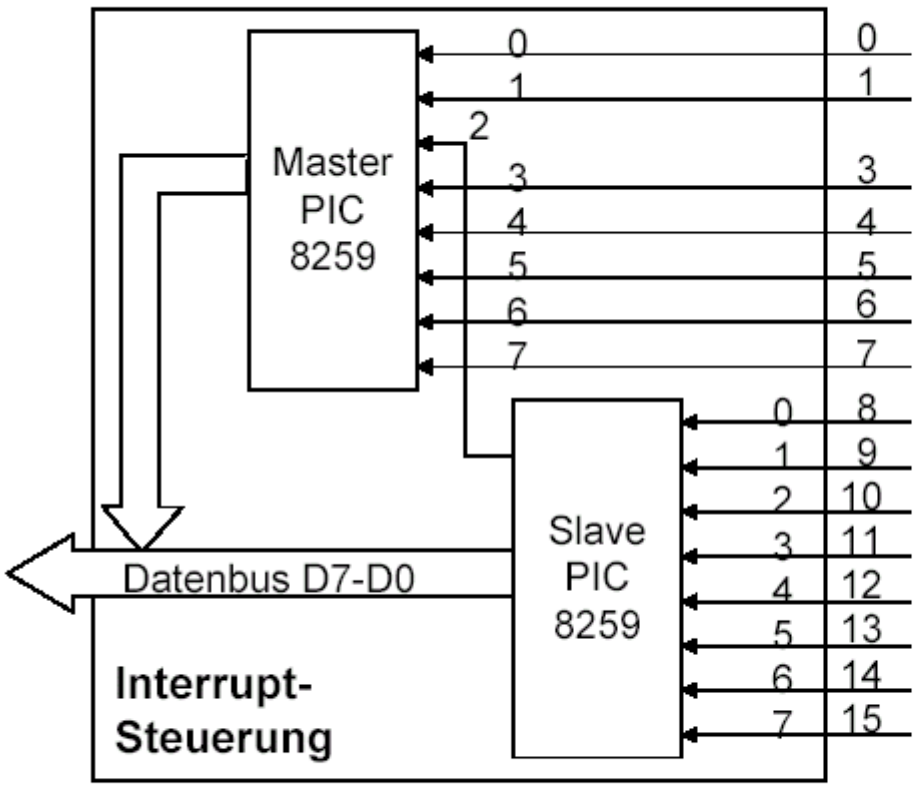
TFH-Berlin



Literaturhinweis:
König, Anne und Manfred: Mit dem PIC-Controller erfolgreich arbeiten
Verlag Markt und Technik, 1996

