- Übersicht 16.04.2008
  - I/O Ports
  - Interrupts
  - Timer / Zähler

I/O Ports

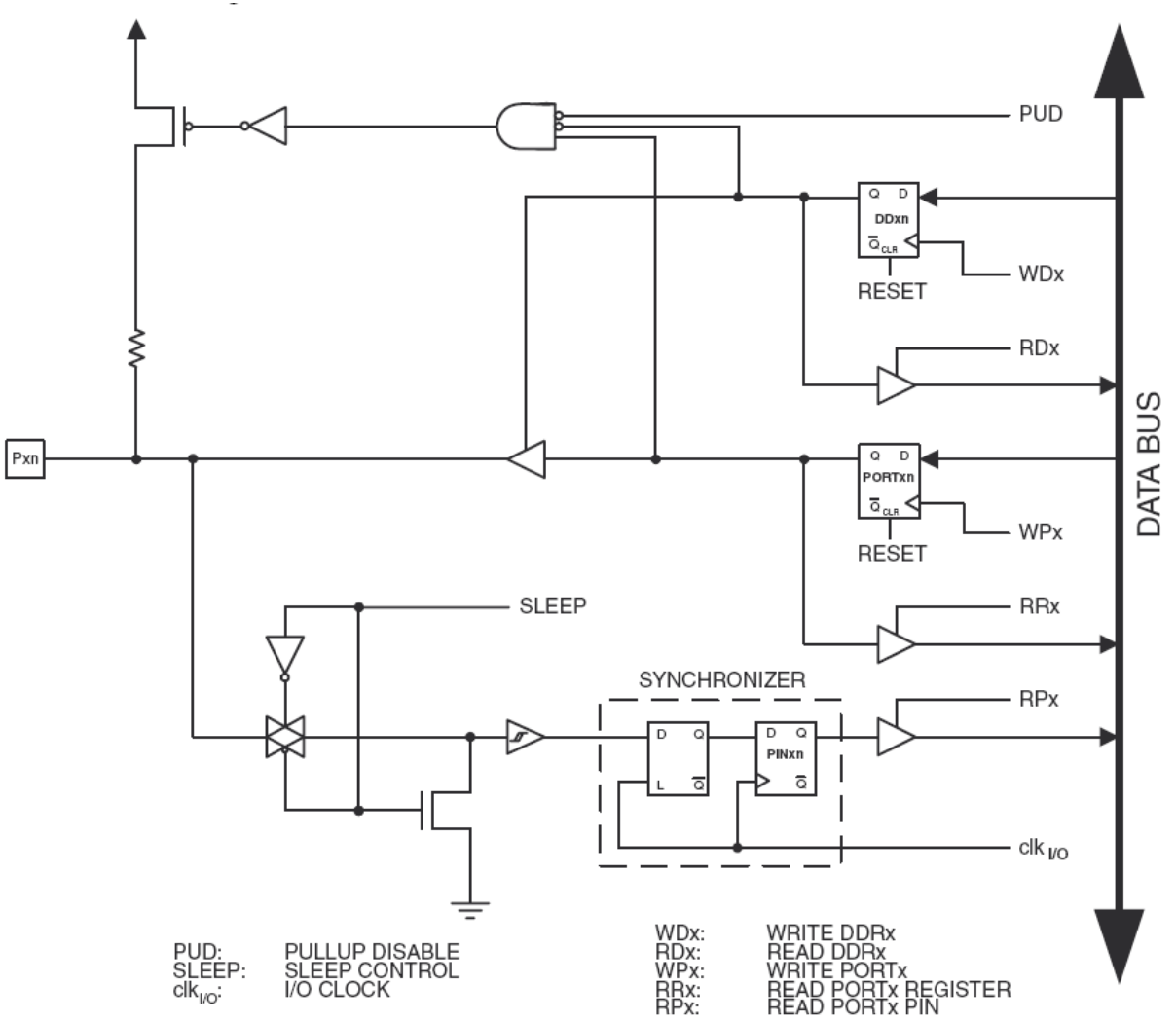
I/O Ports

Hochschule Offenburg
University of Applied Sciences

- ATmega32 Controller besitzt vier I/O Ports mit jeweils acht Pins
- Die Ports sind standartmäßig als digitale I/O Ports geschalten
- Auf jeden Portpin besteht voller Schreib-/Lesezugriff
- Die Portpins können sowohl als digitaler Eingang als auch als digitaler Ausgang geschaltet werden
- Jeder Portpins besitzt einen eigenen Treiber, welcher aber nur aktiv ist, wenn der zugehörige Pin als Ausgang konfiguriert ist
- Sollte ein Portpin als Eingang deklariert werden, so kann für diesen ein Pull-Up-Widerstand geschalten werden
