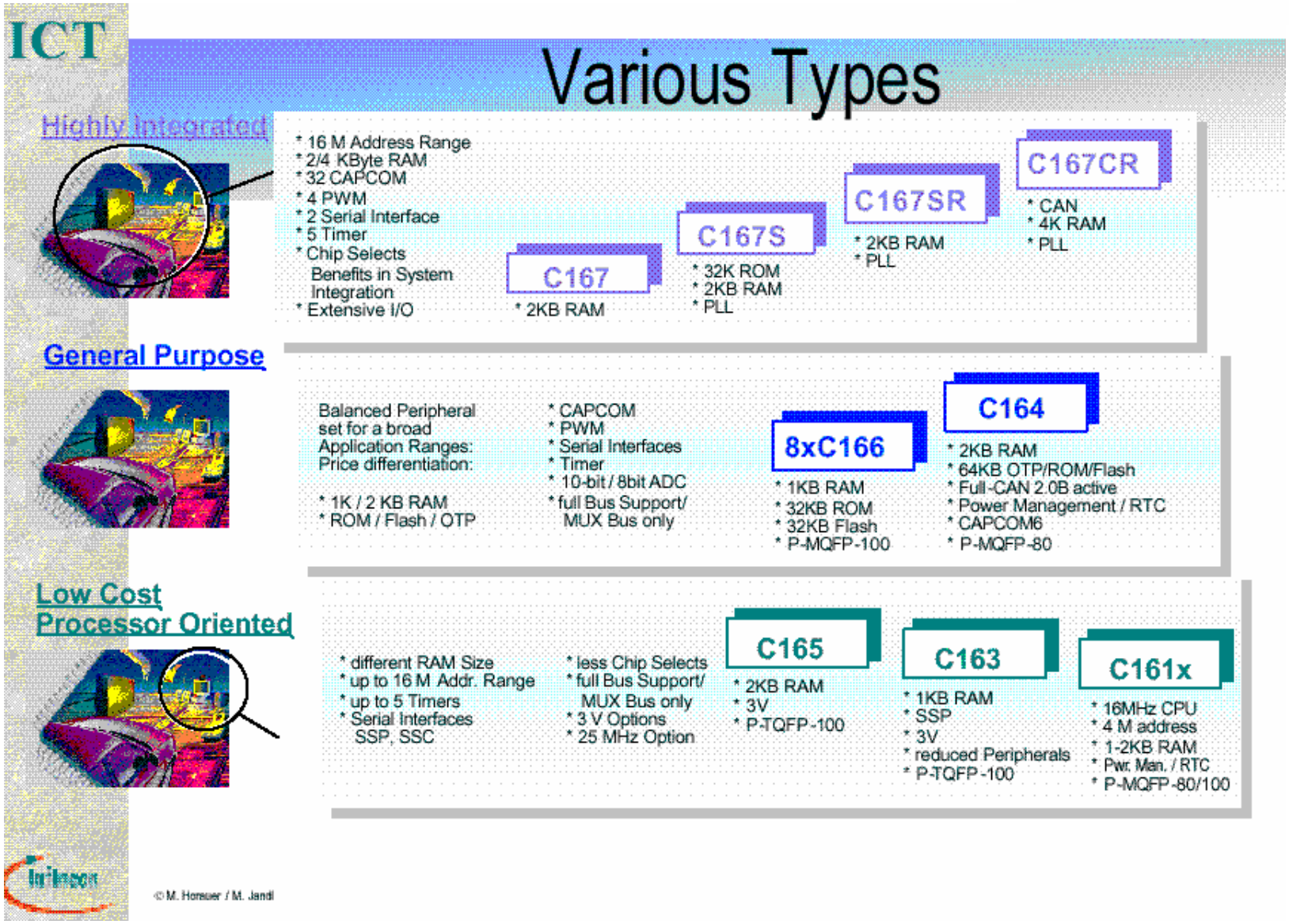
1 C167 Einführung

1.1 µController Familie 80C166 der Fa. Infineon

Entwickelt ca. 1990

Entwicklungsziele:

- schneller moderner 16-bit-Mikrocontroller
- Interruptbehandlung mit schnellen Antwortzeiten für Vielzahl von Quellen
- schnelle und effektive Einzelbit-Verarbeitung
- schneller Datentransfer der On Chip Peripherie mit CPU/Speicher
- verschiedene Buskonfigurationen mit vielen Timingparametern
- geringe Chipfläche


ICT
Various Types
Highly Integrated
* 16 M Address Range
* 2/4 KByte RAM
* 32 CAPCOM
* 4 PWM
* 2 Serial Interface
* 5 Timer
* Chip Selects
Benefits in System Integration
* Extensive I/O
C167
* 2KB RAM
C167S
* 32K ROM
* 2KB RAM
* PLL
C167SR
* 2KB RAM
* PLL
C167CR
* CAN
* 4K RAM
* PLL
General Purpose
Balanced Peripheral set for a broad Application Ranges:
Price differentiation:
* 1K / 2 KB RAM
* ROM / Flash / OTP
* CAPCOM
* PWM
* Serial Interfaces
* Timer
* 10-bit / 8bit ADC
* full Bus Support/ MUX Bus only
8xC166
* 1KB RAM
* 32KB ROM
* 32KB Flash
* P-MQFP-100
C164
* 2KB RAM
* 64KB OTP/ROM/Flash
* Full-CAN 2.0B active
* Power Management / RTC
* CAPCOM6
* P-MQFP-80
Low Cost
Processor Oriented
* different RAM Size
* up to 16 M Addr. Range
* up to 5 Timers
* Serial Interfaces SSP, SSC
* less Chip Selects
* full Bus Support/ MUX Bus only
* 3 V Options
* 25 MHz Option
C165
* 2KB RAM
* 3V
* P-TQFP-100
C163
* 1KB RAM
* SSP
* 3V
* reduced Peripherals
* P-TQFP-100
C161x
* 16MHz CPU
* 4 M address
* 1-2KB RAM
* Pwr. Man. / RTC
* P-MQFP-80/100
Infineon
© M. Horauer / M. Jandl

C166 Family Overview

C166 Family	I/O	ADC	Timer Counter	Capture Compare	PWM	Serial Interface	Watchdog	Add On's
C161 Family	63-76	-/4 Ch. 8 Bit	3-5	-	-	USART SSC	✓	-/I²C
C163 Family	77	-	5	-	-	USART SSP	✓	-
C164 Family	59	-/8 Ch. 8 Bit	5	8 Ch.	6 Outp.	USART SSC	✓	CAN v2.0B
C165 Family	77	-	5	-	-	USART SSP	✓	-
C166 Family	76	-/10 Ch. 10 Bit	7	16 Ch.	-	2 x USART	✓	-
C167 Family	111	-/16 Ch. 10 Bit	9	32 Ch.	4 Outp.	USART SSC	✓	- CAN v2.0B

© M. Horauer / M. Jandl

1.2 Der Mikrocontroller C167CS

16-Bit Single-Chip Microcontroller
C166 Family

C167CS-4R, C167CS-L

- High Performance 16-bit CPU with 4-Stage Pipeline
  - 80/60/50 ns Instruction Cycle Time at 25/33/40 MHz CPU Clock
  - 400/303/250 ns Multiplication (16 × 16 bit), 800/606/500 ns Division (32-/16-bit)
  - Enhanced Boolean Bit Manipulation Facilities
  - Additional Instructions to Support HLL and Operating Systems
  - Register-Based Design with Multiple Variable Register Banks
  - Single-Cycle Context Switching Support
  - 16 MBytes Total Linear Address Space for Code and Data
  - 1024 Bytes On-Chip Special Function Register Area
- 16-Priority-Level Interrupt System with 56 Sources, Sample-Rate down to 40/30/25 ns
- 8-Channel Interrupt-Driven Single-Cycle Data Transfer Facilities via Peripheral Event Controller (PEC)
- Clock Generation via on-chip PLL (factors 1:1.5/2/2.5/3/4/5), via prescaler or via direct clock input

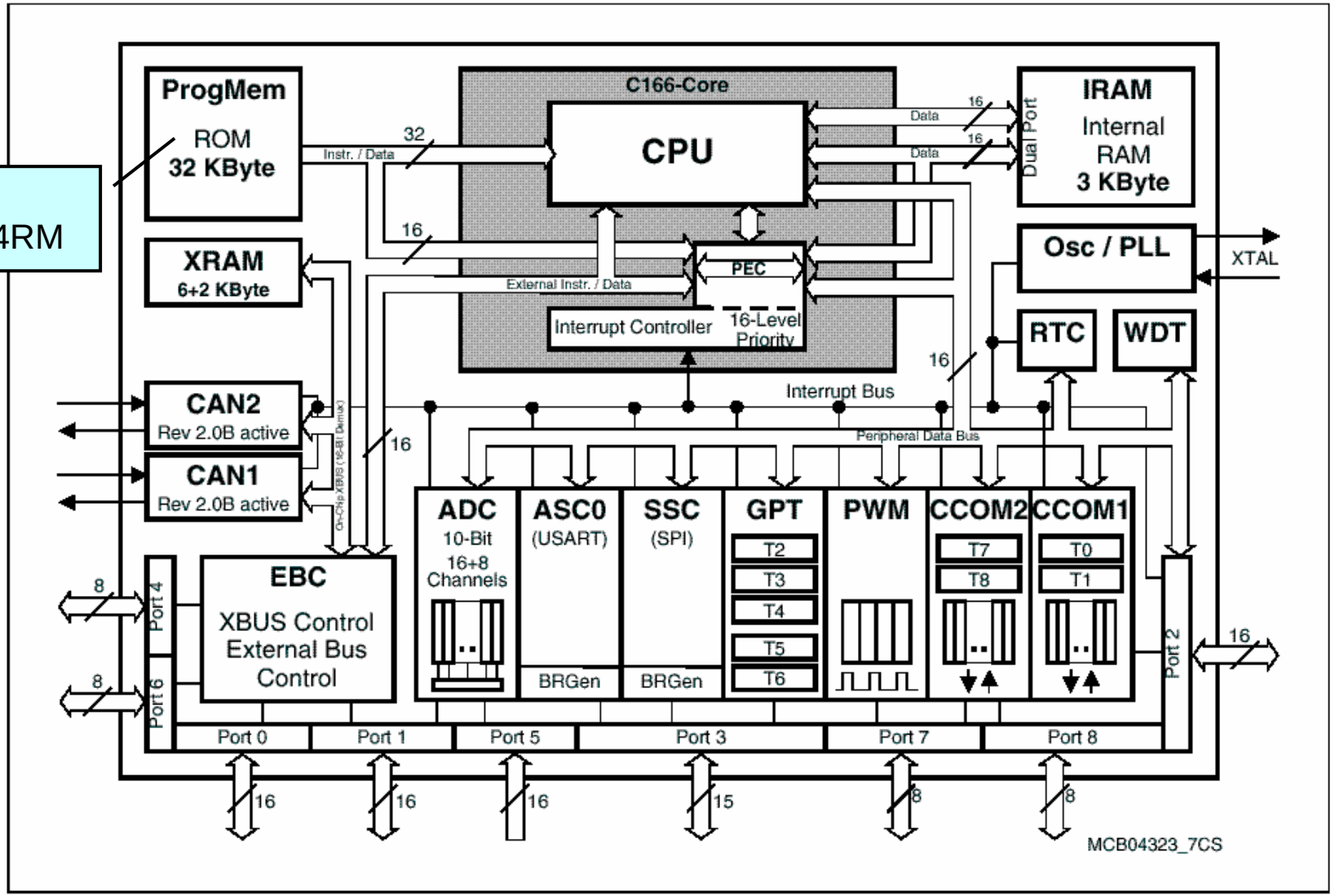
Embedded Systems

- On-Chip Memory Modules
  - 3 KBytes On-Chip Internal RAM (IRAM)
  - 8 KBytes On-Chip Extension RAM (XRAM)
  - 32 KBytes On-Chip Program Mask ROM
- On-Chip Peripheral Modules
  - 24-Channel 10-bit A/D Converter with Programmable Conversion Time down to 7.8 µs
  - Two 16-Channel Capture/Compare Units
  - 4-Channel PWM Unit
  - Two Multi-Functional General Purpose Timer Units with 5 Timers
  - Two Serial Channels (Synchronous/Asynchronous and High-Speed-Synchronous)
  - Two On-Chip CAN Interfaces (Rev. 2.0B active) with 2 × 15 Message Objects (Full CAN/Basic CAN), can work on one bus with 30 objects
  - On-Chip Real Time Clock
- Up to 16 MBytes External Address Space for Code and Data
  - Programmable External Bus Characteristics for Different Address Ranges
  - Multiplexed or Demultiplexed External Address/Data Buses with 8-Bit or 16-Bit Data Bus Width
  - Five Programmable Chip-Select Signals
  - Hold- and Hold-Acknowledge Bus Arbitration Support
- Idle, Sleep, and Power Down Modes with Flexible Power Management
- Programmable Watchdog Timer and Oscillator Watchdog
- Up to 111 General Purpose I/O Lines, partly with Selectable Input Thresholds and Hysteresis

Controller auf „unserem“ Board:
C167CS-LM
(→ kein ROM!!)

1.2.1 Blockschaltbild

bei
CS167CS-4RM



Insgesamt fünf Busse:

* Anschluss für interne Peripherie
* Zwei Busse (gleichzeitig lesend und schreibend) zu IRAM
* 32-Bit-Bus zum ROM
* Bus zum „External Bus Controller“

Die Daten- und Adressregister (GPR = General Purpose Register) als umschaltbare Registerbänke im RAM

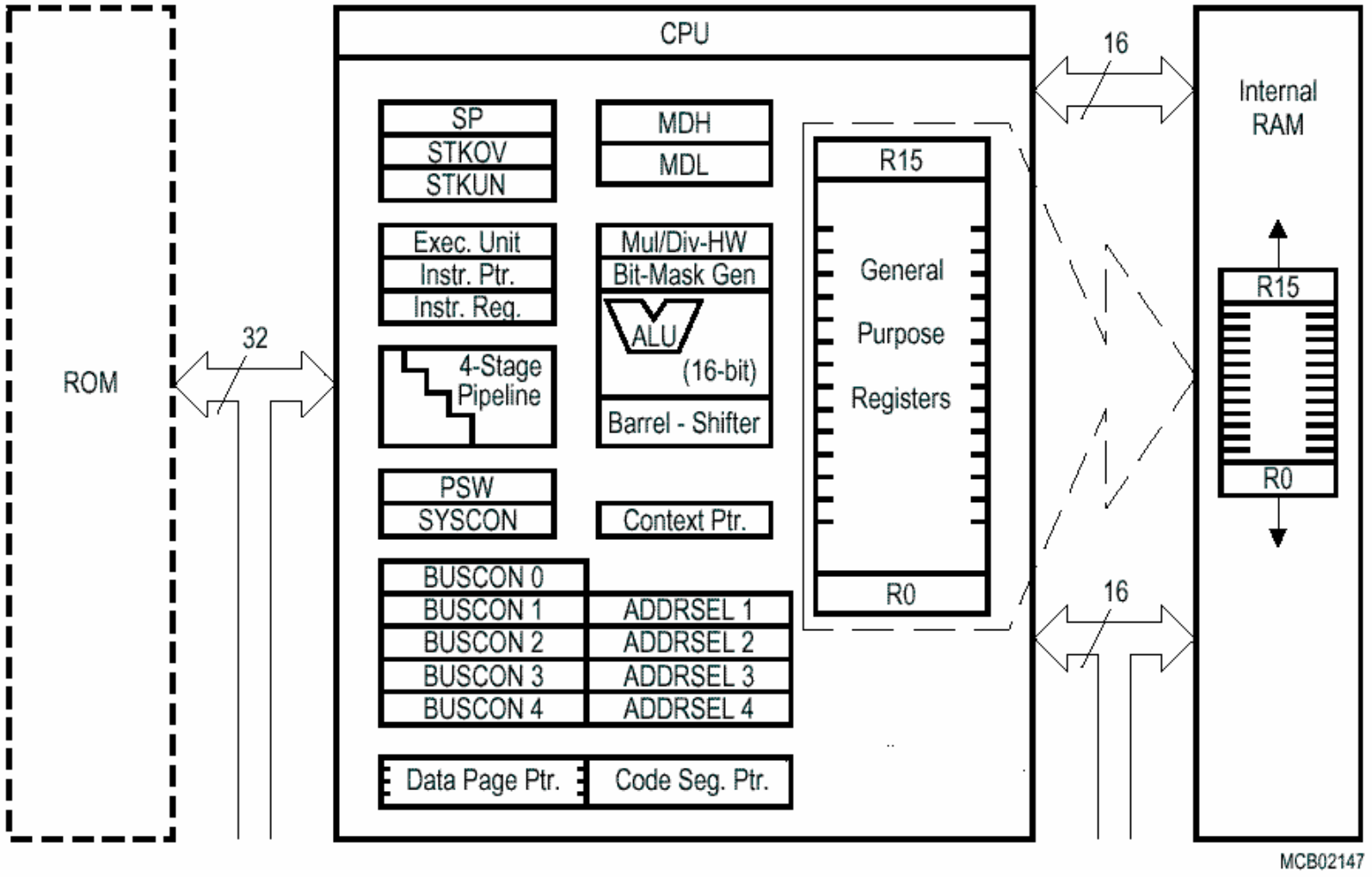
Verbindung des Controllers mit zu steuernden System (Anwendung) über interne Peripherieeinheiten (parallele, serielle und analoge Schnittstellen und Timer)

Die Adressierung der Funktionseinheiten erfolgt über die SFR-(Special Funktion Register) (ebenfalls im RAM)

Mit internem Speicher sind keine externen Bauelemente erforderlich – die Busanschlüsse können dann als Ports genutzt werden

Bootstrap Loader Mode erlaubt laden von Programmen über die serielle Schnittstelle

1.2.2 CPU Block Diagramm



1.2.2.1 System Configuration Register (SYSCON)

SYSCON **RAM:** 0FF12H **SFR:** 089H **Reset:** 0xx0H *bitadressierbar*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STKSZ			ROM *S1*	SGT *DIS*	ROM *EN*	BYT *DIS*	CLK *EN*	WR *CFG*	-	-	-	-	XP *EN*	VIS ...	XPE ...

Die Felder haben folgende Bedeutung:

XPER-SHAHRE	externer Zugriff auf den internen XBUS (0 = gesperrt 1 = frei)
VISIBLE	interner XBUS extern sichtbar (0 = nein 1 = ja)
XPEN	Zugriff auf XBUS Peripherie XRAM (0 = gesperrt 1 = frei)
WRCFG	Schreibkonfiguration (bei Reset invertiert von P0.0 übernommen) 0 = Signale /WR und /BHE 1 = Signale /WRL und /WRH
CLKEN	0 = Stift P3.15 ist Port 1 = Stift P3.15 ist Ausgang CPU-Takt
BYTDIS	Steuersignal /BHE (Busbreite bei Reset von P0.7 übernommen) 0 = Stift P3.12 ist /BHE 1 = Stift P3.12 ist Port
ROMEN	Interner ROM (bei Reset von Anschluß /EA übernommen) 0 = externer Programmspeicher 1 = interner Programmspeicher
SGTDIS	Segmentierung (0 = ein 1 = aus) für CALL-Befehle und Interrupt
ROMS1	Adreßbereich des internen ROM (0 = Segment_0 1 = Segment_1)
STKSZ	Größe und Lage des Systemstapels 000 = 256 Wörter im Bereich von 0FA00H bis 0FBFEH 001 = 128 Wörter im Bereich von 0FB00H bis 0FBFEH 010 = 64 Wörter im Bereich von 0FB80H bis 0FBFEH 011 = 32 Wörter im Bereich von 0FBC0H bis 0FBFEH *100 = 512 Wörter im Bereich von 0F800H bis 0FBFEH* 111 = 1024 Wörter im Bereich von 0F600H bis 0FDFEH

1.2.2.2 Bus Configuration Register (BUSCONx)

Die Register **BUSCONx** sind *bitadressierbar*.

