Übung 04: Timer

Hardware:

- Basis: Arduino, Steckbrett, Kabel

Informationen:

- Datenblatt ATmega2560: Kapitel 17 und Kapitel 18
- Pin Mapping: https://www.arduino.cc/en/Hacking/PinMapping2560

Aufgabe 1: Vorbereitung

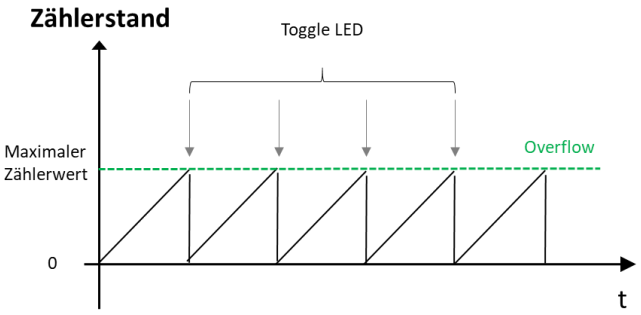
Bauen Sie folgende Schaltung: Eine blaue LED wird über einen Vorwiderstand angeschlossen an `Digital Port 24 (D24)`. Bei `HIGH` an `D24` soll die LED leuchten, bei `LOW` soll sie ausgeschaltet sein.

- Testen Sie mit dem rechten Programm!
- Welchem Pin und Port des Mikrocontrollers entspricht die Buchse `Digital Pin 24` des Arduino MEGA Boards?

```
void setup() {
      pinMode(24, OUTPUT);
}
void loop() {
      digitalWrite(24, HIGH);
}
```

Aufgabe 2: LED Blinkfrequenz mit Prescaler

Bei jedem Overflow des 16-Bit Zählers `Timer4` soll die LED umgeschaltet („getoggelt“) werden. Nach einem Overflow soll der Timer wieder bei 0 beginnen Die Taktrate ohne Prescaler beträgt 16 MHz.



a) Welche Prescaler-Stufen hat der ATmega 2560? (Datenblatt, Seite 164)

b) Die Geschwindigkeit des Zählers lässt sich durch Vorschalten eines *Prescalers* variieren.
Was ist die minimale und die maximale Blinkfrequenz der LED, die man somit erreichen kann?

c) Schreiben Sie nun die Software (***keine Arduino Library!***)
Hinweise:

- Register `TCCR4A` und `TCCR4B` anfangs auf 0x00 initialisieren.
- Prescaler in `TCCR4B` konfigurieren, (Datenblatt S. 156)
- Timer-Interrupt in `TIMSK4` Register aktivieren (Datenblatt S. 161f.)
- „Toggeln“ innerhalb der ISR implementieren. Die passende ISR hat den Namen `ISR(TIMER4_OVF_vect)`, siehe[1]
- Die „`loop`“-Methode muss leer bleiben.
- Beim ersten Test: 64er Prescaler verwenden!

d) Variieren Sie den Prescaler und beobachten Sie die Änderung!

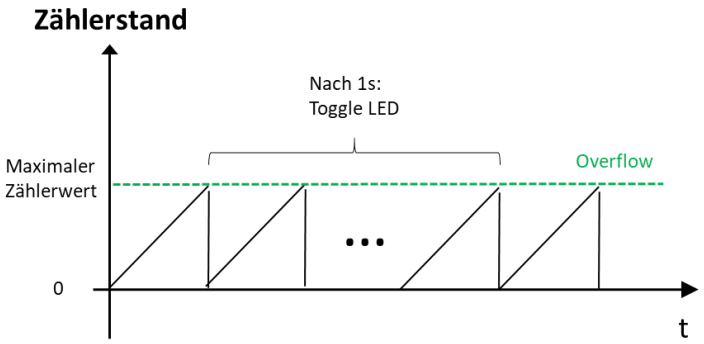
[1] http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html

Aufgabe 3: LED Blinkfrequenz ohne Prescaler

Das Programm wird geändert:

- Die LED soll mit der Frequenz 0,5 Hz blinken und muss somit jede Sekunde „getoggelt“ werden.
- Es darf aber nun ***kein Prescaler*** verwendet werden.

Alle anderen Bedingungen aus Aufgabe 2) bleiben erhalten.

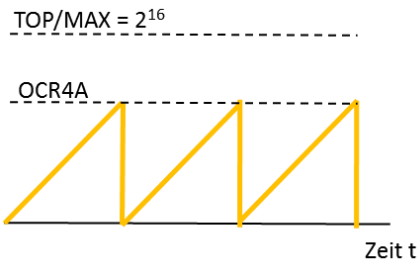


a) Wie lange (in ms) dauert es *ohne* Prescaler bis ein Overflow eintritt? Wie viele Overflows ereignen sich innerhalb von 1 Sekunde bzw. innerhalb von 2 Sekunden?

b) Modifizieren Sie Ihr Programm aus Aufgabe 2!
- Zählen Sie in einer Variablen die Anzahl der aufgetretenen Overflows.
- Schalten Sie die LED erst nach einer ausreichenden Zahl an Overflow-Ereignissen um.

Aufgabe 4: LED Blinkfrequenz mit Output Compare

Das **gleiche** Verhalten wie in Aufgabe 2) und 3) wird nun mit *Output Compare* implementiert. Erreicht der Timer einen Schwellwert, der im *Output Compare Register* (`OCR4A`) konfiguriert ist, wird ein Interrupt ausgelöst und der Zähler auf 0 zurückgesetzt. In der ISR wird die LED umgeschaltet.



a) Überlegen Sie sich mit welchem Wert das `OCR4A` Register bei einem *clk/256 Prescaler* initialisiert werden muss!

b) Schreiben Sie Ihr Programm aus Aufgabe 3 erneut um! Beachten Sie:
- *clk/256-Prescaler* verwenden!
- Verwenden Sie den *Clear Timer on Compare Match (CTC Mode)* um bei Erreichen des Wertes in `OCR4A` den Zähler automatisch zurückzusetzen: Datenblatt S. 145! Dazu müssen die Bits `WGM40` bis `WGM43` passend gesetzt werden. Unschön: Die Bits `WGM40` und `WGM41` befinden sich in Register `TCCR4A` (Seite 154), die beiden anderen Bits in `TCCR4B` (Seite 156).
- Die benötigte ISR nennt sich `ISR(TIMER4_COMPA_vect)`[1].