Kaskadierung von PICs und Belegung der IRQs im PC

TFH-Berlin

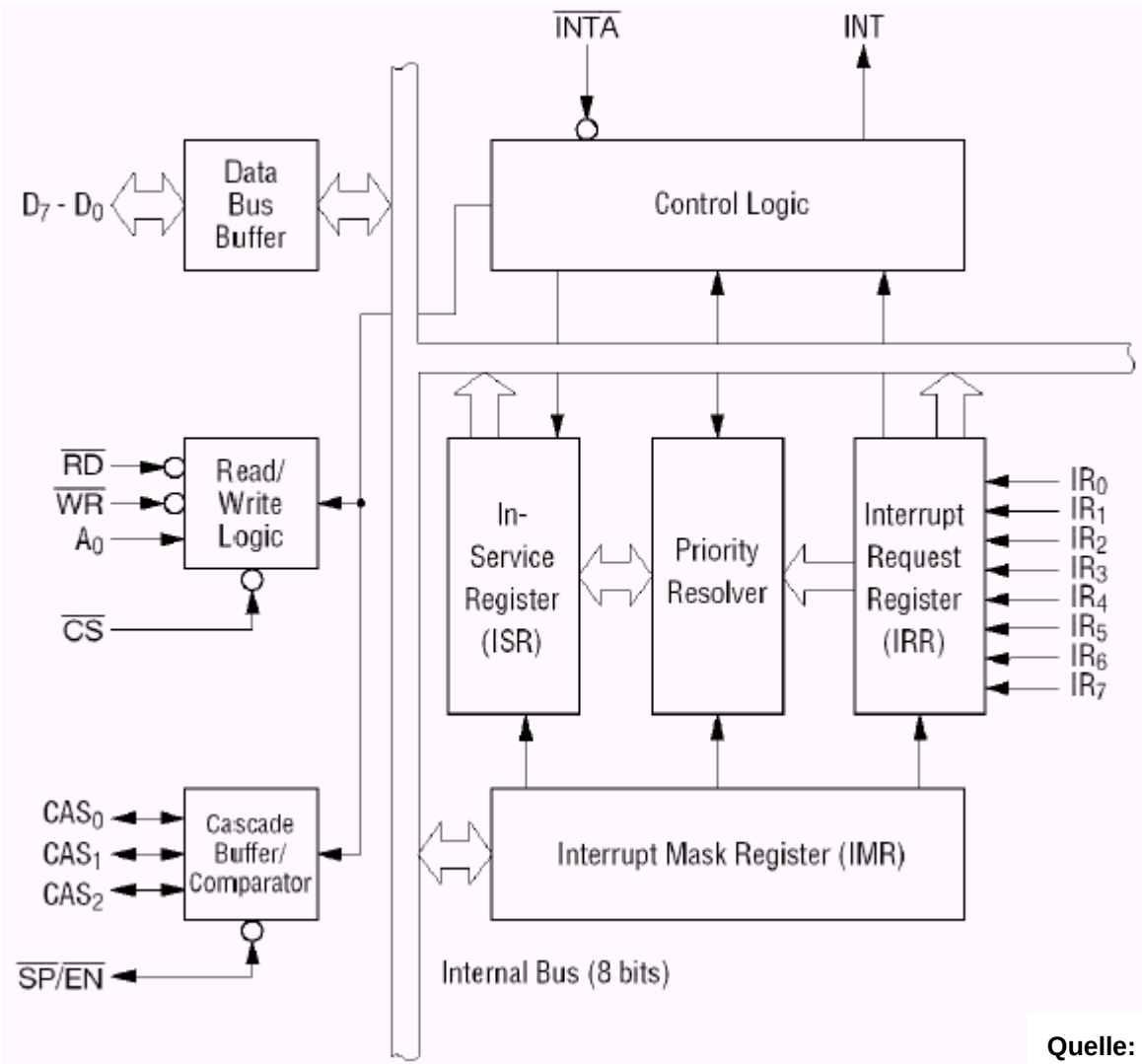


PIC = Programmierbarer Interrupt Controller

Master	Slave	Verwendung im PC
IRQ0		Timer 0
IRQ1		Tastatur
IRQ2		Slave PIC
	IRQ8	Echtzeituhr
	IRQ9	Frei oder reserviert
	IRQ10	Frei
	IRQ11	Frei
	IRQ12	Frei oder PS/2-Maus
	IRQ13	Coprozessor
	IRQ14	1. IDE-Controller
	IRQ15	2. IDE-Controller
IRQ3		COM2
IRQ4		COM1
IRQ5		LPT2
IRQ6		Diskettencontroller
IRQ7		LPT1

Interne Struktur des 8259

TFH-Berlin



Quelle: OKI Semiconductor

Beschreibung der Funktionsblöcke

TFH-Berlin

Interrupt Request Register (IRR) and In-Service Register (ISR): Die Unterbrechungsanforderungen auf den IR-Leitungen werden von zwei hintereinander geschalteten Registern aufgenommen, dem Unterbrechungs-Anforderungs-Register (IRR) und dem Unterbrechungs-Bedienungs-Register. Das IRR speichert die eine Bedienung anfordernde Unterbrechungsebene.
Priority Resolver: Dieser Block bestimmt die Prioritäten der im IRR gesetzten Bits. Die höchste Priorität wird ausgewählt und während eines INTA-Impulses in das entsprechende Bit des ISR übernommen.
Interrupt Mask Register (IMR): Das Unterbrechungs-Masken-Register (IMR) speichert die Bits für die zu maskierenden („sperrenden“) Unterbrechungsleitungen IRn. Seine Ausgänge greifen in das ISR ein. Das Ausmaskieren eines Unterbrechungseingangs höherer Priorität beeinflusst die Unterbrechungsleitungen niedriger Priorität nicht.
Read/Write Logic, Control Logic: Dieser funktionelle Block nimmt die Kommandos von der CPU entgegen. Er enthält die Intitialisierungswort-Register (ICW) und Steuerwort-Register (OCW), die die verschiedenen Steueranweisungen für den Baustein speichern. Zusätzlich wird die Übertragung der Zustandsinformationen auf den Systemdatenbus von diesem funktionellen Block ermöglicht.
Cascade Buffer Comparator: Der Kaskadierungstreiber/Vergleicher speichert und vergleicht die Kennung aller im System vorhandenen 8259A. Die E/A-Anschlüsse (CAS0-2) sind als Ausgänge geschaltet, wenn der Baustein Master ist (‘SP =1), und als Eingänge geschaltet, wenn der Baustein Slave ist (‘SP =0). Der Master gibt die Kennung des die Unterbrechung anfordernden Slave auf CAS0-2 aus. Der so freigegebene Slave legt seine vorprogrammierte Unterbrechungsadresse während der nächsten beiden folgenden ‘INTA-Impulse auf den Datenbus