15	14	13	12	11	10	9	8	7 6	5	4	3 2 1 0
CSW ENx	CSR ENx	-	RDY ENx	-	BUS ACTx	ALE CTLx	-	BTYP	MTT Cx	RWD Cx	MCTC

Die Felder haben folgende Bedeutung:

MCTC	Verlängerung der aktiven Low-Zeit von /RD bzw. /WR (15 - MCTC) 0000 = 15 Wartetakte bis 1111 = keine Wartetakte
RWDCx	Verzögerung fallende Flanke ALE bis fallende Flanke /RD bzw. /WR 0 = eine Taktzeit TCL (Vorgabe) 1 = keine Verzögerung
MTTCx	Verzögerung steigende Flanke /RD bzw. /WR bis neuer Buszyklus 0 = ein Wartetakt (Vorgabe) 1 = keine Verzögerung
BTYP	Externe Buskonfiguration (BTYP0 bei Reset übernommen vom Port P0) 00 = 8bit Demux 01 = 8bit Mux 10 = 16bit Demux 11 = 16bit Mux
ALECTLx	Länge des ALE-Signals 0 = normal eine TCL (Vorgabe) 1 = verlängert um eine Taktzeit TCL
BUSACTx	Externer Bus 0 = gesperrt (Vorgabe) 1 = frei entsprechend ADDRSELx
RDYENx	Eingang READY für Busvergabe 0 = gesperrt (Vorgabe) 1 = frei
CSRENx	Bausteinauswahlsignal /CSx in Lesezyklen 0 = im ganzen Zyklus (Vorgabe) 1 = nur während aktivem Lesesignal
CSWENx	Bausteinauswahlsignal /CSx in Schreibzyklen 0 = im ganzen Zyklus (Vorgabe) 1 = nur während aktivem Schreibsignal

1.2.2.3 Port 0 beim Reset

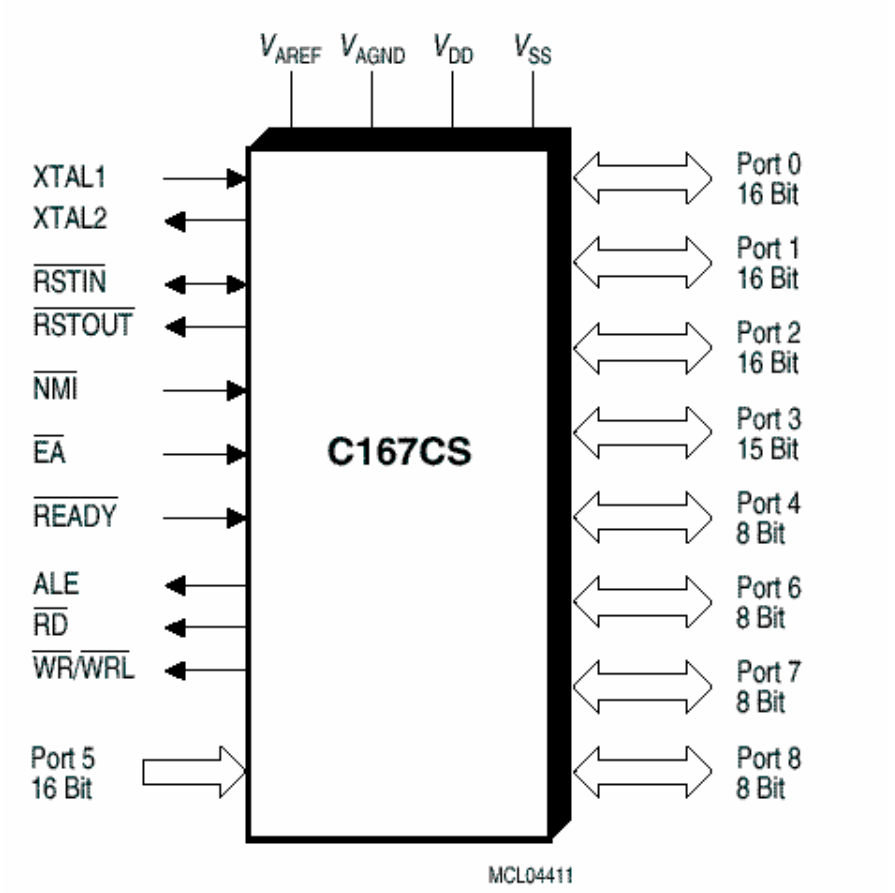
Eingänge des **Ports P0** beim Anlauf des Systems mit Reset:

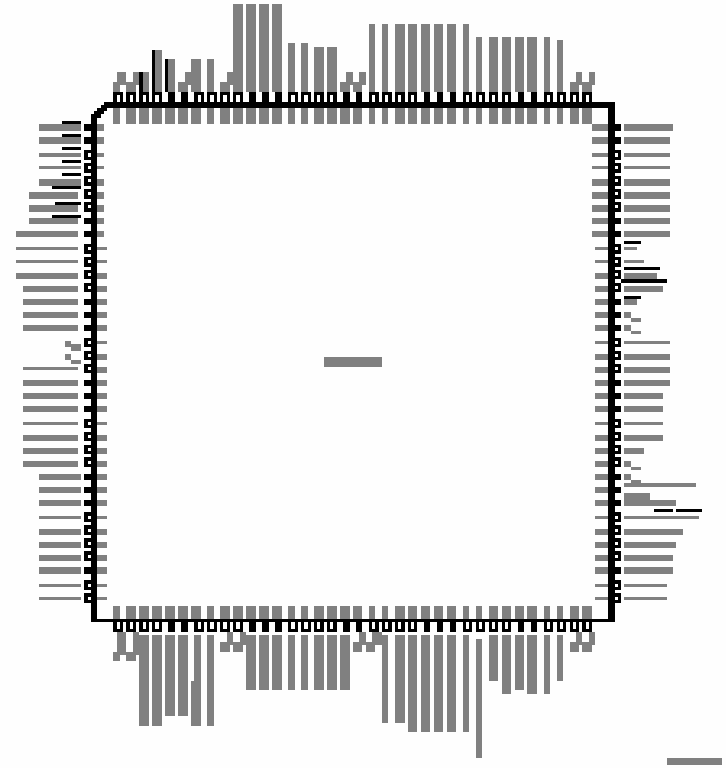
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLKCFG			SALSEL		CSSEL		WRC	BUSTYP		SMOD				ADP	EMU

Anschluß	Feld	Bedeutung	System Kap. 2.5
P0.0	EMU	Emulator Mode (nicht änderbar)	1 (aus)
P0.1	ADP	Adapt Mode (nicht änderbar)	1 (aus)
P0.5 *bis* P0.2	SMOD	Special Modes (nicht änderbar) enthält Bootstrap Loader Mode	1 1 1 1 Bootstrap aus normaler Start
P0.7 P0.6	BUSTYP	8/16 bit bzw. multiplex/demultiplex in Register BUSCON0 (änderbar)	0 0 mit Pull-Down 8-bit demultiplex
P0.8	WRC	Write Configuration (Schreibsignale) in Register RP0H angezeigt in Register SYSCON (invertiert) (änderbar)	1 /WR und /BHE
P0.10 P0.9	CSSEL	Chip Select Lines (Auswahlsignale) in Register RP0H angezeigt (nicht änderbar)	0 0 mit Pull-Down / CS0 /CS1 /CS2
P0.12 P0.11	SALSEL	Segment Address Lines (A16 bis A23) in Register RP0H angezeigt (nicht änderbar)	0 1 mit Pull Down 64 KB Speicher
P0.15 *bis* P0.13	CLKCFG	Clock Generation Control (Taktgenerator) in Register RP0H angezeigt (nicht änderbar)	0 1 1 mit Pull Down direkter Takt
Stift /EA	ROMEM	External Access Enable (externer Speicher) in Register SYSCON (änderbar)	fest auf Low (0) externes EPROM

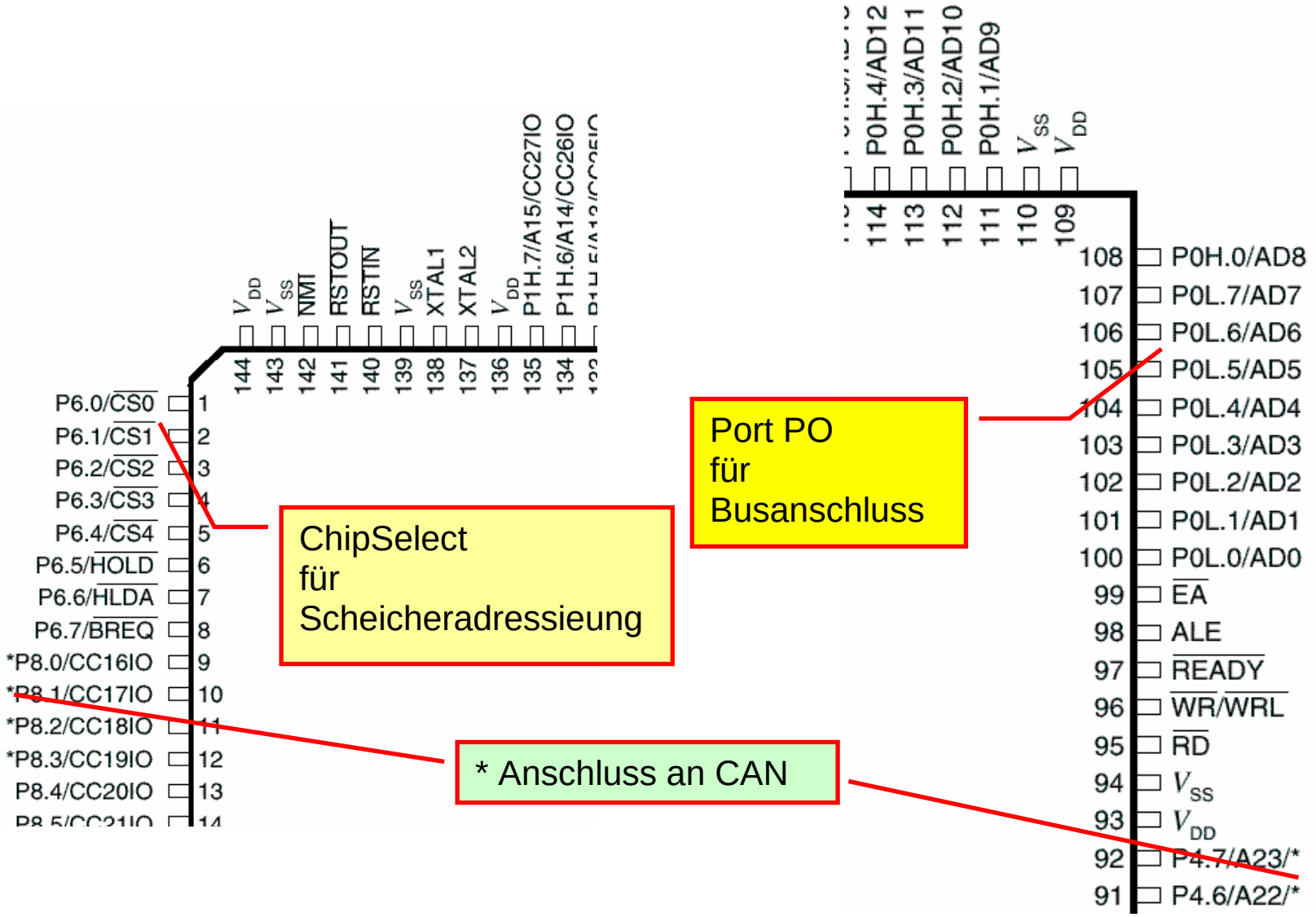
1.2.3 Initialisierung

1.2.3.1 Die Anschlüsse

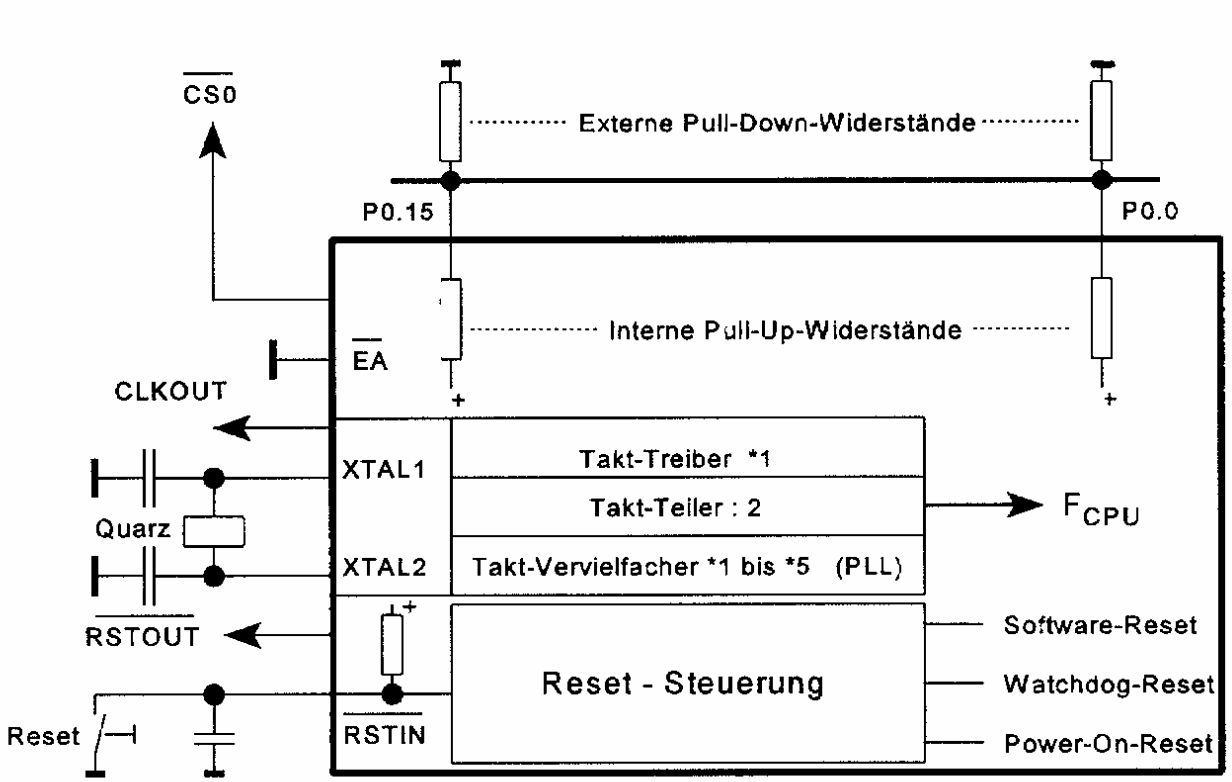




P-MFQP-144-6
(Plastic Metric Quad Flat Package

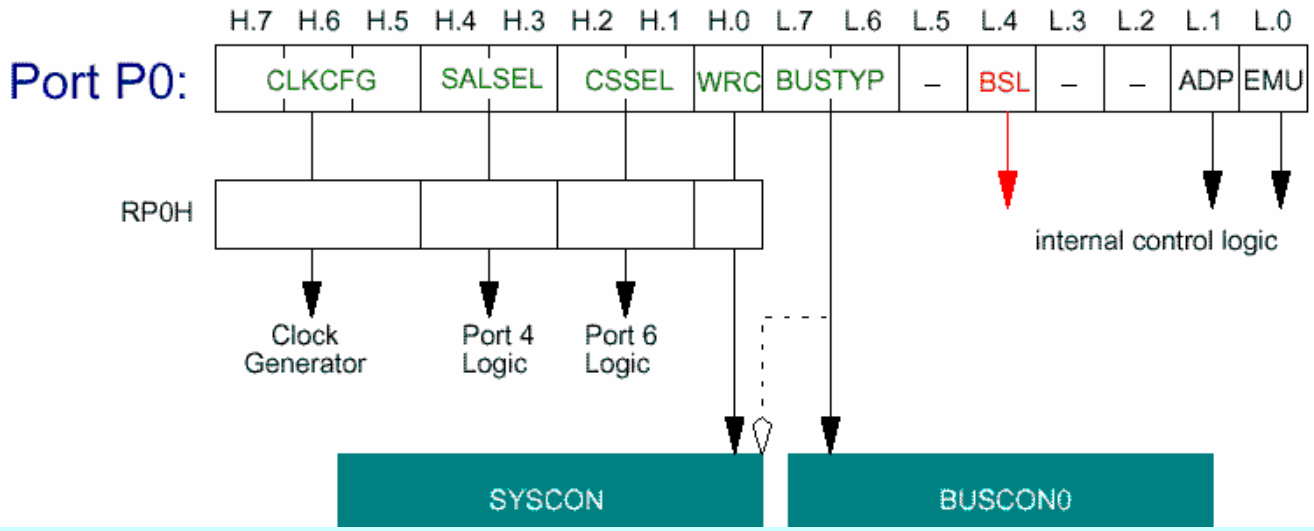

P6.0/CS0 1
P6.1/CS1 2
P6.2/CS2 3
P6.3/CS3 4
P6.4/CS4 5
P6.5/HOLD 6
P6.6/HLDA 7
P6.7/BREQ 8
*P8.0/CC16IO 9
*P8.1/CC17IO 10
*P8.2/CC18IO 11
*P8.3/CC19IO 12
P8.4/CC20IO 13
144 V_{DD}
143 V_{SS}
142 NMI
141 RSTOUT
140 RSTIN
139 V_{SS}
138 XTAL1
137 XTAL2
136 V_{DD}
135 P1H.7/A15/CC27IO
134 P1H.6/A14/CC26IO
114 P0H.4/AD12
113 P0H.3/AD11
112 P0H.2/AD10
111 P0H.1/AD9
110 V_{SS}
109 V_{DD}
108 P0H.0/AD8
107 P0L.7/AD7
106 P0L.6/AD6
105 P0L.5/AD5
104 P0L.4/AD4
103 P0L.3/AD3
102 P0L.2/AD2
101 P0L.1/AD1
100 P0L.0/AD0
99 EA
98 ALE
97 READY
96 WR/WRL
95 RD
94 V_{SS}
93 V_{DD}
92 P4.7/A23/*
91 P4.6/A22/*
ChipSelect für Scheicheradressieung
Port PO für Busanschluss
* Anschluss an CAN

1.2.3.2 Die Takt- und Reset-Steuerung



Wenn ***/*EA** auf *high* (d.h. unbeschaltet) ist, dann wird das Programm aus dem internen Programmspeicher (PROM oder EPROM) gestartet

Einstellen der Systemkonfiguration über Port P0!