- Alle Ports sind nach einem Reset als Eingang deklariert

I/O Port Pins (1/2)

- DDRx Register
  - Deklariert ob Pins als Ausgang oder Eingang agieren
- PORTx Register
  - Schreibt Signale auf die Pins (DDRxn=1)
  - Kann den Pull-UP ein- bzw. ausschalten (DDRxn=0)
- PINx Register
  - Hiermit kann das anliegende Signal der Pins ausgelesen weden



I/O Port Pins (2/2)

Hochschule Offenburg
University of Applied Sciences

DDRxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Jes	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)

Register PORTA

Hochschule Offenburg
University of Applied Sciences

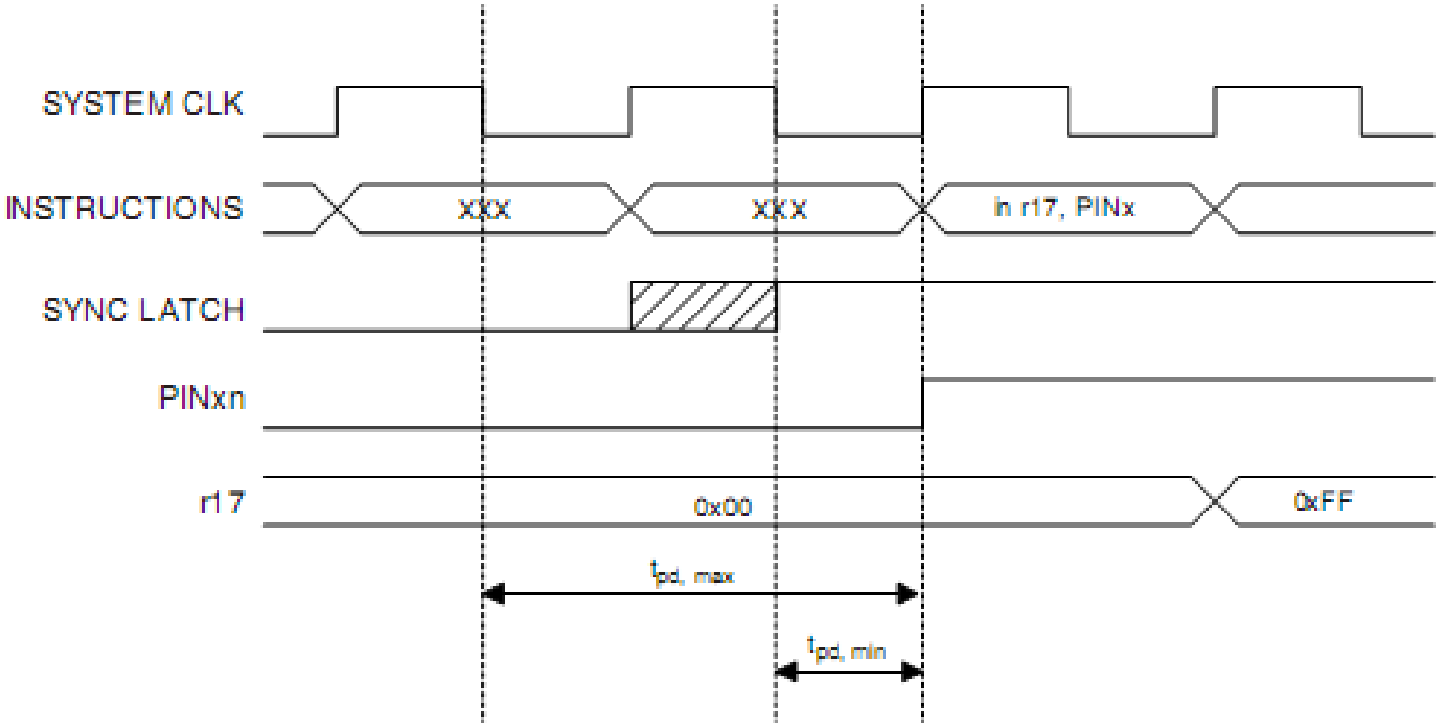
Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Synchronisation PINxn

