AVR MIKROCONTROLLER

Grundlagen ATMega644P

V0.13, P. Klotz

Elemente eines Mikrocontrollers

- Zentrale Recheneinheit
- Programmspeicher
- Ein- und Ausgabeeinheiten
- Datenspeicher (flüchtig, nicht flüchtig)
- Register zur Steuerung des Controllers
- Timer
- A/D und D/A Wandler
- Businterfaces (I2C, SPI, USB, Ethernet, Can-Bus, ...)
- Ansteuerung von Displays

Einsatzgebiete eines Mikrocontrollers

- Momentan einer starken Wandlung unterworfen - immer mehr Einsatzgebiete
- Ersatz von einfacher digitaler und analoger Schaltungen (-> Preisverfall)
- Übernahme von komplexen Steuerungsaufgaben
- Visualisierung bei Maschinen und Messgeräten
- Mikrocontroller ab 4 Pin bis 1000 Pins

Aufbau eines Mikrocontrollers

Programm-speicher

Takt-steuerung

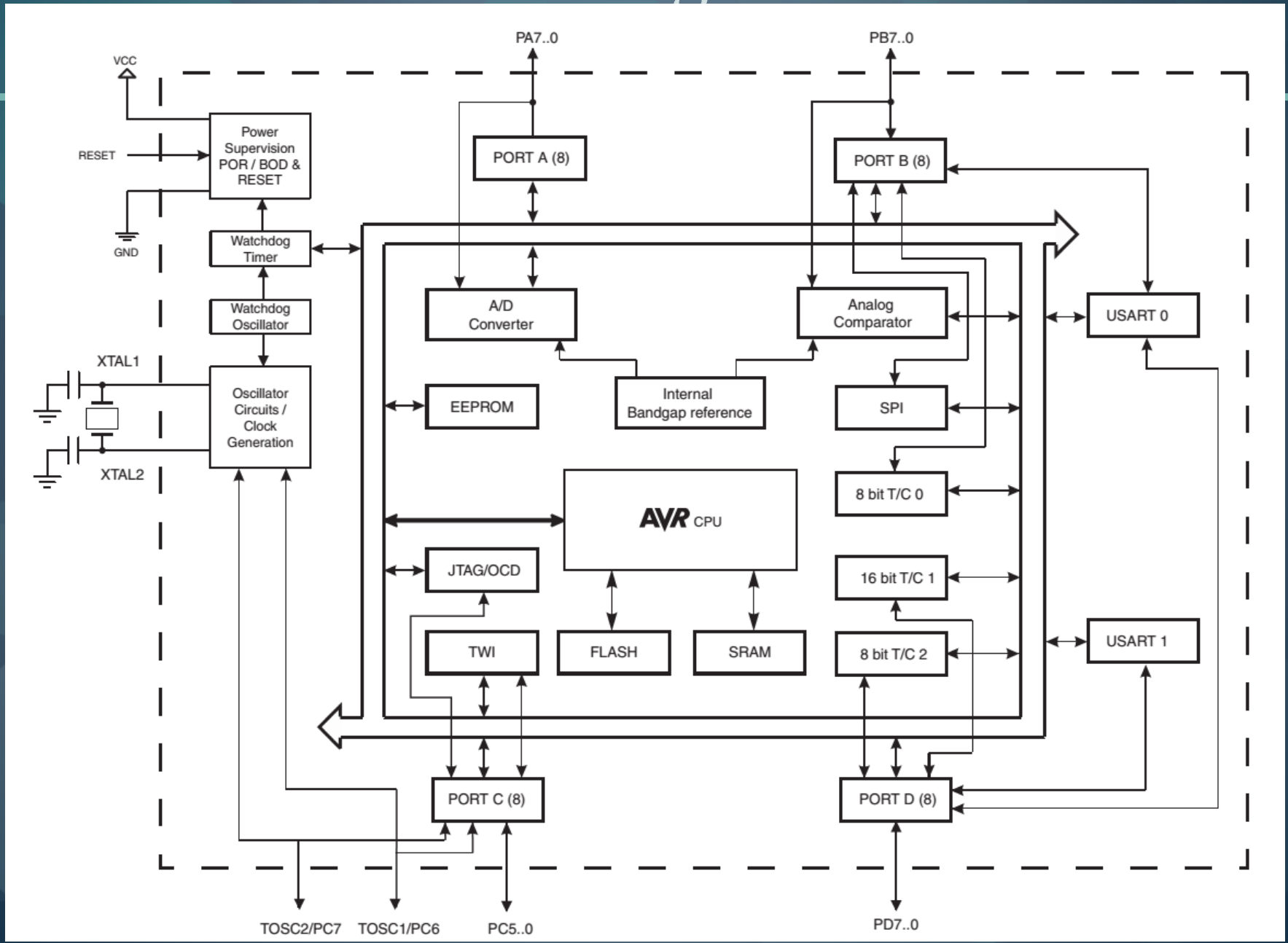
Daten-speicher

CPU und Register

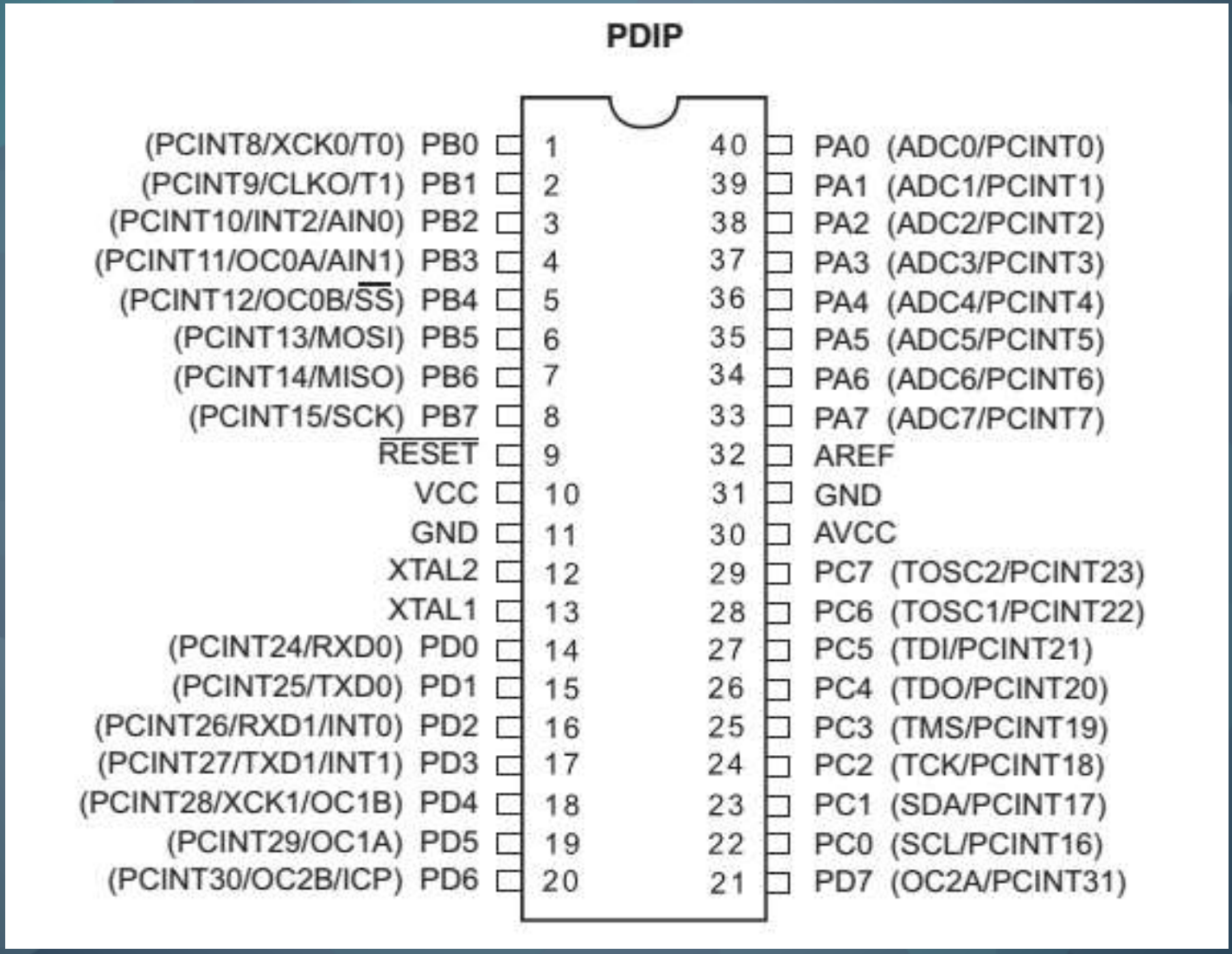
Diverse interne Hardware

Ein-/Ausgabe Pins

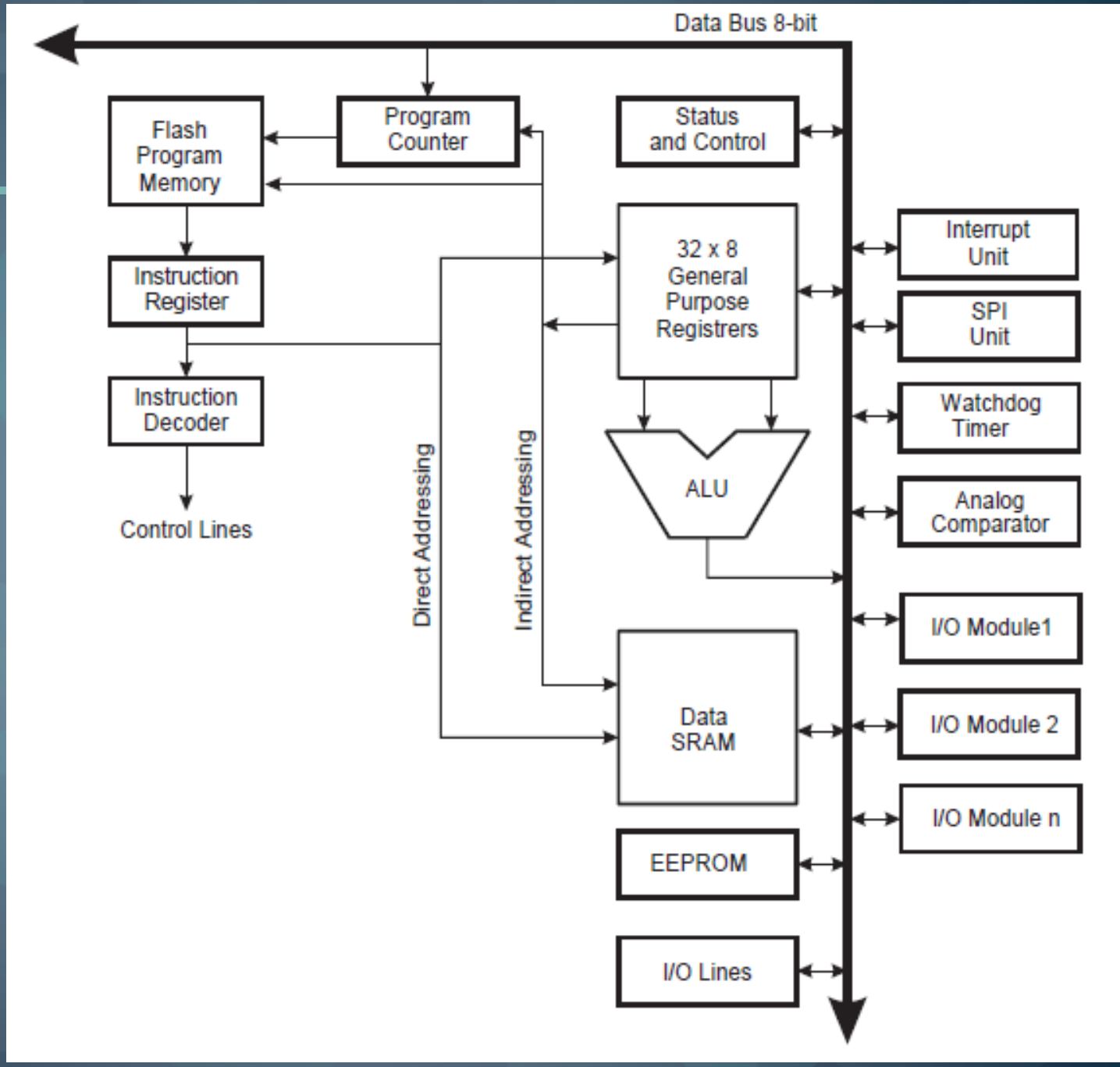
Blockschaltbild ATMega644P



ATMega 644 PA



CPU ATMega644



CPU ATMega644P

- 32 General Purpose Registers
- Zugriff auf 2 Register mit einem Befehl in einem Takt
- 64 kByte Flash – EEPROM
- 2 kByte EEPROM
- 4 kByte RAM
- JTAG Interface zu Debuggen
- BOOT Möglichkeit
- I/O Memory Bereich von 64 Byte

Zusatzkomponenten ATMega644P

- 4 Ports zu je 8 Pins
- Analog / Digital Konverter mit 8 Eingängen, 10 Bit Auflösung
- I2C Interface (TWI)
- SPI-Bus
- 2x 8-Bit Timer, 1 x 16-Bit Timer
- Zähler
- 6 - PWM Kanäle (Puls Width Modulator)
- 2 Serielle Schnittstellen
- Externe und interne Interrupts
- Watchdog
- Interner RC-Oszillator
- 6 verschiedene Sleep -Moden

