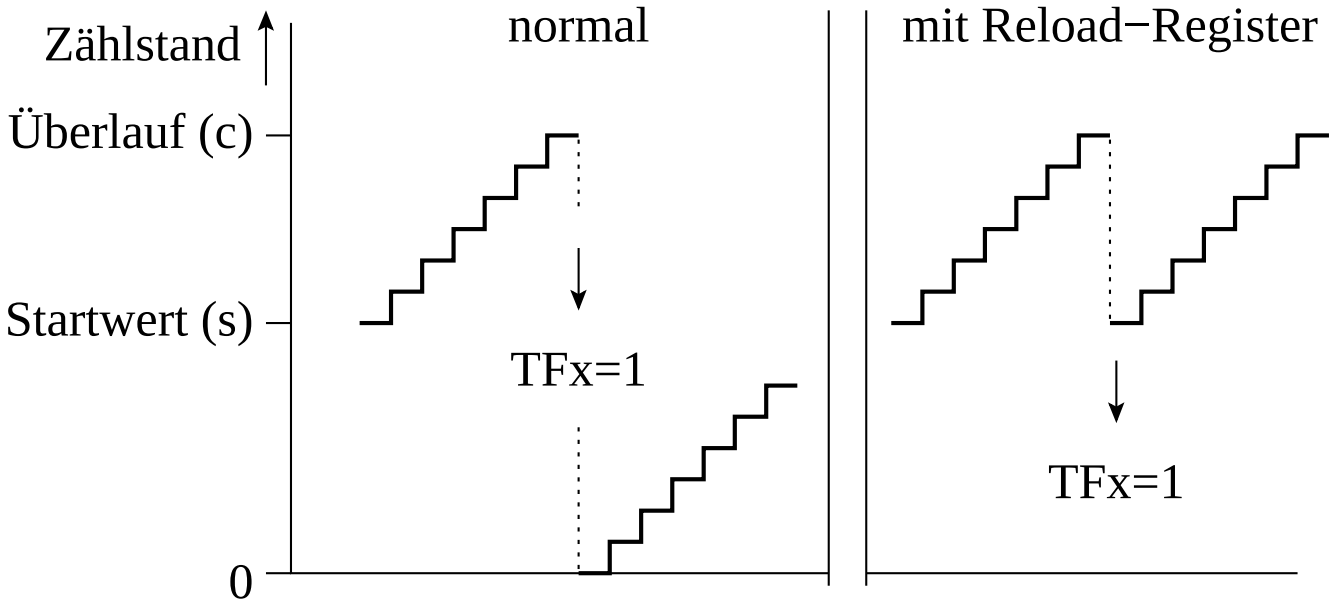
Praktikum Mikrorechner 11 (Timer 1)

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

5. November 2014

Zähler und Zeitgeber



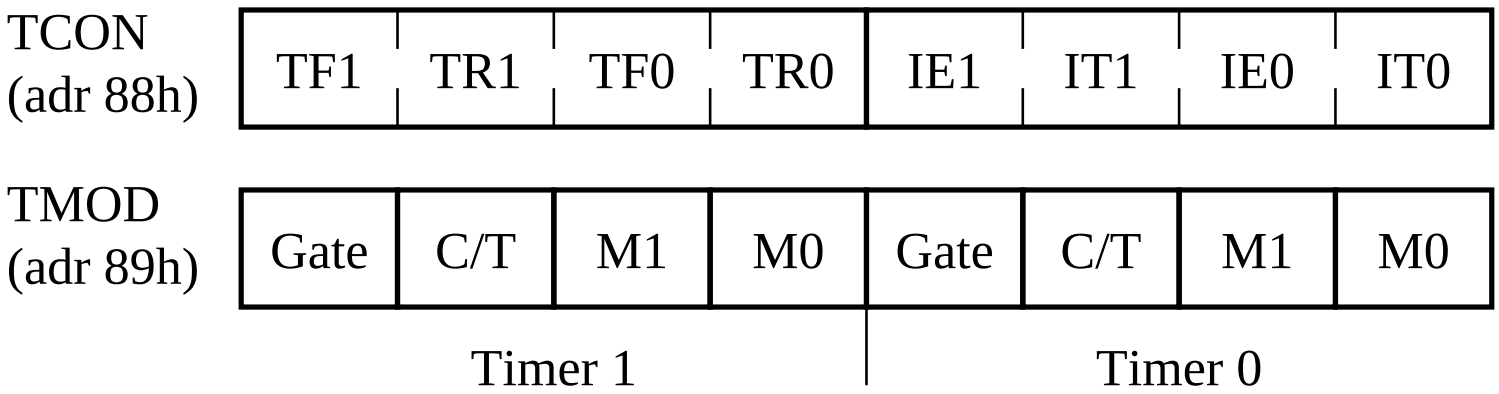
Zählen von Ereignissen, Setzen eines Interruptanforderungsflags bei Überlauf. Mögliche Zählereignisse:

- Betriebsart Timer: Zeitimpulse
- Betriebsart Counter: Eingabeimpulse

Ereigniszahl bis Überlauf: $c - s$

Die Zähler-/Zeitgeber-Kanäle CT0 und CT1

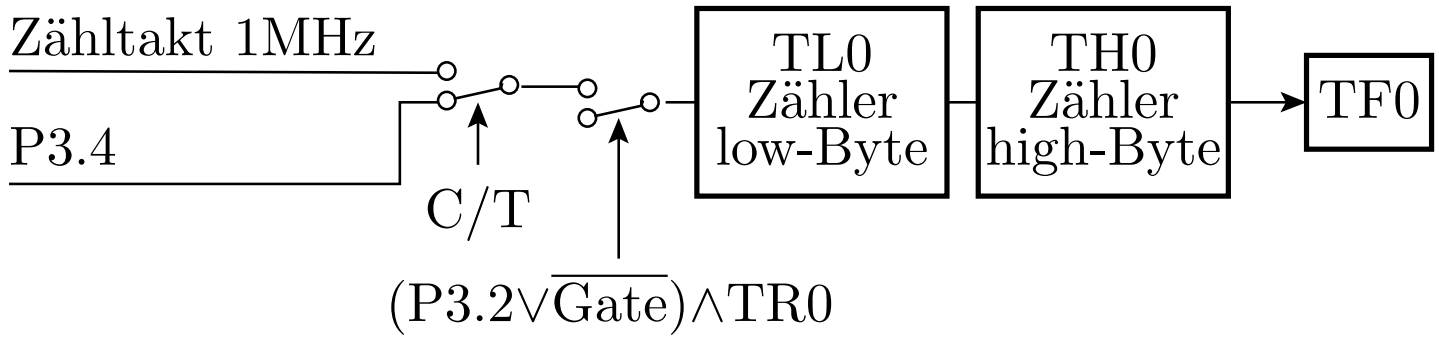
- Zählerregister: TL0 (adr. 8ah), TH0 (adr. 8bh), TL1 (adr. 8ch), TH1 (adr. 8dh)
- Konfigurationsregister



TRx	Zähler einschalten (ein: 1)
M1, M0	Betriebsart
C/T	Counter/Timer-Umschaltung (Counter: 1)
Gate	Zähleingang freigeben (freigegeben: 1)

Betriebsart 1

- 16-Bit-Zähler ohne Auto-Reload
- beider 8-Bit-Zählregister sind zu einem 16-Bit-Zähler zusammengeschaltet
- Bei jedem Zählerüberlauf wird das Interruptflag gesetzt. Gezählt werden:
  - bei C/T=0 und TRx=1 der durch 12 geteilte Prozessortakt
  - bei C/T=1, Gate=1 und TRx=0 die Anzahl der Impulse am Eingang P3.4



Warteschleifen mit Timer im Polling-Betrieb

```
Wiederhole immer
    Wiederhole 10x

        Timer für Wartezeit von 50ms initialisieren
        und starten
        Warten bis Timer-Überlauf
    Invertieren von LED0
```

Startwert für 50ms Wartezeit (50.000 Zählschritte):

- Überlauf des 16-Bit-Zählers: $c = 2^{16}$
- Startwert: $s = 2^{16} - 50.000 = 15536 = 3\text{cb0h}$

- Makro zur Initialisierung des Timers für eine Wartezeit von 50 ms

```
Init_Tmr0_50ms_mac MACRO
  clr TR0          ; Timer anhalten
  clr TF0          ; Überlaufflag löschen
    mov TL0, #0b0h
    mov TH0, #3ch
   setb TR0        ; Timer starten
ENDM
```

```
Ct data 60h
org 100h
  clr EA          ; alle Interrupts aus
  mov tmod, #1    ; Timer 0: C/T=0, M1=0; M0=1
Endlosschleife:
   mov Ct, #10
 Schleife10:
     Init_Tmr0_50ms_mac
     jnb TF0, $ ; Warte bis Zählerüberlauf
     clr TF0
     djnz Ct, Schleife10
   cpl P1.0
   ljmp Endlosschleife
```