- Da ein externes Signal nicht synchron zum Controllertakt geschaltet sein muss, beträgt die Verzögerung bestenfalls 0,5 Takte, schlechtestenfalls 1,5 Takte (abhängig vom Zeitpunkt der Signaländerung relativ zum Takt).



DC Characteristics

Hochschule Offenburg
University of Applied Sciences

Symbol	Parameter	Condition	Min	Max
V_{IL}	Input Low Voltage	V_{CC} = 4,5-5,5V	-0,5V	0,2 V_{CC}
V_{IH}	Input High Voltage	V_{CC} = 4,5-5,5V	0,6 V_{CC}	V_{CC} + 0,5V
V_{OL}	Output Low Voltage	I_{OL} = 20mA V_{CC} = 5V		0,7V
V_{OH}	Output High Voltage	I_{OH} = 20mA V_{CC} = 5V	4,2V	
R_{pu}	I/O Pin Pullup Resistor		20kΩ	50kΩ

Zusammenfassung I/O Ports

Hochschule Offenburg
University of Applied Sciences

- Jeder Port Besitzt drei Register
  - DDRx (Data Direction Register)
  - PORTx (Port Data Register)
  - PINx (Pin Data Register)
- Mit dem zweiten Bit (PUD) des Special Function I/O Register (SFIOR) können alle Pull-Up-Widerstände deaktiviert werden.
- Reaktionszeit auf Eingehende Signale min. 0,5 Takte max. 1,5 Takte
- Max Strom je Port Pin ±40mA

Interrupts

Interrupts

Hochschule Offenburg
University of Applied Sciences

- Bei Interrupts handelt es sich um einen Benachrichtigungs-mechanismus , welcher es einer ALU erlaubt, mit minimaler Verzögerung aufgetretene Ereignisse zu erkennen und zu verarbeiten
- Interrupts werden in drei Kategorien unterteilt
  - Externen Interrupts (z.B.: Reset)
  - Interne Interrupts (z.B.: Timer)
  - Software Interrupts (irrelevant bei Mikrocontrollern)

Interrupt-Verarbeitung Intel x86-Architektur

Hochschule Offenburg
University of Applied Sciences

- Interrupts sperren
- Program Counter (PC) und Statusregister sichern
- Privilege Level prüfen
- PC und Statusregister der Interrupt-Routine laden
- Interrupt freigeben
- Ausführen der Interruptbehandlungsroutine
- Interrupts sperren
- PC und Statusregister wiederherstellen
- Interrupts freigeben

Interrupt-Verarbeitung Atmel AVR

Hochschule Offenburg
University of Applied Sciences

- Interrupts sperren
- Program Counter (PC) sichern
- PC der Interrupt-Routine laden
- Ausführen der Interruptbehandlungsroutine
- PC wiederherstellen
- Interrupts freigeben

Timing der Interruptroutine

- Start der Interruptroutine
  - Beim eintreffen eines Interrupts wird der aktuelle Befehl noch vollständig Bearbeitet (drei Taktzyklen bei einem Sprungbefehl).
  - Vier weitere Takte werden für die Interrupt-Sperrung, das Löschen des jeweiligen Interrupt Flags, das Sichern des PC sowie der Sprung an die Startadresse der Interrupt-Routine benötigt (in einem Sleep-Modus werden acht Takte benötigt)
- Ende der Interruptroutine
  - Es werden wiederum vier Takte benötigt, in denen der urspüngliche PC vom Stack wiederhergestellt wird, der Sprung an die ursprüngliche Adresse erfolgt und die Interrupts freigegeben werden