Architektur CPU

- Harvard Architektur:
  Eigener Daten und Programmspeicher und eigene Datenbusse.
- 1-stufige Befehlspipeline
- 32 x 8-Bit Register
- Abarbeitung der Befehle in einem Taktzyklus
- 6 der 32 Register als 16 Bit Register verwendbar (X-, Y-, und Z-Register)
- Flash Memory ist in 2 Teile unterteilt (zum Booten)
- Speicher ist linear adressierbar
- 64 Byte I/O Speicher für Spezialfunktionen

Befehlsgruppen: Rechnen

- Addieren (Add, Adc) und Subtrahieren (Sub, Sbc) ohne und mit Übertrag,
- Vergleichen mit Konstante (Cpi) / Register (Cp) / Register und Übertrag (Cpc), Um eins
- erhöhen (Inc) oder vermindern (Dec), Logisches Und (And) / Oder (Or) / Exklusiv-Oder (Eor),
- Bits auf 0 (Cbr) oder 1 (Sbr) setzen, Register auf Festwert setzen (Ldi) / auf Null setzen (Clr)
- / auf 255 (Ser) setzen

Befehlsgruppen: Bit und Bit-test

- **Bit im Port auf Eins (*Sbi*) oder Null (*Cbi*) setzen, *Logisches Links-***
- **Schieben (*Lsl*) oder *Rechtsschieben (Lsr), Links- (Rol) oder Rechts- (Ror) Schieben über***
- **Carry, Arithmetisches Rechtsschieben (*Asr*), *Unteres und oberes Nibble tauschen (Swap),***
- **Flaggbit setzen (*Set*) / *rücksetzen (Clt) / auf Registerbit setzen (Bst) / in Register laden (Bld),***
- **Flags im Statusregister setzen (*Sex*) oder *löschen (Clx) mit x={Z,C,N,V,S,H,T,I}, Einer-***
- **(*Com*) und *Zweier-(Neg) Komplement***

Befehlsgruppen: Sprünge

- **Relativer (*Rjmp*) / *Indirekter (Ijmp) Sprung, Relativer (Rcall) / Indirekter (Icall)***
- **Unterprogrammaufruf, Relativsprung bei gesetztem (*Brxs*) / *gelöschtem (Brxc) Statusbit mit***
- **x={Z,C,N,V,S,H,T,I}, Überspringen bei gesetztem (Sbxs) oder gelöschtem (*Sbxc*) *Bit mit {x=r***
- **Register, x=i I/O-Port)**

Befehlsgruppen: Datentransfer

- **Register zu Register (*Mov*), *Registerwort zu Registerwort (Movw)*,**
- **Speicher zu Register direkt (*Ld*) / *indirekt (Ldd)* / *Festadresse (Lds)*, *Register in Speicher***
- **direkt (*St*) / *indirekt (Std)* / *Festadresse (Sts)***
- ***Programmspeicher in Register (Lpm)***
- ***Register* in I/O-Port (*Out*), *I/O-Port in Register (In)***
- ***Register auf Stapel (Push) und vom Stapel (Pop)***

Befehlsgruppen: CPU-Controlle

- **Nichtstun (*Nop*), *Schlafen (Sleep)*, *Wachhund rücksetzen (Wdr)*, *Debug*-Unterbrechung (*Break*)**

Befehlsgruppen: Befehlsaufbau

- **Jeder Befehl belegt ein Wort (=2 Byte) im Programmspeicher.**
- **Die meisten Instruktionen benötigen einen einzigen Takt für die Ausführung, Sprungbefehle zwei Takte.**

Zusammenfassung

- **Der Chef ist die CPU**
- **Hoflieferant der Programmspeicher**
- **Höflinge mit Vitamin B und direktem Zugang zum Chef sind die 32 Register**
- **Indianer sind die Speicher und diverse interne Gerätschaften.**
- **Für die Außenpolitik sorgen externe Pins, die geruhen, etwas ein- oder auszugeben.**

Zusammenfassung

- **Alle Befehle/Instruktionen werden nacheinander bearbeitet**
- **Den Takt der Verarbeitungsmusik gibt der Taktgeber vor**
- **Die aktuelle Bearbeitungsadresse steht immer im Programmzähler (program counter).**
- **Nach außen hin herrscht absolute Ruhe! Kein Pin gibt interne Adressen oder Daten nach außen, es sei denn, es wird so angewiesen!**

Zusammenfassung

- **AVRs holen schon mal den nächsten Befehl, während sie den letzten noch gar nicht richtig bearbeitet haben (Pre-Fetch).**
- **Wildes Umherspringen im Programmablauf wird deshalb mit Strafzeiten von mindestens einer Taktlänge geahndet.**
- **Es gibt mehr als 100 verschiedene Befehle**

Statusregister

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 – I : Global Interrupt Enable
- Bit 6 – T: Bit Copy Storage
- Bit 5 – H: Half Carry Flag
- Bit 4 – S: Sign Bit
- Bit 3 – V: Two's Complement Overflow Flag
- Bit 2 – N: Negative Flag
- Bit 1 – Z : Zero Flag
- Bit 0 – C: Carry Flag

Bit	*Rechnen*	*Logik*	*Vergleich*	*Bits*	*Schieben*	*Sonstige*
Z	ADD, ADC, ADIW, DEC, INC, SUB, SUBI, SBC, SBCI, SBIW	AND, ANDI, OR, ORI, EOR, COM, NEG, SBR, CBR	CP, CPC, CPI	BCLR Z, BSET Z, CLZ, SEZ, TST	ASR, LSL, LSR, ROL, ROR	CLR
C	ADD, ADC, ADIW, SUB, SUBI, SBC, SBCI, SBIW	COM, NEG	CP, CPC, CPI	BCLR C, BSET C, CLC, SEC	ASR, LSL, LSR, ROL, ROR	-
N	ADD, ADC, ADIW, DEC, INC, SUB, SUBI, SBC, SBCI, SBIW	AND, ANDI, OR, ORI, EOR, COM, NEG, SBR, CBR	CP, CPC, CPI	BCLR N, BSET N, CLN, SEN, TST	ASR, LSL, LSR, ROL, ROR	CLR
V	ADD, ADC, ADIW, DEC, INC, SUB, SUBI, SBC, SBCI, SBIW	AND, ANDI, OR, ORI, EOR, COM, NEG, SBR, CBR	CP, CPC, CPI	BCLR V, BSET V, CLV, SEV, TST	ASR, LSL, LSR, ROL, ROR	CLR
S	SBIW	-	-	BCLR S, BSET S, CLS, SES	-	-
H	ADD, ADC, SUB, SUBI, SBC, SBCI	NEG	CP, CPC, CPI	BCLR H, BSET H, CLH, SEH	-	-
T	-	-	-	BCLR T, BSET T, BST, CLT, SET	-	-
I	-	-	-	BCLR I, BSET I, CLI, SEI	-	RETI

Debugger

- **Belegte Anschlüsse:**
  PB7 (SCK)
  PC2 (TCK)
  PC3 (TMS)
  PC4 (TDO)
  PC5 (TDI)

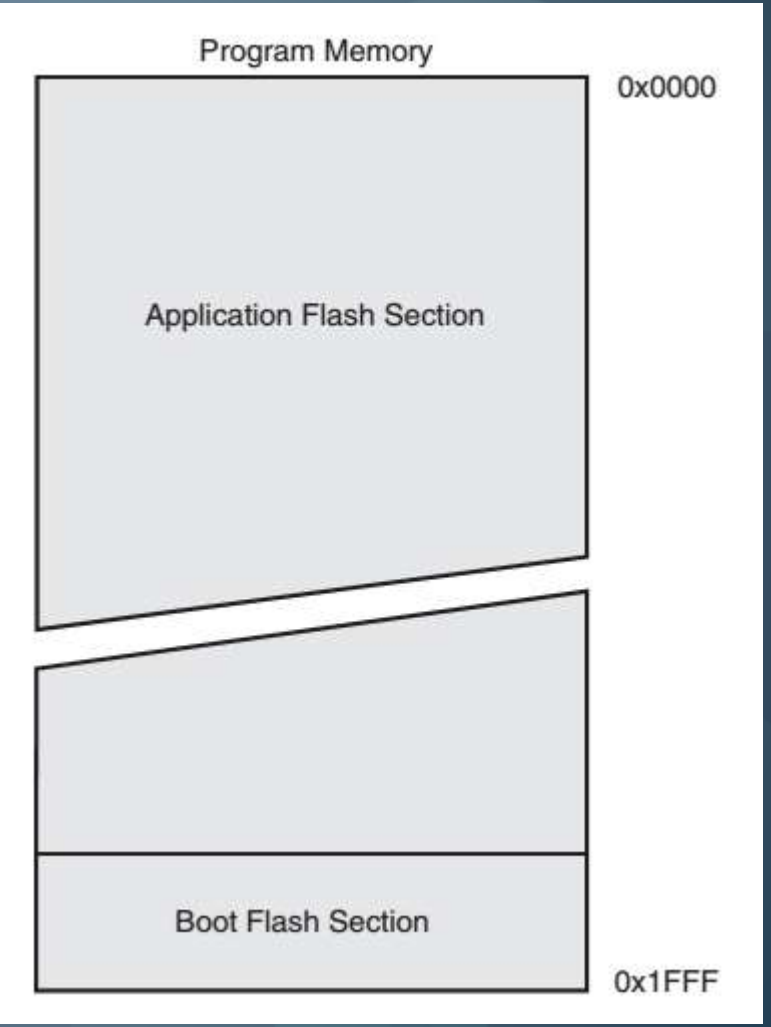
INTERNER SPEICHER

Interne Speicherarten

- Datenspeicher (4 kByte)
- Programmspeicher (64 kByte, 32k x 16 organisiert)
- EEProm-Speicher (2 kByte)