Programmierung des 8259

TFH-Berlin

Zwei Arten von Kommandoworten:

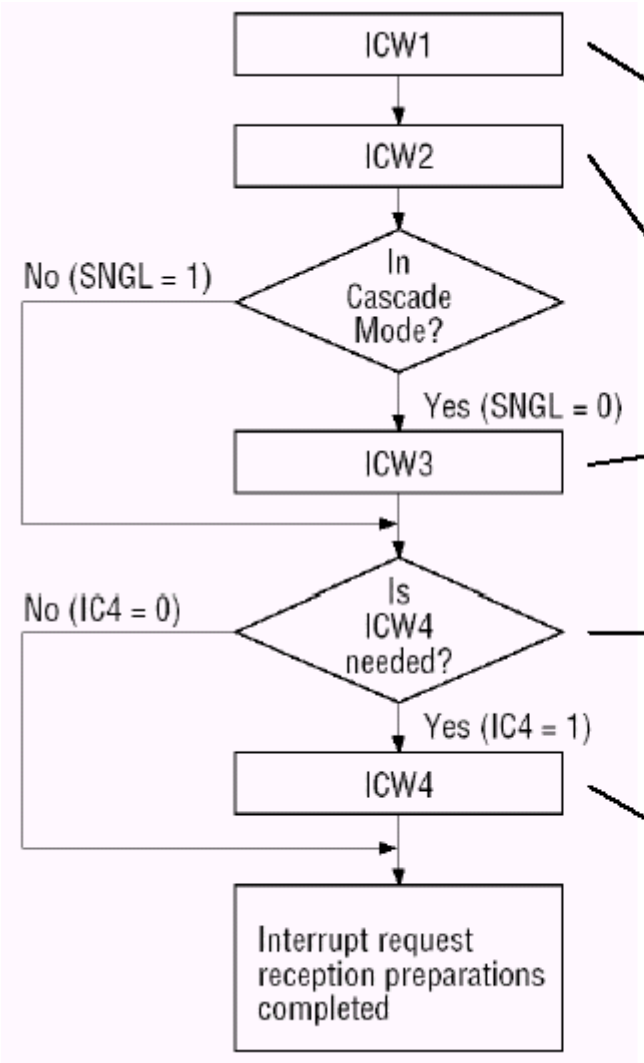
- Intitialisierungskommandos (ICW = Initialization Command Word)
- Operationssteuerkommandos (OCW = Operational Command Word)

Register	Mnemonics			Description	Access Method
	A0	D4	D3		
ICW 1	0	1	X	A write with A0 low and D4 high is interpreted as the beginning of an initialization sequence	Sequential access which starts with ICW1 and timed by the pulsing write signal
ICW 2	1	X	X	This register always follows ICW 1	
ICW 3	1	X	X	The use of this register depends on the value of SNGL in ICW 1	
ICW 4	1	X	X	The use of this register depends on the value of IC4 in ICW 1	
OCW 1	1	X	X	These registers can be accessed randomly	Random access
OCW 2	0	0	0		
OCW 3	0	0	1		