Intialisierung Interrupt

Hochschule Offenburg
University of Applied Sciences

- Ein Programm mit Interruptsteuerung hat folgende Merkmale
  - Das Global Interrupt Enable Bit muss gesetzt sein (SREG |= (1<<7))
    - Bei AVR GCC Compiler wird dies durch das Makro sei() erreicht.
  - Eine Interrupt-Service-Routine muss vorhanden sein

```
# include <avr/interrupt.h>

ISR(INT0_vect) //Vector name
{
   PORTC=0x55;
}
```

  - Interrupt müss initialisiert werden

```
...
MCUCR|=(1<<ISC01)|(1<<ISC00); //Steigende Flanke löst aus
GICR|=(1<<INT0);              //INT0 enable
...
```

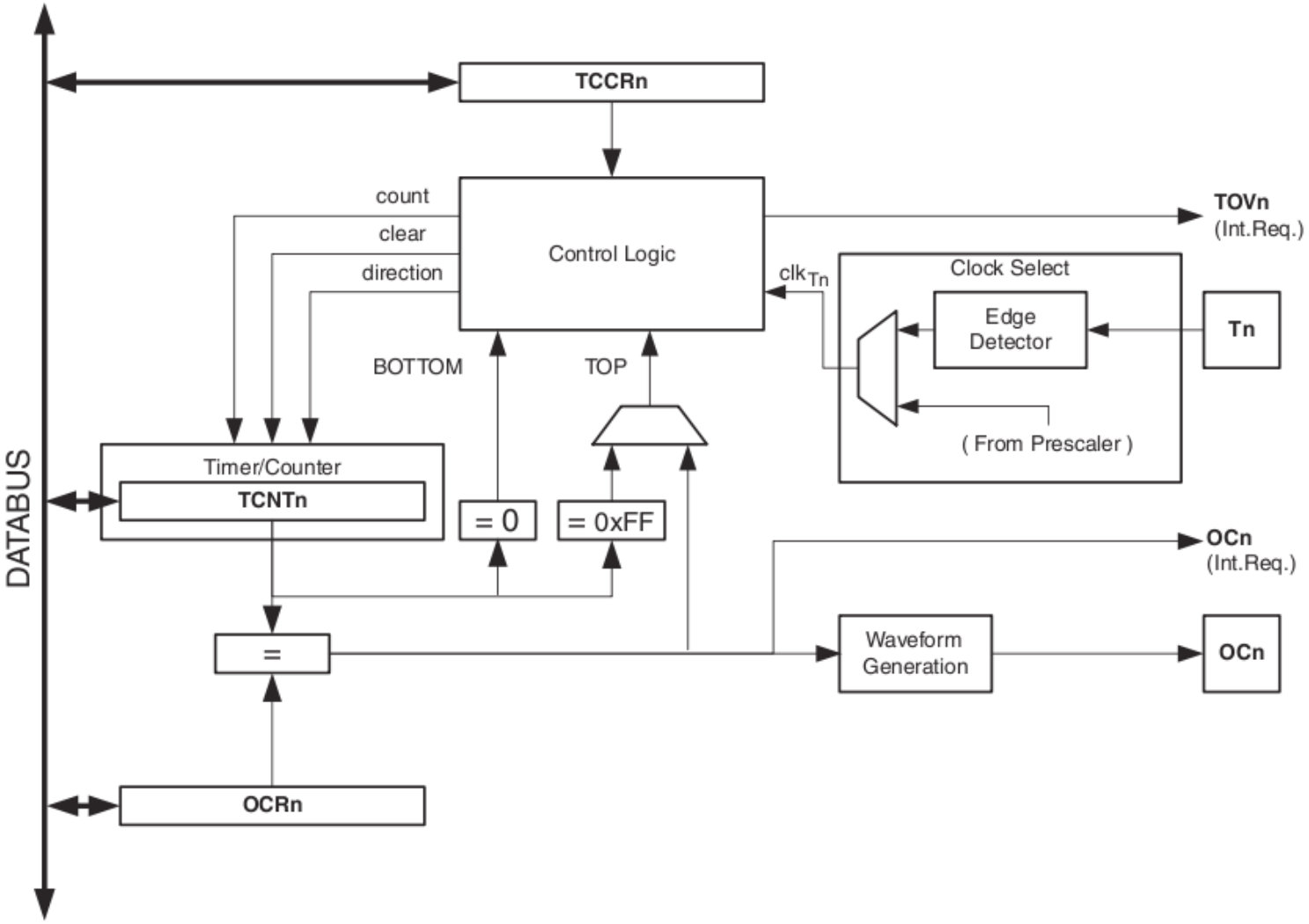
Timer / Zähler

Timer/Zähler

Hochschule Offenburg
University of Applied Sciences

- Spezial Register welches hoch bzw. runter gezählt werden kann.
- Zähler: Registerwert wird durch externes Signal in- oder dekrementiert.
- Timer: Register wird von einem internen Taktsignal in- oder dekrementiert.
- Abhängig vom Zählerstand kann ein
  - Ausgang gesetzt oder gelöscht werden (PWM).
  - ein Interrupt ausgelöst werden.

8-bit Timer/Counter 0