SYSCON
System Control Register **SFR ($FF12_H/89_H$)** **Reset Value: $0XX0_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STKSZ			ROM S1	SGT DIS	ROM EN	BYT DIS	CLK EN	WR CFG	CS CFG	-	OW D DIS	BD RST EN	XP EN	VISI BLE	XPER-SHARE
rw			rw	rw	rwh	rwh	rw	rwh	rw	-	rwh	rw	rw	rw	rw

Das Ändern von SYSCON (System Configuration Register) nach der Ausführung des Befehls *EINIT* nicht mehr möglich

1.2.4 Die Speicherorganisation

Gemeinsamer Adressraum für Daten, Programme und I/O von maximal 16 MByte
(segmentierter Mode – unsegmentierter Mode)

256 Segmente mit je 64 KByte und 1024 Pages von je 16 KByte

Code-Segement:
Code Segement Pointer (CSP) und **Instruction Pointer** (IP)

16-Bit Datenadresse:
14 Bit **Page Offset** (POF) und
2 Bit für die Auswahl eines der vier **Data Page Pointers** (DPP)

1.2.4.1 Segmentierter Adressraum

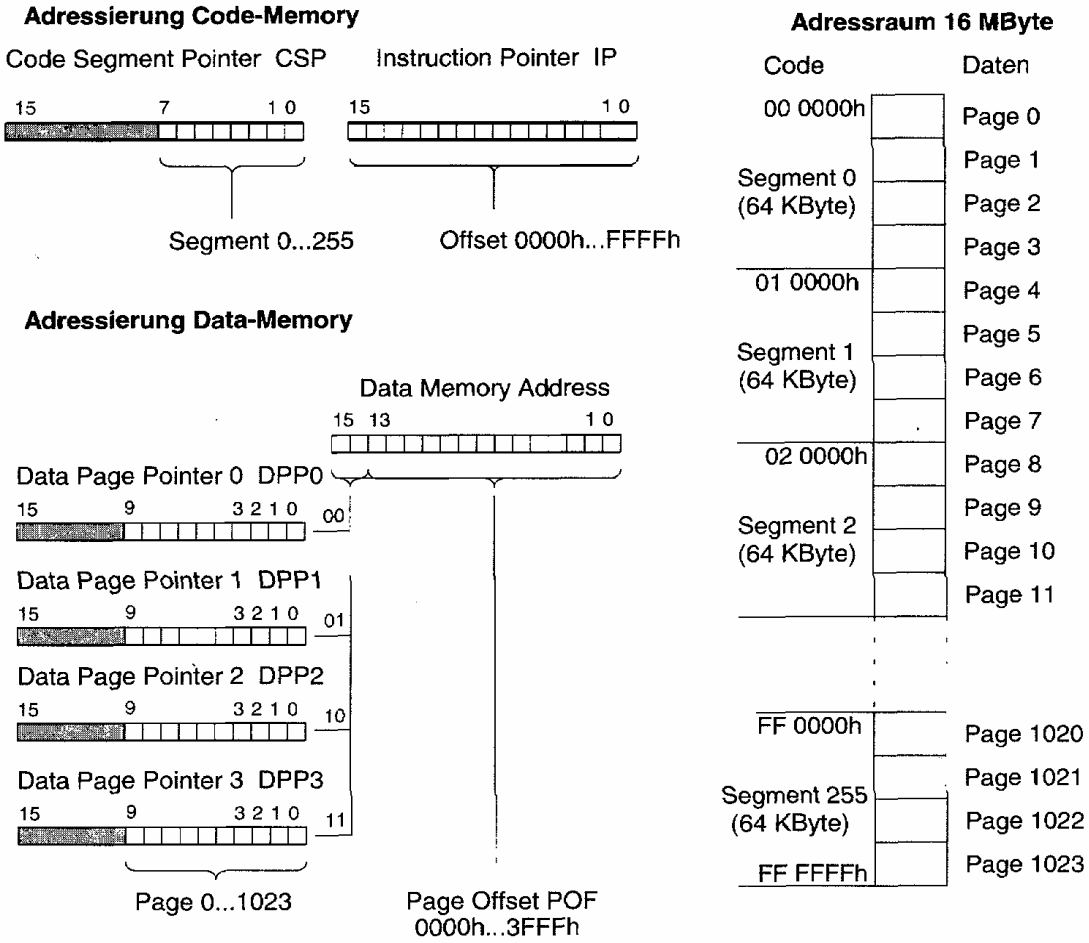


Abb. 2.2: Segmentierter Adressraum

1.2.4.2 Unsegmentierter Adressraum

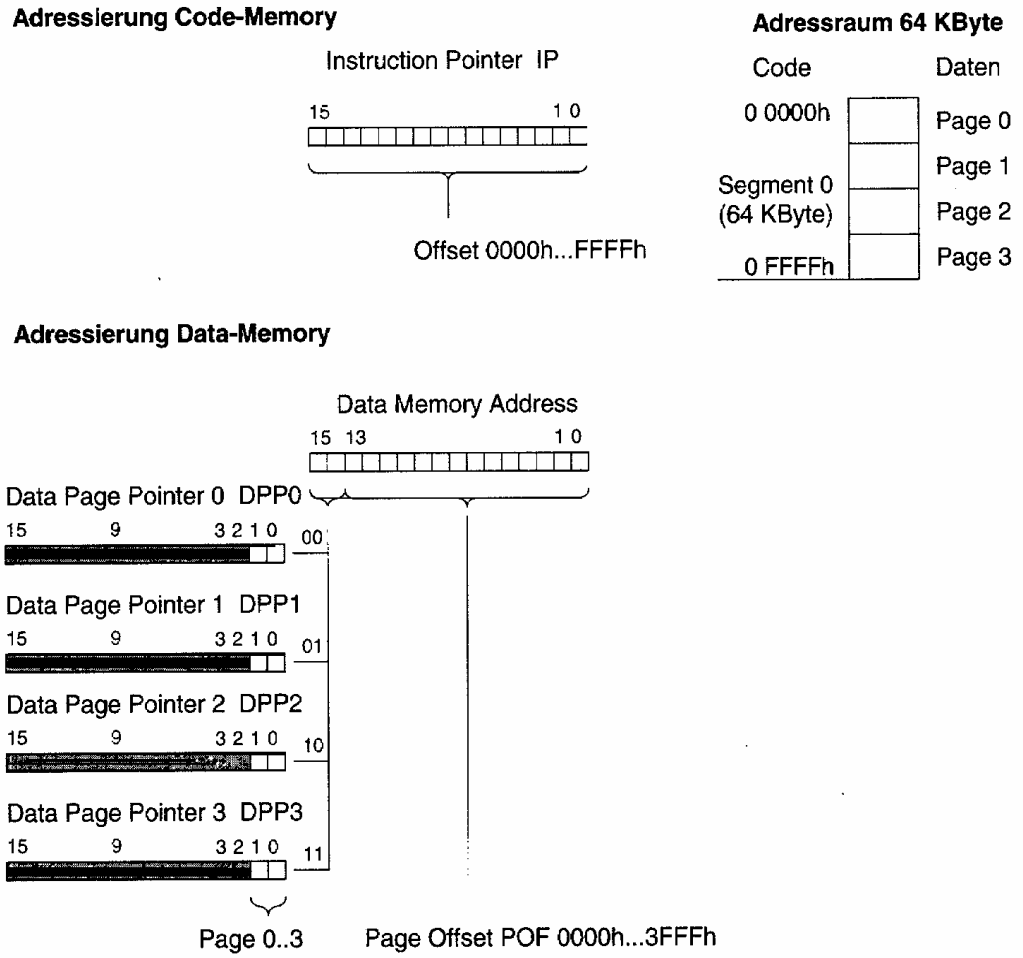


Abb. 2.3: Unsegmentierter Adressraum

1.2.4.3 Die Organisation des Speichersegmentes 0

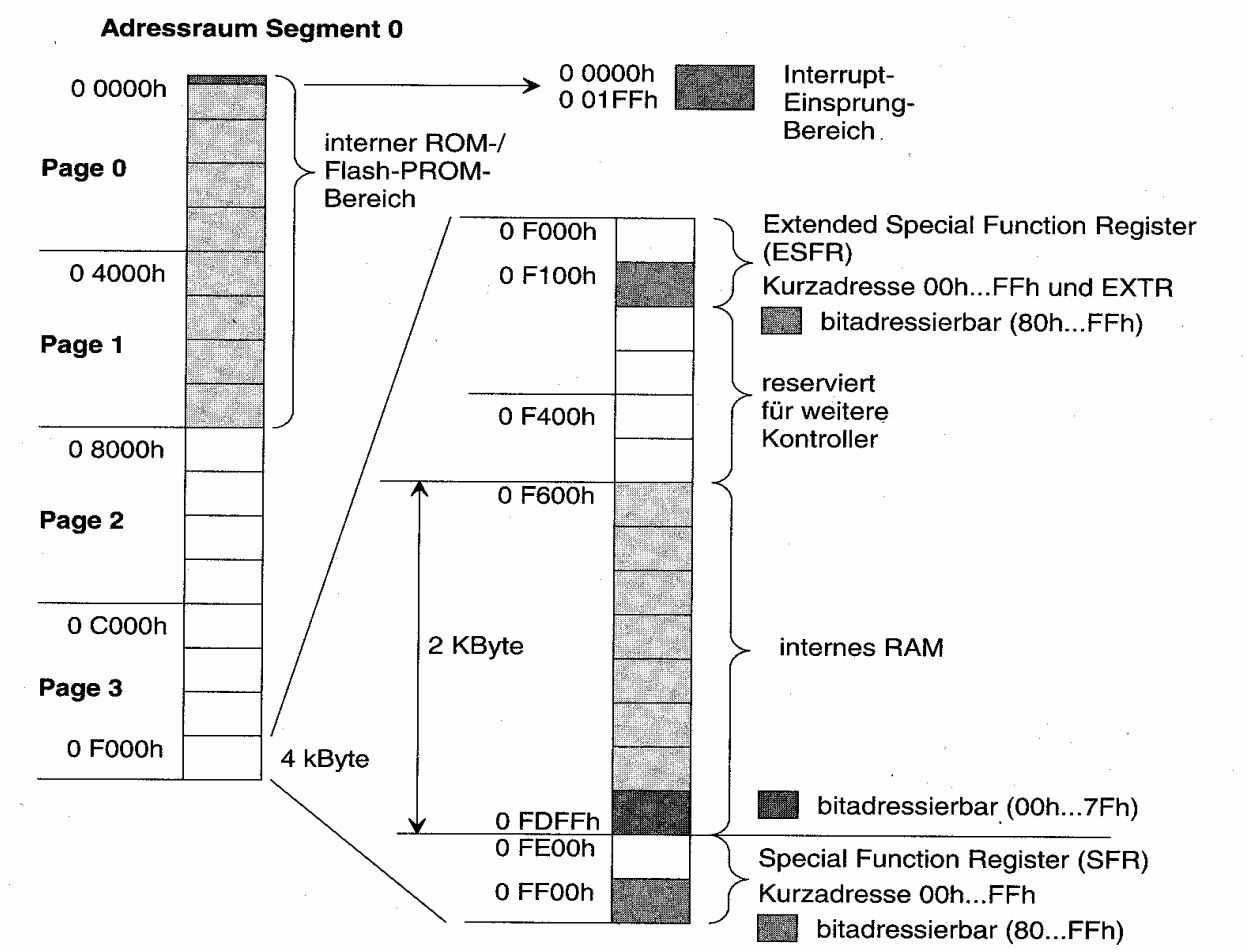


Abb. 2.4: Das Speichersegment 0

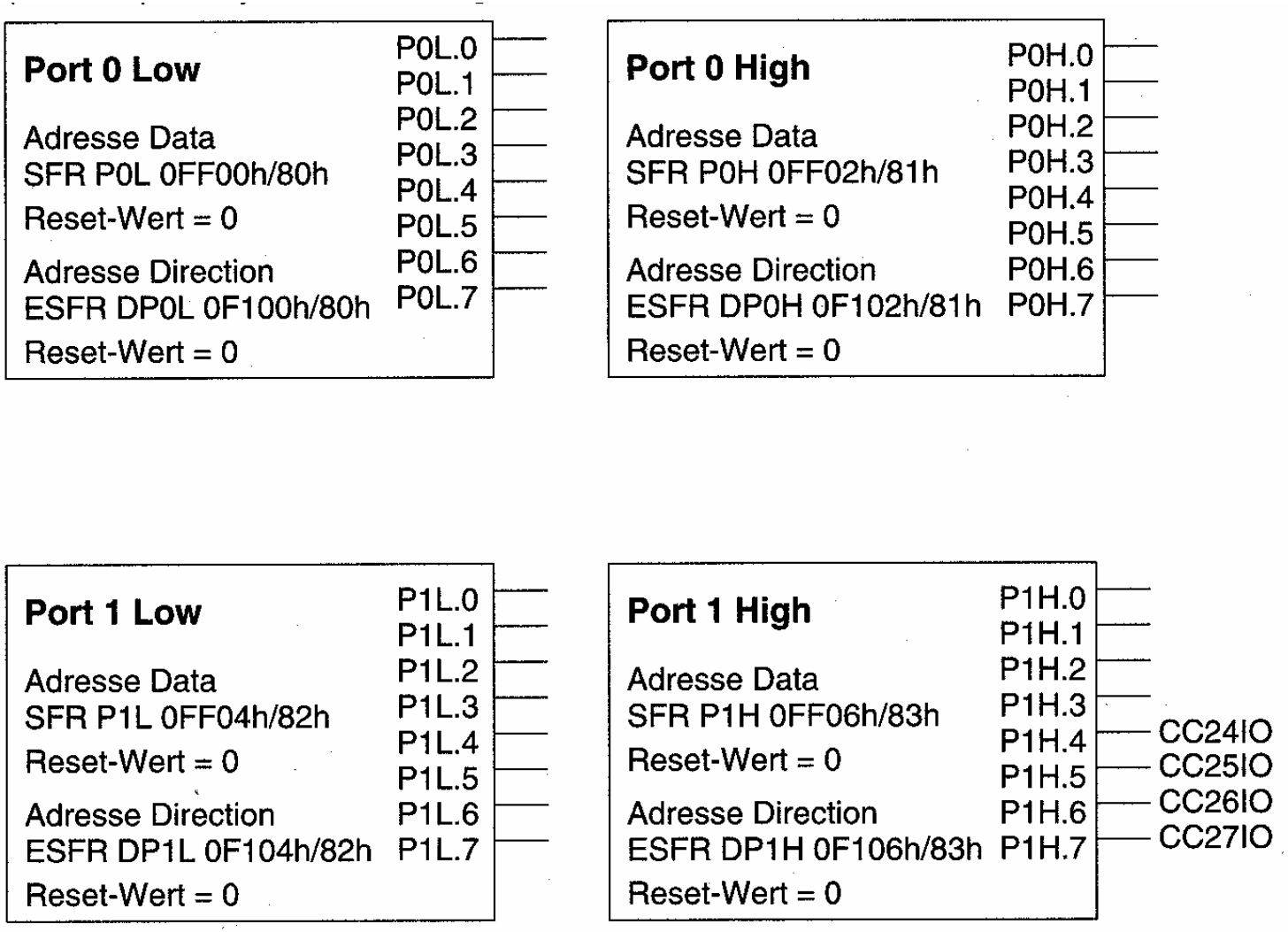
1.2.5 Die Ports

9 Ports mit 111 Bit (Input/Output)
- Input oder Output Port mit Push-Pull Treiber (Port 0,1 und 4)
- Input Port (Port 5)
- Input oder Output Port mit Push-Pull- oder Open-Drain-Treiber

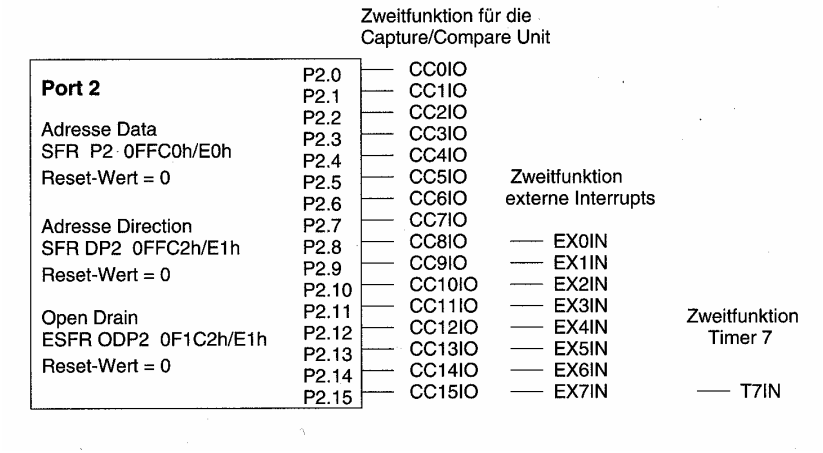
Zweitfunktionen:

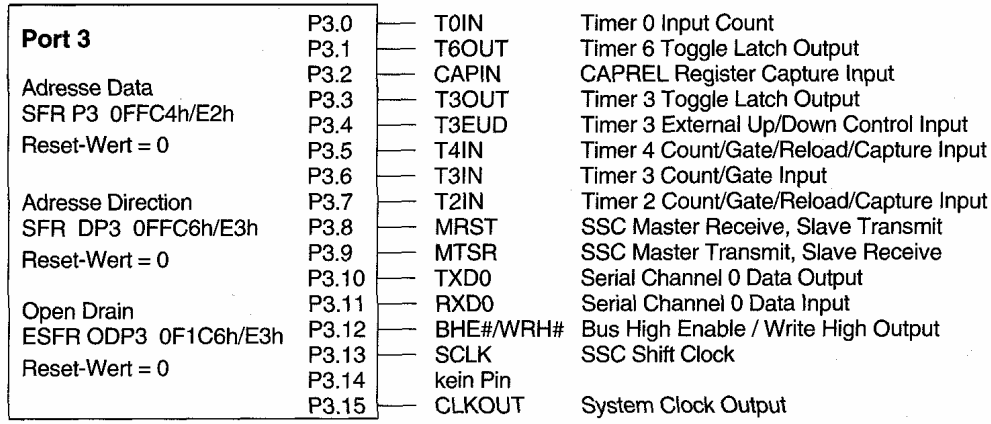
Ports 0 und 1	Bei erweiterten Systemen werden die Ports 0 und 1 als Daten- und Adressbus verwendet.
Port 4	Im segmentierten Betrieb werden über Port 4 die Adressen A16....A23 ausgegeben.
Port 6	Port 6 wird für die Chip-Select-Signale verwendet.
Port 2, 7 und 8	Diese Ports werden im Zusammenhang mit den Funktionen Compare/ Capture und Pulsweitenmodulation eingesetzt.
Port 3	Port 3 enthält Signale für die Steuerung der Timer, der seriellen Schnittstellen und die Bus-Steuer-Signale
Port 5	Port 5 wird vom Analog/Digital-Wandler benutzt

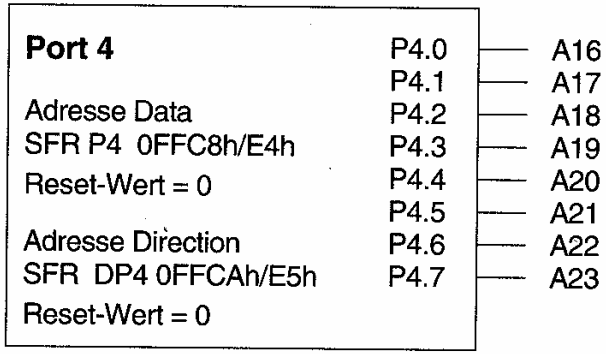
Bei erweiterten Systemen wird Port 0 (und im non-multiplexed-Betrieb auch Port1) als Bus verwendet

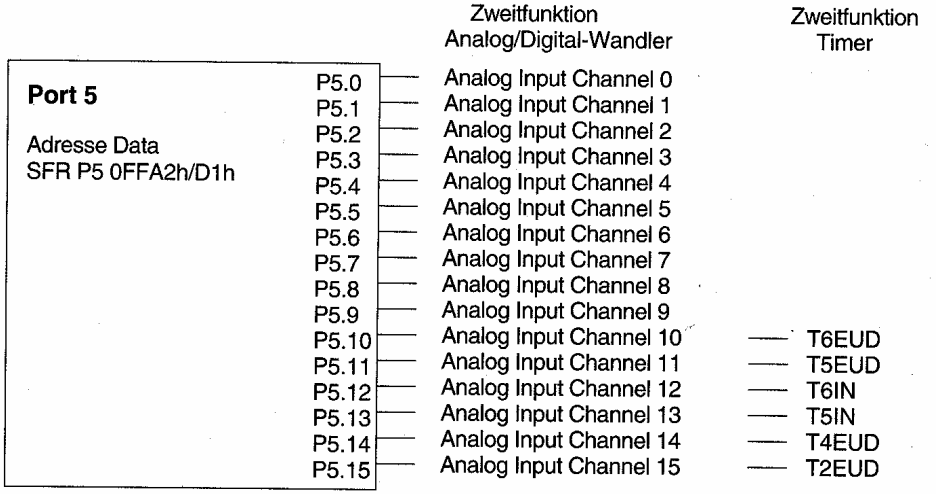


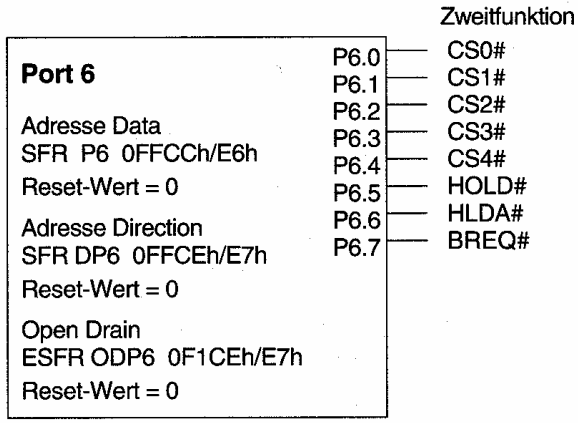
Embedded Systems

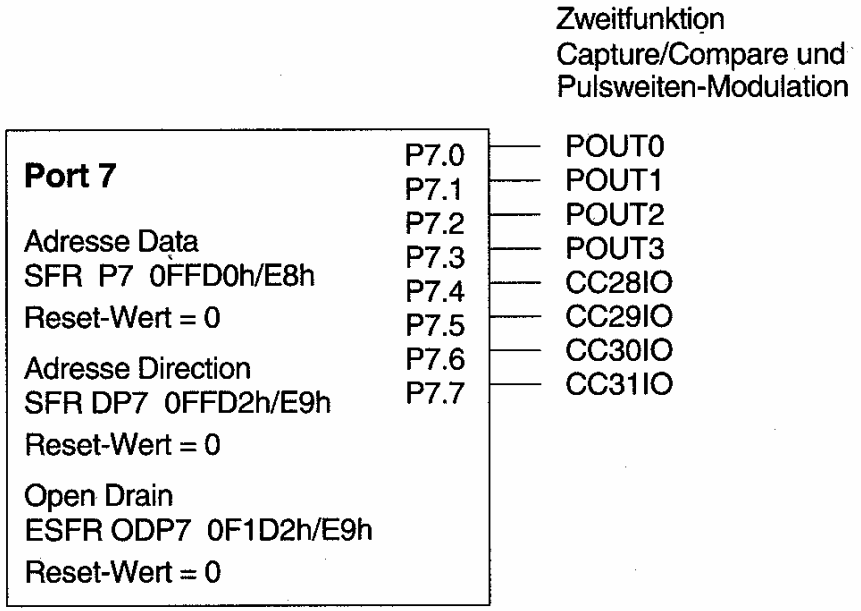


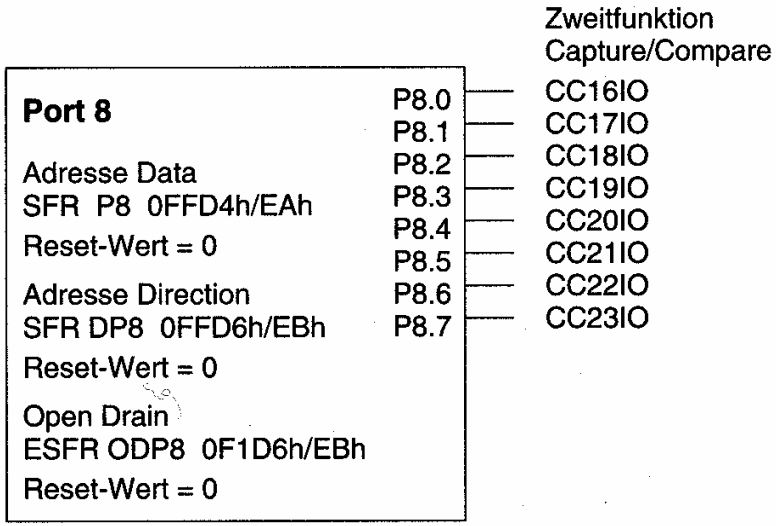







Zweitfunktion
Port 6
Adresse Data
SFR P6 0FFCCh/E6h
Reset-Wert = 0
Adresse Direction
SFR DP6 0FFCEh/E7h
Reset-Wert = 0
Open Drain
ESFR ODP6 0F1CEh/E7h
Reset-Wert = 0
P6.0 CS0#
P6.1 CS1#
P6.2 CS2#
P6.3 CS3#
P6.4 CS4#
P6.5 HOLD#
P6.6 HLDA#
P6.7 BREQ#