Quelle: Altera

Initialisierungssequenz 8259A

TFH-Berlin

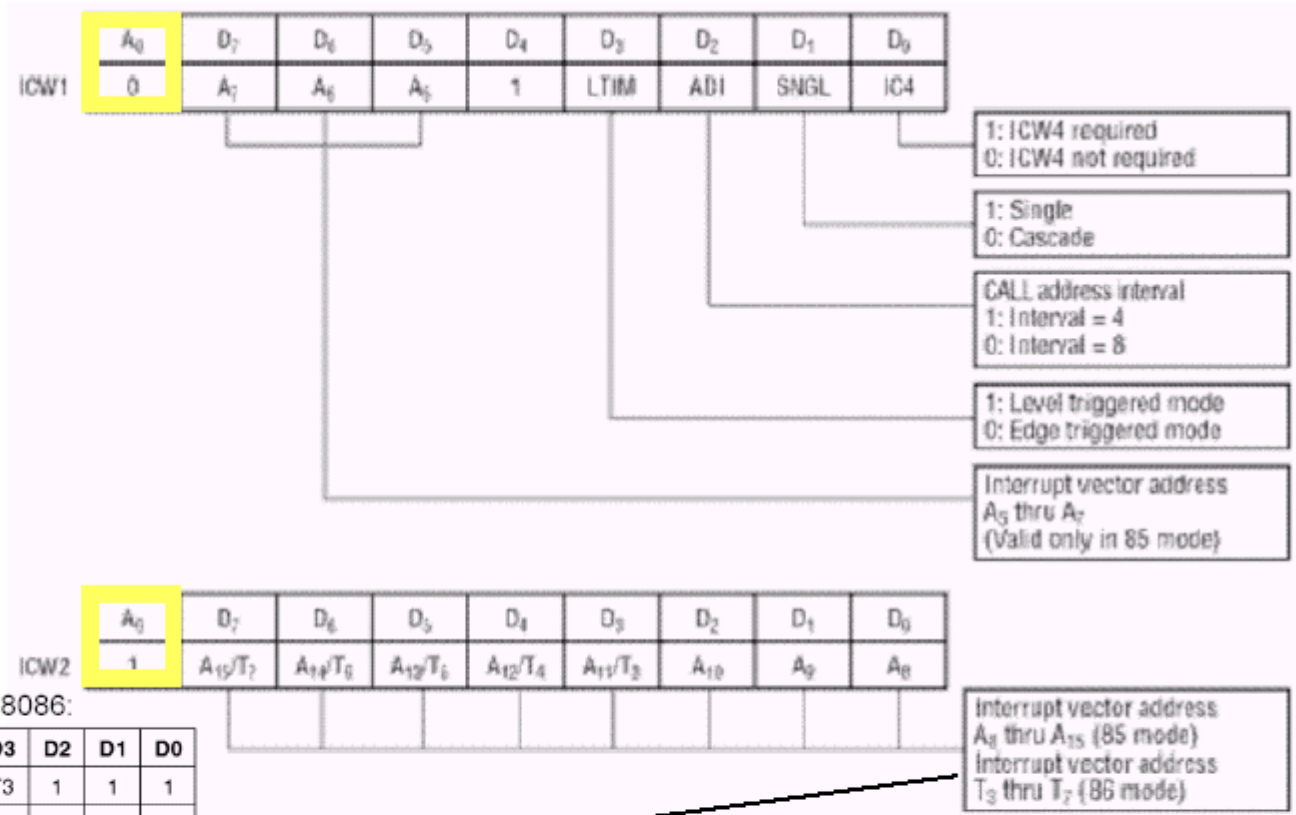


- Startet die Initialisierung
  - Flanken-/Pegelsteuerung
  - Kaskadierung (j/n)
  - ICW4 erforderlich (j/n)
- Nr. der INT-Vektoren festlegen
- Zusammenspiel Master/Slave festlegen
- Wenn nicht erforderlich:
  Alle Funktionen von ICW4 werden automatisch auf 0 gesetzt
  → nicht gepufferte Betriebsart
- Betriebsart festlegen

