

Microcontroller Teil 6 Timer/Counter

DI (FH) Andreas Pötscher

HTL Litec

Timer/Counter werden als Timer verwendet, um exakt zeitgesteuerte Aufgaben zu erledigen. Sie werden als Counter verwendet, um das Auftreten von Hardware-Ereignissen zu zählen.

- ▶ Der ATmega2560 hat zwei 8-Bit-Timer/Counter und vier 16-Bit-Timer/Counter namens Timer/Counter1, 3, 4 und 5. Herzstück jedes Timer/Counter ist ein 8 bzw. 16 Bit breites Zählerstands-Register, das durch bestimmte Zähl-Impulse ständig inkrementiert wird.

- ▶ Der ATmega2560 hat zwei 8-Bit-Timer/Counter und vier 16-Bit-Timer/Counter namens Timer/Counter1, 3, 4 und 5. Herzstück jedes Timer/Counter ist ein 8 bzw. 16 Bit breites Zählerstands-Register, das durch bestimmte Zähl-Impulse ständig inkrementiert wird.
- ▶ Timer/Counter0 und Timer/Counter2 verwenden 8 Bit breite Register. Das bedeutet, dass das Zählerstandsregister Werte von 0 bis 255 annehmen kann. Solche Timer/Counter werden als 8-Bit-Timer/Counter bezeichnet.

- ▶ Der ATmega2560 hat zwei 8-Bit-Timer/Counter und vier 16-Bit-Timer/Counter namens Timer/Counter1, 3, 4 und 5. Herzstück jedes Timer/Counter ist ein 8 bzw. 16 Bit breites Zählerstands-Register, das durch bestimmte Zähl-Impulse ständig inkrementiert wird.
- ▶ Timer/Counter0 und Timer/Counter2 verwenden 8 Bit breite Register. Das bedeutet, dass das Zählerstandsregister Werte von 0 bis 255 annehmen kann. Solche Timer/Counter werden als 8-Bit-Timer/Counter bezeichnet.
- ▶ Timer/Counter1, Timer/Counter3, Timer/Counter4 und Timer/Counter5 besitzen 16 Bit breite Register. Die möglichen Werte des Zählerstandsregisters liegen daher zwischen 0 und 65535.

Der aktuelle Zählerstand ist im Register TCNTn gespeichert, wobei n die Nummer des verwendeten Timer/Counter ist. Für den Timer/Counter 1 also.

```
uint16_t cnt;  
cnt = TCNT1;
```

Clock-Select-Logic - Auswahl der Zähl-Impulse



Figure 1: Clock-Select-Logic von Timer/Counter 1

| Prescaler-Wert | Timerfrequenz f_T | Periodendauer T_T |
|----------------|---------------------|---------------------|
| 1 | 16 MHz | 0,0625 μs |
| 8 | 2 MHz | 0,5 μs |
| 64 | 250 kHz | 4 μs |
| 256 | 62,5 kHz | 16 μs |
| 1024 | 15,625 kHz | 64 μs |

Nach einem Reset beträgt der Zählerstand aller Timer/Counter gleich 0. Angenommen, in der Initialisierungsphase des Hauptprogramms wird beim Timer/Counter1 ein Prescaler-Wert von 256 ausgewählt. Dadurch startet der Timer/Counter1 und sein Zählerstand beginnt sich von null weg zu erhöhen. Wie lange dauert es, bis ein Zählerstand von 32000 erreicht ist?

- ▶ Die Frequenz des Taktsignals, das den Zählerstand erhöht, hat eine Frequenz von
$$f_T = \frac{f_{osc}}{256} = \frac{16MHz}{256} = 62,5kHz$$

- ▶ Die Frequenz des Taktsignals, das den Zählerstand erhöht, hat eine Frequenz von $f_T = \frac{f_{osc}}{256} = \frac{16MHz}{256} = 62,5kHz$
- ▶ Die Periodendauer ist der Kehrwert daraus: $T_T = \frac{1}{f_T} = \frac{1}{62,5kHz} = 16\mu s$

- ▶ Die Frequenz des Taktsignals, das den Zählerstand erhöht, hat eine Frequenz von $f_T = \frac{f_{osc}}{256} = \frac{16MHz}{256} = 62,5kHz$
- ▶ Die Periodendauer ist der Kehrwert daraus: $T_T = \frac{1}{f_T} = \frac{1}{62,5kHz} = 16\mu s$
- ▶ Direkt nach dem Starten des Timer/Counter1 durch Auswahl des Prescaler-Werts von 256 beträgt der Zählerstand noch 0. Eine Zeitspanne von $16\mu s$ später (T_T) wird der Zählerstand von 0 auf 1

- ▶ Die Frequenz des Taktsignals, das den Zählerstand erhöht, hat eine Frequenz von $f_T = \frac{f_{osc}}{256} = \frac{16MHz}{256} = 62,5kHz$
- ▶ Die Periodendauer ist der Kehrwert daraus: $T_T = \frac{1}{f_T} = \frac{1}{62,5kHz} = 16\mu s$
- ▶ Direkt nach dem Starten des Timer/Counter1 durch Auswahl des Prescaler-Werts von 256 beträgt der Zählerstand noch 0. Eine Zeitspanne von $16\mu s$ später (T_T) wird der Zählerstand von 0 auf 1
- ▶ Der Zählerstand erhöht sich genau alle $16\mu s$ um 1. Er ist somit eine Stufenfunktion der Zeit.