Zweitfunktion
Capture/Compare und
Pulsweiten-Modulation
Port 7
Adresse Data
SFR P7 0FFD0h/E8h
Reset-Wert = 0
Adresse Direction
SFR DP7 0FFD2h/E9h
Reset-Wert = 0
Open Drain
ESFR ODP7 0F1D2h/E9h
Reset-Wert = 0
P7.0 POUT0
P7.1 POUT1
P7.2 POUT2
P7.3 POUT3
P7.4 CC28IO
P7.5 CC29IO
P7.6 CC30IO
P7.7 CC31IO


Zweitfunktion
Capture/Compare
Port 8
Adresse Data
SFR P8 0FFD4h/EAh
Reset-Wert = 0
Adresse Direction
SFR DP8 0FFD6h/EBh
Reset-Wert = 0
Open Drain
ESFR ODP8 0F1D6h/EBh
Reset-Wert = 0
P8.0 CC16IO
P8.1 CC17IO
P8.2 CC18IO
P8.3 CC19IO
P8.4 CC20IO
P8.5 CC21IO
P8.6 CC22IO
P8.7 CC23IO

Adressierung und Initialisierung über **S**pezial **F**unction **R**egister (SFR)

bitadressierbar (Px**.**y bzw. Px^y)

Datenregister **P**x (Port) zur Aufnahme der Daten (Ausgabe / Eingabe)

Richtungsregister **DP**x (Direction Port) legen für jeden Anschluss die Richtung fest

Treiberregister **ODP**x (Open Drain Port) bestimmen die Ausgangsschaltung

Das Kontrollregister **PICON** (Port Input Control Register) legt die Eingangspegel für die Ports P2, P3, P7 und P8 fest.

PICON RAM: 0F1C4H **ESFR:** 0E2H **Reset:** --00H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								P8L IN	P7L IN	-	-	P3H IN	P3L IN	P2H IN	P2L IN

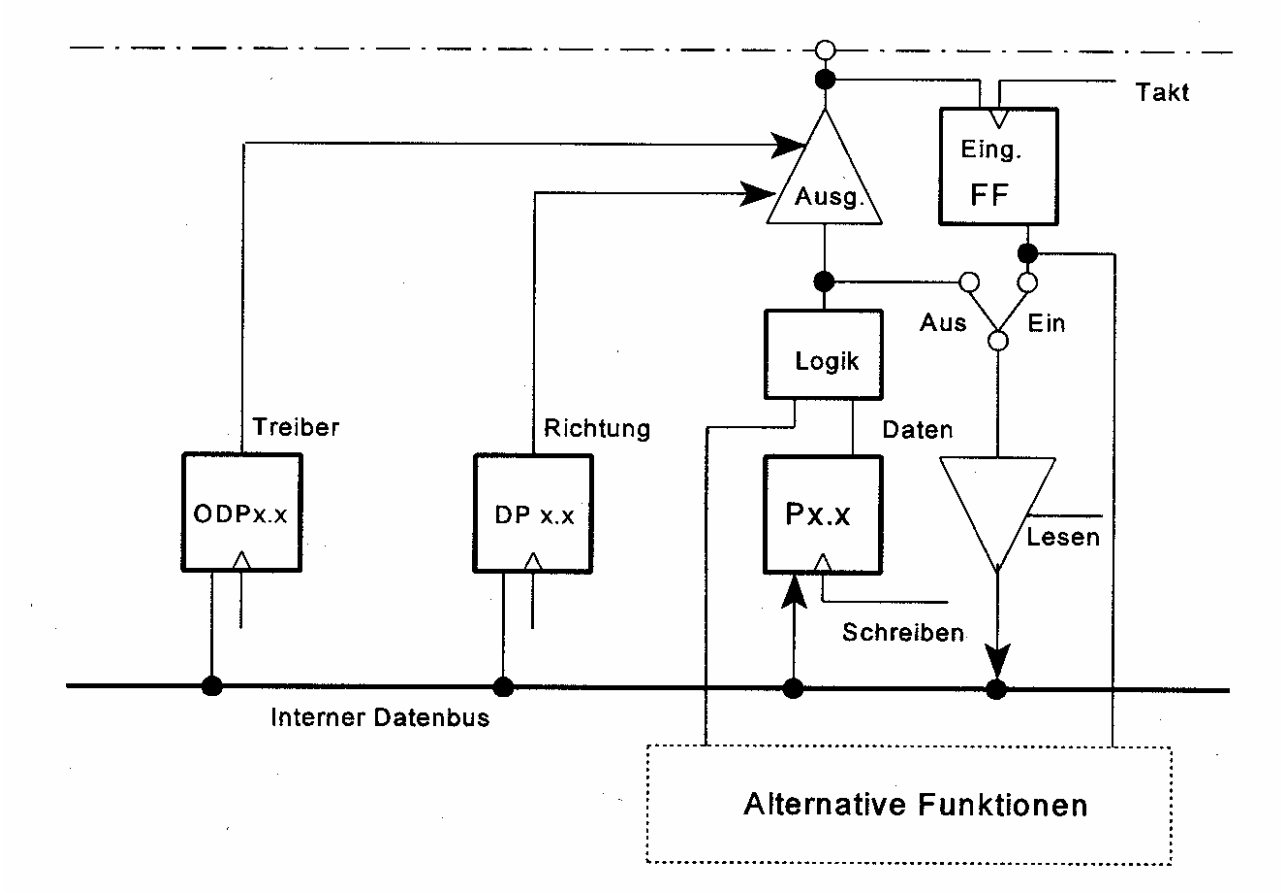
PxLIN Eingangspegel der Low-Bits (Px.7 bis Px.0): 0 = TTL 1 = CMOS

PxHIN Eingangspegel der High-Bits (Px.15 bis Px.8): 0 = TTL 1 = CMOS

Embedded Systems

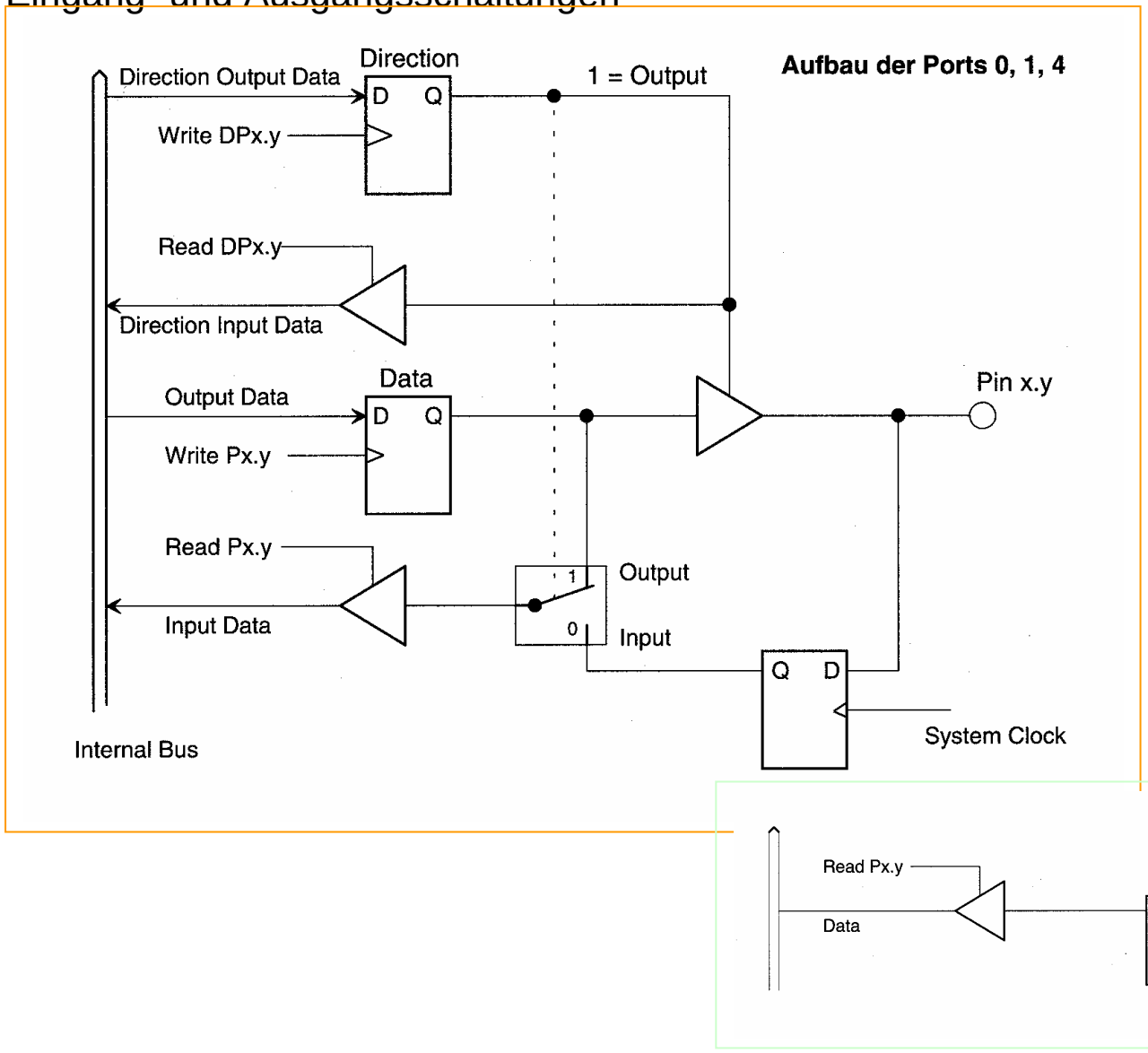
Port	Länge	RAM-A.	SFR-Adr.	Anwendung als Port	Alternativ
P0L **DP0L**	8bit	0FF00H 0F100H	80H SFR 80H ESFR	Datenregister Richtungsregister 0=ein 1=aus	Datenbus/ Adr./Datenb.
P0H **DP0H**	8bit	0FF02H 0F102H	81H SFR 81H ESFR	Datenregister Richtungsregister 0=ein 1=aus	Datenbus/ Adr./Datenb.
P1L **DP1L**	8bit	0FF04H 0F104H	82H SFR 82H ESFR	Datenregister Richtungsregister 0=ein 1=aus	Adreßbus A0 - A7
P1H **DP1H**	8bit	0FF06H 0F106H	83H SFR 83H ESFR	Datenregister Richtungsregister 0=ein 1=aus	Adreßbus A8 - A15
P2 **DP2** **ODP2**	16bit	0FFC0H 0FFC2H 0F1C2H	0E0H SFR 0E1H SFR 0E1H ESFR	Datenregister Richtungsregister 0=ein 1=aus 0 = Push/Pull 1 = Open Drain	CAPCOM- Timer Interrupt
P3 **DP3** **ODP3**	16bit	0FFC4H 0FFC6H 0F1C6H	0E2H SFR 0E3H SFR 0E3H ESFR	Datenregister Richtungsregister 0=ein 1=aus 0 = Push/Pull 1 = Open Drain	Timer und serielle Schn. Takt BHE
P4 **DP4**	8bit	0FFC8H 0FFCAH	0E4H SFR 0E5H SFR	Datenregister Richtungsregister 0=ein 1=aus	Adreßbus A16 - A23
P5	16bit	0FFA2H	0D1H SFR	digitales Datenregister nur lesen	Analogeing.
P6 **DP6** **ODP6**	8bit	0FFCCH 0FFCEH 0F1CEH	0E6H SFR 0E7H SFR 0E7H ESFR	Datenregister Richtungsregister 0=ein 1=aus 0 = Push/Pull 1 = Open Drain	/CS0 ... /CS4 und Busvergabe
P7 **DP7** **ODP7**	8bit	0FFD0H 0FFD2H 0F1D2H	0E8H SFR 0E9H SFR 0E9H ESFR	Datenregister Richtungsregister 0=ein 1=aus 0 = Push/Pull 1 = Open Drain	PWM- Timer Interrupt
P8 **DP8** **ODP8**	8bit	0FFD4H 0FFD6H 0F1D6H	0EAH SFR 0EBH SFR 0EBH ESFR	Datenregister Richtungsregister 0=ein 1=aus 0 = Push/Pull 1 = Open Drain	CAPCOM- Timer Interrupt

Allgemeines Modell einesPortanschlusses



Embedded Systems

Eingang- und Ausgangsschaltungen



Achtung: 4 stufige Pipeline

Fetch – Decode – Execute – Write Back

Falsch:

```
BSET    DP2.4    ; Pin 2.4 als Output definieren
BSET    P2.4     ; Pin 2.4 steht noch auf INPUT (Pipeline!!)
```

Richtig:

```
BSET    DP2.4    ;
NOP              ; Wartebefehl
BSET    P2.4     ; Jetzt ist Pin 2.4 OUTPUT-Pin
```

1.2.6 Der Analog/Digitalwandler

- 24 Eingänge über Multiplexer an A/D-Wandler
  (16 an Port 5 und 8 an Port 1L (im ADCON Bit ADX auf H!))
- Auflösung: 10 Bit
- Vier Betriebsarten:
  - einmalige Wandlung an einem Eingangskanal
  - fortlaufende Wandlung an einem Eingangskanal
  - einmalige Wandlung mehrerer Kanäle
  - fortlaufende Wandlung mehrerer Kanäle
- Channel Injection Modus
- Wait for Read Control

aus [11]

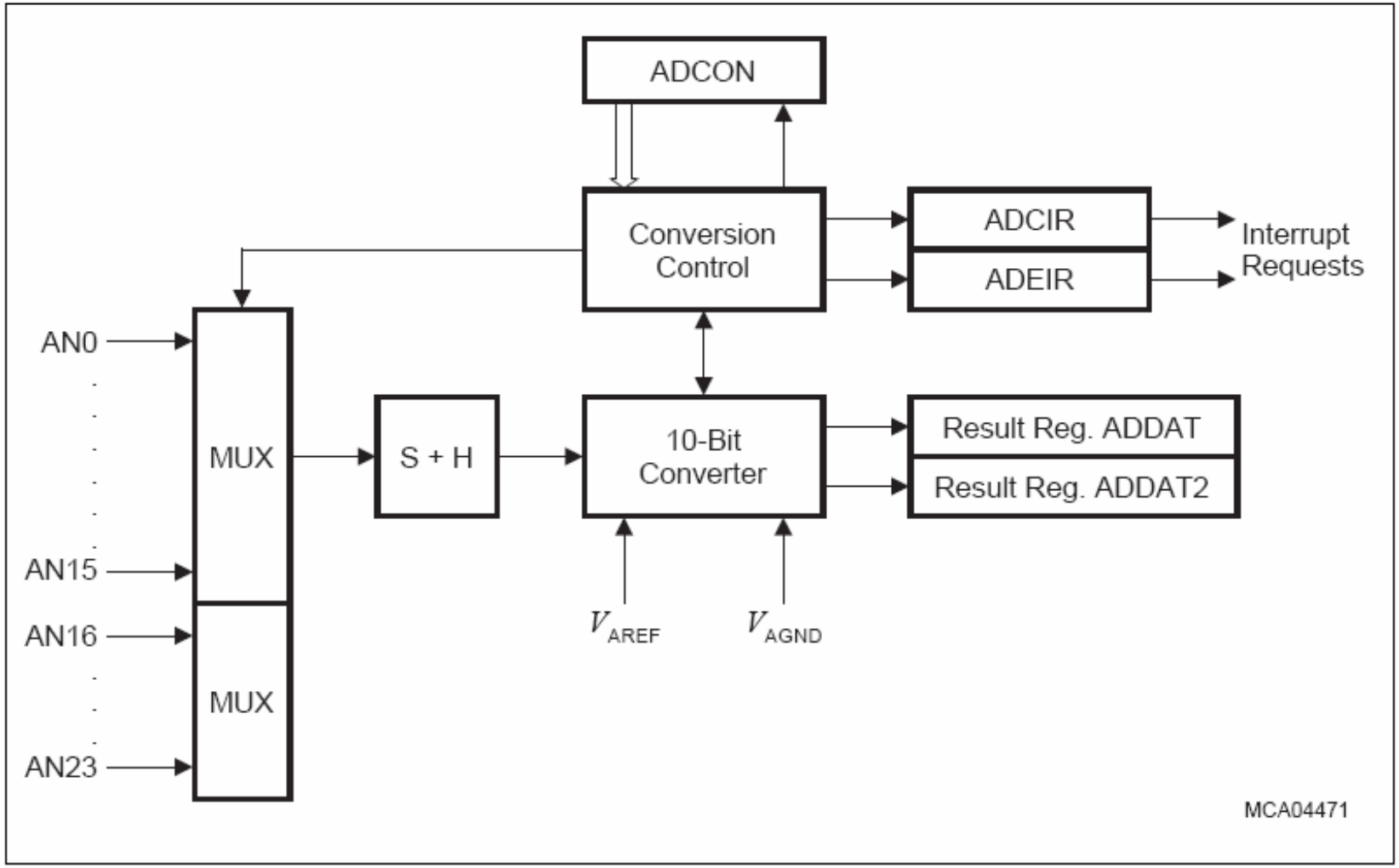


Figure 18-2 Analog/Digital Converter Block Diagram

aus [11]

A set of SFRs and port pins provide access to control functions and results of the ADC.