Quelle: OKI Semiconductor

Initialization Command Word Format – ICW1 und 2

TFH-Berlin



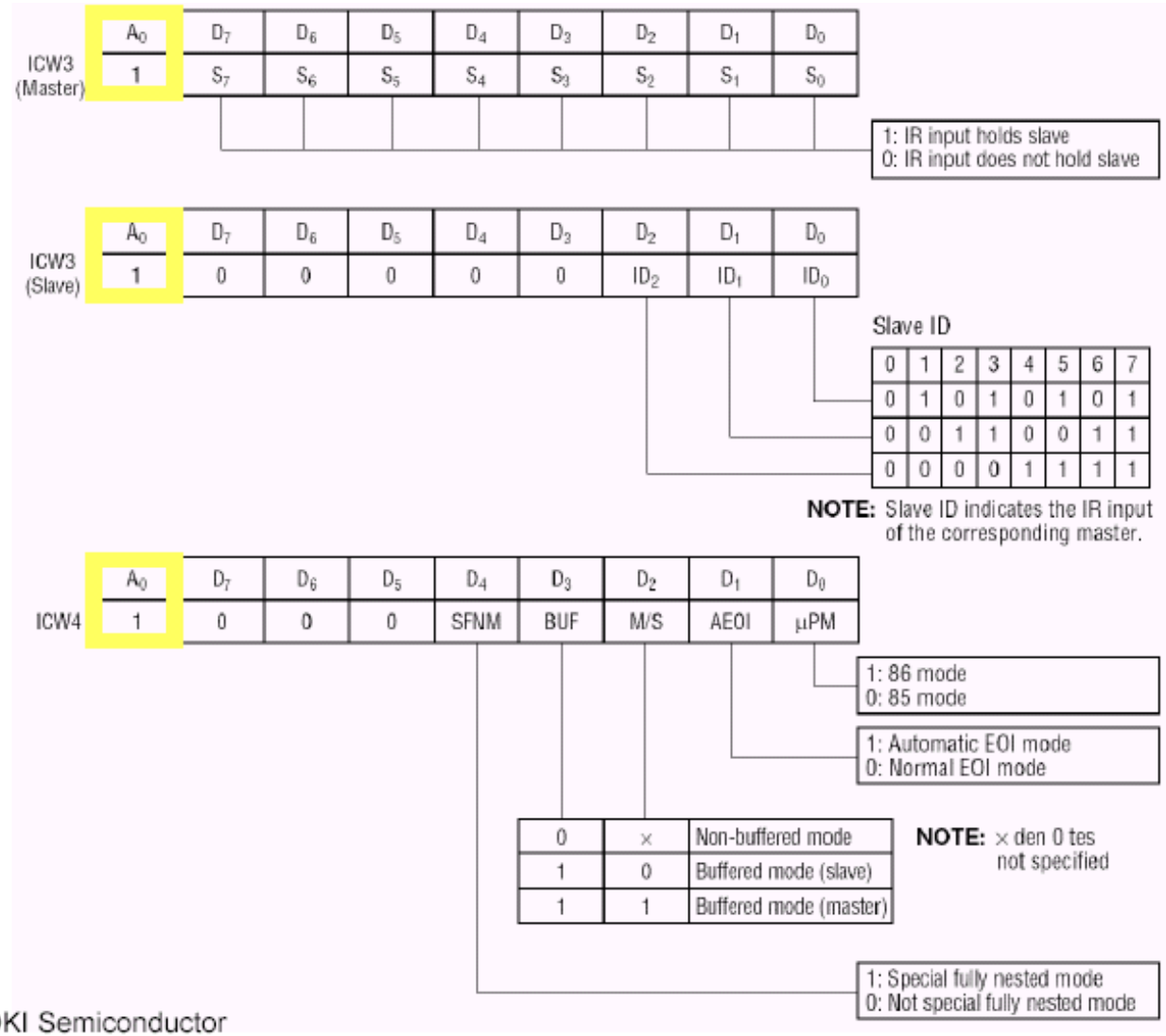
Interrupt-Vektorbyte für 8086:

	D7	D6	D5	D4	D3	D2	D1	D0
IR7	T7	T6	T5	T4	T3	1	1	1
IR6	T7	T6	T5	T4	T3	1	1	0
IR5	T7	T6	T5	T4	T3	1	0	1
IR4	T7	T6	T5	T4	T3	1	0	0
IR3	T7	T6	T5	T4	T3	0	1	1
IR2	T7	T6	T5	T4	T3	0	1	0
IR1	T7	T6	T5	T4	T3	0	0	1
IR0	T7	T6	T5	T4	T3	0	0	0

Quelle: OKI Semiconductor, Intel

Initialization Command Word Format – ICW3 und 4

TFH-Berlin



Quelle: OKI Semiconductor

Beispiel: Intitialisierung von Master- und Slave-PICs im PC

TFH-Berlin

ICW1 für Master und Slave:

0	0	0	1	LTIM	0	SNGL	IC4
0	0	0	1	0	0	0	1

- IC4 = 1: ICW4 notwendig (8086/88-Modus einstellen)
- SNGL = 0: zwei kaskadierte PICs
- LTIM = 0: Flankentriggerung im PC

```
MOV al, 11h   ; 00010001b=11h in Akkumulator laden
OUT 20h, al   ; ICW1 über Port 020h an Master ausgeben und Initialisierung starten
OUT a0h, al   ; ICW1 über Port 0a0h an Slave ausgeben und Initialisierung starten
```

ICW2 für Master:

Off7	Off6	Off5	Off4	Off3	0	0	0
0	0	0	0	1	0	0	0

- Offset 08h einstellen (IRQ0 entspricht INT 08h)

ICW2 für Slave:

Off7	Off6	Off5	Off4	Off3	0	0	0
0	1	1	1	0	0	0	0

- Offset 070h einstellen (IRQ8 entspricht INT 070h)

```
MOV al, 08h   ; 00001000b=08h in Akkumulator laden
OUT 21h, al   ; ICW2 über Port 021h an Master ausgeben
MOV al, 070h  ; 01110000b=070h in Akkumulator laden
OUT a1h, al   ; ICW2 über Port 0a1h an Slave ausgeben
```

ICW3 für Master:

S7	S6	S5	S4	S3	S2	S1	S0
0	0	0	0	0	1	0	0

- S2 = 1: Slave-PIC ist mit IRQ2 des Masters verbunden

```
MOV al, 04h   ; 00000100b=04h in Akkumulator laden
OUT 21h, al   ; ICW3 über Port 021h an Master ausgeben
```

ICW3 für Slave:

0	0	0	0	0	ID2	ID1	ID0
0	0	0	0	0	0	1	0

- ID = 2: Slave-PIC ist mit IRQ2 des Masters verbunden

```
MOV al, 02h   ; 00000010b=02h in Akkumulator laden
OUT a1h, al   ; ICW3 über Port 0a1h an Slave ausgeben
```

ICW4 für Master und Slave

0	0	0	SFNM	BUF	M/S	AEOI	µPM
0	0	0	0	0	0	0	1

- µPM = 1: PIC im 8086/88-Modus betreiben
- AEOI = 0: manueller EOI-Befehl von der CPU
- M/S = 0: bedeutungslos, da BUF=0
- BUF = 0: ungepufferter Modus
- SFNM = 0: normaler Nested-Mode

```
MOV al, 01h    ; 00000001b=01h in Akkumulator laden
OUT 21h, al    ; ICW4 über Port 021h an Master ausgeben
OUT a1h, al    ; ICW4 über Port 0a1h an Slave ausgeben
Damit ist die Initialisierung von Master und Slave im AT beendet.
```

Operation Control Word Format – OCW1 und 2

TFH-Berlin

	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
OCW1	1	M7	M6	M5	M4	M3	M2	M1	M0

Interrupt Mask
1: Mask set
0: Mask reset

	A_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
OCW2	0	R	SL	EOI	0	0	L2	L1	L0

Active IR Level

0	1	2	3	4	5	6	7
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	0	0	0	1	1	1	1

Operation Control Words (OCWs)