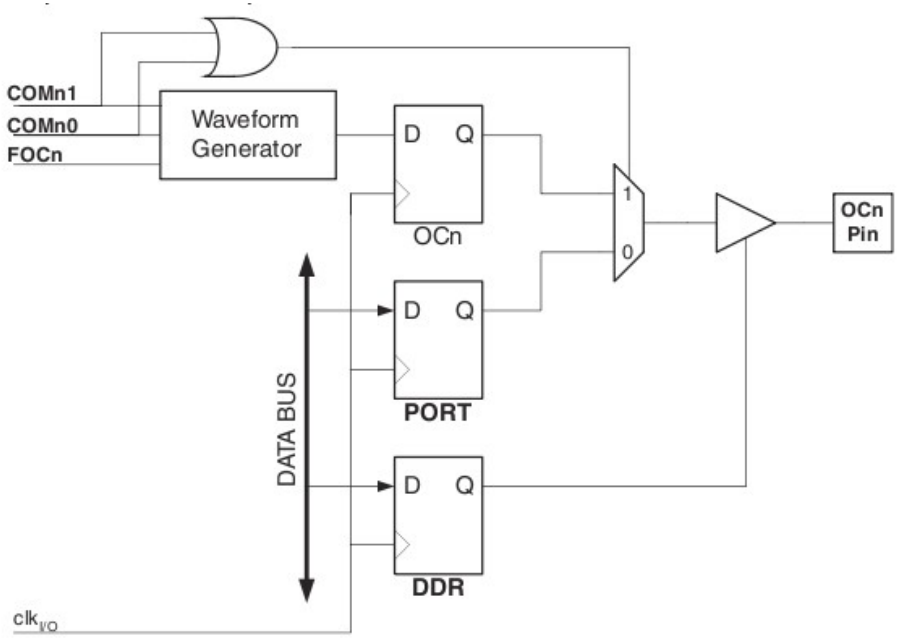
Clock Source

Hochschule Offenburg
University of Applied Sciences

- Externes Signal über T0 (Signal von PINB0).
- Internes Taktsignal mit Vorteiler.
- Einstellung über CS02...CS00 in TCCR0.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Output Unit



- Bestimmt den Zustand am Pin PB3
- Einstellung über COM01 und COM00 in TCCR0
- Abhängig vom Mode
- FOC0 in TCCR0 erzwingt das Setzen eines bestimmten Zustandes

Interrupts

Hochschule Offenburg
University of Applied Sciences

- Interrupt bei Überlauf von TCNT0. TOV (Timer/Conter0 Overflow Flag) in TIFR.
- Interrupt bei Gleichheit von TCNT0 und OCR0. OCF0 (Output Compare Flag 0) in TIFR.
- Enable Bits:
  - TOIE0 (Timer/Counter0 Overflow Interrupt Enable)
  - OCIE0 (Timer/Counter0 Output Compare Match Interrupt Enable)

Normal Mode

Hochschule Offenburg
University of Applied Sciences

- Der Modus wird eingestellt über die Bits WGM00 und WGM01 in TCCR0
- Bei WGM01:0 = 0 ist der Normal Mode aktiv.
- Einfaches Hochzählen von TCNT0.
- TOV0 kann als 9-Bit verwendet werden.