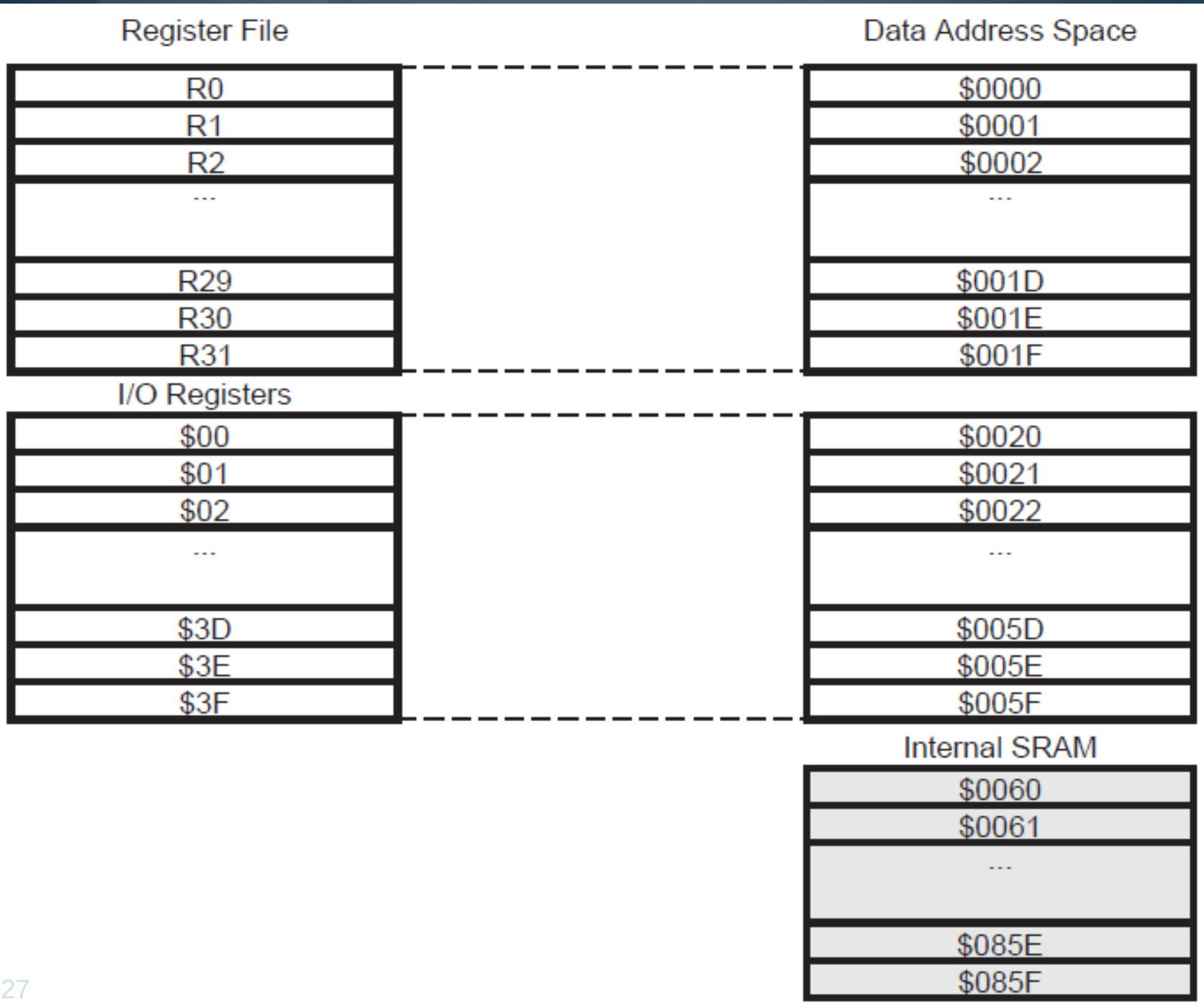
Flash-Speicher

- 64 kByte
- 2-Teile
- Boot – Bereich
- Application
- Programmcounter 16 Bit



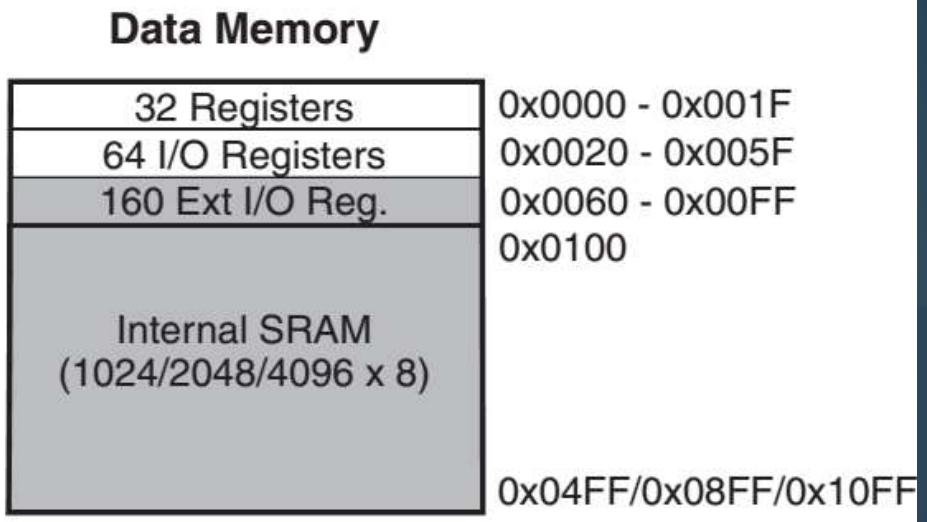
RAM

- 2144 Byte insgesamt
- 96 Byte Register File und I/O Bereich
- 2048 internes Data SRAM
- 5 verschiedene Adressierungsarten (direct, indirect, indirect with displacement, indirect with pre-decrement, indirect with Post-decrement)
- R26 bis R31 indirekte Adresszeiger-Register


Register File
R0
R1
R2
...
R29
R30
R31
Data Address Space
$0000
$0001
$0002
...
$001D
$001E
$001F
I/O Registers
$00
$01
$02
...
$3D
$3E
$3F
$0020
$0021
$0022
...
$005D
$005E
$005F
Internal SRAM
$0060
$0061
...
$085E
$085F

AVR CPU General Purpose Working Registers

7	0	Addr.	
R0		0x00	
R1		0x01	
R2		0x02	
…			
R13		0x0D	
R14		0x0E	
R15		0x0F	
R16		0x10	
R17		0x11	
…			
R26		0x1A	X-register Low Byte
R27		0x1B	X-register High Byte
R28		0x1C	Y-register Low Byte
R29		0x1D	Y-register High Byte
R30		0x1E	Z-register Low Byte
R31		0x1F	Z-register High Byte



EEPROM

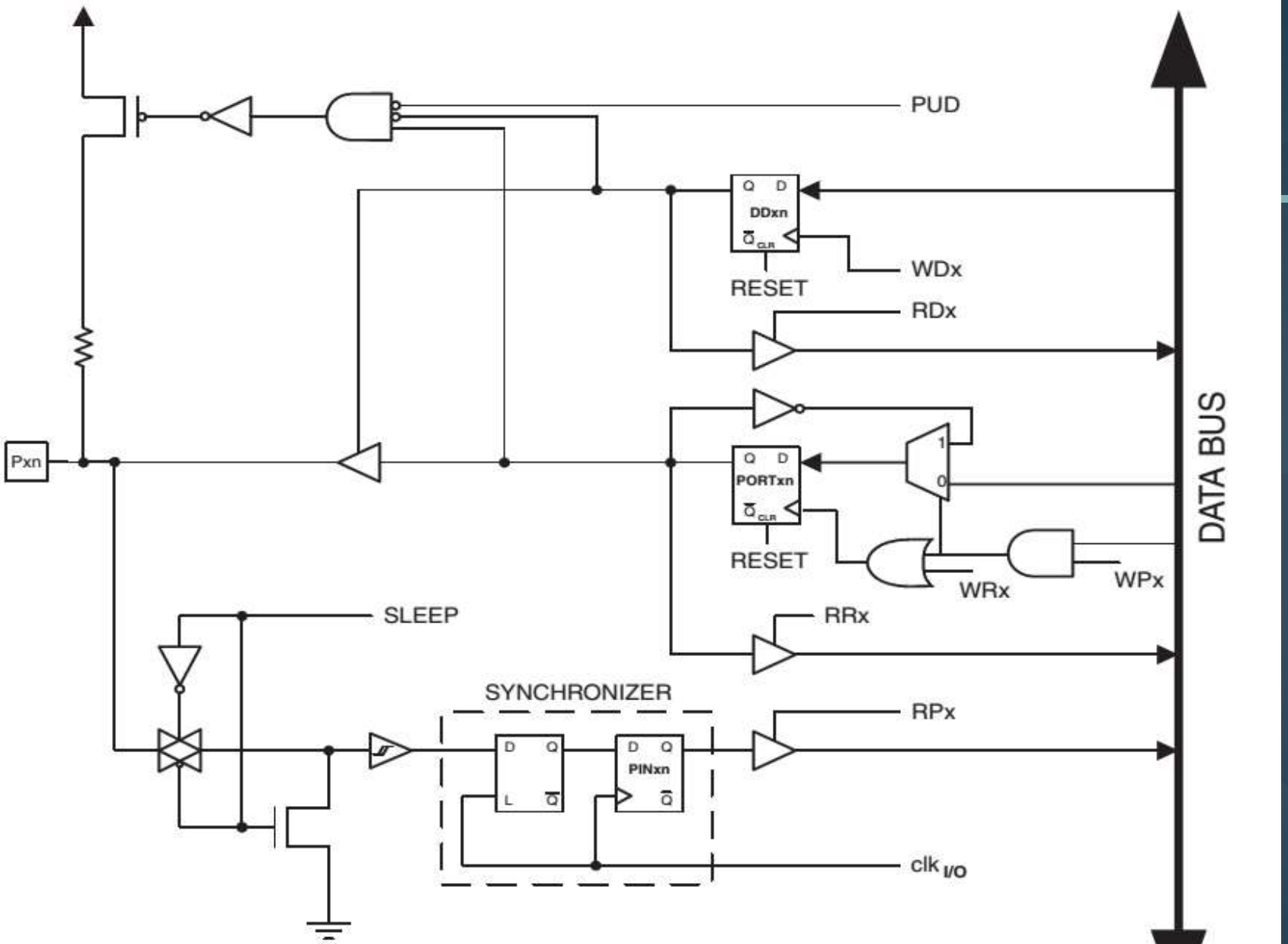
- 2 kByte EEPROM
- 100 000 Schreib / Löschzyklen

I/O Speicher

- Durch IN / OUT Befehle ansprechbar
- I/O Register im Adressbereich von 0 bis 0x1F sind Bitadressierbar (CBI, SBI,SBIS,SBIC instruction)
- Wenn die LD und ST Befehle verwendet werden, muss 0x20 zur Adresse der Register hinzugezählt werden-

I/O

- I/O Port Pins müssen initialisiert werden
- Jeder Pin kann unabhängig als Ausgang oder Eingang verwendet werden
- Pull-up Widerstände zuschaltbar
- 3 zuständige Register:
  PORTx Data Register
  DDRx Data Direction Register
  PINx Port Input Pins (read only)
- PUD Pull-up disable Bit in SFIOR


PUD
DDxn
RESET
WDx
RDx
Pxn
PORTxn
RESET
WRx
WPx
DATA BUS
SLEEP
RRx
SYNCHRONIZER
RPx
PINxn
clk I/O
PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
clk I/O: I/O CLOCK
WDx: WRITE DDRx
RDx: READ DDRx
WRx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN
WPx: WRITE PINx REGISTER

Initialisierung der Pins

- **DDRxn Data Direction Register Jeder Pin**
  "1": Output
  "0": Input
- **Bei Konfiguration als Eingang kann der Pull-up Widerstand durch Beschreiben des PORTxn Bits mit einer "1" eingeschalten werden**
  (aus mit "0")
- **Output: Der Inhalt des PORTxn Bits bestimmt den Zustand des Ausganges**
- **Toggling eines Pins:**
  Beim Schreiben einer 1 auf PINxn wird der Zustand von PORTxn getoggelt.

Initialisierung der Pins

- **Zusammenfassung:**

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Lesen der Pins

- **Das Lesen erfolgt über das PINxn Registerbit**
- **Zur Vermeidung einer Metastabilität wird intern synchronisiert (dadurch ergibt sich eine kleine Verzögerung)**
-

INTERRUPTS

Reset und Interrupt-Handling

Interrupt

- Interrupts sind Programmunterbrechungen, wie der Aufruf einer Funktion.
- Die Funktion wird allerdings nicht von der Software aufgerufen, sondern durch ein externes Ereignis.
- Beispiele:
  Tastendruck
  Empfang eines Zeichens über eine Schnittstelle
  Wandlung eines Analogwertes
  Ablauf eines Timers

Reset und Interrupt-Handling

- Jeder Interrupt hat seinen eigenen Programmvektor (Einsprungadresse) im Speicher
- Jeder Interrupt kann individuell enabled werden.
- Global Enable – Bit (Assembler: sei – Befehl)
- Verschiedene Interruptprioritäten (je niedriger die Interruptadresse ist, desto höher die Priorität)
- Antwort- und Returnzeit mindestens 4 Taktzyklen

Reset und Interrupt-Handling

- 2 Arten von Interrupts
  - 1.Interruptart setzt ein Interrupt-Flag, das durch Beschreiben mit 1 (oder auch automatisch) wieder zurückgesetzt wird
    Das automatische Zurücksetzen erfolgt beim Einsprung in die Interrupt Service Routine (ISR)
  - Bei der 2. Art ist der Interrupt solange aktiv, solange die Interruptbedingung erfüllt ist.

Interrupt-Vektortabelle

Vector No	Program Address(2)	Source	Interrupts definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	INT2	External Interrupt Request 2
5	0x0008	PCINT0	Pin Change Interrupt Request 0
6	0x000A	PCINT1	Pin Change Interrupt Request 1
7	0x000C	PCINT2	Pin Change Interrupt Request 2
8	0x000E	PCINT3	Pin Change Interrupt Request 3
9	0x0010	WDT	Watchdog Time-out Interrupt
10	0x0012	TIMER2_COMPA	Timer/Counter2 Compare Match A
11	0x0014	TIMER2_COMPB	Timer/Coutner2 Compare Match B
12	0x0016	TIMER2_OVF	Timer/Counter2 Overflow
13	0x0018	TIMER1_CAPT	Timer/Counter1 Capture Event
14	0x001A	TIMER1_COMPA	Timer/Counter1 Compare Match A
15	0x001C	TIMER1_COMPB	Timer/Coutner1 Compare Match B
16	0x001E	TIMER1_OVF	Timer/Counter1 Overflow
17	0x0020	TIMER0_COMPA	Timer/Counter0 Compare Match A
18	0x0022	TIMER0_COMPB	Timer/Coutner0 Compare Match B
19	0x0024	TIMER0_OVF	Timer/Counter0 Overflow
20	0x0026	SPI_STC	SPI Serial Transfer Complete

Interrupt-Vektortabelle

Vector No	Program Address(2)	Source	Interrupts definition
21	0x0028	USART_RX	USART Rx Complete
22	0x002A	USART_UDRE	USART Data Register Empty
23	0x002C	USART_TX	USART Tx Complete
24	0x002E	ANALOG_COMP	Analog Comparator
25	0x0030	ADC	ADC Conversion Complete
26	0x0032	EE_READY	EEPROM Ready
27	0x0034	TWI	TWI Transfer complete
28	0x0036	SPM_READY	Store Program Memory Ready
29	0x0038	USART1_RX	USART1 Rx Complete
30	0x003A	USART1_UDRE	USART1, Data Register Empty
31	0x003C	USART1_TX	USART1, Tx Complete

Vorraussetzungen zum Interrupt-Handling

- Der Interruptbetrieb erfordert einen Stack (SPH:SPL sind auf RAMEND zu setzen)
- Jeder Interrupt muss durch das entsprechende Enable –Bit freigeschalten werden.
- Jedem Interrupt wird eine Interrupt - Service-Routine (ISR) zugeordnet
- Jede ISR wird mit RETI beendet
- Jede ISR sichert das Statusregister
- Jeder nicht verwendete Interrupt soll mit RETI abgeschlossen werden

Interruptregister

- MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	JTD	BODS	BODSE	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Die Bits IVSEL und IVCE bestimmen den Ort der Interruptvektoren (Programm-Speicher oder Boot Bereich)

Externe Interrupts

- Externe Interrupts werden durch die Pins INT0, INT1, INT2 oder durch PCINT0 – PCINT31 ausgelöst.
- Auslösung durch fallende, steigende oder Low – Pegel, PCINT31 – PCINT0 nur flankengesteuert
- EICRA (External Interrupt Control Register A)
  EIMSK (External Interrupt Mask Register)
  EIFR (External Interrupt Flag Register)
  PCICR (Pin Change Interrupt Control Register)
  PCIFR (Pin Change Control Register)
  PCMSK0 – 3 (Pin Change Mask Register 0 to 3)
- Low-Level Interrupts an INT0 und INT1 können zum Aufwecken aus dem Sleep-Mode verwendet werden.