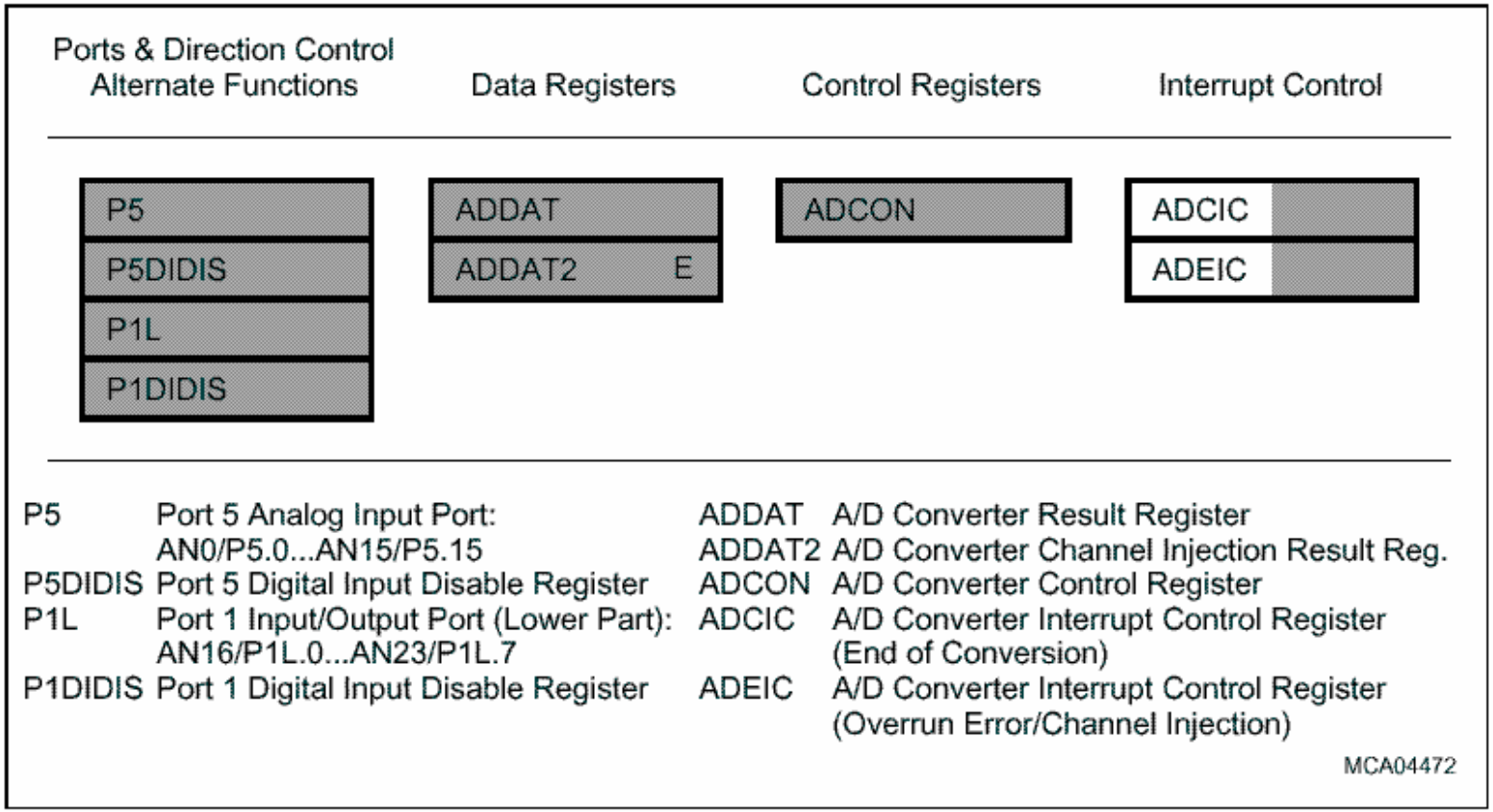
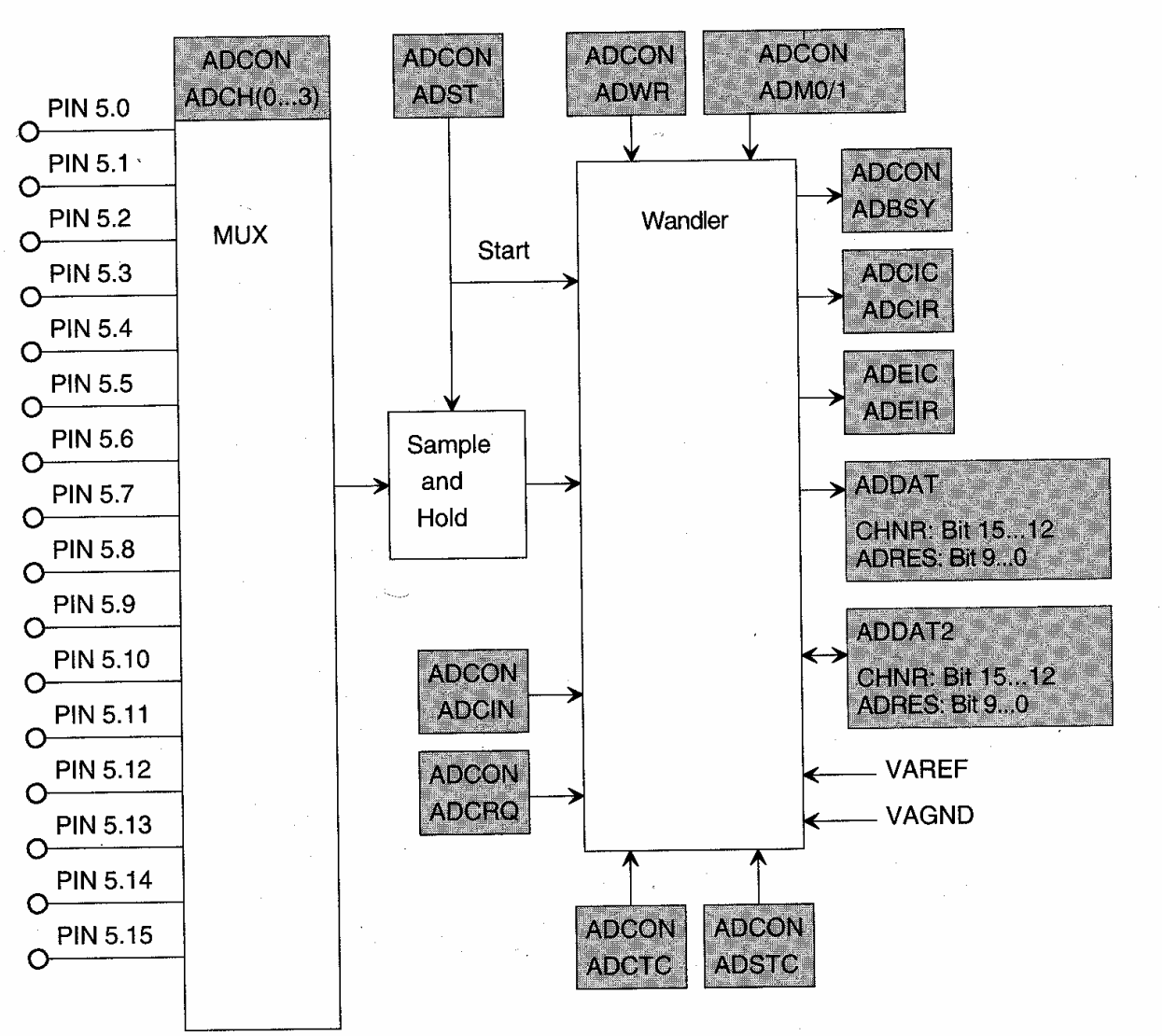


Figure 18-1 SFRs and Port Pins Associated with the A/D Converter

Embedded Systems

Aufbau des A/D-Wandlers (aus [12])



Das Register **ADCON** (A/D Converter Control) (aus [13])

ADX
0: Kanal 0 …15
1: Kanal 16….23

ADCON **RAM:** 0FFA0H **SFR:** 0D0H **Reset:** 0000H *bitadressierbar*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCTC		ADSTC		AD CRQ	AD CIN	AD WR	AD BSY	AD ST		ADM		ADCH			

ADCH Kanalauswahl bzw. Startkanal von 0000 (AN0) bis 1111 (AN15)

ADM Betriebsart 00: ein Kanal, einmalige Messung
01: ein Kanal, fortlaufende Messungen
10: mehrere Kanäle, einmalige Messung (von ADCH bis 0000)
11: mehrere Kanäle, fortlaufende Messungen (ADCH bis 0000)

ADST Start der Wandlung durch Setzen auf **1** durch das Programm

ADBSY wird während der Wandlung von der Steuerung auf **1** gesetzt

ADWR warten bis Daten gelesen (0 = nein, 1 = ja)

ADCIN Einfügen einer Einzelmessung freigeben (0 = gesperrt, 1 = frei)

ADCRQ Einzelmessung einfügen (1 = ja) Programm oder CAPCOM Kanal CC31

ADSTC Sample-Zeit (00

ADCTC Wandlungszeit (

ADC Sample Time Control

00: $t_{BC} \times 8$
01: $t_{BC} \times 16$
10: $t_{BC} \times 32$
11: $t_{BC} \times 64$

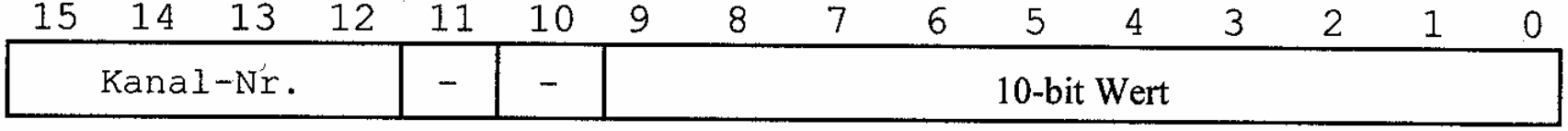
ADC Conversion Time Control

00: $f_{BC} = f_{CPU} / 4$
01: $f_{BC} = f_{CPU} / 2$
10: $f_{BC} = f_{CPU} / 16$
11: $f_{BC} = f_{CPU} / 8$

Die Datenregister (**ADDAT** und **ADDAT2**) aus [13]:

Das Register **ADDAT** enthält nach einer normalen Wandlung den gewandelten Wert und die Nr. des gemessenen Kanals.

ADDAT **RAM:** 0FEA0H **SFR:** 050H **Reset:** 0000H



Das Register **ADDAT2** enthält nach einer *eingefügten* Wandlung den gewandelten Wert. ***Vor*** der Anforderung ist die Nr. des einzufügenden Kanals einzutragen.

ADDAT2 **RAM:** 0F0A0H **ESFR:** 050H **Reset:** 0000H

Das Interrupt-Steuerregister **ADCIC** (aus [13]

ADCIC RAM: 0FF98H **SFR:** 0CCH **Reset:** 0000H *bitadressierbar*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								**ADC IR**	**ADC IE**	ILVL				GLVL	

GLVL	Gruppen-Ebene von 00 (niedrigster) bis 11 (höchster) Rang in der Gruppe
ILVL	Interrupt-Ebene von 0000 (gesperrt) bis 1111 (höchste Priorität)
ADCIE	Freigabe des Interrupts (0 = gesperrt, 1 = frei)
ADCIR	Ende der Wandlung (1 = aufgetreten)

Das Interrupt-Fehlerregister **ADEIC**

ADEIC RAM: 0FF9AH **SFR:** 0CDH **Reset:** 0000H *bitadressierbar*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								**ADE IR**	**ADE IE**	ILVL				GLVL	

GLVL	Gruppen-Ebene von 00 (niedrigster) bis 11 (höchster) Rang in der Gruppe
ILVL	Interrupt-Ebene von 0000 (gesperrt) bis 1111 (höchste Priorität)
ADEIE	Freigabe des Fehler-Interrupts (0 = gesperrt, 1 = frei)
ADEIR	Überlauffehler (1 = aufgetreten)

1.2.7 Das Interrupt-System

Es werden folgende Arten von Interrupts unterschieden:

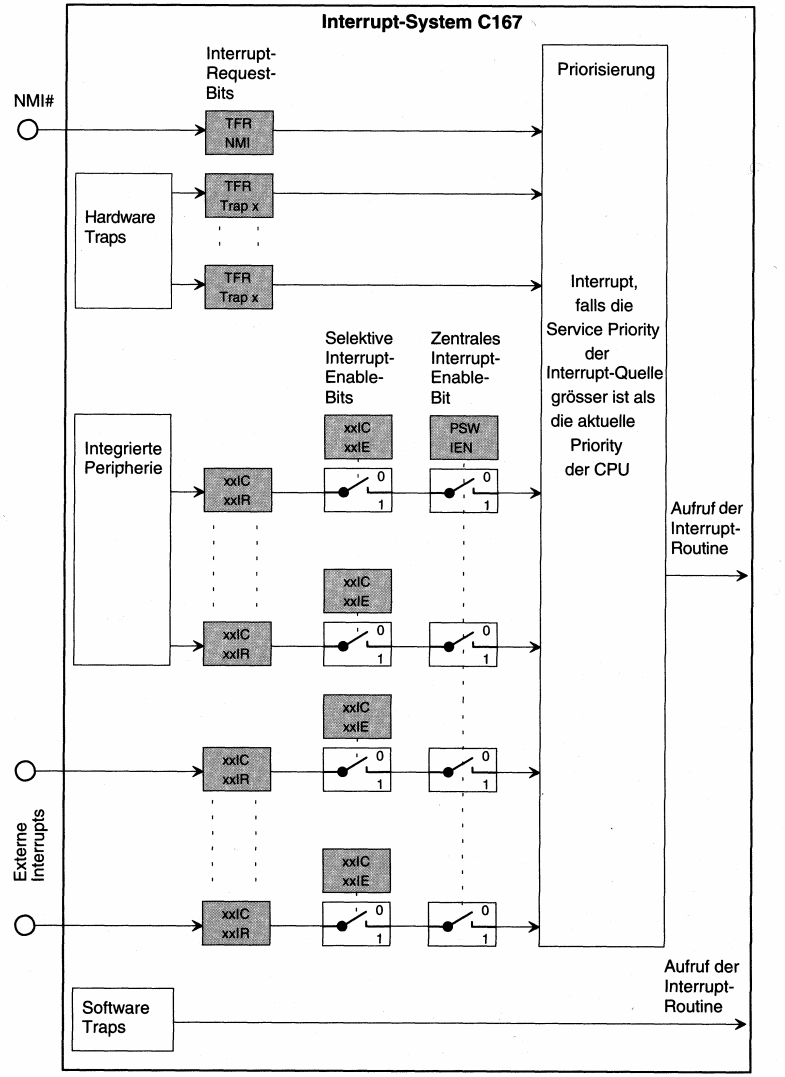
- **Non Maskable Interrupt (NMI)**
  (nicht maskierbar, ausgelöst durch negative Flanke an Pin NMI)

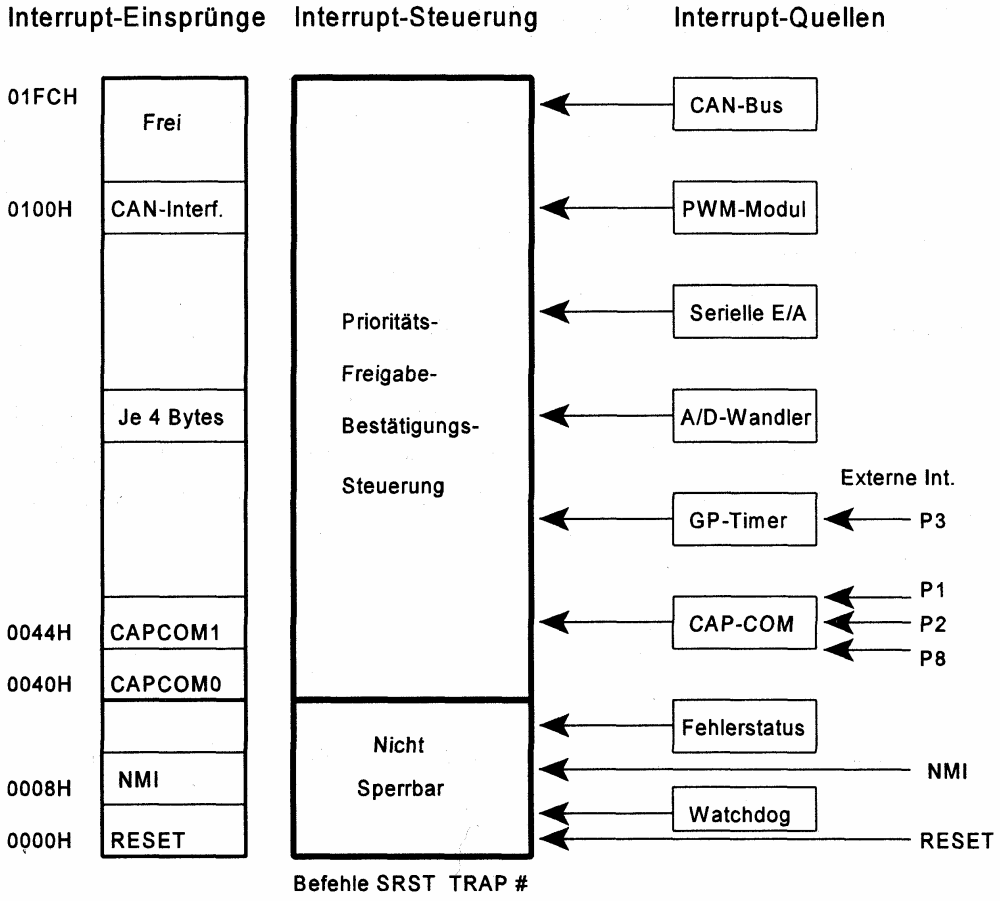
- **Hardware Traps**
  (nicht maskierbar, ausgelöst durch "Exceptions" wie z.B. "Stack Overflow", "Illegal Instruction",....)

- **Hardware Interrupts der integrierten Peripherie**
  (maskierbar, ausgelöst durch integrierte Peripherie, unterschiedliche Priorität)

- **Externe Hardware Interrupts**
  (maskierbar, ausgelöst durch positiv oder negative Flanken an Interrupt Pins, unterschiedliche Priorität)

- **Software Traps**
  (nicht sperrbar, ausgelöst mit dem Befehl TRAP #Interruptnummer)


Interrupt-System C167
Interrupt-
Request-
Bits
Priorisierung
NMI#
TFR
NMI
Hardware
Traps
TFR
Trap x
TFR
Trap x
Interrupt,
falls die
Service Priority
der
Interrupt-Quelle
grösser ist als
die aktuelle
Priority
der CPU
Selektive
Interrupt-
Enable-
Bits
Zentrales
Interrupt-
Enable-
Bit
Integrierte
Peripherie
xxIC
xxIE
PSW
IEN
xxIC
xxIR
Aufruf der
Interrupt-
Routine
Externe
Interrupts
Software
Traps
Aufruf der
Interrupt-
Routine


Interrupt-Einsprünge
Interrupt-Steuerung
Interrupt-Quellen
01FCH
Frei
0100H
CAN-Interf.
Je 4 Bytes
0044H
CAPCOM1
0040H
CAPCOM0
0008H
NMI
0000H
RESET
Prioritäts-
Freigabe-
Bestätigungs-
Steuerung
Nicht
Sperrbar
Befehle SRST TRAP #
CAN-Bus
PWM-Modul
Serielle E/A
A/D-Wandler
Externe Int.
GP-Timer
P3
P1
P2
P8
CAP-COM
Fehlerstatus
NMI
Watchdog
RESET

1.2.7.1 Prioritätsstruktur

Trap-Bedingungen	Trap-Request-Flag	Vektor-Adresse	Trap-Nummer	Symb. Name des Vektors	Trap-Priorität
Reset-Funktionen:					
– Hardware-Reset	–	00000H	00H	RESET	III
– Software-Reset	–	00000H	00H	RESET	III
– Watchdog-Reset	–	00000H	00H	RESET	III
HW-Traps der Klasse A:					
– Nicht maskierb. Interrupt	NMI	00008H	02H	NMITRAP	II
– Stack-Überlauf	STKOF	00010H	04H	STOTRAP	II
– Stack-Unterlauf	STKUF	00018H	06H	STUTRAP	II
HW-Traps der Klasse B:					
– Undefinierter Opcode	UNDOPC	00028H	0AH	BTRAP	I
– Fehler bei geschütztem Befehl	PRTFLT	00028H	0AH	BTRAP	I
– Falscher Wort-Zugriff	ILLOPA	00028H	0AH	BTRAP	I
– Falscher Befehlszugriff	ILLINA	00028H	0AH	BTRAP	I
– Verbotener Zugriff auf externen Bus	ILLBUS	00028H	0AH	BTRAP	I
Software-Traps: Befehl TRAP	–	Alle; ergibt sich aus Operand #trap7	Alle (00H bis 7FH)	–	Aktuelle CPU-Priorität

Request Priority

Die Prioritäten sind fest zugeordnet (für Reset und HW-Traps, s.u.) oder durch den Interrupt Level (ILVL = 0...15) und durch den Group Level /GLVL = 0...3) im zuständigen Interrupt Control Register (xxIC) festgelegt. Bei gleichem ILVL wird der Intrerrupt mit dem höherem GLVL zuerst bedient.