Beispiel 1

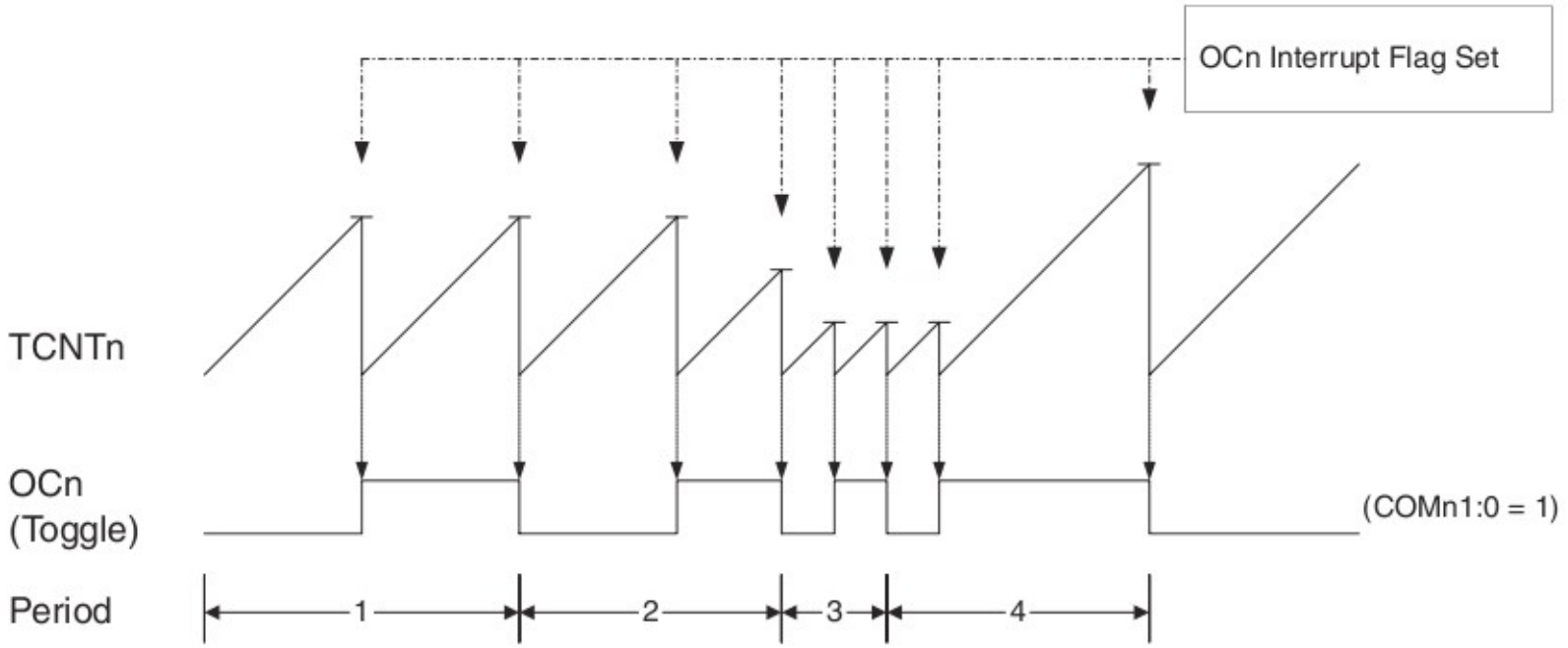
Hochschule Offenburg
University of Applied Sciences

```
#include <stdlib.h>
#include <avr/io.h>

int main(void)
{
    ...
    DDRB  &= ~_BV(PB0);           //PB0 Eingang
    PORTB |=  _BV(PB0);           //Pull-Up aktiv

    //Clock Source auf extern.(Fallende Flanke)
    TCCR0 = _BV(CS02) | _BV(CS01);
    for(;;)
    {
        ...
        if( TCNT0 == 25)
        {
            do somting
            TCNT0 = 0;
        }
    }
}
```

Clear Timer on Compare Match (CTC)Mode

Hochschule Offenburg
University of Applied Sciences

- WGM01:0 = 2
- TCNT0 wird bis OCR0 inkrementiert.
- TOV tritt im Normalfall nicht auf.