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

OCW2

0	R	SL	EOI	0	0	L2	L1	L0

OCW3

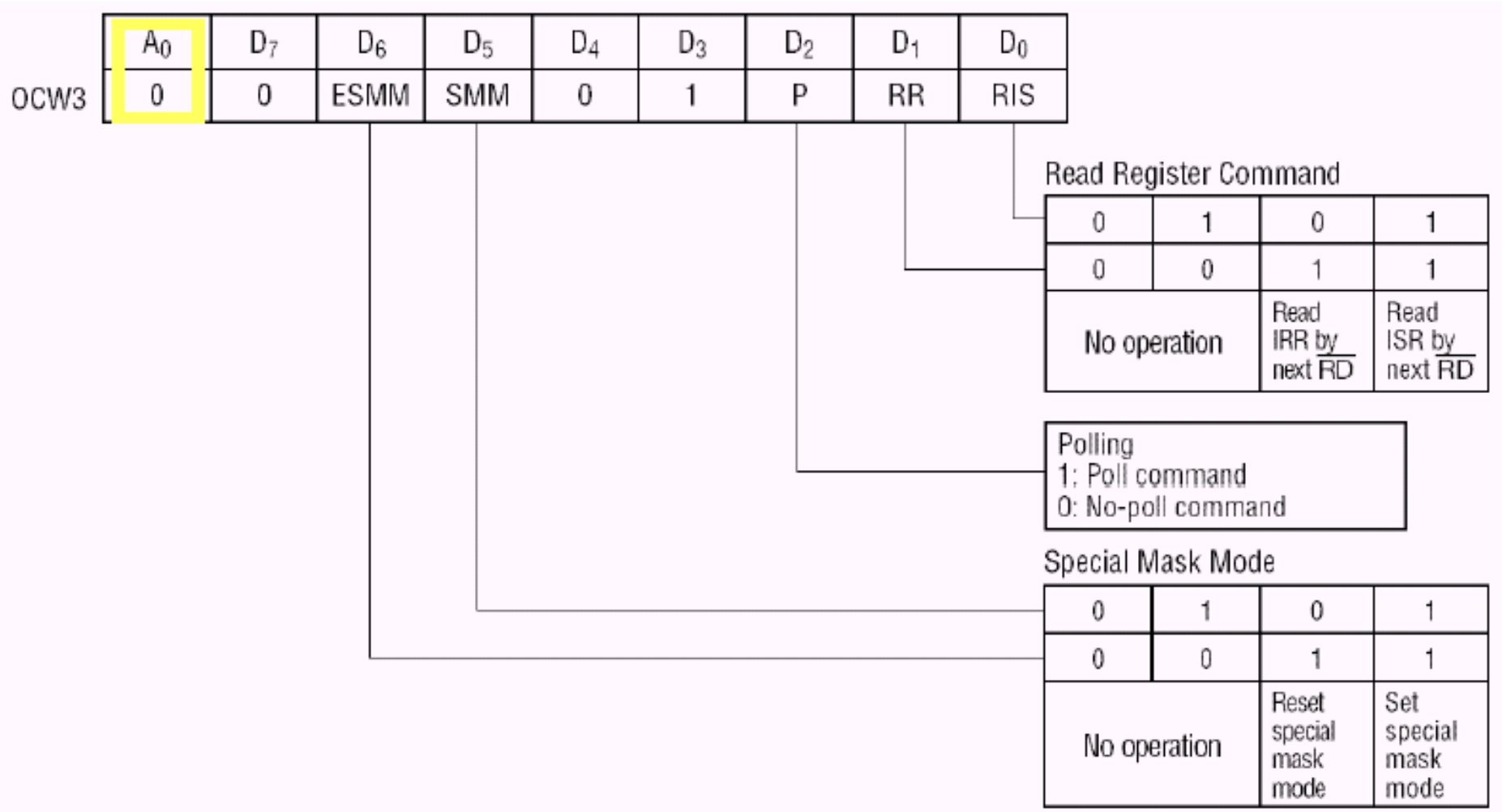
0	0	ESMM	SMM	0	1	P	RR	RIS

0	0	1	Non-specific EOI command	End of interrupt
0	1	1	Specific EOI command (NOTE)	
1	0	1	Rotate on non-specific EOI command	Automatic rotation
1	0	0	Rotate in automatic EOI mode (SET)	
0	0	0	Rotate in automatic EOI mode (Clear)	
1	1	1	Rotate on specific EOI command (NOTE)	Specific rotation
1	1	0	Set priority command (NOTE)	
0	1	0	No operation	

NOTE: L0 thru L2 used

Operation Control Word Format – OCW3

TFH-Berlin



Quelle: OKI Semiconductor

Priorisierung der Interrupts

TFH-Berlin

1. Voll verschachteltes Verfahren
   a. Fully Nested (default)
      - Feste Prioritäten von IR0 (höchste) bis IR7 (niedrigste)
      - Bei Bearbeitung eines Interrupts sind Interrupts höherer Prioritäten erlaubt (nur wenn in der CPU das IF gesetzt ist)
      - Das End of Interrupt command (EOI) gibt den PIC für den nächsten Interrupt frei – bzw. in der AEOI-Betriebsweise der INTA-Impuls.
   b. Special Fully Nested (more than one PIC)
2. Rotierende Priorität
   a. Automatische Rotation (round robin),
      Betriebsart A, erzeugt gleiche Prioritäten für alle IRs

	Vor Rotatation	Nach Rotation
"IS"	01010000	01000000
Priorität	76543210	21076543

   b. Spezifische Rotation (Rotieren durch Software)
      Betriebsart B, programmieren der niedrigsten Priorität, SEOI
3. Spezial-Masken-Verfahren
   Jeder Interrupt kann maskiert („stillgelegt") werden
4. Abfrage-Verfahren (Polling)
   IF in der CPU gelöscht. Programm liest von PIC die Vektornummer des höchstprioren Interrupts

Hinweise OCW

TFH-Berlin

I/O-Adressen und I/O-Daten des PIC 8259:

Der 8259 belegt nur zwei Portadressen, welche üblicherweise mit der Adressleitung A0 unterschieden werden.
Die beiden Interrupt-Controller werden über zwei Register gesteuert, die im PC unter folgenden Portadressen decodiert werden:

	Master	Slave - Controller		
	IRQ0-7	IRQ8-15	Lesedaten	Schreibdaten
Steuerkanal	**20h**	**A0h**	**IRR, ISR, Int.-Vektor**	**ICW1, OCW2, OCW3**
Datenkanal	**21h**	**A1h**	**IMR**	**ICW2, ICW3, ICW4, OCW1**

(ICWx = Initialisation Control Word, OCWx = Operation Control Word)

Im PC wird der 8259A im **End-of-Interrupt-Kommando-Modus (EOI)** betrieben und deswegen muss eine Interrupt-Service-Routine die Beendigung einer Interrupt-Bearbeitung durch ein EOI-Kommando (20H) an den Master und gegebenenfalls an den Slave melden, damit diese das zugehörige ISR-Bit löschen können und dann der Prioritätengeber weitere Interrupt-Anforderungen bedienen kann.