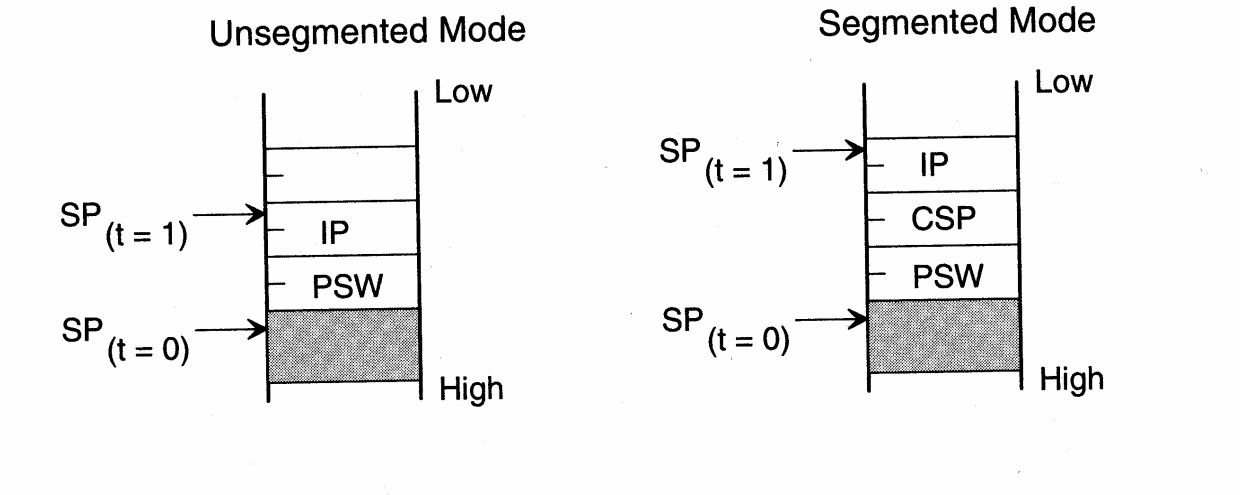
Service Priority

Die HW-Traps Klasse A haben die höchste Priorität gefolgt von den HW-Traps der Klasse B. Die Service Priorität der Hardware Interrupts ist durch den ILVL im Interrupt Control Register festgelegt (je höher ILVL desto höher die Priorität). Die CPU übernimmt den ILVL im PSW. Die CPU startet mit ILVL=0 im PSW. Nur ein Interrupt mit höherer Service Priority als aktuelle kann eine Unterbrechung auslösen.

1.2.7.2 Aufruf einer Interrupt Routine

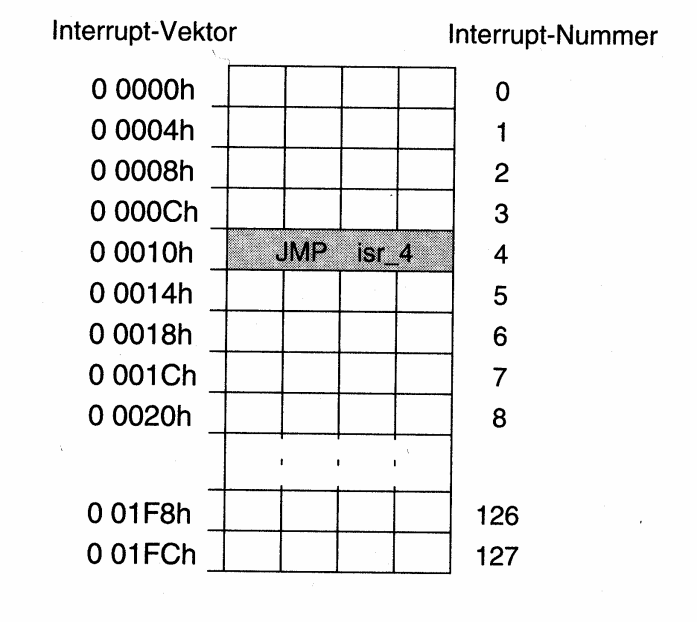
Bei einem "erlaubten" Interrupt werden folgende Aktivitäten ausgeführt:

- Speichern des Processor Status Word (**PSW**) und des Instruction Pointers (**IP**) auf den Stack. Falls das Bit Segmentation Disable (**SGTDIS**) im System Configuration Register (**SYSCON**) Null ist, wird zusätzlich der Code Segment Pointer (**CSP**) auf dem Stack gesichert.



- Die Interrupt-Sevice-Priorität des unterbrechenden Interrupt wird in das **PSW** übertragen.

- Das Interupt-Request-Bit der Hardware-Interrupt-Quelle wird gelöscht.
- Falls eine Multiplikation oder Division durch einen Interrupt unterbrochen wurde, so wird das Bit Multply/Divide in Progress (MULIP) im PSW gesetzt. Werden in der Interrupt-Routine die Funktionen Multiplikation oder Division benützt, so müssen die Register MDC, MDL und MDH auf den Stack gesichert werden, damit der unterbrochene Befehl nach der Interrupt-Routine korrekt weitergeführt werden kann.
- Mit der dem Inerrupt zugeordneten Interrupt-Nummer wird in die Interrupt-Vektor-Tabelle "gesprungen". Für jede Interupt-Nummer sind in der Vektor-Tabelle 4 Bytes reserviert, in denen ein Sprungbefehl zur Interrupt-Service-Routine stehen muss.



1.2.7.3 Nicht sperrbare Interrupts

Nr.	Adresse	Vektor	Flag	Priorität	Quelle
00H	0000H	RESET		III	Reset-Eingang Stift Nr. 140 Software-Reset Befehl SRST Watch Dog Timer Reset
02H	0008H	NMITRAP	NMI	II	NMI-Eingang Stift Nr. 142
04H	0010H	STOTRAP	STKOF	II	Stapelüberlauf Register STKOV
06H	0018H	STUTRAP	STKUF	II	Stapelunterlauf Register STKUV
0AH	0028H	BTRAP	UNDOPC PRTFLD ILLOPA ILLINA ILLBUS	I	undefinierter Befehlscode Fehler bei geschütztem Befehl Wortzugriff auf ungerade Adr. Sprung auf ungerade Adresse unzulässiger Buszugriff

TFR **RAM:** 0FFACH **SFR:** 0D6H **Reset:** 0000H *bitadressierbar*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMI	STK OF	STK UF	-	-	-	-	-	UND OPC	-	-	-	PRT FLT	ILL OPA	ILL INA	ILL BUS

Die Felder haben folgende Bedeutung:

ILLBUS	externer Buszugriff ohne entsprechende Definition
ILLINA	Sprungbefehl auf eine ungerade Adresse
ILLOPA	Wortzugriff auf eine ungerade Adresse
PRTFLD	geschützter Befehl nicht im richtigen Format
UNDOPC	nichtdefinierter Operationscode
STKUF	Unterlauf des Stapelzeigers (Kontrolle in Register STKUN)
STKOF	Überlauf des Stapelzeigers (Kontrolle in Register STKON)
NMI	fallende Flanke am Eingang NMI aufgetreten

Programmbeispiel: Bei jedem NMI werden die am Port 7 anliegenden Daten an Port 2 übertragen

```
; k5b7.asm Bild 5-7: NMI-Interrupt ohne Schleifenkontrolle
%target 167
%list
        ORG     0008H           ; Einsprung NMI-Interrupt
        mov     P2,P7           ; P2 <= Port P7
        bclr    NMI             ; Flag in TFR löschen
        reti                    ; Rücksprung von Interruptprogramm

        ORG     200H            ; Hauptprogramm
haupt   PROC    far             ; keine Interrupt-Freigabe !!!
        mov     DP2,#0ffffh     ; Port P2 ist Ausgang
loop:   jmp     loop            ; hier tut sich nichts
        rets                    ; Abbruch mit RESET-Taste
haupt   ENDP                    ; Ende des Hauptprogramms
        END                     ; Ende des Quelltextes
```

```
/* k5b7.c Bild 5-7:  NMI-Interrupt ohne Laufkontrolle       */
/* Vektor 0x2 vom System belegt: Aenderungen mit Monitor !! */
#include <reg167.h>

void nmi_int (void) interrupt 0x2 // NMI Interruptprogramm
{
 P2 = P7;                         // P2 <= Port P7
 NMI = 0;                         // Flag in TFR loeschen
}

int main(void)                    // Hauptfunktion
{
 DP2 = 0xffff;                    // Port P2 ist Ausgang
 while (1);                       // Abbruch nur mit RESET-Taste
 return 0;                        // zurueck nach System
}
```

Bild 5-7: NMI-Interrupt-Programme (Assembler und C)

1.2.7.4 Hardware Interrupts

Der Prozessor besitzt folgende Interrupt-Quellen:

- 32 Interrupts der Capture/Compare Unit, welche auch als externe Interrupts verwendet werden können
- 10 Interrupts der Timer Units, wovon drei auch als externe Interrupts verwendet werden können
- 7 Interrupts der zwei seriellen Schnitstellen
- Interrupts des A/D-Wandlers
- Interrupts der CAN - Schnittstellen
- je 1 Interrupt für PWM und PLL

Portleitungen	altern. Funkt.	Steuer-Reg.	Anzeige	Freigabe	Mode
P2.0 - P2.15	Capture/ Compare	CC0IC-CC15IC	CC0IR-CC15IR	CC0EI-CC15EI	CCM0-CCM3
P8.0 - P8.7	Capture/ Compare	CC16IC-CC23IC	CC16IR-CC23IR	CC16EI-CC23EI	CCM4-CCM5
P1H4 - P1H7	Capture/ Compare Adreßbus	CC24IC-CC27IC	CC24IR-CC27IR	CC24EI-CC27EI	CCM6
P7.4 - P7.7	Capture/ Compare	CC28IC-CC31IC	CC28IR-CC31IR	CC28EI-CC31EI	CCM7
P3.7	Timer T2				
P3.5	Timer T4				
P3.2	Timer GPT2				

Jeder Interrupt-Quelle ist ein Interrupt Control Register zugeordnet (x**IC**)

xxIC **RAM:** 0Fxxx SFR / ESFR **RESET:** xx00H *bitadressierbar*

15 – 8	7	6	5 4 3 2	1 0
–	xxIR	xx**IE**	ILVL	GLVL

Die Felder haben folgende Bedeutung:

GLVL	Gruppen-Ebene von 00 (niedrigster) bis 11 (höchster) Rang in der Gruppe
ILVL	Interrupt-Ebene von 0000 (gesperrt) bis 1111 (höchste) Priorität
xxIE	Einzel-Freigabe: 0 = gesperrt, 1 = freigegeben
xxIR	Interrupt-Anzeige: 0 = keine Anforderung, 1 = Anforderung aufgetreten

Mit ILVL wird die Service Priority festgelegt
Mit ILVL = 1110 oder 1111 → PEC
Bei gleichem ILVL bestimmt GLVL die Reihenfolge

Bei einem entsprechendem Ereignis wird xxIR gesetzt (Interrupt-Anforderung)
Nur wenn xxIE Bit gesetzt ist, kann eine Unterbrechung erfolgen

I E N im PSW muss gesetzt sein → globale Freigabe von Interrupts

PSW **RAM:** 0FF10H **SFR:** 88H **Reset:** 0000H *bitadressierbar*

15	14	13	12	11	10	9	8	7							0
ILVL				**IEN**	H	-	-	-	U	M	E	Z	V	C	N

Die Felder der Interrupt-Steuerung haben folgende Bedeutung:

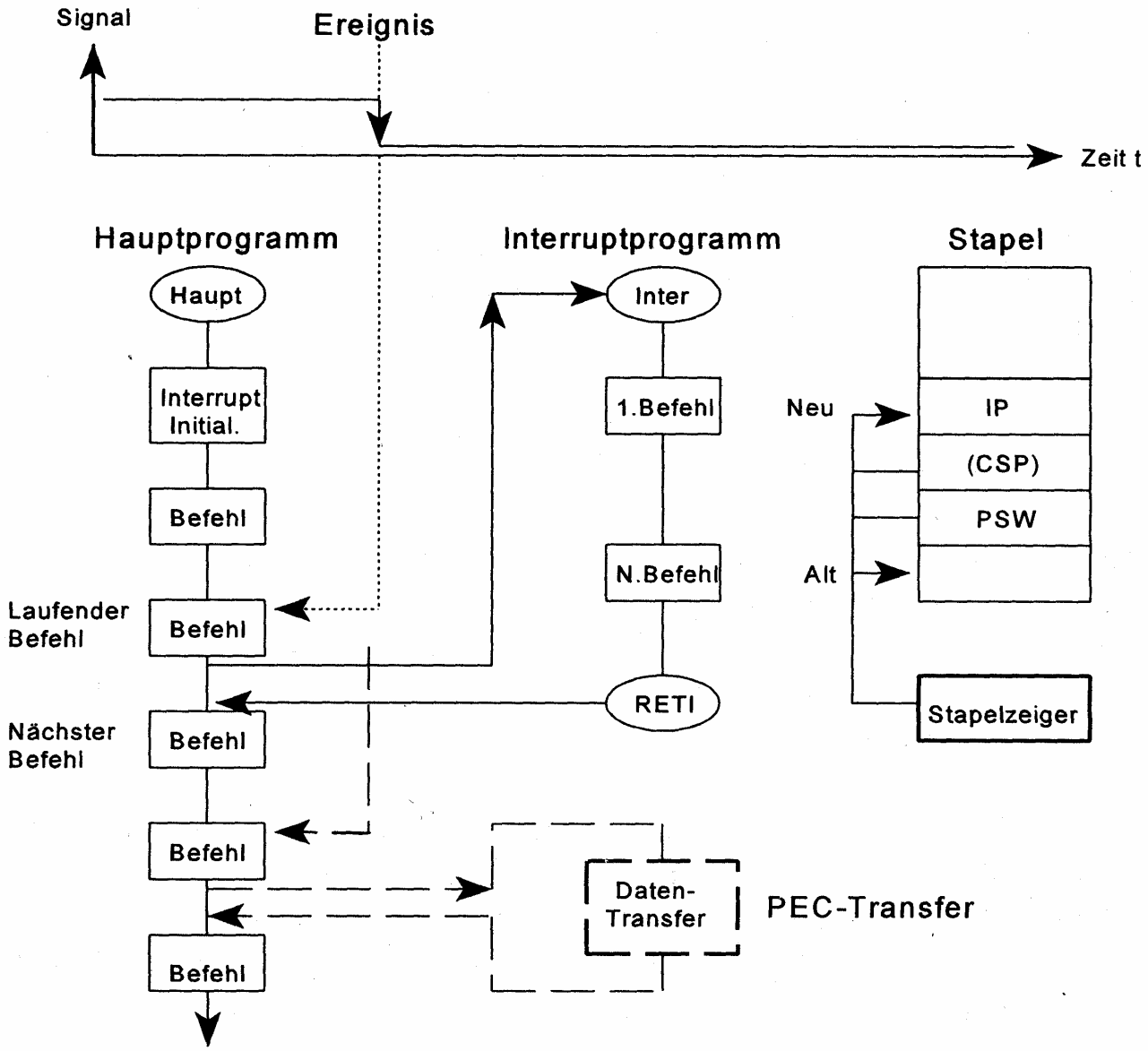
IEN Gesamt-Freigabe: 0 = alle Interrupts gesperrt, 1 = globale Freigabe

ILVL laufende Interrupt-Priorität von Steuerung eingetragen

Bei einem "erlaubten" Interrupt wird der aktuelle Befehl (außer Multiplikation und Division) durchgeführt, xxIR wird gelöscht und es erfolgt ein direkter Sprung an die entsprechende Adresse der Interrupt Vektor Tabelle (hier steht ein Sprungbefehl zur entsprechenden Interrupt-Routine (oder einige Befehle)).

Die Interrupt-Service-Routine wird abgearbeitet – Mit dem Befehl RETI (Return from Interrupt) werden IP, ggfs. CSP und PSW vom Stack geholt und das Programm an der unterbrochenen Stelle weiterbearbeitet.

Embedded Systems



1.2.7.5 Externe Interrupts

Wenn an einem Pin eine externe Interrupt-Quelle angeschlossen wird, muss der entsprechende Pin als Input (DPx.y) geschaltet werden und es muss festgelegt werden, bei welcher Situation ein Interrupt ausgelöst werden soll (→ Eintrag in den Capture/Compare Mode Registern CCM0 bis CCM7).

CCMx **RAM:** *Tabelle* **SFR:** *Tabelle* **RESET:** 0000H *bitadressierbar*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCx	CCMODx		ACCx		CCMODx			ACCx		CCMODx		ACCx		CCMODx	

CCMODx Auswahl der Betriebsart

- 000: Capture und Compare gesperrt, Anschluß als Interrupt verwendbar
- 001: Capture bei steigender Flanke am Anschluß CCxIO
- 010: Capture bei fallender Flanke am Anschluß CCxIO
- 011: Capture bei beiden Flanken am Anschluß CCxIO
- 100: *Compare-Mode 0:* mehrere Interrupts während der Timer Periode
  Doppel-Register-Mode möglich für CC8..CC15 und CC24..CC31
- 101: *Compare-Mode 1:* mehrere Interrupts während der Timer Periode
  Anschluß CCxIO bei Gleichheit umschalten
  Doppel-Register-Mode möglich für CC0..CC7 und CC16..CC23
- 110: *Compare-Mode 2:* nur ein Interrupt während der Timer Periode
- 111: *Compare-Mode 3:* nur ein Interrupt während der Timer Periode
  Anschluß CCxIO bei Gleichheit auf 1 gesetzt
  Anschluß CCxIO bei Timer-Überlauf auf 0 gelöscht

ACCx Zuordnung des Timers: 0 = T0 bzw. T7, 1= T1 bzw. T8

Zuordnung der Capture/Compare Mode Register:

Nr.	Vektor-Adresse	IC-Register	Mode-Register	Interupt-Quelle
10H	0040H CC0INT	CC0IC	CCM0 .0 - .3	P2.0
11H	0044H CC1INT	CC1IC	CCM0 .4 - .7	P2.1
12H	0048H CC2INT	CC2IC	CCM0 .8 -.11	P2.2
13H	004CH CC3INT	CC3IC	CCM0 .12-.15	P2.3
14H	0050H CC4INT	CC4IC	CCM1 .0 - .3	P2.4
15H	0054H CC5INT	CC5IC	CCM1 .4 - .7	P2.5
16H	0058H CC6INT	CC6IC	CCM1 .8 -.11	P2.6
17H	005CH CC7INT	CC7IC	CCM1 .12-.15	P2.7
18H	0060H CC8INT	CC8IC	CCM2 .0 - .3	P2.8 Fast Mode
19H	0064H CC9INT	CC9IC	CCM2 .4 - .7	P2.9 Fast Mode
1AH	0068H CC10INT	CC10IC	CCM2 .8 -.11	P2.10 Fast Mode
1BH	006CH CC11INT	CC11IC	CCM2 .12-.15	P2.11 Fast Mode
1CH	0070H CC12INT	CC12IC	CCM3 .0 - .3	P2.12 Fast Mode
1DH	0074H CC13INT	CC13IC	CCM3 .4 - .7	P2.13 Fast Mode
1EH	0078H CC14INT	CC14IC	CCM3 .8 -.11	P2.14 Fast Mode
1FH	007CH CC15INT	CC15IC	CCM3 .12-.15	P2.15 Fast Mode
30H	00C0H CC16INT	CC16IC	CCM4 .0 - .3	P8.0
31H	00C4H CC17INT	CC17IC	CCM4 .4 - .7	P8.1
32H	00C8H CC18INT	CC18IC	CCM4 .8 - 11	P8.2

Die Portleitungen P2.8 bis P2.15 können im Fast Mode alle 50 ns satt alle 400 ns abgetastet werden. Dazu müssen die entsprechenden Einstellungen im External Interrupt Control Register (EXICON) vorgenommen werden.