Gleiche Aufgabe mit Interrupt

```
org 0bh          ; Startadresse Timer0-Int.
  ljmp Tmr0      ; Sprung zur isr
; ------------------------------------
org 100h
  clr EA         ; alle Interrupts aus
  mov sp, #0c0h; Stack einrichten
  mov tmod, #1 ; Timer 0: C/T=0, M1=0; M0=1
  Init_Tmr0_50ms_mac
  setb ET0       ; Timer 0: Interrupt ein
  mov Ct, #10
  set EA         ; Interrupt global ein
; ------------------------------------
  ljmp $         ; Endlosschleife
; ------------------------------------
```

Interruptroutine

```
Tmr0:
    Init_Tmr0_50ms_mac
    djnz Ct, Tmr0_Ex
        mov Ct, #10
        cpl P1.0
Tmr0_EX:
  reti
```

Tastenabfrage und Entprellung mit 2 verketteten Interrupts

```
extInt0: (fallende Flanke)
  * P1 = P1 +1
  * deaktiviere extInt0
  * Starte Timer für Wartezeit 50ms
  * reaktiviere tmr0-Interrupt

tmr0: (Betriebsart 1)
  * deaktiviere tmr0-Interrupt
  * reakiviere extInt0
```

Achtung: Ereignisgesteuerter Programmfluss; bei Programmierfehlern Risiko für. Deadlock (Systemverklemmungen)

```
;-------------------------------
Wartezeit equ 3cb0h
TCON  data 88h
ITCON data 9ah
;-------------------------------
; Sprung zu den Interruptroutinen
;-------------------------------
org 3                ; externer Interrupt 0
ljmp isrEX0

org 0bh              ; Timer 0 Interrupt
  ljmp isrTmr0
```

```
; Hauptprogramm
;------------------------------
org 100h
  clr  EA          ; alle Interrupts aus
  mov  SP, #0c0h   ; Stack einrichten
  mov  P1, #0      ; Anfangswert für P1
  setb P3.2        ; P3.2 als Eingang vorbereiten
  setb it0         ; ext. Int. 0 auf Flanke
  mov  itcon, #2   ; fallende Flanke
  setb EX0         ; ext. Interrupt ein
;..............................
  mov  TMOD, #1    ; Timer 0, Betriebsart 1
  clr  TR0         ; Timer 0 anhalten
  clr  ET0         ; Timer 0 Interrupt aus
  setb EA          ; Interrupt global ein
;..............................
  ljmp $           ; Endlosschleife
```

- Serviceroutine für den externen Interrupt

```
isrEX0:
 clr EX0          ; externer Interrupt aus
 inc P1;          ; Ausgabe um 1 erhöhen
 Init_Tmr0_50ms_mac
 setb ET0         ; Timer 0 Interrupt ein
reti
```

- Serviceroutine für den Timer0-Interrupt

```
isrTmr0:
 clr  TR0         ; Timer 0 aus
 clr  ET0         ; Timer 0 Interrupt aus
 setb EX0         ; Ext. Interrupt 0 ein
 reti
```

Messung der Lüftergeschwindigkeit

Zählen der fallenden Flanken am Ausgang der Lüfterlichtschranke (P3.3) in einem Messzeitintervall von 1s

- Messdauer mit tmr0-isr + Zusatzbyte erzeugen. Startwert:

$$2^{24} - 10^6 = 15.777.216 = \text{f0.bd.c0h}$$

- mit ext1-isr fallende Flanken an P3.3 zählen
- Ausgabe des Zählerstand (Umdrehungen pro s mal 9 Flügel) wenn Tmr0-Interrupt und Überlauf führendes Byte (r0)

- Makro zur Initialisierung des Timers und des zusätzlichen registers r0 für eine Wartezeit von 50 ms

```
Init_mac MACRO
  clr  TR0          ; Timer anhalten
  mov  r0,  #0f0h   ; höchstwertiges Zählerbyte
  mov  TH0, #0bdh   ; mittleres Zählerbyte
  mov  TL0, #0c0h   ; niederwertigstes Zählerbyte
  mov  r1, #0       ; Ereigniszähler löschen
  setb TR0          ; Timer starten
ENDM
```

- Einsprungpunkt der Serviceroutine des Timer0-Interrupts

```
org 0bh             ; Timer 0 Interrupt
  ljmp isrTmr0
```

- Die Serviceroutine des exterenen Interrupts zählt nur eine Variable weiter:

```
org 13h              ; externer Interrupt 1
  inc r1             ; Ereigniszähler erhöhen
  reti
```

```
; Hauptprogramm
;  1. Initialisierung
;-----------------------------------------
org 100h
  clr  EA          ; alle Interrupts verbieten
  mov  SP, #0c0h   ; Stack einrichten
  mov  P1, #0      ; Anfangswert für P1
  setb P3.3        ; Lichtschranke => Eingang
  setb EX1         ; Freigabe des ext. Int. 0
  setb IT1         ; Interrupt bei Flanke
  mov  ITCON,#8    ; fallende Flanke
```

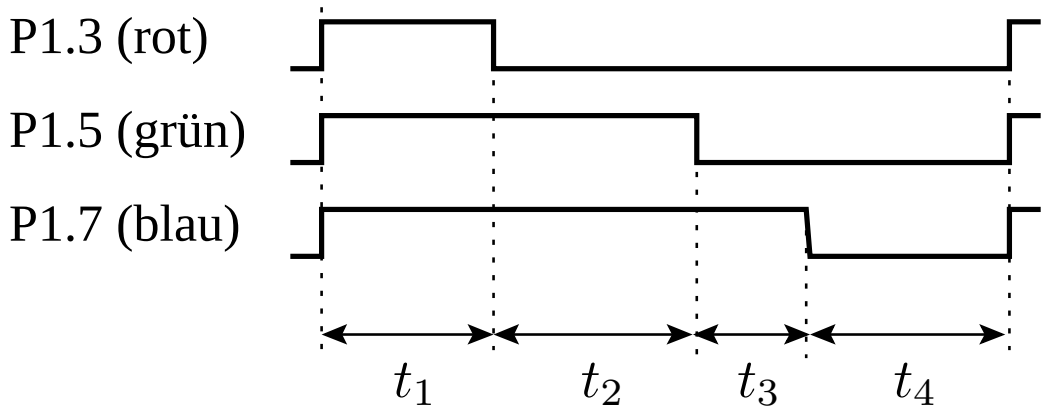
itcon (adr. 9ah)

IT2	IE2	I2ETF	I2ETR	I1ETF	I1ETR	I0ETF	I0ETR

```
  mov TMOD, #1 ; Timer 0, Modus 1
  Init_mac     ; Timer neu initialisieren
  setb ET0     ; Timer0-Interrupt ein
  setb TR0     ; Timer0 ein
  setb EA      ; Interrupts ein
  ljmp $       ; Endlosschleife
;==============================

isrTmr0:
  inc r0
  cjne r0, #0, isrTmr0_Ex ; jedes 10. Mal
    mov P1, r1 ; Ereigniszähler ausgeben
    mov r1, #0
    Init_mac   ; Timer neu initialisieren
isrTmr0_Ex:
  reti
```

PWM-Ansteuerung 3-Farb-LED



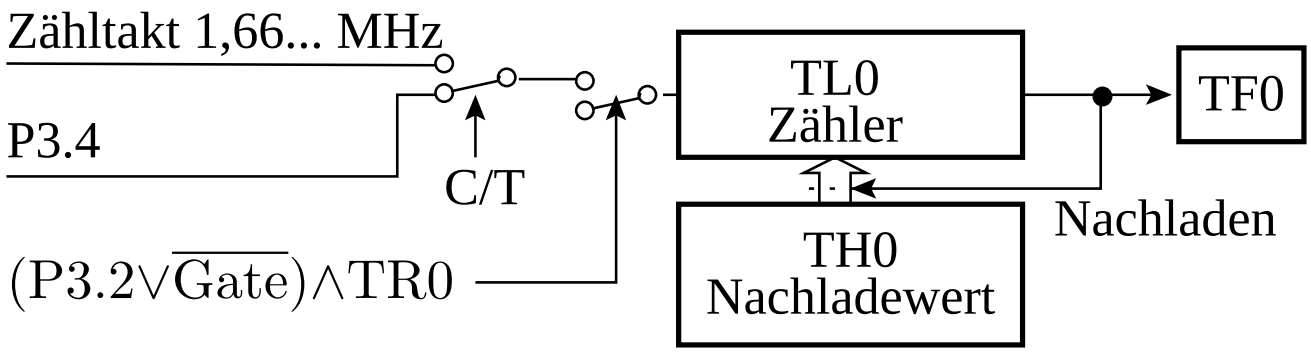

```
Lut_Time:
  movc a, @a+pc
  ret
db 123, 22, 7, 12 ; Zeiten t1 bis t4

Lut_Ausgabe:
  movc a, @a+pc
  ret
db 00000100, 00010100, 01010100, 00000000
```

- Das Programm besteht fast nur aus einer Interruptserviceroutine

```
istTmr0:
  clr EA     ; Int. aus
  clr TR0    ; Tmr0 Stop
  push acc
  mov a, State
  lcall Lut_Time
  mov TL0, a
  mov TH0, #0
  mov a, State
  lcall Lut_Ausgabe
  mov P1, a
  mov a, State
  cjne a, #4, isrTmr0_1
        mov State, #1
        sjmp isrTmr0_EX
isrTmr0_1:
        inc State
ist_Tmr0_EX:
  setb TR0 ; Tmr0 Start
  setb EA  ; Int. ein
  reti
```

Betriebsart 2: 8-Bit-Zähler mit Auto-Reload



Das Interruptflag TF0 (bzw. TF1) wird exakt nach 100h-TH0 Zählpulsen gesetzt; exakte Reload-Zeit.

Beispiel: 1kHz-Ton im Köpfhörer mit Timer 0

- Timer-Initialisierung C/T=0, /Gate=0, M1=1, M0=0 => TMOD=2
- Reloadwert: 100h-250=6
- Invertierung von P1.4 (Speaker) aller 500µs, d.h. nur bei jedem 2. Interrupt

```
Reload250us equ 6
Speaker bit  P1.4
Flags   data 22h
Ct1     bit  Flags.0
;---- Interrupteinsprung ---------
org 0bh      ; Timer 0 Interrupt
  ljmp Int_Tmr0
;--------------------------------
;     Initialisierung
org 100h
  clr EA     ; alle Interrupts aus
  mov tmod, #02
  mov TL0, #Reload250us
  mov TH0, #Reload250us
  clr Ct1
```

```
  setb ET0   ; Timer0-Interrupt ein
  setb TR0   ; Timer0 ein
  setb EA    ; Interrupts ein
  ljmp $     ; Endlosschleife
```

- Interrutserviceroutine (Invertierung von P1.4 (Speaker) aller 500µs, d.h. nur bei jedem 2. Interrupt)

```
Int_Tmr0:
  cpl Ct1
  jb  Ct1, Int_Tmr0_Ex
  cpl Speaker
Int_Tmr0_Ex:
  reti
```

Echtzeituhr

Weiterzählen von P1 im Sekundentakt, ohne dass isrExt1 stört

- Aufruf Tmr0 exakt aller 250µs, Autoreload, Betriebsart 2, hohe Priorität
- isrEx0, Event=LowPegel als Störquelle des Zeitablaufs

```
cTeiler equ 4000
ct0 data 60h
ct1 data 61h
InitTeiler MACRO
  mov Ct0, #low(4000)
  mov Ct1, #high(4000)
ENDM
org 03h    ; Externer interrupt 0
  nop
  nop
  reti
```

```
org 0bh    ; Timer 0 Interrupt
  lcall updateClock
  reti
org 100h
  clr  EA
  clr  TR0
  mov  Ct0, #low(cTeiler)
  mov  Ct1, #high(cTeiler)
  mov  TL0, #250
  mov  TH0, #250
  setb TR0
  setb ET0
  clr  IT0    ; Interrupt durch Low-Pegel
  setb EX0
  setb EA
  ljmp $
```

Die eigentliche Echtzeituhr ist hier ein Unterprogramm der Interruptserviceroutine des Tmr0-Interrupts

```
updateClock:
   djnz Ct0, updateClock_Ex
     djnz Ct1, updateClock_Ex
       inc P1
       mov Ct0, #low(cTeiler)
       mov Ct1, #high(cTeiler)
updateClock_Ex:
  ret
```

Aufgabe 11.1: Experimente mit Timern

- Testen Sie zwei von den beschriebenen Beispielprogrammen aus.