EICRA – External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 5:0 – ISC21, ISC20 – ISC01, ISC00 : Interrupt Sense Control für INT2, INT1 und INT0 Interrupt Pin

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any edge of INTn generates asynchronously an interrupt request.
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

EIMSK External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	INT2	INT1	IINT0	EIMSK
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 2 – INT2: External Interrupt Request 2 Enable
- Bit 1 – INT1: External Interrupt Request 1 Enable
- Bit 0 – INT0: External Interrupt Request 0 Enable

EIFR –External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	INTF2	INTF1	IINTF0	EIFR
Read/Write	R/W	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	PCIE3	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 3 – PCIE3: Pin Change Interrupt Enable 3

  When the PCIE3 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 3 is enabled. Any change on any enabled PCINT31..24 pin will cause an interrupt. PCINT31..24 pins are enabled individually by the PCMSK3 Register.
- • Bit 2 – PCIE2: Pin Change Interrupt Enable 2 : PCINT23..16 pin
- • Bit 1 – PCIE1: Pin Change Interrupt Enable 1 : PCINT15..8 pin
- • Bit 0 – PCIE0: Pin Change Interrupt Enable 0 : PCINT7..0 pin

The corresponding interrupt of Pin Change Interrupt Request is executed from the PCIn Interrupt Vector. PCINTxx pins are enabled individually by the PCMSKn Register.

PCIFR – Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)			–	–	PCIF3	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 3:0 – PCIFn: Pin Change Interrupt Flag n
  When a logic change on any PCINT31..24 (PCINT23..16, PCINT15..8, PCINT7..0) pin triggers an interrupt request, PCIFn becomes set (one). If the I-bit in SREG (sei) and the PCIEn bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- Bits 7:0 – PCINTn: Pin Change Enable Mask [n = 7:0] Each PCINT[7:0] bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is set and the PCIE0 bit in PCICR is set, in change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is cleared, pin change interrupt on the corresponding I/O pin is disabled.
- PCMSK1 to PCMSK3 for the other pins.

8-BIT TIMER / COUNTER

Timer / Counter mit PWM

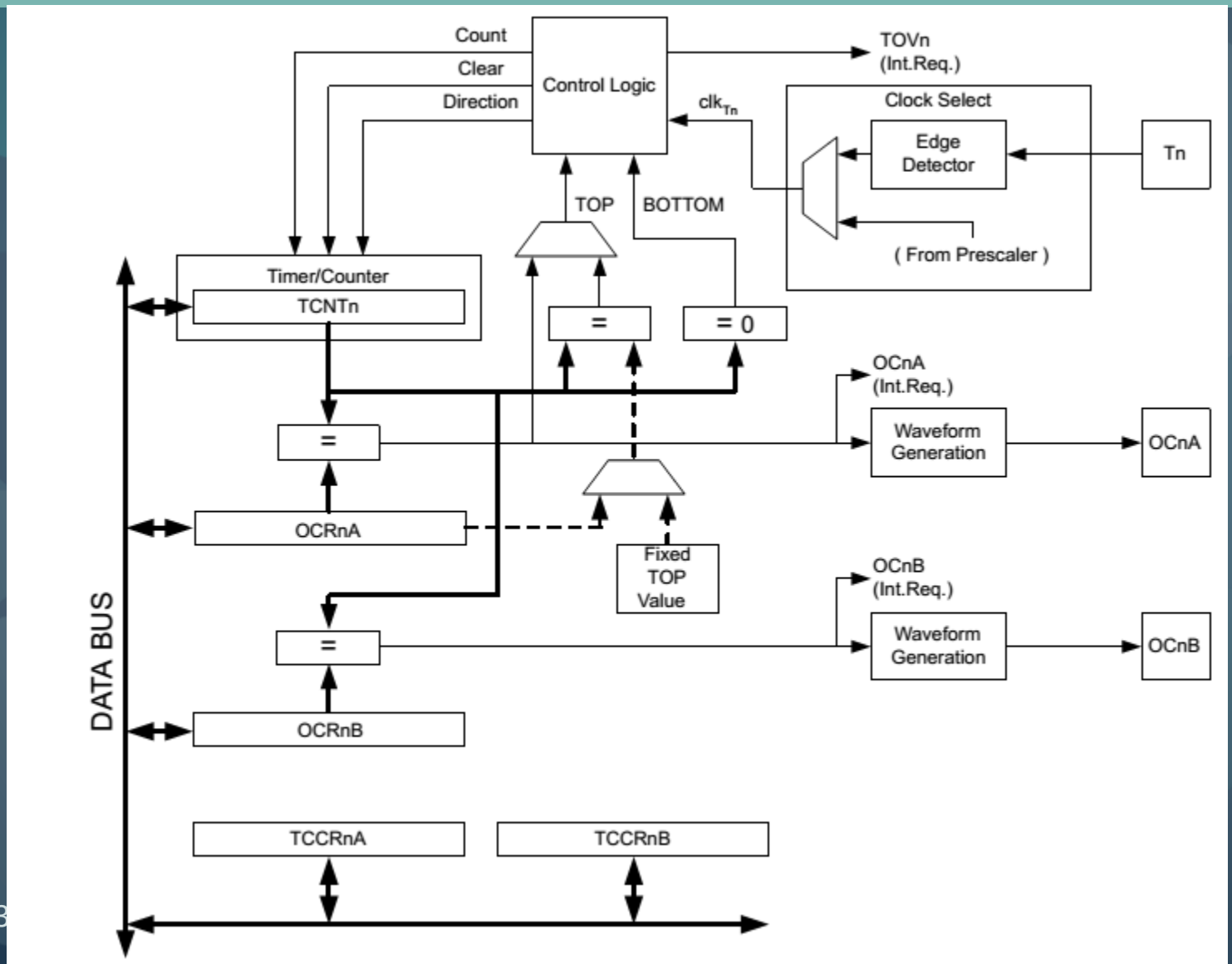
Timer / Counter

- Vorgänge in bestimmten Zeitabständen durchzuführen
- Laufen unabhängig von der CPU
- Hohe Genauigkeit (abhängig von der Taktquelle)
- Ein Timer ist ein Register im µC, das inkrementiert (oder auch dekrementiert) wird
- Bei bestimmten Zählerwerten wird ein Interrupt ausgelöst (z.B. bei einem Overflow 0xFF -> 0x00)
- Das Timerregister kann auch Signale zählen (von einem I/O Pin)
- 8-Bit Timer: Auflösung 256
- 16-Bit Timer: Auflösung 65536
- Eingang für Takt: CPU Taktfrequenz, Vorteiler-Ausgang, Signal vom I/O Pin

Eigenschaften

- 8-Bit Timer / Counter mit PWM Unterstützung
- Overflow und Compare Match Interrupt
- Variable PWM (Pulsweitenmodulation)
- Glitch-freier, Phasenkorrekter Pulseweiten-Modulator (PWM)
- Frequenzgenerator
- Ereigniszähler
- 10 Bit Clock – Vorteiler
- 3 unabhängige Interruptquellen

Schaltung (vereinfacht)



Register

- TCNT0 : Timer/Counter Register
- TC0A, TC0B: Output Compare Register
- TCCR0A, TCCR0B: Timer/Counter Control Register, Prinzipielle Funktion des Timers
- OCR0A, OCR0B: Output Compare Register
- TIFR0: Timer Interrupt Flag Register
- TIMSK0: Timer Interrupt Mask Register

Clock

- Der Timer kann durch verschiedene Quellen getaktet werden:
- Interner Clock mit Vorteiler
- externer Clock auf Pin To

Vorteiler

- Der Vorteiler dient dazu, den CPU-Takt vorerst um einen einstellbaren Faktor zu reduzieren. Die so geteilte Frequenz wird den Eingängen der Timer zugeführt.
- Wenn wir mit einem CPU-Takt von 4 MHz arbeiten und den Vorteiler auf 1024 einstellen, wird also der Timer mit einer Frequenz von 4 MHz / 1024, also mit ca. 4 kHz versorgt. Wenn also der Timer läuft, so wird das Daten- bzw. Zählregister (TCNTn) mit dieser Frequenz inkrementiert.

Funktionsarten des Timers

- Normal Mode:
  Der Counter zählt bis 0xFF und fängt wieder von 0 an. Beim Overflow wird ein Interrupt generiert (falls freigegeben)
- Clear Timer on Compare Match (CTC) Mode:
  Der Counter wird auf 0 gesetzt, wenn der Zähler den Inhalt des OCR0A/ OCR0B Registers erreicht. Dadurch kann die Rücksetzfrequenz variiert werden.

Output Compare Unit

- Das Zählerregister TCNT0 wird permanent mit dem Output Compare Register (OCR0A/B) verglichen.
- Der 8 – Bit Komparator meldet einen Match, dadurch wird das OCFA/B Flag beim nächsten Taktzyklus gesetzt.
- Wenn OCIEA/B = 1 und das Global Interrupt Flag gesetzt ist, wird ein Interrupt erzeugt.
- Beim Ausführen des Interrupts wird OCFA/B automatisch gelöscht.
- Zum manuell löschen muss das OCFA/B Bit mit 1 beschrieben werden.

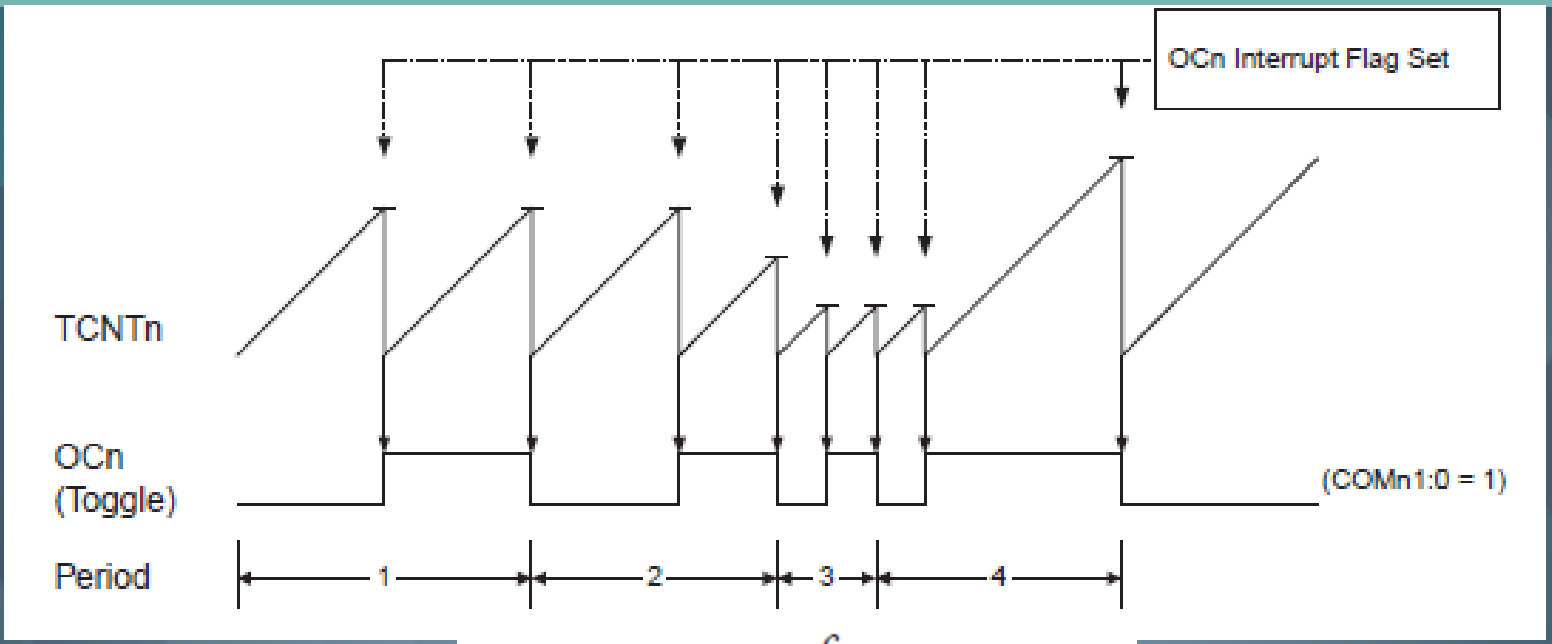
Normal

- TCNT0 zählt immer aufwärts
- TCNT hat normalen Überlauf
- Timer/Counter Overflow Flag (TOV0) wird beim Überlauf gesetzt, und damit ein Interrupt ausgelöst
- WGM02:0 = 0

Output Compare Modus CTC - Mode

- WGM02:0 = 2
- TCNT0 zählt immer aufwärts
- TCNT wird beim Erreichen des Wertes vom OCR0A -Register auf 0 gesetzt.
- Das OCR0A – Register bestimmt daher den maximalen Wert für das TCNT0 Register
- Das OCF0 Flag wird beim Rücksetzen von TCNT gesetzt und damit ein Interrupt ausgelöst (falls er freigegeben ist, OCIE0 Flag im TIMSK-Register)
- Mit diesem Mode kann daher eine variable Interruptzeit eingestellt werden.
- Wenn COM0A1:0 = 1, OC0A wird getoggelt (Port B, Pin 3), damit kann eine variable Frequenz ausgegeben werden. Datadirection muss auf OUT gesetzt werden.