EXICON **RAM:** 0F1C0H **ESFR:** 0E0H **RESET:** 0000H *bitadressierbar*

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
EXI7ES P2.15	EXI6ES P2.14	EXI5ES P2.13	EXI4ES P2.12	EXI3ES P2.11	EXI2ES P2.10	EXI1ES P2.9	EXI0ES P2.8

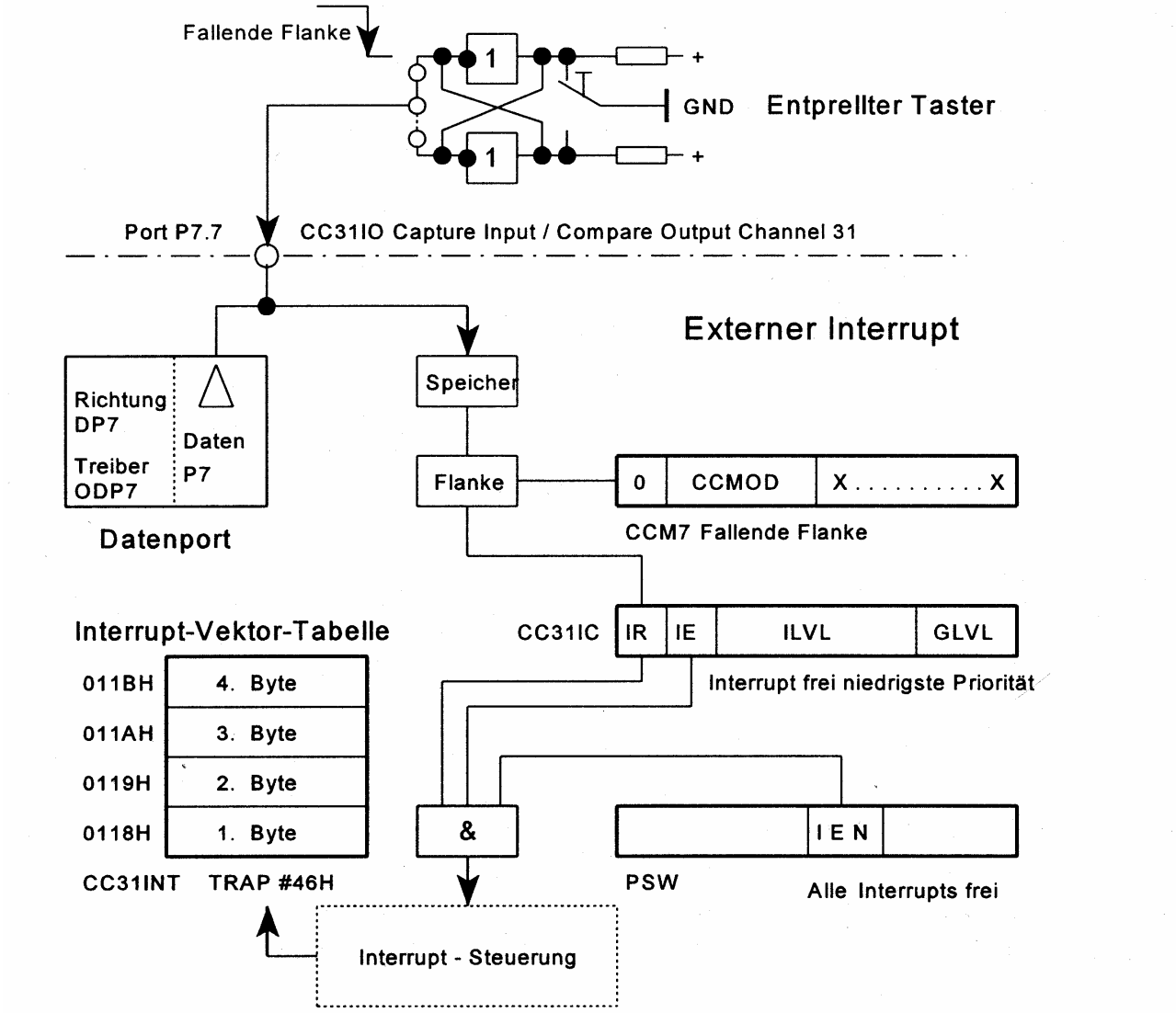
Das Feld EXIxES hat folgende Bedeutung:

0 0 = Fast Interrupt gesperrt
0 1 = Interruptauslösung durch die steigende Flanke
1 0 = Interruptauslösung durch die fallende Flanke
1 1 = Interruptauslösung durch beide Flanken

Embedded Systems

Beispiel:

Vier entprellte Taster (T7 bis T4) sind an den Pins P7.7 bis P7.4 angeschlossen. An den Pins P7.3 bis P7.0 sind vier Leuchtdioden angeschlossen. Bei einem Tastendruck soll eine entsprechende Diode für 3 Sekunden leuchten. Alle Interrupts werden auf eine fallende Flanke ausgelöst. P7.7 soll die höchste Priorität haben. P7.6 soll das gleiche ILVL haben – P7.5 und P7.4 eine Ebene niedriger.



Embedded Systems

```
; k5b9.asm Bild 5-9: Freigabe- und Prioritäts-Steuerung
%target 167
%list
zeit    EQU     90              ; Wartezeit 90 * 0.033 = ca. 3 sek
CC31IC  EQU     0f194h          ; Capture/Compare Kanal 31
CC30IC  EQU     0f18ch          ; Capture/Compare Kanal 30
CC29IC  EQU     0f184h          ; Capture/Compare Kanal 29
; CC28IC ist bereits vordefiniert Capture/Compare Kanal 28
; Interrupt-Einsprünge
        ORG     118H            ; Taste P7.7
        jmp     p77             ;
        ORG     114H            ; Taste P7.6
        jmp     p76             ;
        ORG     110H            ; Taste P7.5
        jmp     p75             ;
        ORG     0f0H            ; Taste P7.4
        jmp     p74             ;
; Hauptprogramm
        ORG     200H            ; Hauptprogramm
haupt   PROC    far             ;
        mov     DP2,#0ffffh     ; Port P2 ist Ausgang
        mov     P2,#0           ; P2 löschen
        mov     DP7,#0fh        ; P7.3 - P7.0 sind Ausgänge
        mov     P7,#00h         ; P7.3 - P7.0 LED aus
        mov     CCM7,#2222h     ; P7.7 P7.6 P7.5 P7.4 fallende Flanke
        mov     r0,#49h         ; 0 1 0010 01  ILevel 2  GLevel 1
        mov     CC31IC,r0       ; Interrupt P7.7 frei
        mov     r0,#48h         ; 0 1 0010 00  ILevel 2  GLevel 0
        mov     CC30IC,r0       ; Interrupt P7.6 frei
        mov     r0,#45h         ; 0 1 0001 01  ILevel 1  GLevel 1
        mov     CC29IC,r0       ; Interrupt P7.5 frei
        extr    #1              ; Zugriff auf ESFR-Bereich
        mov     CC28IC,#44h     ; P7.4: 0 1 0001 00 ILevel 1 GLevel 0
        bset    IEN             ; alle Interrupts frei
loop:   jmp     loop            ; schlafen
        rets                    ; zurück nach System
haupt   ENDP                    ; Ende des Hauptprogramms
```

```
; Interruptprogramme liegen hinter dem Hauptprogramm
p77     PROC    near            ; bei jeder fallenden Flanke P7.7
        bset    P7.3            ; LED P7.3 an
        add     P2,#100h        ; Port P2 um 100h erhöhen
        call    warte           ;
        bclr    P7.3            ; LED P7.3 aus
        reti                    ; Rücksprung von Interruptprogramm
p77     ENDP                    ; Ende des Interruptprogramms
p76     PROC    near            ; bei jeder fallenden Flanke P7.6
        bset    P7.2            ; LED P7.2 an
        add     P2,#1           ; Port P2 um 1 erhöhen
        call    warte           ;
        bclr    P7.2            ; LED P7.2 aus
        reti                    ; Rücksprung von Interruptprogramm
p76     ENDP                    ; Ende des Interruptprogramms
p75     PROC    near            ; bei jeder fallenden Flanke P7.5
        bset    P7.1            ; LED P7.1 an
        sub     P2,#100h        ; Port P2 um 100h vermindern
        call    warte           ;
        bclr    P7.1            ; LED P7.1 aus
        reti                    ; Rücksprung von Interruptprogramm
p75     ENDP                    ; Ende des Interruptprogramms
p74     PROC    near            ; bei jeder fallenden Flanke P7.4
        bset    P7.0            ; LED P7.0 an
        sub     P2,#1           ; Port P2 um 1 vermindern
        call    warte           ;
        bclr    P7.0            ; LED P7.0 aus
        reti                    ; Rücksprung von Interruptprogramm
p74     ENDP                    ; Ende des Interruptprogramms

; Warte-Unterprogramm für alle Interrupt-Programme
warte   PROC    near            ; wartet ca. zeit * 0.033 sek
        push    r0              ; R0 retten
        push    r1              ; R1 retten
        mov     r1,#zeit        ; äussere Schleife
warte1: mov     r0,#0           ; innere Schleife
warte2: sub     r0,#1           ;
        jmpr    cc_nz,warte2    ; bis innere Schleife == 0
        sub     r1,#1           ;
        jmpr    cc_nz,warte1    ; bis äussere Schleife == 0
        pop     r1              ; R1 zurück
        pop     r0              ; R0 zurück
        ret                     ; Rücksprung von Unterprogramm
warte   ENDP                    ;
        END                     ; Ende des Quelltextes
```

```
/* k5b9.c Bild 5-9: Freigabe- und Prioritaets-Steuerung  */
#include <reg167.h>
sbit led70 = P7^0;                  // LED P7.0
sbit led71 = P7^1;                  // LED P7.1
sbit led72 = P7^2;                  // LED P7.2
sbit led73 = P7^3;                  // LED P7.3

void warte(void)                    // Wartefunktion
{ unsigned long i; for (i=0; i<1575000; i++); } // ca. 3 sek

void p77 (void) interrupt 0x46      // P7.7 Interruptprogramm
{
 led73 = 1;                         // LED P7.3 an
 P2 += 0x100;                       // P2 = P2 + 0x100
 warte();
 led73 = 0;                         // LED P7.3 aus
}
```

```
void p76 (void) interrupt 0x45      // P7.6 Interruptprogramm
{
 led72 = 1;                         // LED P7.2 an
 P2 += 0x1;                         // P2 = P2 + 0x1
 warte();
 led72 = 0;                         // LED P7.2 aus
}
```

```
void p75 (void) interrupt 0x44      // P7.5 Interruptprogramm
{
 led71 = 1;                         // LED P7.1 an
 P2 -= 0x100;                       // P2 = P2 - 0x100
 warte();
 led71 = 0;                         // LED P7.1 aus
}
void p74 (void) interrupt 0x3c      // P7.4 Interruptprogramm
{
 led70 = 1;                         // LED P7.0 an
 P2 -= 0x1;                         // P2 = P2 - 0x1
 warte();
 led70 = 0;                         // LED P7.0 aus
}
int main(void)                      // Hauptfunktion
{
 DP2 = 0xffff;                      // Port P2 ist Ausgang
 P2 = 0;                            // Port P2 loeschen
 DP7 = 0x000f;                      // P7.3 bis P7.0 sind Ausgaenge
 P7 = 0;                            // P7.3 bis P7.0 LED aus
 CCM7 = 0x2222;                     // P7.7 bis P7.4 fallende Flanke
 CC31IC = 0x49;                     // P7.7 ILevel 2 GLevel 1
 CC30IC = 0x48;                     // P7.6 ILevel 2 GLevel 0
 CC29IC = 0x45;                     // P7.4 ILevel 1 GLevel 1
 CC28IC = 0x44;                     // P7.4 ILevel 1 GLevel 0
 IEN = 1;                           // alle Interrupts freigeben
 while (1);                         // Abbruch mit Reset
 return 0;                          // zurueck nach System
}
```

1.2.7.6 Der Periphal Event Controller (PEC) Interupt

Interrupt-gesteuerter DMA-ähnlicher Datentransfer über acht getrennte PEC-Kanäle

Jedem Kanal (0....7) sind die Steuerregister (Peripheral Event Channel Control **PECCx**), ein Source Pointer (**SRCPx**) und ein Destination Pointer (**DSTPx**) zugeordnet.

Jeder beliebige Hardware Interrupt kann einem der acht PEC-Kanäle zugeordnet werden.

Die Zuordnung geschieht durch ILVL=1111 oder ILVL=1110 und den Group Level (GLVL)

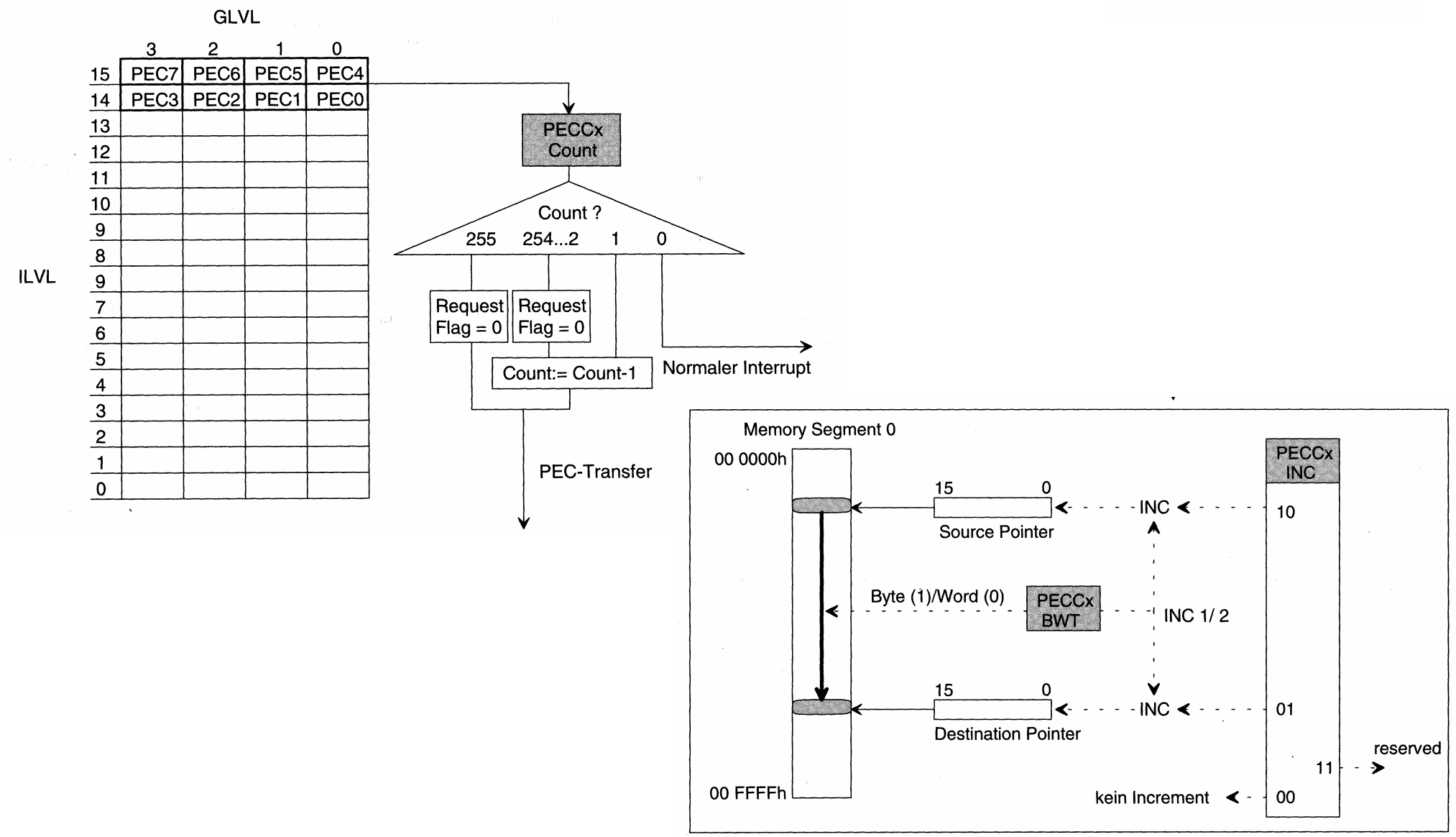
Wird ein Interrupt, der dem Level 14 oder 15, zugeordnet ist, ausgelöst, so wird aufgrund des GLVL ein PEC-Kanal ausgewählt und je nach Inhalt von PECCx ein Datentransfer oder ein Interrupt ausgelöst.

Ist COUNT im PECCx gleich 0, dann wird ein Interrupt ausgelöst.

Ist COUNT im PECCx im Bereich 254 bis 1, wird COUNT dekrementiert, das Repuest Flag (bei COUNT $\neq$ 1) gelöscht, ggfs. SRCPx oder DSTPx (je nach INC im PECCx) erhöht und ein PEC-Transfer durchgeführt.

Ist COUNT im PECCx gleich 255 werden PEC-transfers durchgeführt ohne COUNT zu verändern.

Embedded Systems



Embedded Systems

PECCx **RAM:** *Tabelle* **SFR:** *Tabelle* **RESET:** 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
–	–	–	–	–	INC		BWT	COUNT							

Die Felder haben folgende Bedeutung:

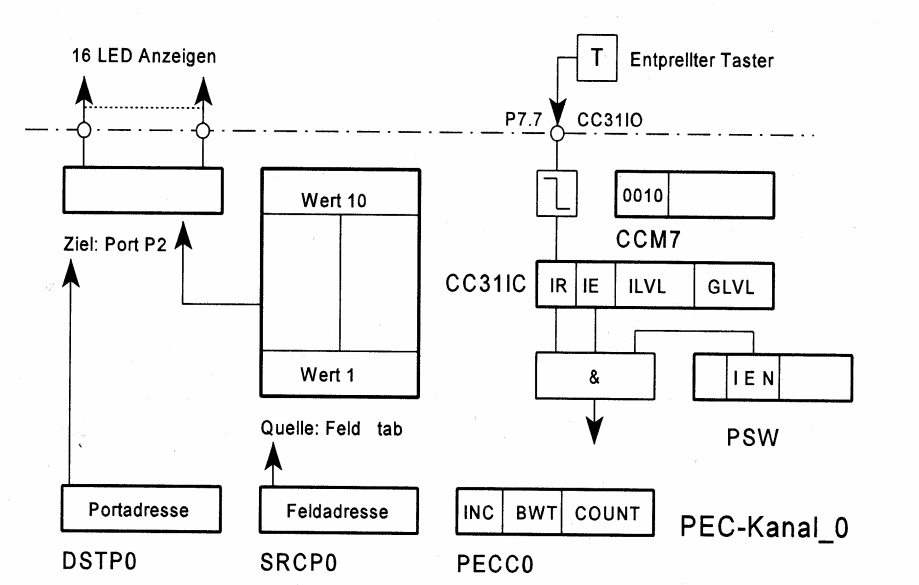
COUNT wird mit dem Zähleranfangswert initialisiert
Anfangswert **0FFH:** Dauertransfer ohne Änderung des Zählers
Anfangswert **00H:** kein Transfer, IR=1: Interrupt entsprechend Level
Event vermindert den Zähler um 1 bis COUNT = 0, dann IR=1: Interrupt

BWT Byte/Wort-Auswahl: 0 = Worttransfer 1 = Bytetransfer

INC Steuerung des Zielzeigers DSTPx bzw. Quellzeigers SRCPx
00: beide Zeiger werden nicht verändert
01: erhöhe den Zielzeiger DSTPx um 1 (Byte) oder 2 (Wort)
10: erhöhe den Quellzeiger SRCPx um 1 (Byte) oder 2 (Wort)

Nr.	ILVL	GLVL	PECCx-Register (SFR)	Zielzeiger Adr.	Quellzeiger Adr.
7	1111	11	PECC7 FECEH (67H)	DSTP7 FCFEH	SRCP7 FCFCH
6	1111	10	PECC6 FECCH (66H)	DSTP6 FCFAH	SRCP6 FCF8H
5	1111	01	PECC5 FECAH (65H)	DSTP5 FCF6H	SRCP5 FCF4H
4	1111	00	PECC4 FEC8H (64H)	DSTP4 FCF2H	SRCP4 FCF0H
3	1110	11	PECC3 FEC6H (63H)	DSTP3 FCEEH	SRCP3 FCECH
2	1110	10	PECC2 FEC4H (62H)	DSTP2 FCEAH	SRCP2 FCE8H
1	1110	01	PECC1 FEC2H (61H)	DSTP1 FCE6H	SRCP1 FCE4H
0	1110	00	PECC0 FEC0H (60H)	DSTP0 FCE2H	SRCP0 FCE0H

Beispiel:
Es soll bei jeder fallenden Flanke am Eingang P7.7 ein Wort aus einer Tabelle (10 Worte) an P2 ausgegeben werden.




```
int main(void)                          // Hauptfunktion
{
 unsigned int i;
 for (i=0; i<10; i++) tab[i] = i + 1;  // Werte von 1 bis 10
 DP2 = 0xffff;                          // Port P2 ist Ausgang
 P2 = 0;                                // Port P2 loeschen
 PECC0 = 0x40a;                         // 10 0 0000 1010 SRCP+ 10 Woerter
 DSTP0 = (unsigned int) &P2;            // Zielzeiger <= Adresse Port 2
 SRCP0 = (unsigned int) &tab;           // Quellzeiger <= Tabelle
 CCM7 = CCM7 | 0x2000;                  // P7.7 fallende Flanke
 CC31IC = 0x78;                         // 0 1 1110 00 P7.7 PEC-Kanal
 IEN = 1;                               // alle Interrupts freigeben
 while (lauf);                          // solange P7.0 High
 IEN = 0;                               // alle Interrupts sperren
 return 0;                              // zurueck nach System
}

void neu (void) interrupt 0x46          // P7.7 Interruptprogramm
{
 PECC0 = 0x40a;                         // 10 0 0000 1010 SRCP+ 10 Woerter
 DSTP0 = (unsigned int) &P2;            // Zielzeiger <= Adresse Port 2
 SRCP0 = (unsigned int) &tab;           // Quellzeiger <= Tabelle
}
```

