

Übungen zu Systemnahe Programmierung in C

Abschnitt 8.2: Timer

15.06.2020

Tim Rheinfels

Benedict Herzog

Bernhard Heinloth

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT



- Häufige Aufgaben in der Mikrocontrollerprogrammierung:
 - Regelmäßige Aktualisierung der Ausgabe (z.B. Bildwiederholrate)
 - Regelmäßiges Auslesen eines Wertes (z.B. serielle Konsole)
 - Pulswweitenmodulation (PWM)
 - Passives Warten
 - ...

⇒ Timer helfen bei der Umsetzung



- Ein Timer modifiziert pro Timertakt seinen Zähler
 - Inkrement (default)
 - Dekrement
 - Bei vorher konfigurierten Ereignissen wird ein Interrupt ausgelöst
 - Zähler erreicht einen bestimmten Wert
 - Zähler läuft über
 - (externes Ereignis tritt auf)
 - Der ATmega328PB bietet 5 verschiedene Timer:
 - `TIMER{0,2}`: 8-bit Zähler
 - `TIMER{1,3,4}`: 16-bit Zähler
- ⇒ Für die Übungsaufgaben: `TIMER0`
- ⇒ In der libspicboard: `TIMER{1,2,4}`

■ Wie schnell läuft der Timer:

- TCCR0B: TC0 Control Register B
- CSxx: Clock Select Bits
- Prescaler: Anzahl der CPU-Takte bis Zähler inkrementiert wird
- Was passiert, wenn die CPU in den Schlafmodus geht?

CS02	CS01	CS00	Beschreibung
0	0	0	Timer aus
0	0	1	prescaler 1
0	1	0	prescaler 8
0	1	1	prescaler 64
1	0	0	prescaler 256
1	0	1	prescaler 1024
1	1	0	Ext. Takt (fallende Flanke)
1	1	1	Ext. Takt (steigende Flanke)

CS02	CS01	CS00	Beschreibung
0	0	0	Timer aus
0	0	1	prescaler 1
0	1	0	prescaler 8
0	1	1	prescaler 64
1	0	0	prescaler 256
1	0	1	prescaler 1024
1	1	0	Ext. Takt (fallende Flanke)
1	1	1	Ext. Takt (steigende Flanke)

```
01 static void init(void) {  
02     // Timer mit prescaler 64 aktivieren  
03     TCCR0B |= (1 << CS01) | (1 << CS00);  
04     TCCR0B &= ~(1 << CS02);  
05  
06     // [...]  
07 }
```



■ Wann löst der Timer einen Interrupt aus:

- **Overflow:** Wenn der Zähler überläuft
- **Match:** Wenn der Zähler einen bestimmten Wert erreicht
 - ⇒ Register OCR0A (TIMER0 Output Compare Register A)
 - ⇒ Register OCR0B (TIMER0 Output Compare Register B)
- Interrupts einzeln demaskierbar
- TIMSK0: TIMER0 Interrupt Mask Register

Bit	ISR	Beschreibung
TOIE0	TIMER0_OVF_vect	TIMER0 Overflow (Interrupt Enable)
OCIE0A	TIMER0_COMPA_vect	TIMER0 Output Compare A (...)
OCIE0B	TIMER0_COMPB_vect	TIMER0 Output Compare B (...)

Bit	ISR	Beschreibung
TOIE0	TIMER0_OVF_vect	TIMER0 Overflow (Interrupt Enable)
OCIE0A	TIMER0_COMPA_vect	TIMER0 Output Compare A (...)
OCIE0B	TIMER0_COMPB_vect	TIMER0 Output Compare B (...)

```
01 ISR(TIMER0_OVF_vect) {  
02     // [...]  
03 }  
04  
05 static void init(void) {  
06     // Überlaufunterbrechung aktivieren  
07     TIMSK0 |= (1 << TOIE0);  
08  
09     // [...]  
10 }
```



- Zur Erinnerung: $\text{prescaler} \in \{1, 8, 64, 256, 1024\}$

- **Beispiel:**

- 8-bit Timer mit Überlaufinterrupt
 - CPU Frequenz: 16 MHz (ATmega328PB)
 - Ziel: Mit Periode 1 s zählen
- ⇒ Welcher prescaler ist am ressourcenschonendsten?
- ⇒ Wie viele Überlaufinterrupts bis 1 s vergangen ist?
- ⇒ Welcher Fehler entsteht?