Ablauf eines Interrupts

TFH-Berlin

- Gerät fordert Interrupt am Programmierbaren Interrupt-Controller (PIC 8259A) an.
- PIC aktiviert Eingang INTR des Prozessors
- Wenn Int.-Flag gesetzt, bestätigt der Prozessor die Anforderung mit zwei aufeinanderfolgenden Impulsen am Ausgang INTA (Interrupt Acknowledge), gleichzeitig Deaktivierung der Busleitungen
- Der Prozessor liest vom Datenbus eine 8-Bit-Zahl, die Nummer des Interrupts, die vom PIC dort aufgelegt wurde. Der Interrupt vermerkt den anliegenden Interrupt nun als ”in Bearbeitung”
- Der Prozessor multipliziert die Interrupt-Nummer mit vier. Das Ergebnis wird als Adresse benutzt, um unter dieser Adresse in der Interrupt-Vektoren Tabelle die Einsprungadresse (Interrupt-Vektor, 4 Byte) der zugehörigen Service-Routine auszulesen.
  Beispiel: INT 5; es wird an Adresse 20d ein Interrupt-Vektor aus der Tabelle gelesen, dies ist die Einsprungadresse der Serviceroutine zu INT 5
- Der Prozessor führt folgende Aktionen aus:
  - Abspeichern von Flags, CS und IP auf den Stack
  - Trapflag und Interruptflag löschen
  - Interruptvektor nach CS:IP laden, Programm setzt in der Interrupt-Service-Routine (ISR) fort
- ISR sendet End-of-Interrupt-Steuerwort (EOI)an den Interrupt-Controller
- ISR endet mit IRET-Befehl: Rücksprungadresse und Flags vom Stack zurückholen, Wiederherstellung der Flags (auch TF und IF) und Rücksprung in unterbrochenes Programm
- Der PIC empfängt das EOI und löscht das ISR-Bit dieses Interrupts, er gilt nun als abgearbeitet

Beispiel: Initialisieren und Interrupts maskieren

TFH-Berlin

```
;Initialisierung des 8259A auf einem XT (alter PC mit einem 8259A):

INTA00   EQU 20H    ;Portadresse des 8259A im PC
INTA01   EQU 21H

PC_INIT:
   MOV   AL,13H     ;ICW1: Flankentrigger, nur ein 8259A, ICW4 erf.
   OUT   INTA00,AL
   MOV   AL,8       ;ICW2: INTs beginnen mit Nr. 8, laufen also von
   OUT   INTA01,AL  ;       Nr. 8 bis 15
                    ;Kein ICW3, da nur Master vorhanden!
   MOV   AL,1       ;ICW4: Prozessor ist 8086
   OUT   INTA01,AL
   MOV   AL,FFH     ;OCW1: Alle Interrupts werden maskiert
   OUT   INTA01,AL  ;OCW1 ausgeben

;Beispiel: Freigabe des Interrupts 0
  IN  AL,INTA01     ;Vorhandene Interrupt-Maske lesen
  AND  AL,0FEH      ;Bit 0 für Interrupt 0 löschen
  OUT  INTA01,AL    ;Maske ausgeben, Interrupt 0 ist freigegeben
```

Beispiel: Programmierung von EOI und IRR

TFH-Berlin

```
;Senden des non-specific end-of-interrupt Kommandos (EOI)
;an den Master

EOI   EQU   20H

      MOV   AL,EOI     ; Ausgabe an OCW2
      OUT   INTA00,AL

Senden des EOI wenn der IRQ vom Slave kam:

      MOV   AL,20H     ;EOI Kommando
      OUT   0A0H,AL    ;Senden an Slave PIC
      OUT   20H,AL     ;Senden an Master PIC

Lesen des Interrupt Request Register (IRR).

      MOV   AL,0AH     ;OCW3: OCW3-Bit und IRR-Bit setzen
      OUT   INTA00,AL ;       und ausgeben
      IN    AL,INTA00 ;Lesen des IRR
      ;Ausgabe von 0BH statt 0AH: ISR wird gelesen
```