Embedded Systems

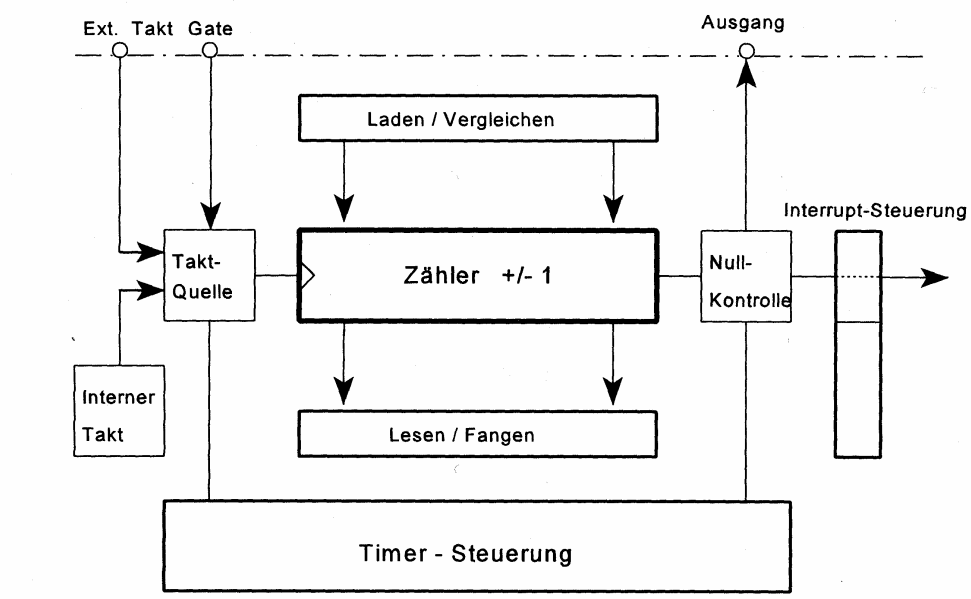
```
; k5b12.asm Bild 5-12: PEC-Transfer mit Interrupt P7.7
%target 167
%list
CC31IC  EQU     0f194h          ; Capture/Compare-Interrupt-Steuerung
        ORG     118h            ; Einsprung P7.7-Interrupt
        jmp     neuini          ; nach Interruptprogramm

        ORG     200H            ; Hauptprogramm
haupt   PROC    far             ;
        mov     DP2,#0ffffh     ; Port P2 ist Ausgang
        mov     P2,#0           ; P2 löschen
        mov     PECC0,#40ah     ; 10 0 00001010 SRCP+ 10 Wörter
        mov     r0,#P2          ; R0 = Adresse Port 2
        mov     DSTP0,r0        ; Zielzeiger <= Port 2
        mov     r0,#tab         ; R0 = Adresse Tabelle
        mov     SRCP0,r0        ; Quellzeiger <= Tabelle
        or      CCM7,#2000h     ; 0 010 xxxxx  P7.7 fallende Flanke
        mov     r0,#78h         ; 0 1 1110 00  P7.7 PEC-Kanal Nr. 0
        mov     CC31IC,r0       ; Interrupt P7.7 frei
        bset    IEN             ; alle Interrupts frei
loop:   jb      P7.0,loop       ; solange P7.0 High: weiter schlafen
        bclr    IEN             ; alle Interrupts wieder gesperrt
        rets                    ; zurück nach System
haupt   ENDP                    ; Ende des Hauptprogramms
```

```
; Interruptprogramm am Ende der Tabelle ausgeführt
neuini  PROC    near            ; PEC-Kanal wieder initalisieren
        push    r0              ; R0 gerettet
        mov     PECC0,#40ah     ; 10 0 00001010 SRCP+ 10 Wörter
        mov     r0,#P2          ; R0 = Adresse Port 2
        mov     DSTP0,r0        ; Zielzeiger <= Port 2
        mov     r0,#tab         ; R0 = Adresse Tabelle
        mov     SRCP0,r0        ; Quellzeiger <= Tabelle
        pop     r0              ; R0 zurück
        reti                    ; Rücksprung von Interruptprogramm
neuini  ENDP                    ; Ende des Interruptprogramms
; Datenbereich mit Ausgabetabelle
tab     DW 1,2,3,4,5,6,7,8,9,10 ; 10 Wörter mit Werten von 1 bis 10
        END                     ; Ende des Quelltextes
```

1.2.8 Die Timereinheiten

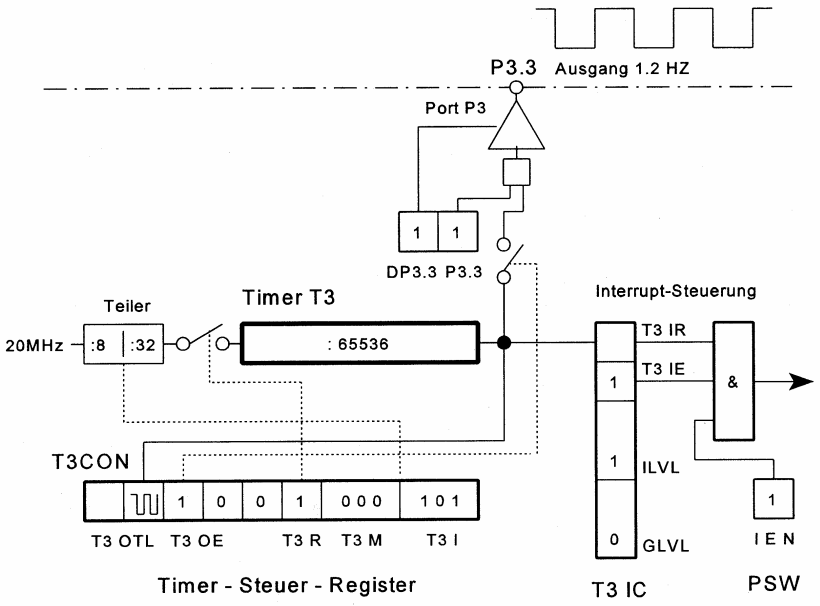
Programmierbarere Hardware-Zähler, die bei Nulldurchgang Interrupts auslösen können.

Die wichtigsten Betriebsarten sind:



- programmierbarer Frequenzteiler (TIMER-Mode) mit internem Takt
- Frequenzteiler mit Gate-steuerung (Gated Timer Mode)
- Ereigniszähler (Count-Mode) für externe Signalflanken
- Auslesen des Zählerstandes bei bestimmten Ereignis (Capture-Mode)
- nachladbarer Frequenzteiler (Reload-Mode)
- Ausgabe eines Signals bei bestimmten Zählerstand (Compare-Mode)

Embedded Systems



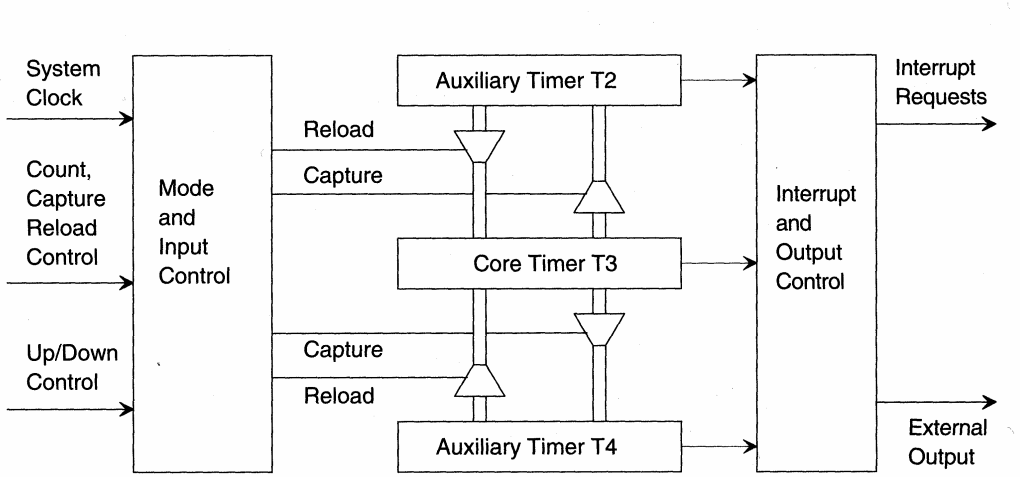

```
; k6b3.asm  Bild 6-3: Timer T3 Timerbetrieb Frequenzausgabe P3.3
%target 167
%list
        ORG     200h            ; Hauptprogramm
haupt   PROC    far             ;
        bset    DP3.3           ; P3.3 Richtung Ausgabe
        bset    P3.3            ; P3.3 Datenbit = 1
        mov     T3CON,#245h     ; 00000 0 1 0 0 1 000 101 T3R=1
loop:   jb      P7.0,loop       ; solange P7.0 High
        bclr    T3R             ; Timer T3 sperren
        rets                    ; zurück nach System
haupt   ENDP                    ;
        END                     ; Ende des Quelltextes
```

```
/* k6b3.c  Bild 6-3: Timer T3 Timerbetrieb Frequenzausgabe P3.3 */
#include <reg167.h>
sbit lauf = P7^0;               // Laufkontrolle
sbit richt = DP3^3;             // Richtung P3.3
sbit daten = P3^3;              // Daten P3.3

int main(void)
{
 richt = 1;                     // P3.3 ist Ausgang
 daten = 1;                     // P3.3 Daten sind high
 T3CON = 0x245;                 // 00000 0 1 0 0 1 000 101 T3R=1
 while (lauf);                  // solange P7.0 High
 T3R = 0;                       // Timer  T3 sperren
 return 0;                      // zurueck nach System
}
```

1.2.8.1 General Purpose Timer Units (GPT1 und GPT2)

Zwei unabhängig arbeitende programmierbare Timereinheiten

GPT1 mit den Timern T2, T4 und dem Core-Timer T3



TxCON **RAM:** *Tabelle* **SFR:** *Tabelle* **RESET:** 0000H *bitadressierbar*

15	14	13	12	11	10	9	8	7	6	5 4 3	2 1 0
-	-	-	-	-	T3 OTL	T3 OE	Tx UDE	Tx UD	TxR	TxM	TxI

Die Felder haben folgende Bedeutung:

TxI Taktsteuerung (Input) je nach Mode und Timer

Timer und Timer mit Gate	***Ereigniszähler (Count)***
000: Teiler 8*1=8	000: keine Flanke (Timer gesperrt)
001: Teiler 8*2=16	001: steigende Flanke an TxIN
010: Teiler 8*4=32	010: fallende Flanke an TxIN
011: Teiler 8*8=64	011: beide Flanken an TxIN
100: Teiler 8*16=128	100: ***nur*** T2 T4: keine Flanke von T3OTL
101: Teiler 8*32=256	101: ***nur*** T2 T4: steigende Flanke von T3OTL
110: Teiler 8*64=512	110: ***nur*** T2 T4: fallende Flanke von T3OTL
111: Teiler 8*128=1024	111: ***nur*** T2 T4: beide Flanken von T3OTL

TxM Betriebsart (Mode)

- 000: Timer ohne Gate
- 001: Ereigniszähler (Count)
- 010: Timer mit Gate aktiv Low
- 011: Timer mit Gate aktiv High
- 1xx: für Timer T3 ***nicht*** verwendbar!
- 100: ***nur*** Timer T2 und T4: Reload Mode (nachladen von T3)
- 101: ***nur*** Timer T2 und T4: Capture Mode (speichern von T3)
- 11x: für keinen der Timer verwendbar!

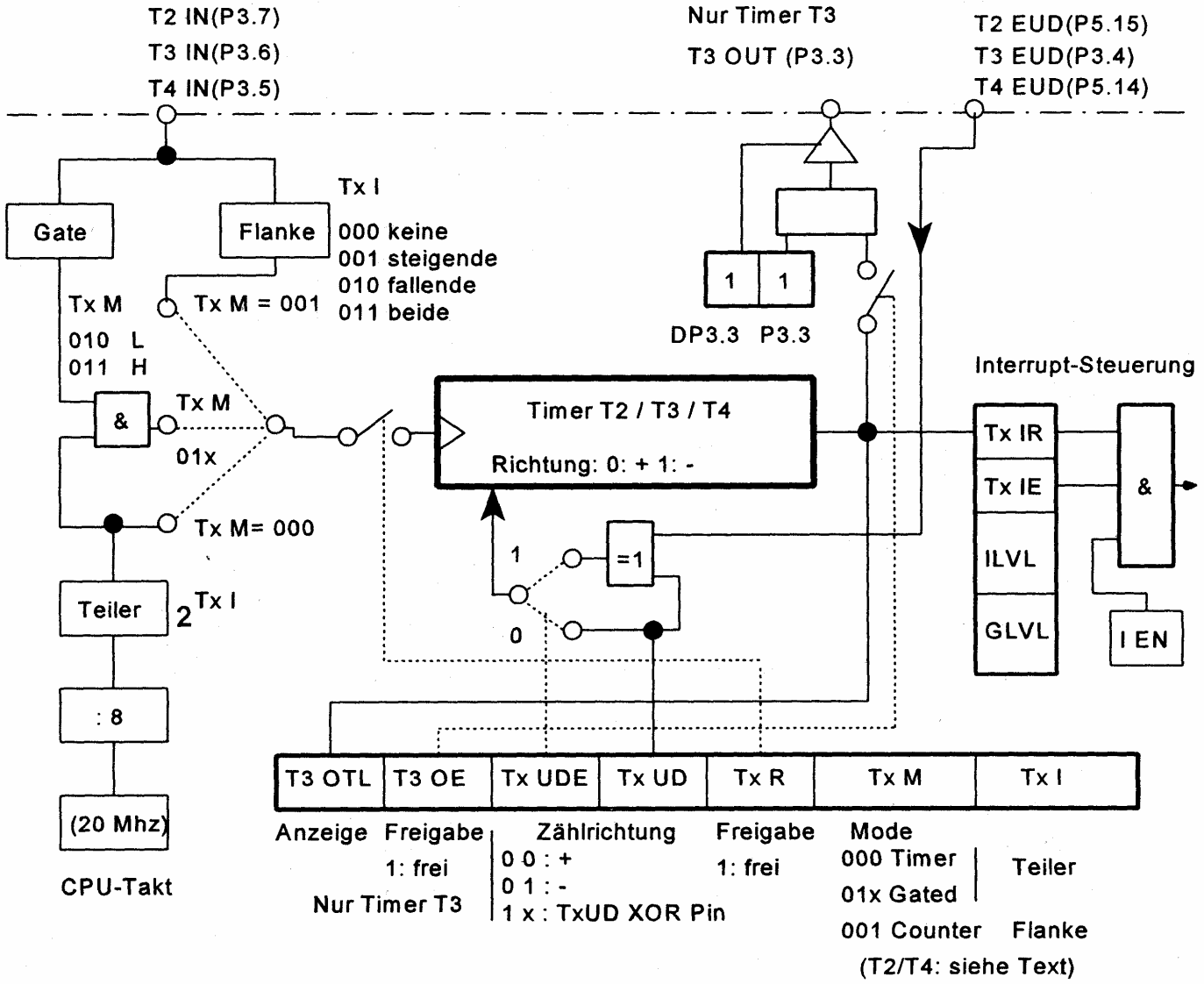
TxR Laufkontrolle 0 = aus, 1 = Run (Lauf)

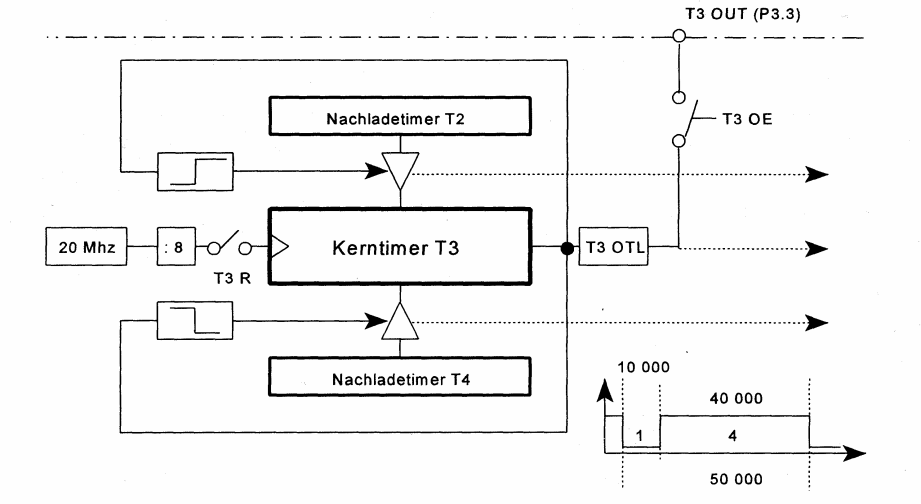
TxUD Zählrichtung 0 = auf (Up), 1 = ab (Down)

TxUDE Mode der Zählrichtung 0 = nur von TxUD abhängig, 1 = XOR mit TxEUD

T3OE Ausgangssteuerung (Timer T3) 0 = P3.3 ist Port, 1 = P3.3 ist Timerausgang

T3OTL Timer T3 Umschaltung bei jedem Nulldurchgang bzw. durch Befehl


T2 IN(P3.7)
T3 IN(P3.6)
T4 IN(P3.5)
Nur Timer T3
T3 OUT (P3.3)
T2 EUD(P5.15)
T3 EUD(P3.4)
T4 EUD(P5.14)
Tx I
Gate
Flanke
000 keine
001 steigende
010 fallende
011 beide
Tx M
Tx M = 001
010 L
011 H
1
1
DP3.3
P3.3
Interrupt-Steuerung
&
Tx M
01x
Timer T2 / T3 / T4
Richtung: 0: + 1: -
Tx IR
Tx IE
ILVL
GLVL
&
I EN
Tx M= 000
1
=1
0
Teiler
2
Tx I
: 8
(20 Mhz)
CPU-Takt
T3 OTL
T3 OE
Tx UDE
Tx UD
Tx R
Tx M
Tx I
Anzeige
Freigabe
1: frei
Nur Timer T3
Zählrichtung
0 0 : +
0 1 : -
1 x : TxUD XOR Pin
Freigabe
1: frei
Mode
000 Timer
01x Gated
001 Counter
(T2/T4: siehe Text)
Teiler
Flanke




```
; k6b9.asm  Bild 6-9: Timer T3 Pulsweitenmodulation PWM-Ausgang
; 50 Hz T=20ms T/2=10 000 000 ns : 400 ns = 25 000 Teiler für 1:1
%target 167
%list
        ORG     200h            ; Hauptprogramm
haupt   PROC    far             ;
        bset    DP3.3           ; P3.3 Richtung Ausgabe
        bset    P3.3            ; P3.3 Datenbit = 1
        mov     T2CON,#25h      ; 00000 0 0 0 0 0 100 101 Rel. steig.
        mov     T2,#25000       ; Nachladewert von T2 Anfangswert
        mov     T4CON,#26h      ; 00000 0 0 0 0 0 100 110 Rel. fall.
        mov     T4,#25000       ; Nachladewert von T4 Anfangswert
        mov     T3CON,#2c0h     ; 00000 0 1 0 1 1 000 000 T3R=1
loop:   mov     T2,P2           ; T2 Nachladewert von P2
        mov     r0,#50000       ;
        sub     r0,P2           ; ohne Unterlaufkontrolle !!!!!
        mov     T4,r0           ; T4 Nachladewert = 50 000 - P2
        jb      P7.0,loop       ; solange P7.0 High
        bclr    T3R             ; Timer T3 sperren
        rets                    ; zurück nach System
haupt   ENDP                    ;
        END                     ; Ende des Quelltextes
```

```
/* k6b9.c  Bild 6-9: Timer T3 Pulsweitenmodulation PWM-Ausgang */
// 50 Hz T=20ms T/2=10 000 000 ns : 400 ns = 25 000 Teiler fuer 1:1
#include <reg167.h>
sbit lauf = P7^0;               // Laufkontrolle
sbit richt = DP3^3;             // Richtungsbit P3.3
sbit daten = P3^3;              // Datenbit P3.3
int main (void)
{
 richt = 1;                     // P3.3 Richtung Ausgabe
 daten = 1;                     // P3.3 Datenbit = 1
 T2CON = 0x25;                  // 00000 0 0 0 0 0 100 101 Reload
 T2 = 25000;                    // Nachladewert von T2 Anfangswert
 T4CON = 0x26;                  // 00000 0 0 0 0 0 100 110 Reload
 T4 = 25000;                    // Nachladewert von T4 Anfangswert
 T3CON = 0x2c0;                 // 00000 0 1 0 1 1 000 000 T3R=1
 while (lauf)
 {
  T2 = P2;                      // T2 Nachladewert von P2
  T4 = 50000 - P2;              // T4 Nachladewert = 50 000 - P2
 }
 T3R = 0;                       // Timer T3 sperren
 return 0;                      // zurueck nach System
}
```

1.2.8.2 Die Capture/Compare Units

Betriebsarten:

TIMER

CAPTURE (Zählerstand in Register CCxx)

COMPARE (Ausgabe eines Signales an CCxxIO, wenn CCxx mit Zählerstand übereinstimmt)