Output Compare Modus
CTC - Mode



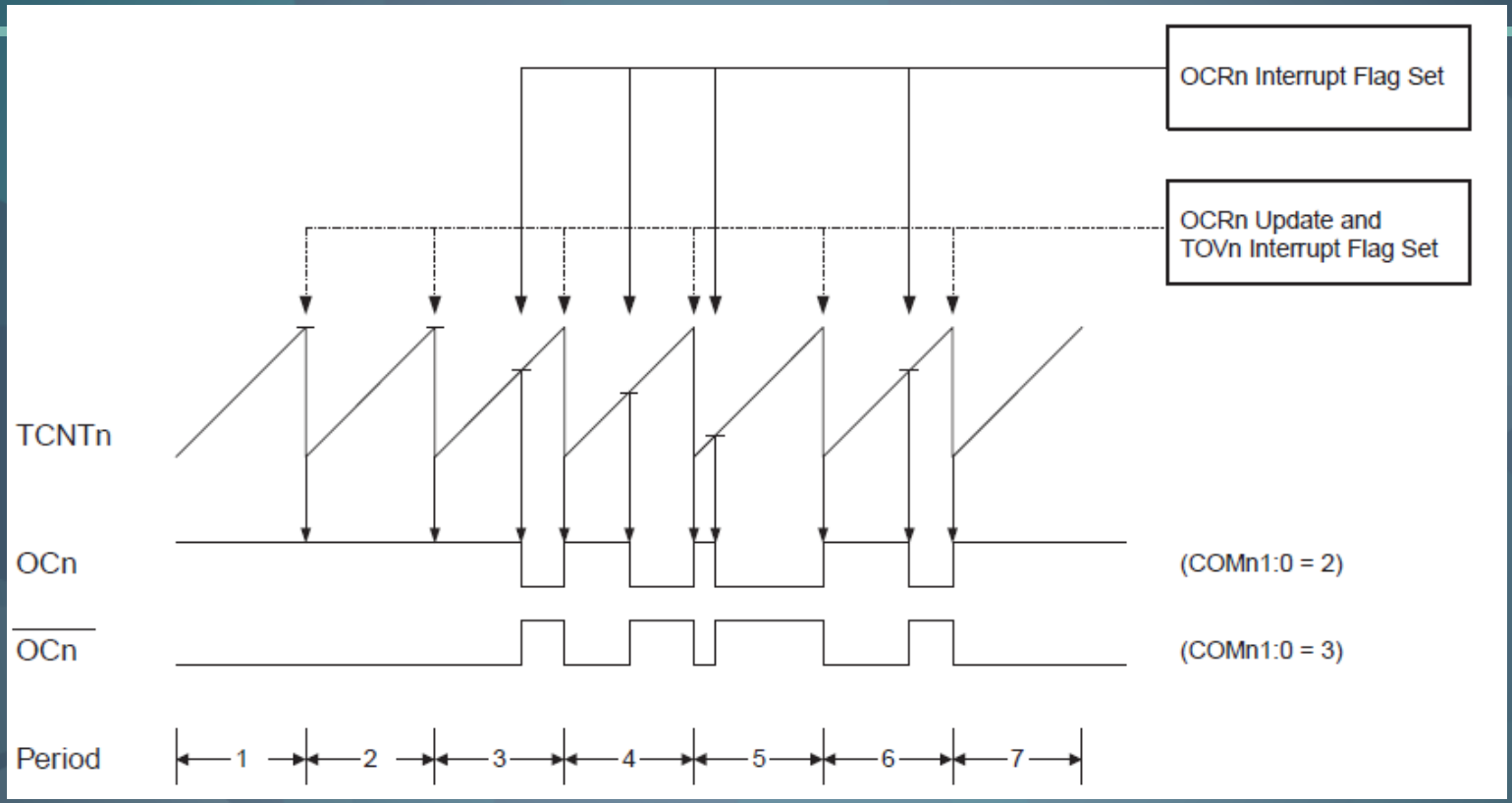
$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

- N: Vorteilerfaktor (1, 8, 64, 256, or 1024)
- Wird das OCR0A - Register im Interrupt gesetzt, kann die Frequenz dynamisch geändert werden.

Output Compare Moden
FAST PWM - Mode

- WGM02:0 = 3
- TCNT0 zählt immer von 0 bis 0xFF
- Non-inverting Mode (COM0A1:0 = 2) : Das OCF0 Flag wird bei Gleichstand TCNT0 = OCR0A gelöscht und beim Überlauf vom TCNT0 Register gesetzt.
- Inverting Mode (COM01:0 = 3) : Das OCF0 Flag wird bei Gleichstand TCNT0 = OCR0A gesetzt und beim Überlauf vom TCNT0 Register gelöscht.
- Timer/Counter Overflow Flag (TOV0) wird beim Überlauf gesetzt, und damit ein Interrupt ausgelöst. Im Interrupt kann das OCR0A Register neu gesetzt werden.

Output Compare Moden FAST PWM - Mode



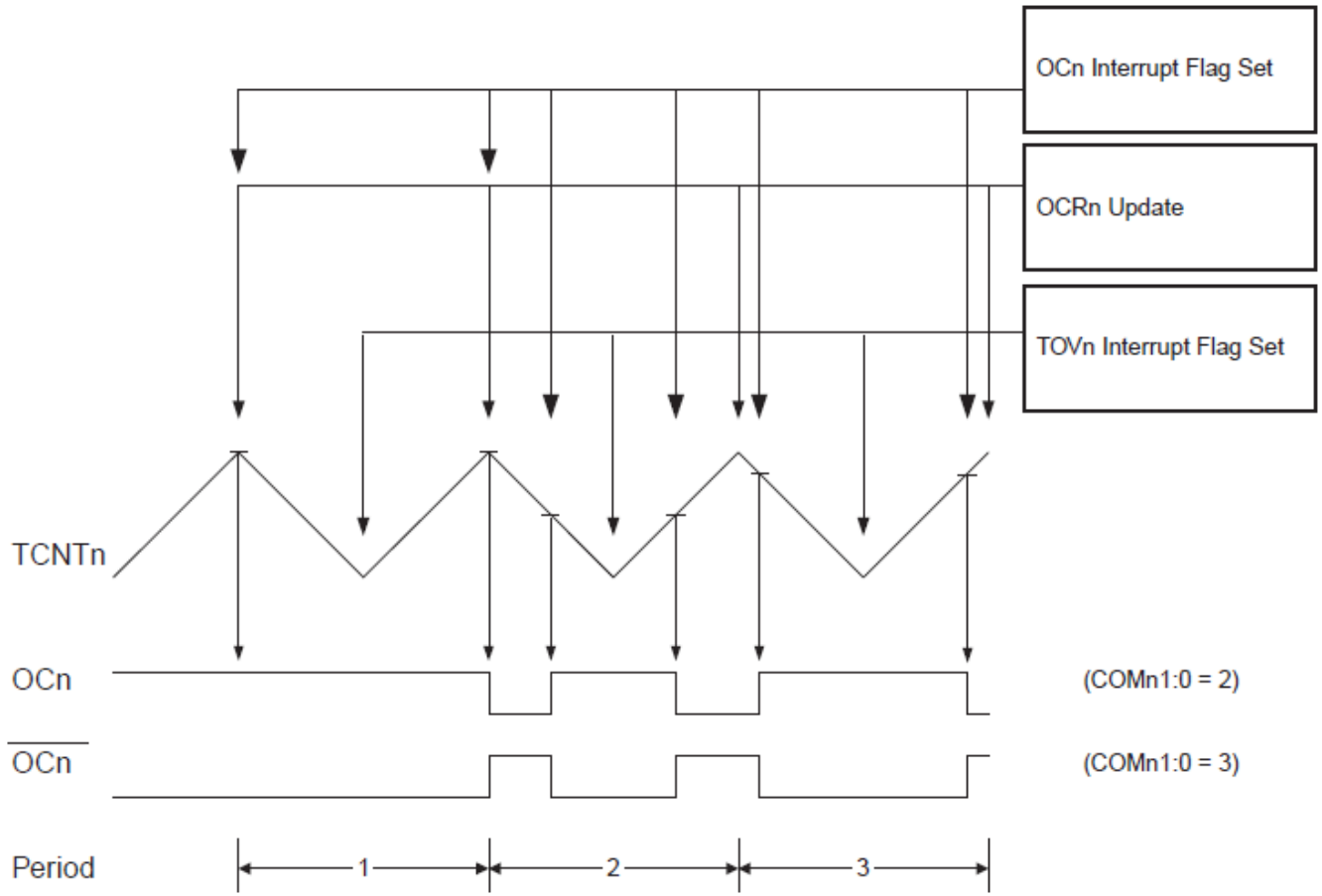
$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

- N: Vorteilerwert: 1,8,64,1024

Output Compare Moden
Phase Correct PWM - Mode

- WGM02:0 = 1
- TCNT0 zählt von 0 bis 0xFF und dann von 0xFF bis 0
- Non-inverting Mode (COM0A1:0 = 2) : Das OCF0 Flag wird bei Gleichstand TCNT0 = OCR0A und beim Hinaufzählen gelöscht und beim beim Hinunterzählen und Gleichstand von TCNT0 = OCR0A gesetzt.
- Inverting Mode (COM0A1:0 = 3) : Genau umgekehrt.
- Timer/Counter Overflow Flag (TOV0) wird beim Überlauf gesetzt, und damit ein Interrupt ausgelöst. Im Interrupt kann das OCR0 Register neu gesetzt werden.

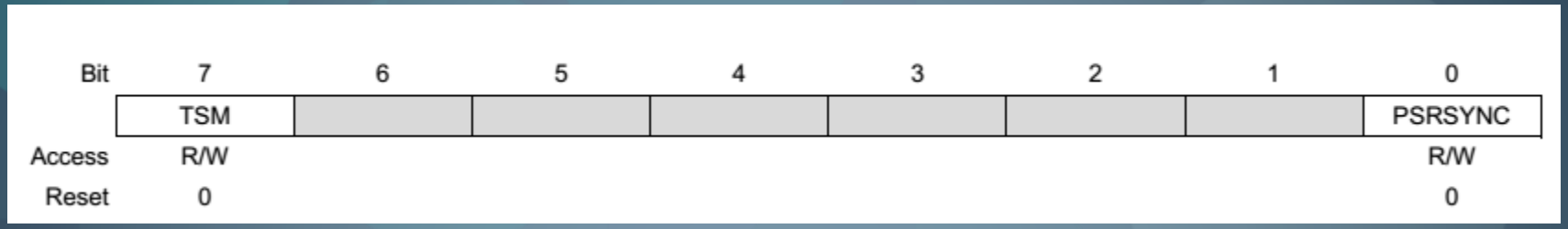
Output Compare Moden
Phase - Correct PWM - Mode



$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

- N: Vorteilerwert: 1,8,64,1024

Vorteiler – Reset-Bit im GTCCR



Bit	7	6	5	4	3	2	1	0
	TSM							PSRSYNC
Access	R/W							R/W
Reset	0							0

- GTCCR : General Timer/Counter Control Register
- TSM: Timer/Counter Synchronization Mode Prescaler gestopped, wenn PSRSYNC auf 1
- PSRSYNC : Wenn eine 1 darauf geschrieben wird, wird der Prescaler für Timer 0 und Timer 1 zurückgesetzt. Das Bit wird wieder automatisch gelöscht, wenn TSM auf 0 ist.

Register TCCR0B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 0 – 2: CS02, CS01, CS00 (Clock Select Bits)

CS02	CS01	CS00	Resultat
0	0	0	Stopp, Der Timer/Counter wird angehalten.
0	0	1	CPU-Takt
0	1	0	CPU-Takt / 8
0	1	1	CPU-Takt / 64
1	0	0	CPU-Takt / 256
1	0	1	CPU-Takt / 1024
1	1	0	Externer Pin **T0**, fallende Flanke
1	1	1	Externer Pin **T0**, steigende Flanke

Register TCCR0A

Bit	7	6	5	4	3	2	1	0
	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

- COM0An: Compare Output Mode for Channel A [n = 1:0] The function of the COM0A[1:0] bits depends on the WGM0[2:0] bit
- Compare Output Mode, non PWM:

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match.
1	1	Set OC0A on Compare Match .

Register TCCR0A

- Compare Output Mode, fast PWM:

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM (inverting mode)

Register TCCR0A

- WGM00, WGM01, WGM02 Waveform Generation Mode Bits
- 2 Operationsmoden: Normal mode, Clear Timer on Compare Match (CTC)
- 2 Puls Width Modulation (PWM) Moden

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Register TCCR0A

- Bit 7, 6, (5,4): COM0A1:0, COM0B1:0 Compare Match Output Mode Bits
- Non-PWM Mode:

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match.
1	1	Set OC0A on Compare Match .

Register TCCR0

- Bit 7, 6, (5,4): COM0A1:0, COM0B1:0 Compare Match Output Mode Bits
- Fast-PWM Mode:

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM (inverting mode)

- Weitere Modi im Referezmanual nachlesen

Register
TCNT0 – Timer Counter Register

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Das TCNT0 Register kann gelesen und beschrieben werden.