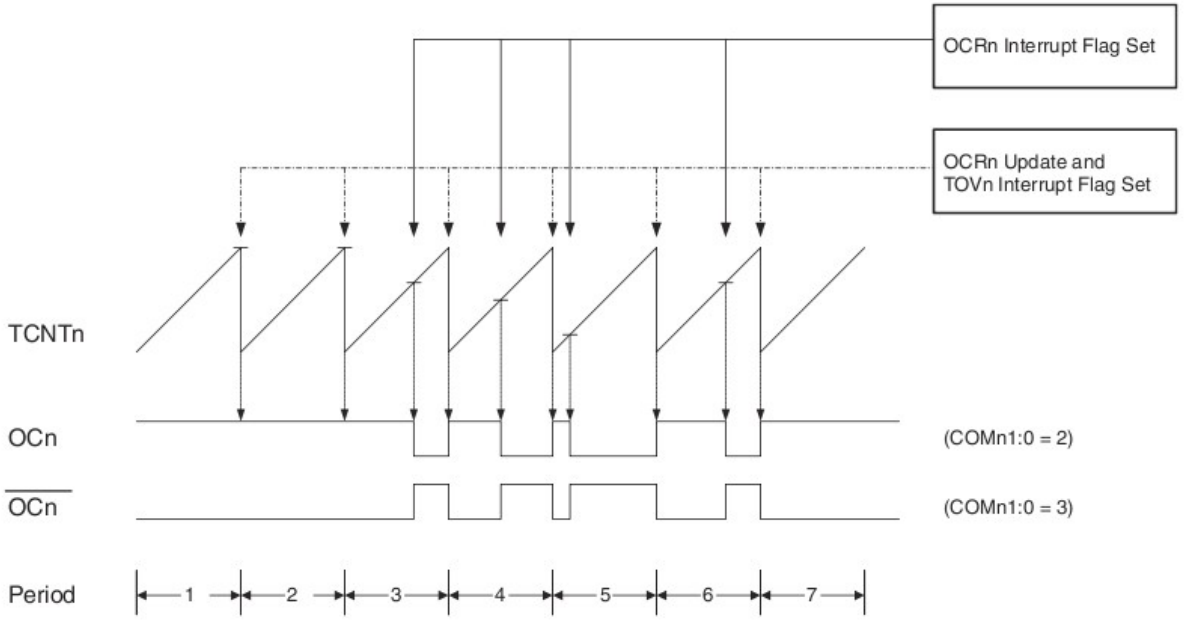
Beispiel 2

Hochschule Offenburg
University of Applied Sciences

```
ISR(TIMER0_COMP_vect)  // wird jede ms ausgeführt
{
    static uint16_t i=0;
    i++;
    if( i== 500)
    {
        i= 0;
        PORTA ^=_BV(PA4);
    }
}
int main(void)
{

    // CTG Mode, Prescaler=64 -> clk=250kHz
    TCCR0 = _BV(WGM01) | _BV(CS01) | _BV(CS00);

    //On Compare match Interrupt enable for timer 0
    TIMSK |= _BV(OCIE0);

    OCR0= 249; // 250kHz/(249+1) = 1kHz
    for(;;)
}
```

Fast PWM Mode

- WGM01:0 = 3
- TCNT0 wird bis 0xFF inkrementiert
- $f_{PWM} = f_{clk} / (N*256)$
  N = 1, 8, 64, 256, 1024
- Pulsbreite wird über OCR0 eingestellt.
- OCR0 wird gebuffert.



Beispiel 3

Hochschule Offenburg
University of Applied Sciences

```
int main(void)
{

    //Fast PWM Mode, clear OC0 on compare match,Prescaler=256 -> clk=62,5kHz
    TCCR0 = _BV(WGM00) | _BV(COM01)  | _BV(CS02)  |_BV(CS00);

    OCR=63 //Pulsbreite 25%

    DDRB |= _BV(PB3);  // PB3 Ausgang

}
```

Phase Correct PWM Mode

- WGM01:0 = 1
- TCNT0 wird bis 0xFF inkrementiert
- $f_{PWM} = f_{clk} / (N*510)$
- Pulsbreite wird über OCR0 eingestellt.
- OCR0 wird gebuffert.