Figure 2: Zählerstand am Anfang

- ▶ Insgesamt dauert es bis ein Zählerstand von 32000 erreicht wird dann $32000 * T_T = 32000 * 16\mu s = 512ms$

- ▶ Insgesamt dauert es bis ein Zählerstand von 32000 erreicht wird dann $32000 * T_T = 32000 * 16\mu s = 512ms$
- ▶ Bei einem Zählerstand von 32000 ist die Stufenfunktion Aufgrund der vielen Punkte nicht mehr zu sehen.

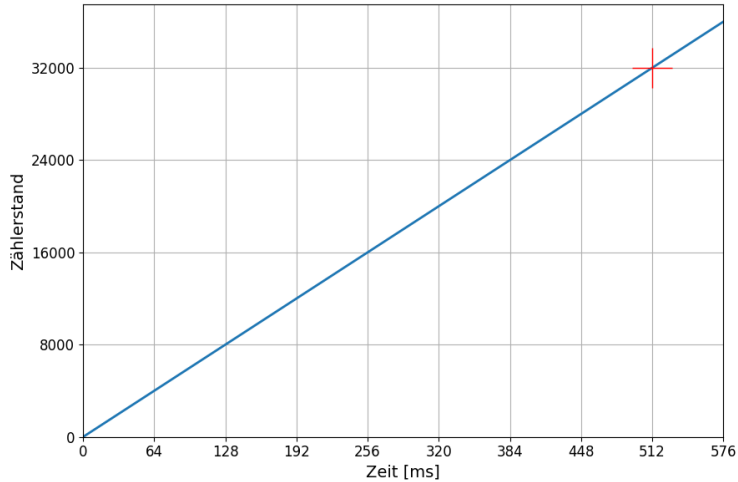


Figure 3: Nach 512 ms wird ein Wert von 32000 erreicht

Die Auswahl der Taktquelle geschieht durch das Setzen der Bits CS_n2 , CS_n1 und CS_n0 (CS steht für Clock Select) im Timer/Counter Control Register B ($TCCR_nB$). Dabei steht das „n“ für die Nummer des jeweiligen Timer/Counters und ist entsprechend zu ersetzen.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|---|---|---|-------|-------|------|------|------|
| TCCR1B: | — | — | — | WGM13 | WGM12 | CS12 | CS11 | CS10 |

| CSn2 | CSn1 | CSn0 | Taktquelle für Timer/Counter |
|------|------|------|--|
| 0 | 0 | 0 | keine Taktquelle der Timer ist gestoppt. |
| 0 | 0 | 1 | Prescaler 1 $f_T = 16MHz$ |
| 0 | 1 | 0 | Prescaler 8 $f_T = 2MHz$ |
| 0 | 1 | 1 | Prescaler 64 $f_T = 250kHz$ |
| 1 | 0 | 0 | Prescaler 256 $f_T = 62,5kHz$ |
| 1 | 0 | 1 | Prescaler 1024 $f_T = 15,625kHz$ |
| 1 | 1 | 0 | Fallende Flanke am GPIO Pin mit der Funktion "Tn" |
| 1 | 1 | 1 | Steigende Flanke am GPIO Pin mit der Funktion "Tn" |

- ▶ Timer/Counter3 soll als Timer mit einem Prescaler-Wert von 256 betrieben werden.

- ▶ Timer/Counter3 soll als Timer mit einem Prescaler-Wert von 256 betrieben werden.
- ▶ Im Register TCCR3B müssen die Bits CS32, CS31, CS30 gesetzt werden.

- ▶ Timer/Counter3 soll als Timer mit einem Prescaler-Wert von 256 betrieben werden.
- ▶ Im Register TCCR3B müssen die Bits CS32, CS31, CS30 gesetzt werden.

| CSn2 | CSn1 | CSn0 | Taktquelle für Timer/Counter |
|------|------|------|-------------------------------|
| 1 | 0 | 0 | Prescaler 256 $f_T = 62,5kHz$ |

```
TCCR3B |= (0x01<<CS32);
```

```
TCCR3B &= ~( (0x01<<CS31) | (0x01<<CS30) );
```

Eine Variable vom Datentyp `uint16_t` (also ein 16-Bit breiter, vorzeichenloser Integer) kann Werte im Bereich von 0 bis $2^{16} - 1$ annehmen, also von 0 bis 65.535.