Register
OCR0A/B – Output Compare Register

Bit	7	6	5	4	3	2	1	0
	OCR0A[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- Der Wert des OCR0A/B Registers wird dauernd mit dem TCNT0 Register verglichen. Wenn der Wert übereinstimmt, kann ein Interrupt erzeugt werden.
- Das Register wird auch dazu verwendet, um eine PWM am Ausgang zu erzeugen.
- Das Register ist bei gewissen Betriebsarten doppelt-gebuffert

TIMSK0 – TC0 Timer Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
						OCIEB	OCIEA	TOIE
Access						R/W	R/W	R/W
Reset						0	0	0

- Bit 2 – OCIB: Timer/Counter0 Output Compare Match Interrupt Enable
  Dieses Bit schaltet den Interrupt für eine Übereinstimmung zwischen dem TCNT0 und dem OCR0B Register ein.
- Bit 1 – OCIA: Timer/Counter0 Output Compare Match Interrupt Enable
  Dieses Bit schaltet den Interrupt für eine Übereinstimmung zwischen dem TCNT0 und dem OCR0A Register ein.
- Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable
  Dieses Bit schaltet den Interrupt für einen Überlauf des TCNT0 Registers ein.

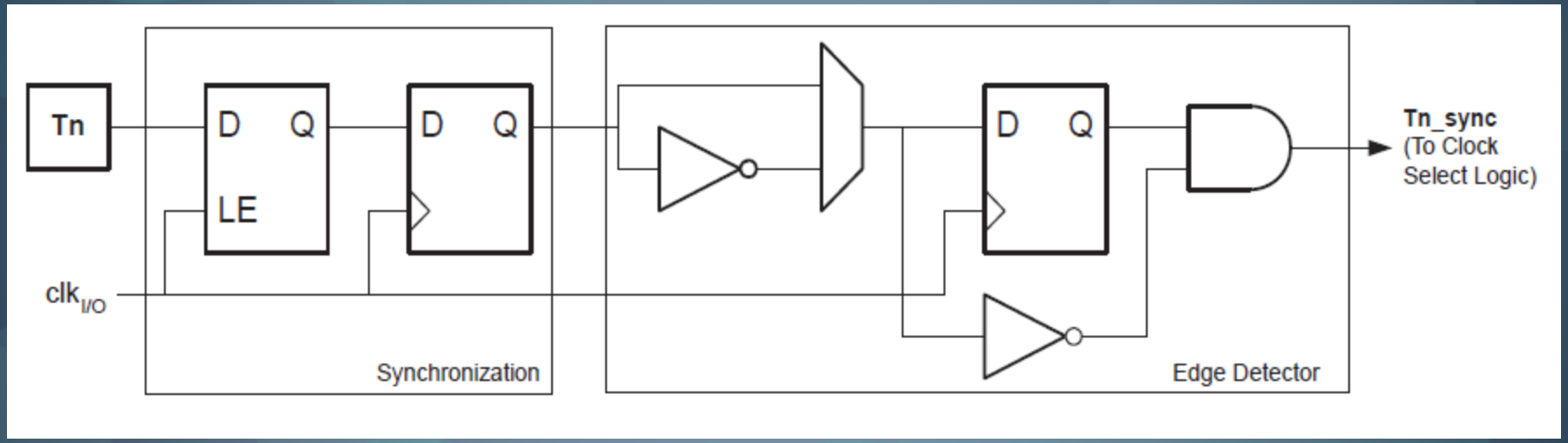
Register
TIFR0 – Timer Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0
						OCFB	OCFA	TOV
Access						R/W	R/W	R/W
Reset						0	0	0

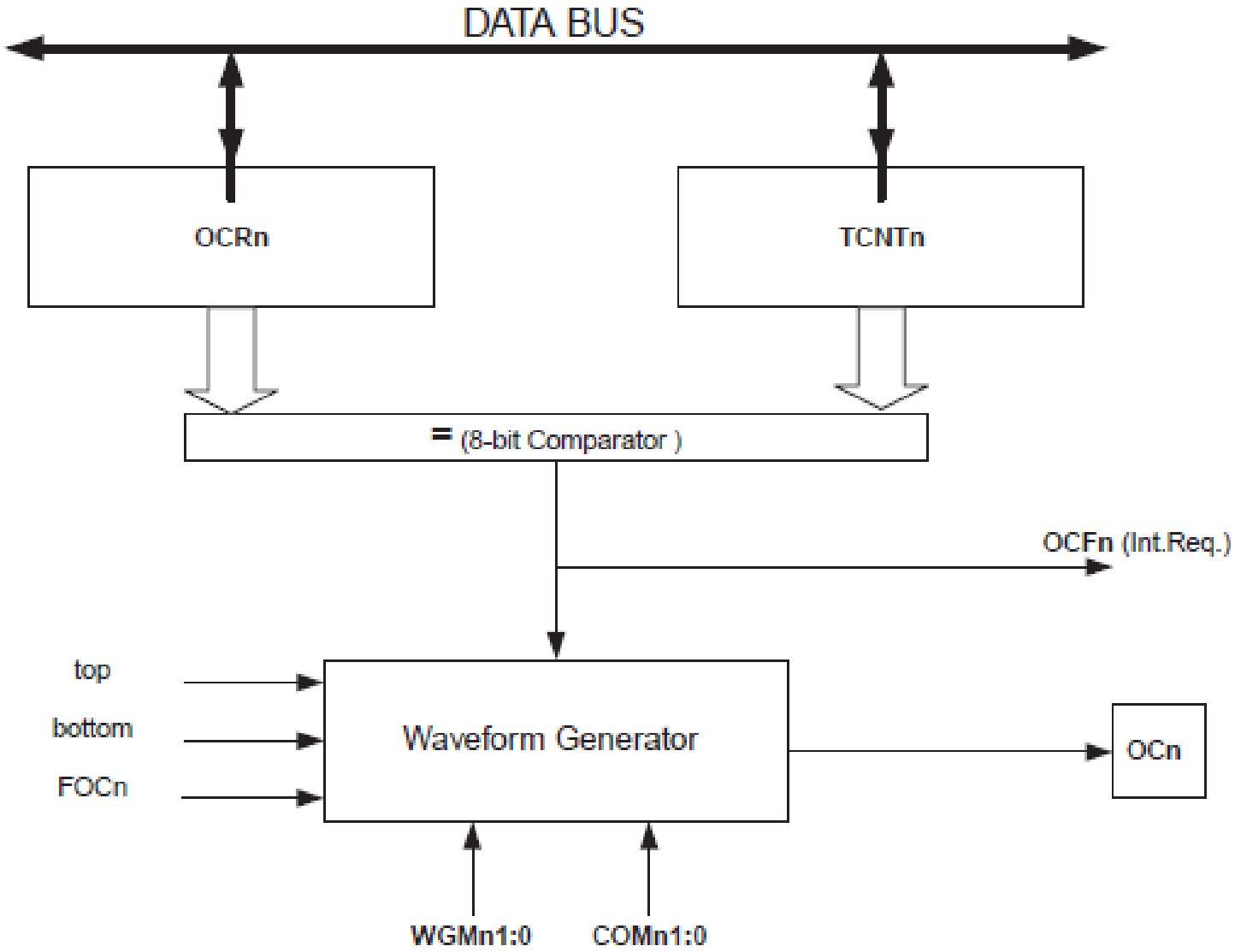
- Bit 2 – OCFB: Output Compare B Match Flag.
  Dieses Bit wird gesetzt, wenn das Zählerregister TCNT0 und das OCR0B Register überein stimmen.
- Bit 1 – OCFA: Output Compare A Match Flag.
  Dieses Bit wird gesetzt, wenn das Zählerregister TCNT0 und das OCR0A Register überein stimmen.
- Bit 0 – TOV0: Timer/Counter0 Overflow Flag

  Dieses Bit wird gesetzt, wenn das TCNT0 Register einen Überlauf hat.

Externer Clock



Output Compare Unit, Blockschaltbild

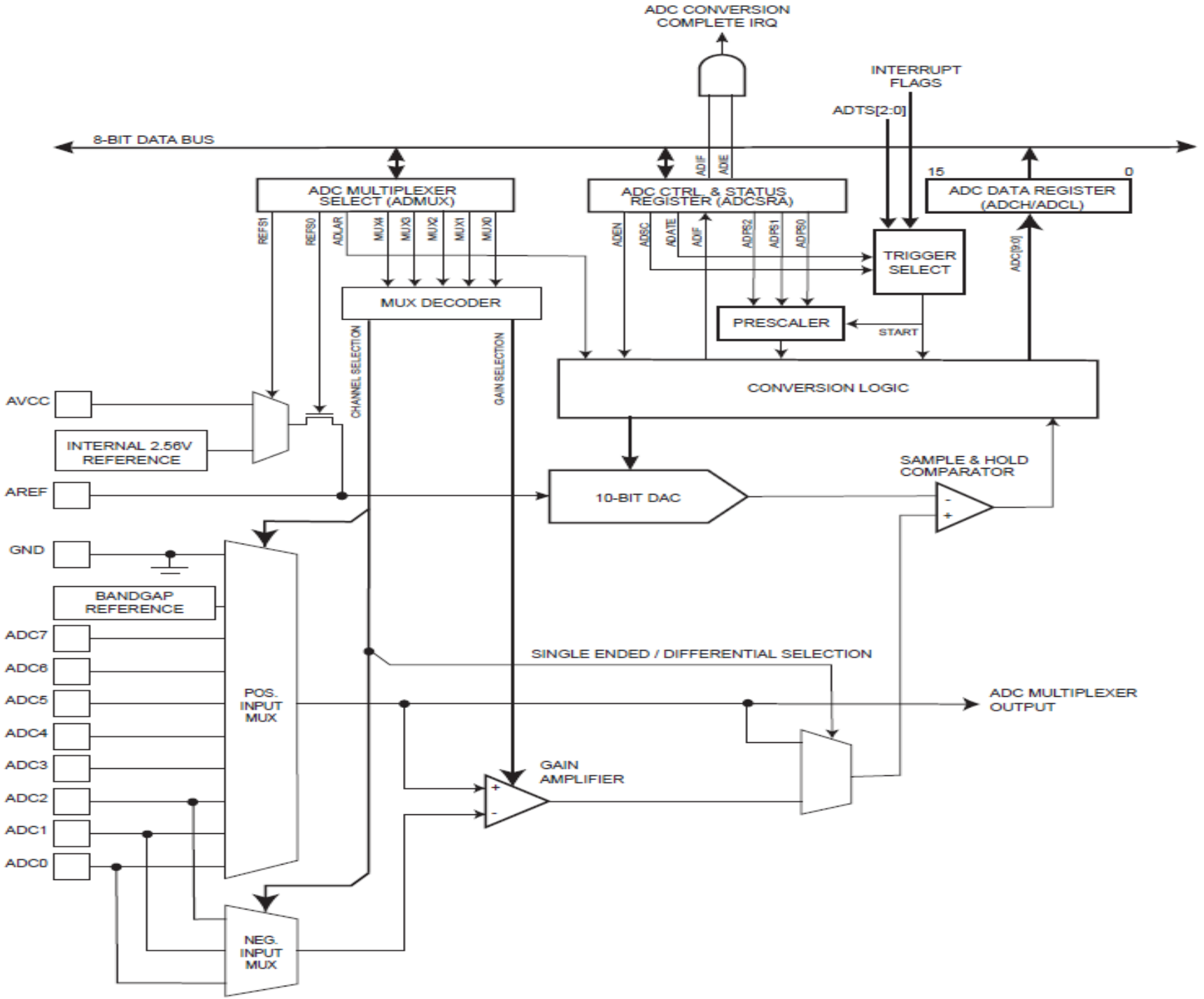


ADC

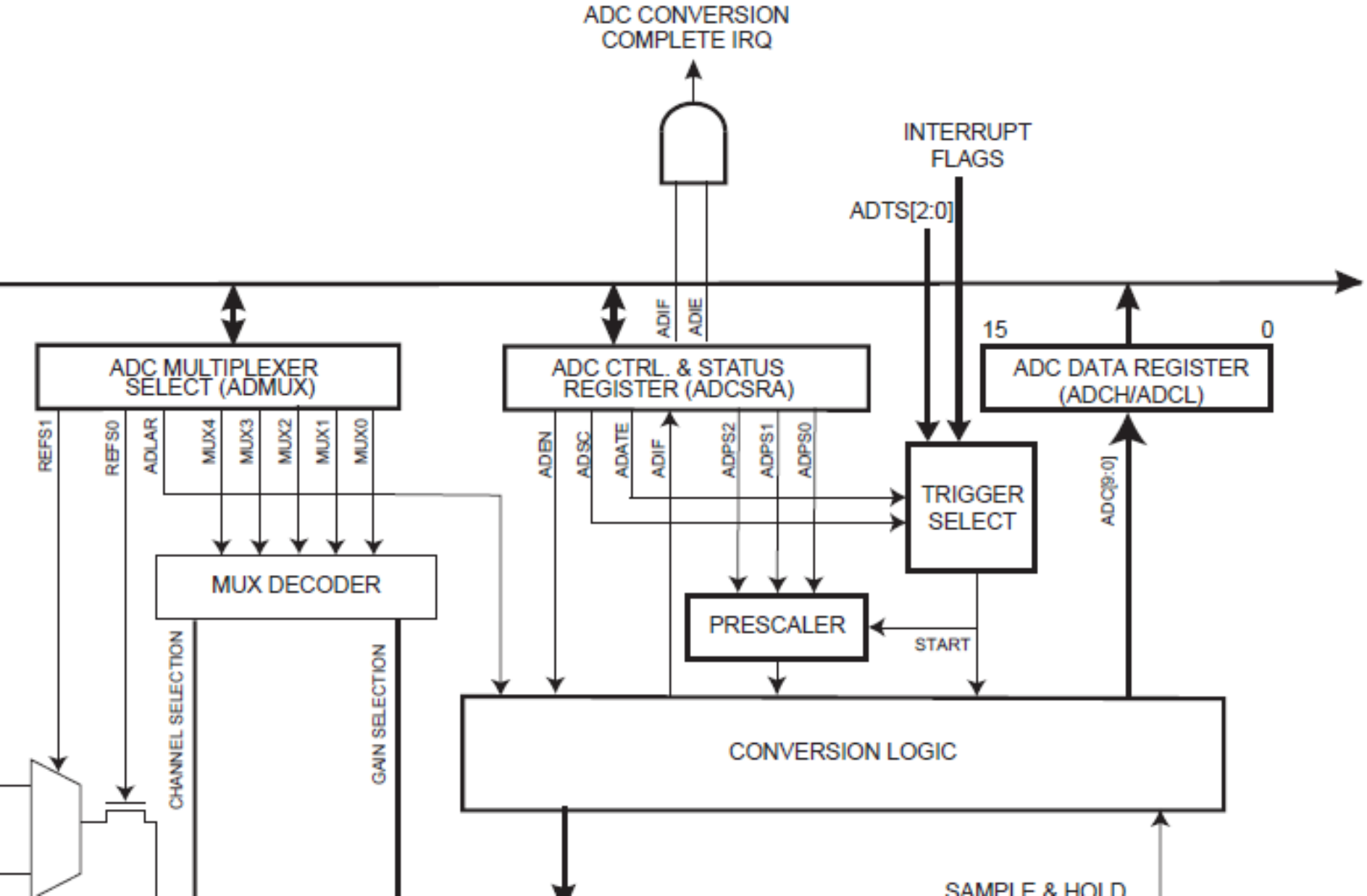
Analog – Digital Konverter

ADC
Analog – Digital Konverter

- 10 Bit Auflösung
- 0.5 LSB Integrale Nichtlinearität
- +/- 2 LSB absolute Genauigkeit
- 13 – 260 us Konversionszeit
- bis 15 000 Samples / s
- 8 Multiplexed Eingänge (Port A)
- 7 Differentielle Eingänge
- 2 differentielle Eingänge mit variabler Verstärkung von 1x, 10x und 200x
- 0 – Vcc ADC Referenzspannung
- Eingebaute 2.56V Referenzspannung
- Free Running – Single Conversion
- Interrupt, wenn Wandlung fertig ist
- Sleep Mode Rauschunterdrückung


ADC CONVERSION COMPLETE IRQ
INTERRUPT FLAGS
ADTS[2:0]
8-BIT DATA BUS
ADIF
ADIE
15
0
ADC MULTIPLEXER SELECT (ADMUX)
ADC CTRL. & STATUS REGISTER (ADCSRA)
ADC DATA REGISTER (ADCH/ADCL)
REFS1
REFS0
ADLAR
MUX4
MUX3
MUX2
MUX1
MUX0
ADEN
ADSC
ADATE
ADIF
ADPS2
ADPS1
ADPS0
TRIGGER SELECT
ADC[9:0]
MUX DECODER
PRESCALER
START
CHANNEL SELECTION
GAIN SELECTION
CONVERSION LOGIC
AVCC
INTERNAL 2.56V REFERENCE
AREF
10-BIT DAC
SAMPLE & HOLD COMPARATOR
GND
BANDGAP REFERENCE
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
POS. INPUT MUX
SINGLE ENDED / DIFFERENTIAL SELECTION
ADC MULTIPLEXER OUTPUT
GAIN AMPLIFIER
NEG. INPUT MUX

ADC


ADC CONVERSION COMPLETE IRQ
INTERRUPT FLAGS
ADTS[2:0]
ADIF
ADIE
15
0
ADC MULTIPLEXER SELECT (ADMUX)
ADC CTRL. & STATUS REGISTER (ADCSRA)
ADC DATA REGISTER (ADCH/ADCL)
REFS1
REFS0
ADLAR
MUX4
MUX3
MUX2
MUX1
MUX0
ADEN
ADSC
ADATE
ADIF
ADPS2
ADPS1
ADPS0
TRIGGER SELECT
ADC[9:0]
MUX DECODER
PRESCALER
START
CHANNEL SELECTION
GAIN SELECTION
CONVERSION LOGIC
SAMPLE & HOLD

ADC
Analog – Digital Konverter

- Sukzessive Aproximation
- Minimale Spannung: GND
  Maximale Spannung: AREF – 1 LSB
- AREF soll mit einem Kondensator versehen werden (Rauschunterdrückung)
- Jeder ADC Eingang, wie auch GND und eine konstante Bandgap-Referenz kann als Eingang ausgewählt werden.
- Einige ausgewählte ADC – Eingänge können sowohl als negativer wie positiver Eingang im differentiellen Mode gewählt werden.

ADC
Analog – Digital Konverter - Start

- Einstellen der Referenzspannung im ADMUX Register (REFS1, REFS0 Bit)
- Der ADC wird durch ADEN im ADCSRA Register eingeschalten
- Ergebnis der Wandlung in den Registern: ADCH, ADCL
  (zuerst ADCL und dann ADCH lesen)
- Start der Konversion:
  Schreiben einer 1 auf das ADSC Bit im ADCSRA Control Register

ADC
ADMUX Multiplexer, Ref-Voltage,

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

MUX4:0	Single Ended Input
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

- Alle anderen Kombinationen im atmega32.pdf File auf Seite 223 nachlesen (Kap. 22.9.1)

ADC
ADCSRA - ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN**: ADC Enable
  Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off.
- **Bit 6 – ADSC**: ADC Start Conversion
  In Single Conversion mode, write this bit to one to start each conversion. In Free Running Mode, write this bit to one to start the first conversion. ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero.
- **Bit 5 – ADATE**: ADC Auto Trigger Enable
- **Bit 4 – ADIF**: ADC Interrupt Flag
  This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag.
- **Bit 3 – ADIE**: ADC Interrupt Enable
  When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

ADC
ADCSRA - ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 2:0** – ADPS2:0: ADC Prescaler Select Bits
  These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADC

- Die erweiterten Funktionalitäten bitte im atmega32.pdf Manual nachlesen

- Differentielles Messen
- Autotriggermode

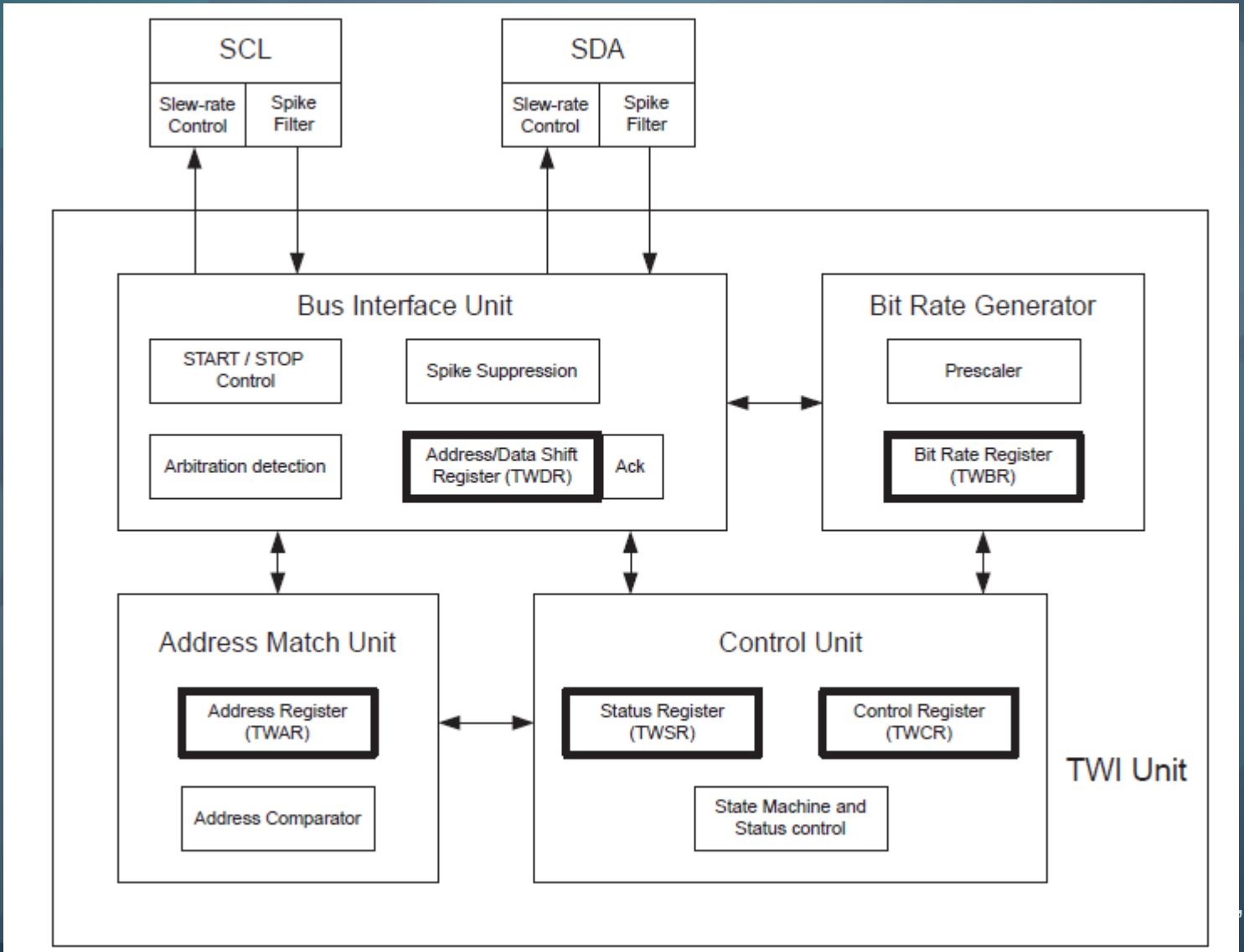
I2C BUS

Two Wire Interface

I2C Bus (Two-wire Serial Interface)

- Simple Yet Powerful and Flexible Communication Interface,
- Both Master and Slave Operation Supported
- Device Can Operate as Transmitter or Receiver
- 7-bit Address Space allows up to 128 Different Slave Addresses
- Multi-master Arbitration Support
- Up to 400 kHz Data Transfer Speed
- Slew-rate Limited Output Drivers
- Noise Suppression Circuitry Rejects Spikes on Bus Lines
- Fully Programmable Slave Address with General Call Support
- Address Recognition causes Wake-up when AVR is in Sleep Mode

I2C Bus (Two-wire Serial Interface)



I2C Bus (Two-wire Serial Interface)

- SCL, SDA Pins
  Interne pull-up Widerstände können verwendet werden. Dadurch kann die Verwendung von externen pull-up Widerständen eventuell vermieden werden.
- Bit Generator (Masterbetrieb)
  TWI Bit Rate Register (TWBR) und Prescaler Bit im TWI Status Register (TWSR)

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

  - TWBR = Wert des TWI Bit Rate Register
  - TWPS = Wert des Prescaler-Bits im TWI Status Register

I2C Bus (Two-wire Serial Interface)

- Bus Interface Unit
  Address Shift Register (TWDR)
  Sendebyte oder Empfangsbyte
  ACK/NACK Register durch das TWI Control Register (TWCR) auslesbar.
- Start / Stop Controller
  Start, Stop und Repeated Start Condition
  Wakeup – Funktion
- Address Match Unit
  Überprüft die Übereinstimmung mit der 7-Bit Adresse im TWI Address Register (TWAR)

I2C Bus (Two-wire Serial Interface)

- Control Unit
  Überwacht den I2C Bus
  Bei einem Ereignis wird das TWI Interrupt Flag gesetzt (TWINT) und das TWI Status Register beschrieben, um den Event zu signalisieren. Die Übertragung wird angehalten, solange das TWINT Flag gesetzt ist.