Was passiert, wenn eine solche Variable ihren maximalen Wert hat und anschließend inkrementiert wird? Welchen Wert enthält die Variable `a` nach Ausführung des folgenden Programmcodes?

```
uint16_t a;  
a = 65535;  
a++;
```

Der Wert $65535 + 1 = 65536$ (also 2^{16}) liegt außerhalb des darstellbaren Wertebereichs einer `uint16_t`-Variablen. Man spricht in diesem Fall von einem sogenannten Überlauf (Overflow). Die Variable erhält dann wieder den Wert 0.



Figure 4: Tacho kurz vor dem Überlauf

- ▶ Gleichzeitig wird bei einem Overflow ein Interrupt-Flag gesetzt.

- ▶ Gleichzeitig wird bei einem Overflow ein Interrupt-Flag gesetzt.
- ▶ Beim Timer/Counter1 heißt dieses Flag TOV1 (Timer Overflow 1).

- ▶ Gleichzeitig wird bei einem Overflow ein Interrupt-Flag gesetzt.
- ▶ Beim Timer/Counter1 heißt dieses Flag TOV1 (Timer Overflow 1).
- ▶ Ist zusätzlich das zugehörige Interrupt-Enable-Flag gesetzt, wird das Hauptprogramm unterbrochen und die zugehörige Interrupt Service Routine (ISR) ausgeführt – in diesem Fall die Routine mit dem Vektor `TIMER1_OVF_vect`.

- ▶ Timer/Counter1 verwendet als Clock-Source die mit einem Prescaler-Wert von 8 geteilte CPU-Taktfrequenz (16MHz). In welchem Zeitabständen erfolgen die Overflows des Zählerstand-Registers?

- ▶ Timer/Counter1 verwendet als Clock-Source die mit einem Prescaler-Wert von 8 geteilte CPU-Taktfrequenz (16MHz). In welchem Zeitabständen erfolgen die Overflows des Zählerstand-Registers?
- ▶ Der Zählerstand wird wegen des Prescaler-Werts von 8 mit einer Frequenz von $f_T = \frac{f_{osc}}{8} = 2MHz$ inkrementiert.

- ▶ Timer/Counter1 verwendet als Clock-Source die mit einem Prescaler-Wert von 8 geteilte CPU-Taktfrequenz (16MHz). In welchem Zeitabständen erfolgen die Overflows des Zählerstand-Registers?
- ▶ Der Zählerstand wird wegen des Prescaler-Werts von 8 mit einer Frequenz von $f_T = \frac{f_{osc}}{8} = 2MHz$ inkrementiert.
- ▶ Die Periodendauer ist der Kehrwert daraus: $T_T = \frac{1}{f_T} = \frac{1}{2MHz} = 0,5\mu s$

- ▶ Timer/Counter1 verwendet als Clock-Source die mit einem Prescaler-Wert von 8 geteilte CPU-Taktfrequenz (16MHz). In welchem Zeitabständen erfolgen die Overflows des Zählerstand-Registers?
- ▶ Der Zählerstand wird wegen des Prescaler-Werts von 8 mit einer Frequenz von $f_T = \frac{f_{osc}}{8} = 2MHz$ inkrementiert.
- ▶ Die Periodendauer ist der Kehrwert daraus: $T_T = \frac{1}{f_T} = \frac{1}{2MHz} = 0,5\mu s$
- ▶ Die Zeitdauer zwischen zwei Overflows beträgt **65536** bzw. 2^{16} mal dieser Zeitspanne (Zählerstände 0 bis **65535** bzw. 0 bis $2^{16} - 1$).

- ▶ Timer/Counter1 verwendet als Clock-Source die mit einem Prescaler-Wert von 8 geteilte CPU-Taktfrequenz (16MHz). In welchem Zeitabständen erfolgen die Overflows des Zählerstand-Registers?
- ▶ Der Zählerstand wird wegen des Prescaler-Werts von 8 mit einer Frequenz von $f_T = \frac{f_{osc}}{8} = 2MHz$ inkrementiert.
- ▶ Die Periodendauer ist der Kehrwert daraus: $T_T = \frac{1}{f_T} = \frac{1}{2MHz} = 0,5\mu s$
- ▶ Die Zeitdauer zwischen zwei Overflows beträgt 65536 bzw. 2^{16} mal dieser Zeitspanne (Zählerstände 0 bis 65535 bzw. 0 bis $2^{16} - 1$).
- ▶ Die Zeitdauer zwischen den Overflows T_{OVF} berechnet sich dann aus $T_{OVF} = T_T * 2^{16} = 32,768ms$

- ▶ Timer/Counter1 verwendet als Clock-Source die mit einem Prescaler-Wert von 8 geteilte CPU-Taktfrequenz (16MHz). In welchem Zeitabständen erfolgen die Overflows des Zählerstand-Registers?
- ▶ Der Zählerstand wird wegen des Prescaler-Werts von 8 mit einer Frequenz von $f_T = \frac{f_{osc}}{8} = 2MHz$ inkrementiert.
- ▶ Die Periodendauer ist der Kehrwert daraus: $T_T = \frac{1}{f_T} = \frac{1}{2MHz} = 0,5\mu s$
- ▶ Die Zeitdauer zwischen zwei Overflows beträgt 65536 bzw. 2^{16} mal dieser Zeitspanne (Zählerstände 0 bis 65535 bzw. 0 bis $2^{16} - 1$).
- ▶ Die Zeitdauer zwischen den Overflows T_{OVF} berechnet sich dann aus $T_{OVF} = T_T * 2^{16} = 32,768ms$
- ▶ Die Frequenz der Overflows $f_{OVF} = \frac{1}{T_{OVF}} = \frac{1}{32,768ms} = 30,52Hz$

Blinklicht mit Timer Overflow Interrupt.

WokWi Link

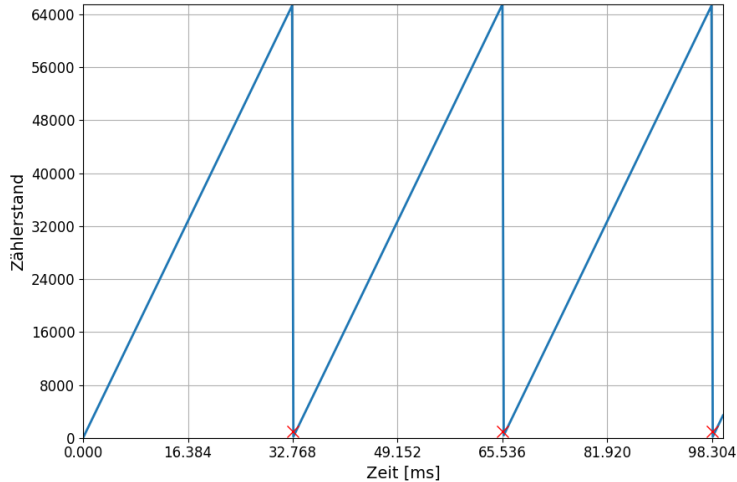


Figure 5: Alle 32,768 ms gibt es einen Timer overflow. Das Overflow Interrupt Flag wird gesetzt.

- ▶ Pro Timer/Counter gibt es mehrere sogenannte Output-Compare-Register.

- ▶ Pro Timer/Counter gibt es mehrere sogenannte Output-Compare-Register.
- ▶ Diese werden vom Programm beschrieben und von der Timer Hardware verwendet.

- ▶ Pro Timer/Counter gibt es mehrere sogenannte Output-Compare-Register.
- ▶ Diese werden vom Programm beschrieben und von der Timer Hardware verwendet.
- ▶ Beim ATmega2560 existieren pro Timer/Counter drei Output-Compare-Register mit den Bezeichnungen `OCRnA`, `OCRnB` und `OCRnC`, wobei „n“ wiederum die Nummer des jeweiligen Timer/Counter ersetzt.

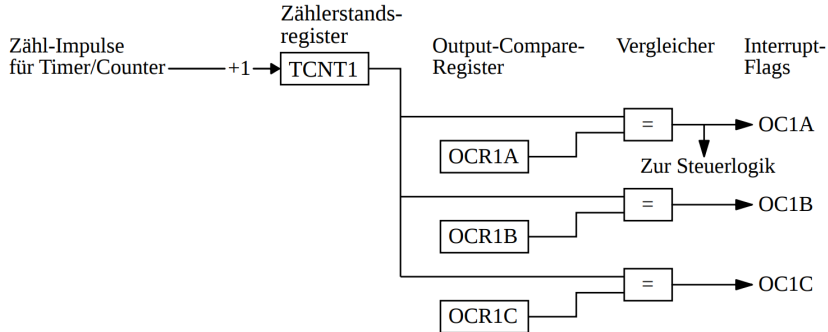


Figure 6: Die Compare Match Units beim ATmega 2560

- ▶ Der aktuelle Wert des Zählerstandsregisters wird kontinuierlich mit den drei Werten der Output-Compare-Register verglichen.

- ▶ Der aktuelle Wert des Zählerstandsregisters wird kontinuierlich mit den drei Werten der Output-Compare-Register verglichen.
- ▶ Immer dann, wenn der Wert des Zählerstandsregisters mit dem eines Output-Compare-Registers übereinstimmt, wird **beim darauffolgenden Zählimpuls** ein entsprechendes Interrupt-Flag gesetzt.

- ▶ Jedes Mal, wenn der Zählerstand des Timer/Counter4 den Wert von 9999 erreicht, soll ein Interrupt ausgelöst werden.

- ▶ Jedes Mal, wenn der Zählerstand des Timer/Counter4 den Wert von 9999 erreicht, soll ein Interrupt ausgelöst werden.
- ▶ Der Wert von OCR1B muss auf $9999 - 1$ gesetzt werden.

- ▶ Jedes Mal, wenn der Zählerstand des Timer/Counter4 den Wert von 9999 erreicht, soll ein Interrupt ausgelöst werden.
- ▶ Der Wert von OCR1B muss auf $9999 - 1$ gesetzt werden.
- ▶ `OCR1B = 9998;`

Die Interrupt Enable Flags befinden sich bei den Timern für jeden Timer in einem eigenen Register. Für Timer 1 im TIMSK1 Register.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------------|---|---|---|---|--------|--------|--------|-------|
| TIMSK1: | — | — | — | — | OCIE1C | OCIE1B | OCIE1A | TOIE1 |

Die Flags OCIE1x sind für die Output Compare Interrupts. Das Flag TOIE1 für den Overflow Interrupt.

| Interrupt-Ereignis: | TCNT1-Overflow | Compare-Match x (TCNT1 == OCR1x+1) |
|----------------------------|-----------------------|---|
| Interrupt Flag | TOV1 | OCF1x |
| Interrupt Enable Flag | TOIE1 | OCIE1x |
| Interrupt Vektor | TIMER1_OVF_vect | TIMER1_COMPx_vect |

Beispiel Timer Overflow Interrupt und OCR Interrupt

WokWi Link

- ▶ Bei den 16-Bit Timer/Counter Modulen gibt es insgesamt 15 Betriebsarten (Modi)

- ▶ Bei den 16-Bit Timer/Counter Modulen gibt es insgesamt 15 Betriebsarten (Modi)
- ▶ Beim Programmstart ist der Normal Mode aktiv.

- ▶ Bei den 16-Bit Timer/Counter Modulen gibt es insgesamt 15 Betriebsarten (Modi)
- ▶ Beim Programmstart ist der Normal Mode aktiv.
- ▶ Dabei verhält sich der Timer wie bereits beschrieben.

- ▶ Bei den 16-Bit Timer/Counter Modulen gibt es insgesamt 15 Betriebsarten (Modi)
- ▶ Beim Programmstart ist der Normal Mode aktiv.
- ▶ Dabei verhält sich der Timer wie bereits beschrieben.
- ▶ Alternativ dazu wird hier der CTC-Mode behandelt.

- ▶ Clear Timer on Compare Match.

- ▶ Clear Timer on Compare Match.
- ▶ Im CTC Modus wird der Zählerstand nur lange inkrementiert, bis er dem Wert des OCR1A Registers entspricht.

- ▶ Clear Timer on Compare Match.
- ▶ Im CTC Modus wird der Zählerstand nur lange inkrementiert, bis er dem Wert des OCR1A Registers entspricht.
- ▶ Wenn dieser Wert erreicht wird, wird mit dem nächsten Zählimpuls das Zählregister auf 0 zurückgesetzt

- ▶ Clear Timer on Compare Match.
- ▶ Im CTC Modus wird der Zählerstand nur lange inkrementiert, bis er dem Wert des OCR1A Registers entspricht.
- ▶ Wenn dieser Wert erreicht wird, wird mit dem nächsten Zählimpuls das Zählregister auf 0 zurückgesetzt
- ▶ Der größte mögliche Wert ist also nicht 65535 sondern OCR1A

- ▶ Clear Timer on Compare Match.
- ▶ Im CTC Modus wird der Zählerstand nur lange inkrementiert, bis er dem Wert des OCR1A Registers entspricht.
- ▶ Wenn dieser Wert erreicht wird, wird mit dem nächsten Zählimpuls das Zählregister auf 0 zurückgesetzt
- ▶ Der größte mögliche Wert ist also nicht 65535 sondern OCR1A
- ▶ Der CTC Modus eignet sich zum genauen Festlegen von Interrupt Intervallen.

Die Interrupt Frequenz berechnet sich nach folgender Formel:

$$f_{OCR1A} = \frac{f_{osc}}{N * (1 + OCR1A)}$$

- ▶ Als erstes kann die Timer Frequenz berechnet werden.

$$f_{T1} = \frac{f_{osc}}{N} = \frac{16MHz}{64} = 250kHz$$

- ▶ Als erstes kann die Timer Frequenz berechnet werden.

$$f_{T1} = \frac{f_{osc}}{N} = \frac{16MHz}{64} = 250kHz$$

- ▶ Daraufhin die Periodendauer. $T_{T1} = \frac{1}{f_{T1}} = \frac{1}{250kHz} = 4\mu s$

- ▶ Als erstes kann die Timer Frequenz berechnet werden.
$$f_{T1} = \frac{f_{osc}}{N} = \frac{16MHz}{64} = 250kHz$$
- ▶ Daraufhin die Periodendauer. $T_{T1} = \frac{1}{f_{T1}} = \frac{1}{250kHz} = 4\mu s$
- ▶ Es werden immer wieder die Zählerstände von 0 bis 6 durchlaufen. Also insgesamt 7 Zählerstände. $T_{OCR1A} = T_{T1} * (OCR1A + 1) = 4\mu s * (6 + 1) = 28\mu s$

- ▶ Als erstes kann die Timer Frequenz berechnet werden.

$$f_{T1} = \frac{f_{osc}}{N} = \frac{16MHz}{64} = 250kHz$$

- ▶ Daraufhin die Periodendauer. $T_{T1} = \frac{1}{f_{T1}} = \frac{1}{250kHz} = 4\mu s$

- ▶ Es werden immer wieder die Zählerstände von 0 bis 6 durchlaufen. Also insgesamt 7 Zählerstände. $T_{OCR1A} = T_{T1} * (OCR1A + 1) = 4\mu s * (6 + 1) = 28\mu s$

- ▶ Die Interrupt Frequenz beträgt dann. $f_{OCR1A} = \frac{1}{T_{OCR1A}} = \frac{1}{28\mu s} = 35,714kHz$

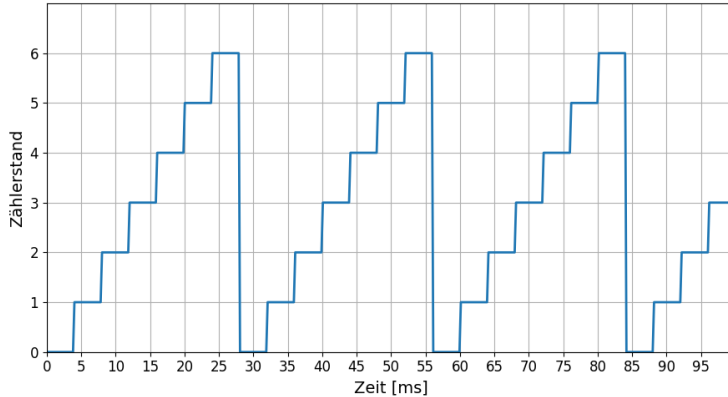


Figure 7: Zählerstand des Timer 1 im CTC Mode bei $OCR1A = 6$

Die Timer Modi werden über die WGM_n (Wave Generation Mode) Bits konfiguriert.

| WGM_n3 | WGM_n2 | WGM_n1 | WGM_n0 | Timer Mode |
|-------------------------|-------------------------|-------------------------|-------------------------|-------------------|
| 0 | 0 | 0 | 0 | Normal Mode |
| ... | ... | ... | ... | |
| 0 | 1 | 0 | 0 | CTC Mode |

Die WGM Bits sind auf zwei Register verteilt.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------|---|---|---|---|---|---|-------|-------|
| TCCR1A: | — | — | — | — | — | — | WGM11 | WGM10 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------|---|---|---|-------|-------|------|------|------|
| TCCR1B: | — | — | — | WGM13 | WGM12 | CS12 | CS11 | CS10 |

Timer/Counter1 soll im CTC-Mode betrieben werden. Dabei soll er als Timer mit einem Prescaler-Wert von 8 arbeiten

- ▶ Für den CTC Mode muss das Bit WGM12 gesetzt werden. `TCCR1B |= 0x01 << WGM12;`

Timer/Counter1 soll im CTC-Mode betrieben werden. Dabei soll er als Timer mit einem Prescaler-Wert von 8 arbeiten

- ▶ Für den CTC Mode muss das Bit WGM12 gesetzt werden. `TCCR1B |= 0x01 << WGM12;`
- ▶ Für den Prescaler von 8 muss das Bit CS11 gesetzt werden. `TCCR1B |= 0x01 << CS11;`

Oftmals ist es notwendig eine Funktion mit einer genauen Frequenz zu wiederholen. Z.B:

- ▶ Auslesen eines Messwertes immer zum gleichen Zeitpunkt.

Oftmals ist es notwendig eine Funktion mit einer genauen Frequenz zu wiederholen. Z.B:

- ▶ Auslesen eines Messwertes immer zum gleichen Zeitpunkt.
- ▶ Berechnen von Stellgrößen bei einem digitalen Regler.

Oftmals ist es notwendig eine Funktion mit einer genauen Frequenz zu wiederholen. Z.B:

- ▶ Auslesen eines Messwertes immer zum gleichen Zeitpunkt.
- ▶ Berechnen von Stellgrößen bei einem digitalen Regler.
- ▶ Erzeugen von Signalen mit konstanter Frequenz.

Interrupts mit genauer Frequenz auslösen

- ▶ Dazu wird der CTC Mode verwendet.

- ▶ Dazu wird der CTC Mode verwendet.
- ▶ In den meisten Fällen ist aber die Frequenz f_{OCR1A} bekannt und es soll daraus der Prescaler N und der OCR1A Wert bestimmt werden.

- ▶ Dazu wird der CTC Mode verwendet.
- ▶ In den meisten Fällen ist aber die Frequenz f_{OCR1A} bekannt und es soll daraus der Prescaler N und der OCR1A Wert bestimmt werden.
- ▶ $OCR1A = \frac{f_{osc}}{N * f_{OCR1A}} - 1$

Timer/Counter1 soll im CTC-Mode betrieben werden. Es soll dabei eine Interruptfrequenz f_{OCR1A} von 100 Hz eingestellt werden.

Welche OCR1A Werte ergeben sich für die 5 möglichen Prescaler.

| Prescaler | OCR1A berechnet | OCR1A möglich |
|-----------|-----------------|---------------|
| 1 | | |
| 8 | | |
| 64 | | |
| 256 | | |
| 1024 | | |

Für die 5 möglichen Prescaler ergeben sich folgende OCR1A Werte.

| Prescaler | OCR1A berechnet | OCR1A möglich |
|-----------|-----------------|---------------|
| 1 | 159999 | |
| 8 | 19999 | |
| 64 | 2499 | |
| 256 | 624 | |
| 1024 | 156,25 | |

Von den 5 berechneten OCR1A Werte können folgende eingestellt werden.

| Prescaler | OCR1A berechnet | OCR1A möglich |
|-----------|-----------------|---------------|
| 1 | 159999 | — |
| 8 | 19999 | 19999 |
| 64 | 2499 | 2499 |
| 256 | 624 | 624 |
| 1024 | 156,25 | 156 |

Für manche Frequenzen kann der Wert aber mit keinem einzigen Prescaler genau eingestellt werden. Z.B. für eine Frequenz von 300 Hz

| Prescaler | OCR1A berechnet | OCR1A möglich | f_{OCR1A} berechnet |
|-----------|-----------------|---------------|-----------------------|
| 1 | 53332,33 | 53332 | 300,0018 Hz |
| 8 | 6665,66 | 6666 | 299,9850 Hz |
| 64 | 832,33 | 832 | 300,1200 Hz |
| 256 | 207,33 | 207 | 300,4807 Hz |
| 1024 | 51,0833 | 51 | 300,4807 Hz |

Für solche Fälle ist es am besten, den kleinsten möglichen Prescaler zu verwenden, da auch dort der Fehler zur gewünschten Frequenz am kleinsten ist.

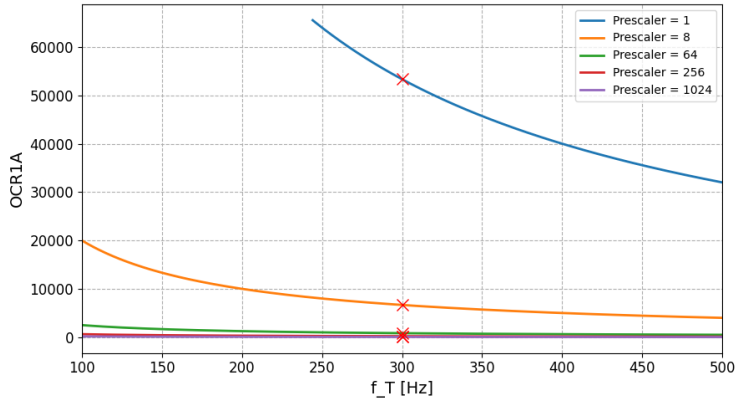


Figure 8: Die berechneten OCR1A Werte für alle möglichen Prescaler. Die Werte für 300 Hz sind hervorgehoben

Beispiel Interrupt mit 300 Hz auslösen.

WokWi Link

- ▶ Soft PWM steht für Software Pulsweitenmodulation.

- ▶ Soft PWM steht für Software Pulsweitenmodulation.
- ▶ Mit einer Pulsweitenmodulation kann im weitesten Sinne ein analoger Ausgang nachgebildet werden.

- ▶ Soft PWM steht für Software Pulsweitenmodulation.
- ▶ Mit einer Pulsweitenmodulation kann im weitesten Sinne ein analoger Ausgang nachgebildet werden.
- ▶ Dabei wird ein digitaler Ausgang mit einer fixen Frequenz ein- und ausgeschaltet.

- ▶ Soft PWM steht für Software Pulsweitenmodulation.
- ▶ Mit einer Pulsweitenmodulation kann im weitesten Sinne ein analoger Ausgang nachgebildet werden.
- ▶ Dabei wird ein digitaler Ausgang mit einer fixen Frequenz ein- und ausgeschaltet.
- ▶ Das Verhältnis von Einschalt- und Periodendauer bestimmt dann den Ausgangswert.

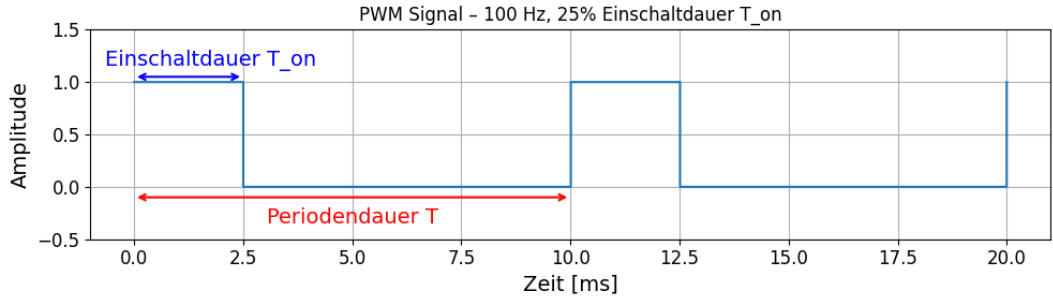


Figure 9: Pulsweitenmodulation

- ▶ Für eine Software PWM kann sehr gut ein Timer im CTC Mode verwendet werden.

- ▶ Für eine Software PWM kann sehr gut ein Timer im CTC Mode verwendet werden.
- ▶ Dazu soll eine PWM mit einer Frequenz von 1kHz verwendet werden. Als ersten Schritt muss der Prescaler und der OCR1A Wert bestimmt werden.

- ▶ Für eine Software PWM kann sehr gut ein Timer im CTC Mode verwendet werden.
- ▶ Dazu soll eine PWM mit einer Frequenz von 1kHz verwendet werden. Als ersten Schritt muss der Prescaler und der OCR1A Wert bestimmt werden.
- ▶ $OCR1A = \frac{f_{osc}}{N * f_{OCR1A}} - 1$

- ▶ Für eine Software PWM kann sehr gut ein Timer im CTC Mode verwendet werden.
- ▶ Dazu soll eine PWM mit einer Frequenz von 1kHz verwendet werden. Als ersten Schritt muss der Prescaler und der OCR1A Wert bestimmt werden.
- ▶ $OCR1A = \frac{f_{osc}}{N * f_{OCR1A}} - 1$
- ▶ Wir nehmen den kleinsten Prescaler von 1 an.

- ▶ Für eine Software PWM kann sehr gut ein Timer im CTC Mode verwendet werden.
- ▶ Dazu soll eine PWM mit einer Frequenz von 1kHz verwendet werden. Als ersten Schritt muss der Prescaler und der OCR1A Wert bestimmt werden.
- ▶ $OCR1A = \frac{f_{osc}}{N * f_{OCR1A}} - 1$
- ▶ Wir nehmen den kleinsten Prescaler von 1 an.
- ▶ $OCR1A = \frac{16000kHz}{1 * 1kHz} - 1 = 15999$

- ▶ Für die PWM sind dann zwei Interrupts notwendig.

- ▶ Für die PWM sind dann zwei Interrupts notwendig.
- ▶ Einer zum Einschalten und einer zum Ausschalten der LED.

- ▶ Für die PWM sind dann zwei Interrupts notwendig.
- ▶ Einer zum Einschalten und einer zum Ausschalten der LED.
- ▶ Es werden der Output Compare A und Output Compare B Interrupt verwendet.

- ▶ Für die PWM sind dann zwei Interrupts notwendig.
- ▶ Einer zum Einschalten und einer zum Ausschalten der LED.
- ▶ Es werden der Output Compare A und Output Compare B Interrupt verwendet.
- ▶ Im OCA Interrupt wird die Led eingeschaltet und im OCB ausgeschaltet.

- ▶ Für die PWM sind dann zwei Interrupts notwendig.
- ▶ Einer zum Einschalten und einer zum Ausschalten der LED.
- ▶ Es werden der Output Compare A und Output Compare B Interrupt verwendet.
- ▶ Im OCA Interrupt wird die Led eingeschaltet und im OCB ausgeschaltet.
- ▶ Für 25% Einschaltdauer ergibt sich dann für OCR1B ein Wert von 4000;

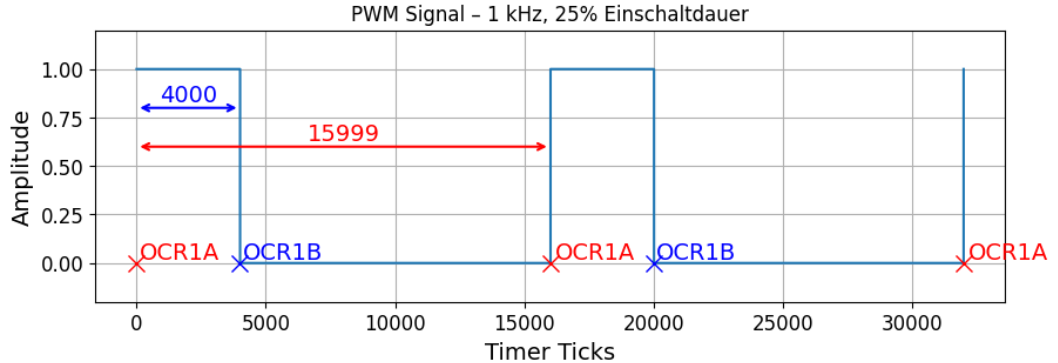


Figure 10: Pulsweitenmodulation mit Timer

Als Beispiel zum Dimmen einer Led.

WokWi Link