I2C Bus (Two-wire Serial Interface)

- Events der Control Unit:
  - After the TWI has transmitted a START/REPEATED START condition
  - After the TWI has transmitted SLA+R/W
  - After the TWI has transmitted an address byte
  - After the TWI has lost arbitration
  - After the TWI has been addressed by own slave address or general call
  - After the TWI has received a data byte
  - After a STOP or REPEATED START has been received while still addressed as a slave
  - When a bus error has occurred due to an illegal START or STOP condition

I2C Bus (Two-wire Serial Interface)

- Ablauf der I2C Bus – Übertragung siehe atmega32.pdf Seite 184, Kapitel "Using the TWI"

USART

Serielle Schnittstelle

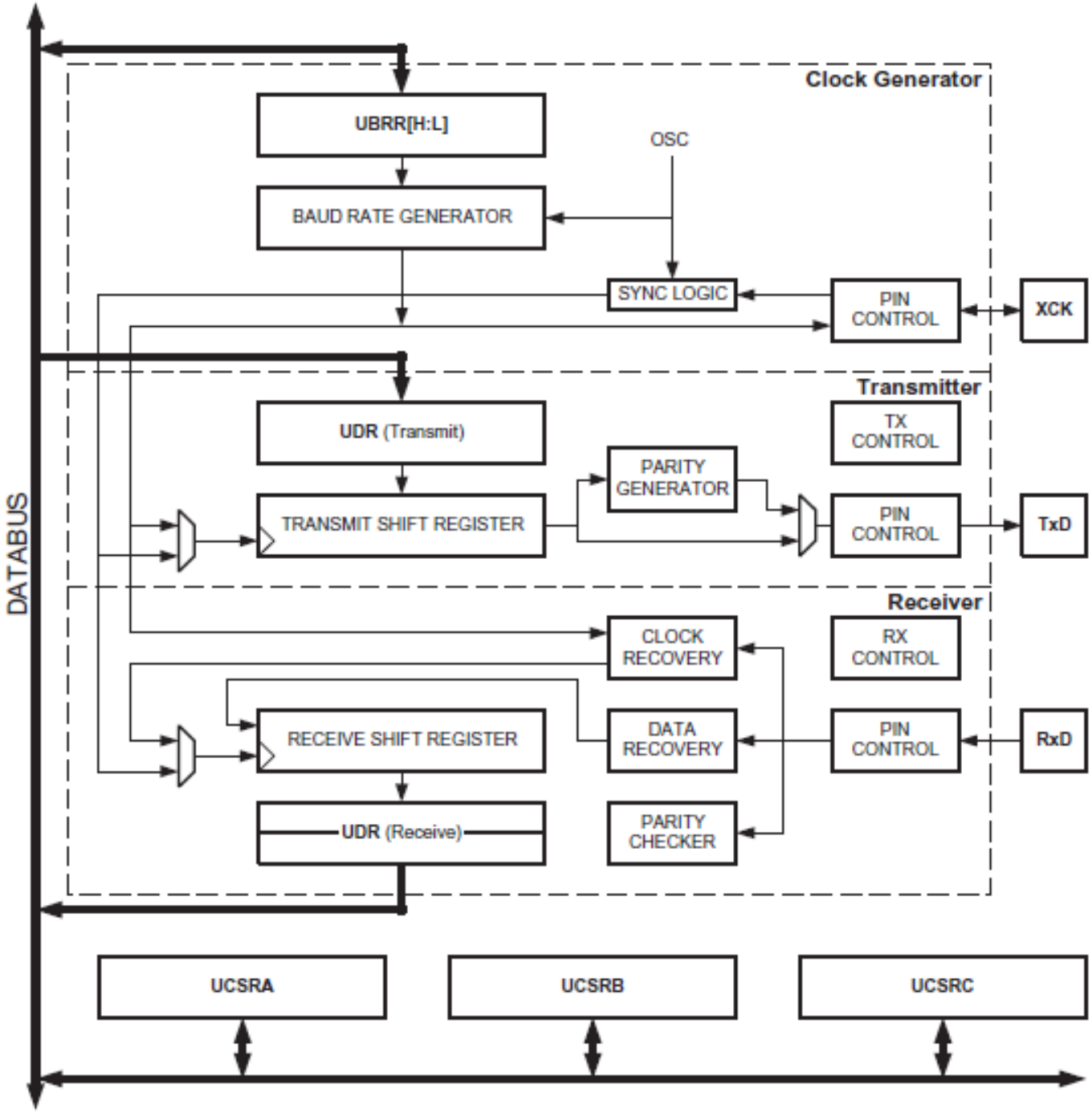
USART: Eigenschaften

- Two Programmable Serial USART (USART0 and USART1)
- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous Operation
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware

USART: Eigenschaften

- **Data OverRun Detection**
- **Framing Error Detection**
- **Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete**
- **Multi-processor Communication Mode**
- **Double Speed Asynchronous Communication Mode**

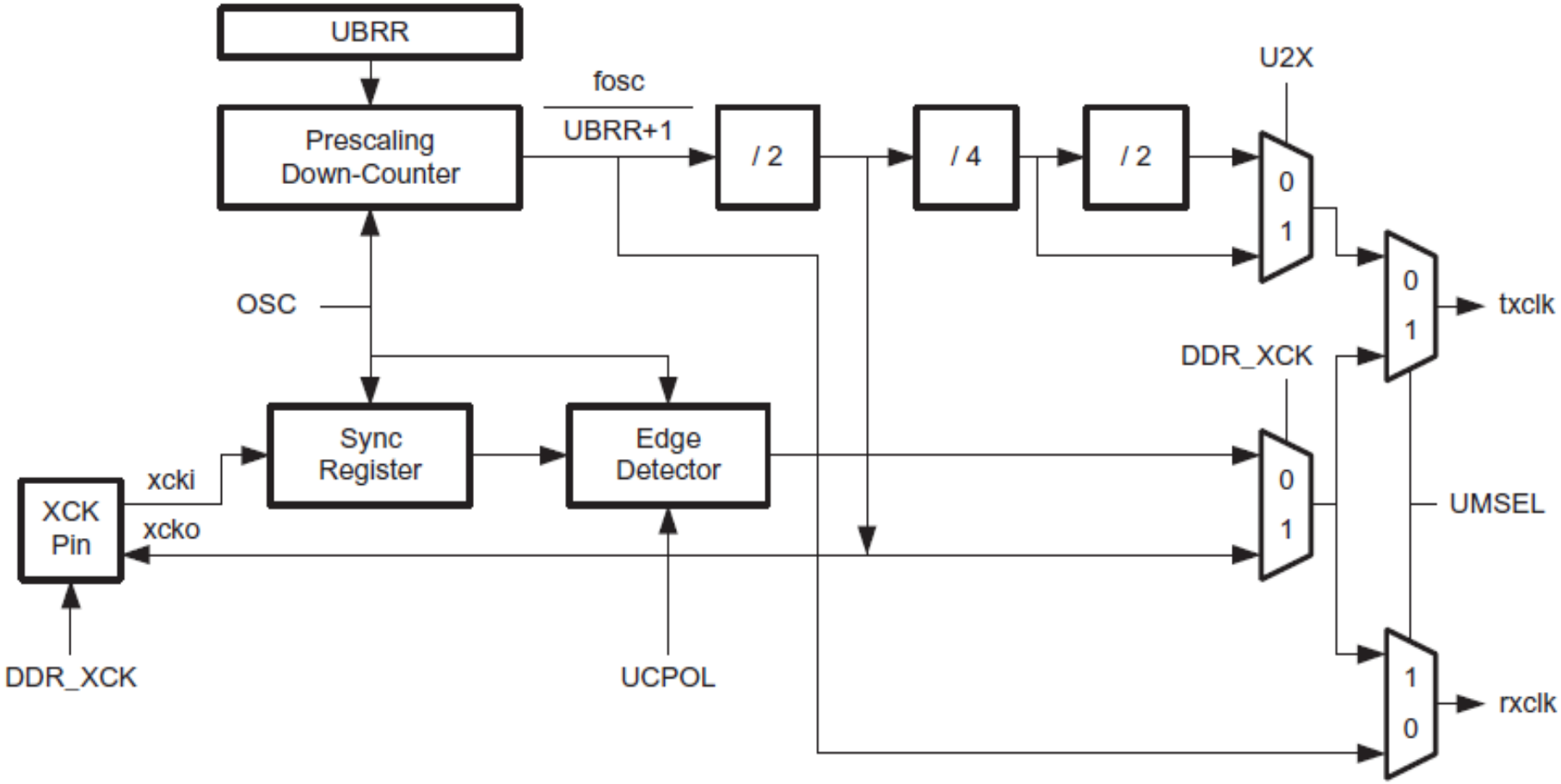
USART:
Blockschaltbild



USART: Takterzeugung

- n: 0 or 1
- 4 Moden:
  - Normal Asynchronous
  - Double Speed Asynchronous
  - Master Synchronous
  - Slave Synchronous mode.
- Steuerbits:
  UMSELn (UCSRnC – Register): async / sync
  U2X (UCSRnA – Register): normal / double speed

USART: Takterzeugung Blockschaltbild



USART: Taktratenberechnung

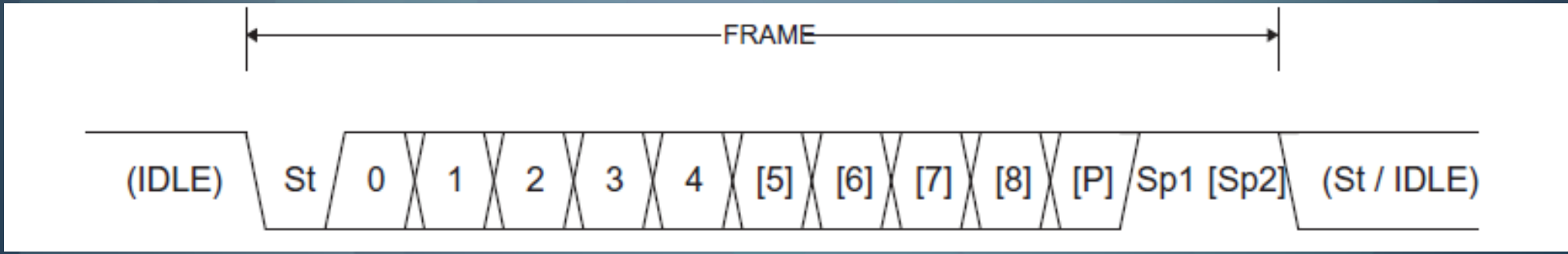
Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

USART: Taktratenberechnung

- BAUD Baud rate (in bits per second, bps)
- fOSC System Oscillator clock frequency
- UBRRn Contents of the UBRRHn and UBRRLn Registers, (0 - 4095)

USART: Frame - Formate

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits



USART Register

- UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

USART: UDR – USART I/O Data Register

- The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register orUDRn.
- The transmit buffer can only be written when the UDREn Flag in the UCSRnA Register is set.
- When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxDn pin.
- The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed.

USART Register

- UCSRnA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

USART Register

- Bit 7 – RXCn: USART Receive Complete
  This flag bit is set when there are unread data in the receive
- Bit 6 – TXCn: USART Transmit Complete
  This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out
- Bit 5 – UDREn: USART Data Register Empty
  The UDREn Flag indicates if the transmit buffer (UDRn) is ready to receive new data.
- Bit 4 – FEn: Frame Error
  This bit is set if the next character in the receive buffer had a Frame Error when received.
- Bit 3 – DORn: Data OverRun
  This bit is set if a Data OverRun condition is detected.
- Bit 2 – UPEn: USART Parity Error
  This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPMn1 = 1).
- Bit 1 – U2Xn: Double the USART Transmission Speed
  This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.
- Bit 0 – MPCMn: Multi-processor Communication Mode

USART Register

- UCSRnB – USART Control and Status Register nB

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

USART Register

- Bit 7 – RXCIEn: RX Complete Interrupt Enable n
  Writing this bit to one enables interrupt on the RXCn Flag. A USART Receive Complete interrupt will be generated only if the RXCIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXCn bit in UCSRnA is set.
- Bit 6 – TXCIEn: TX Complete Interrupt Enable n
  Writing this bit to one enables interrupt on the TXCn Flag. A USART Transmit Completeinterrupt will be generated only if the TXCIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXCn bit in UCSRnA is set.
- Bit 5 – UDRIEn: USART Data Register Empty Interrupt Enable n
  Writing this bit to one enables interrupt on the UDREn Flag. A Data RegisterEmpty interrupt will be generated only if the UDRIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDREn bit in UCSRnA is set.
- Bit 4 – RXENn: Receiver Enable n
  Writing this bit to one enables the USART Receiver.
- Bit 3 – TXENn: Transmitter Enable n
  Writing this bit to one enables the USART
- Bit 2 – UCSZn2: Character Size n
  The UCSZn2 bits combined with the UCSZn1:0 bit in UCSRnC sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use.
- Bit 1 – RXB8n: Receive Data Bit 8 n
  Bit 0 – TXB8n: Transmit Data Bit 8 n

USART Register

- UCSRnC – USART Control and Status Register nC

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

USART: UCSRnC

- Bits 7:6 – UMSELn1:0 USART Mode Select
  These bits select the mode of operation of the USARTn as shown in Table 16-4.

Table 16-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)(1)

USART: UCSRnC

- Bits 5:4 – UPMn1:0: Parity Mode
  These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPMn setting. If a mismatch is detected, the UPEn Flag in UCSRnA will be set.

Table 16-5. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

USART: UCSRnC

- Bit 3 – USBSn: Stop Bit Select
  This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 16-6. USBS Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

USART: UCSRnC

- Bit 2:1 – UCSZn1:0: Character Size
  The UCSZn1:0 bits combined with the UCSZn2 bit in UCSRnB sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use.

Table 16-7. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- Bit 0 – UCPOLn: Clock Polarity
  This bit is used for synchronous mode only.

USART: UBRRnL and UBRRnH
USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRHn
	UBRR[7:0]								UBRRLn
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

- Bit 15:12 – Reserved Bits
- Bit 11:0 – UBRR11:0: USART Baud Rate Register
  This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

USART: UBRRnL and UBRRnH

USART Baud Rate Registers

Baud Rate (bps)	f_{osc} = 16.0000 MHz			
	U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max. (1)	1 Mbps		2 Mbps	

USART: Initialisierung in C

```
void USART_Init( unsigned int baud )
{
  UBRRHn = (unsigned char)(baud>>8);
  UBRRLn = (unsigned char)baud;
  /* Enable receiver and transmitter*/
  UCSRnB = (1<<RXENn)|(1<<TXENn);
  /* Set frame format: 8data, 2stop bit*/
  UCSRnC = (1<<USBSn)|(3<<UCSZn0);
}
```

USART: Sending Frames with 5 to 8 Data Bit

```
voidUSART_Transmit( unsigned chardata )
{
  /* Wait for empty transmit buffer*/
  while( !( UCSRnA & (1<<UDREn)) )
  ;
  /* Put data into buffer, sends the data*/
  UDRn = data;
}
```

USART: Receiving Frames with 5 to 8 Data Bits

```
unsigned charUSART_Receive( void)
{
  /* Wait for data to be received*/
  while( !(UCSRnA & (1<<RXCn)) )
  ;
  /* Get and return received data from
    buffer*/
  return UDRn;
}
```

USART: