Technische Hochschule Rosenheim
Technical University of Applied Sciences

Embedded Systems
Kapitel 4: Timer

Prof. Dr. Wolfgang Mühlbauer
Fakultät für Informatik
wolfgang.muehlbauer@fh-rosenheim.de

Sommersemester 2020

Motivation

- Abwarten einer festen Zeitspanne
  - Ggfs. Ausführen anderer Aufgaben während des Wartens.
- Messen von Zeitintervallen
  - Zeitdauer bis sich Zustand an GPIO Pin ändert.
- Zählen von (externen) Ereignissen
  - Wie viele steigende Flanken gab es?
- Erzeugen von Impulsfolgen
  - Bsp1: Generierung der Baudrate, z.B. bei UART
  - Bsp2: Pulsweitenmodulation zur Ansteuerung von Motoren
    - Siehe nächstes Kapitel.
- **Lösung**: Hardware **Timer/Counter** auf Mikrocontrollern

gemeinfrei

Inhalt

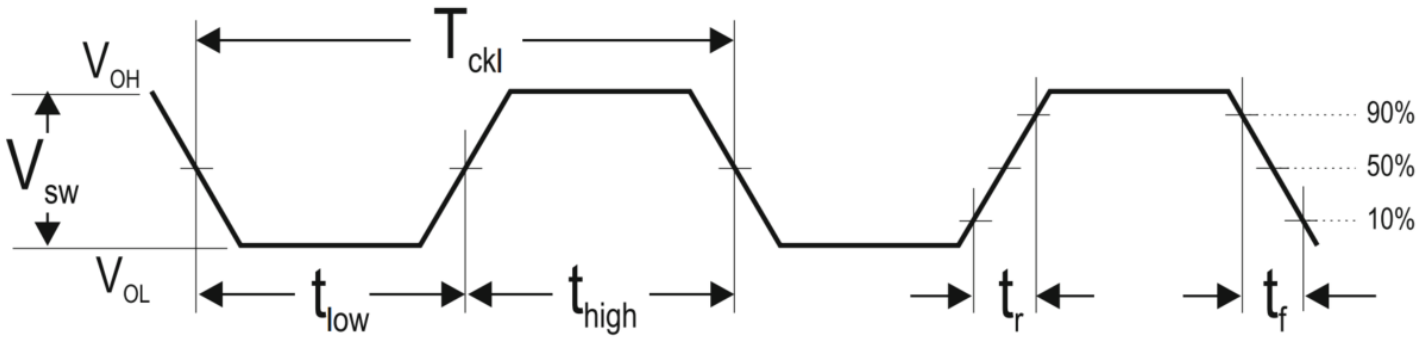
- **Aufbau und Grundanwendungen eines Timers**
- Prescaler
- Input Capture und Output Compare
- Timer beim ATmega2560

Takt und Taktgeneratoren

- Takt erlaubt ***Synchronisation*** von Schaltkreisen / Flipflops
- Takt durch *interne* oder *externe* ***Oszillatoren*** erzeugt.

Quarzoszillator

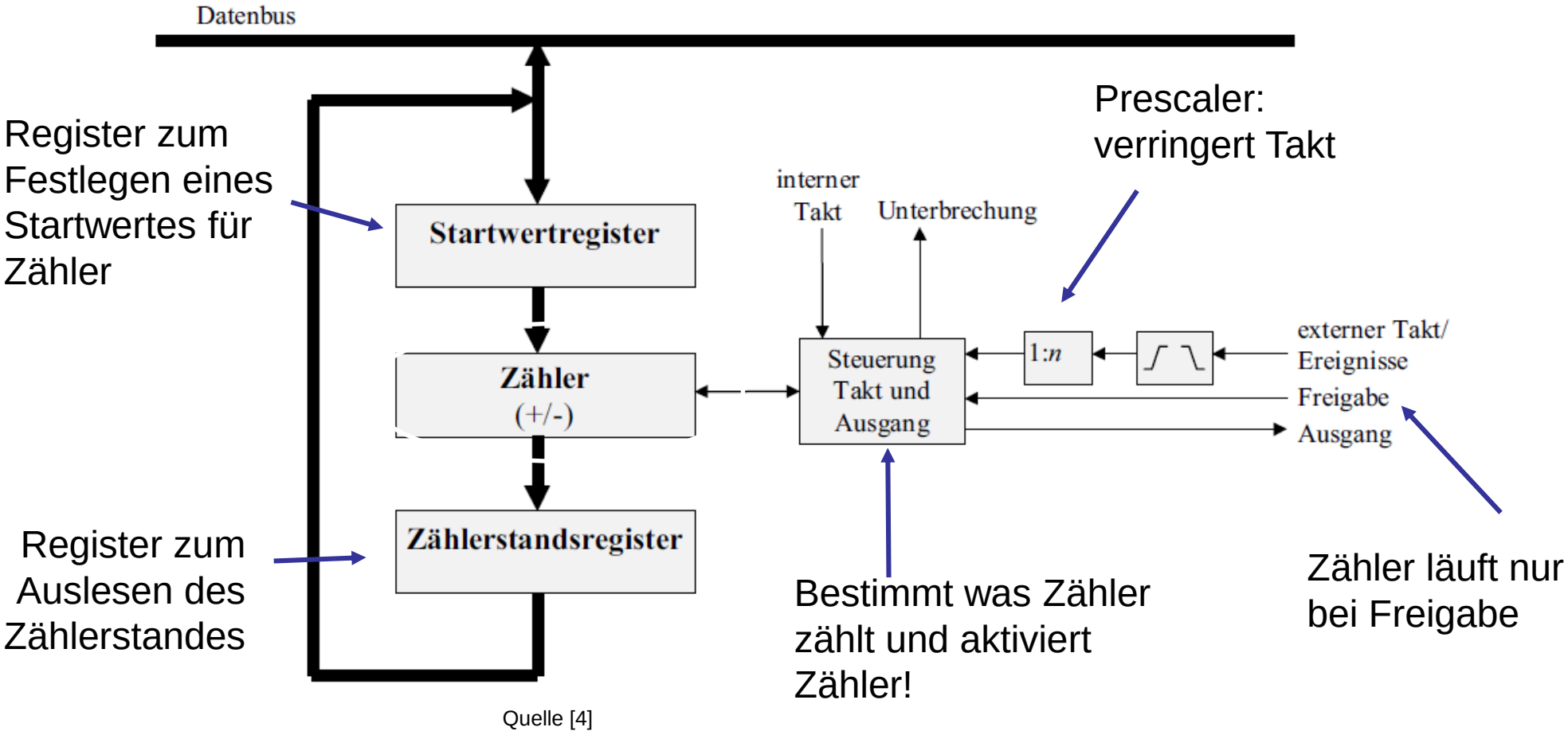
- ***Kenngrößen***
  - *Frequenz*
  - *Duty Cycle:* Dauer von t_{high} im Verhältnis zur Periode.
  - *Clock Stability:* Abweichung (Toleranz) von der Nominalfrequenz.
  - *Clock Jitter:* Zufällige Schwankungen in der Frequenz.
  - *Clock Drift:* Systematische Frequenzänderung über Zeit / Alterung



Quelle [3]

Aufbau: Mikrocontroller-Timer

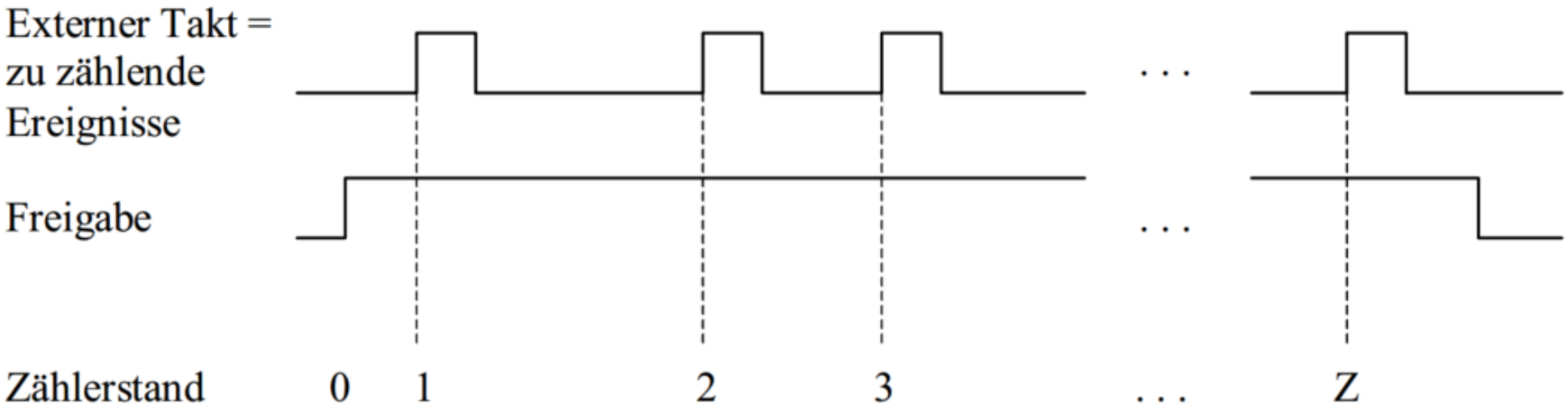
- **Counter** (*dt. Zähler*) implementieren **Timer** (*dt. Zeitgeber*).



Quelle [4]

Zählen von Ereignissen

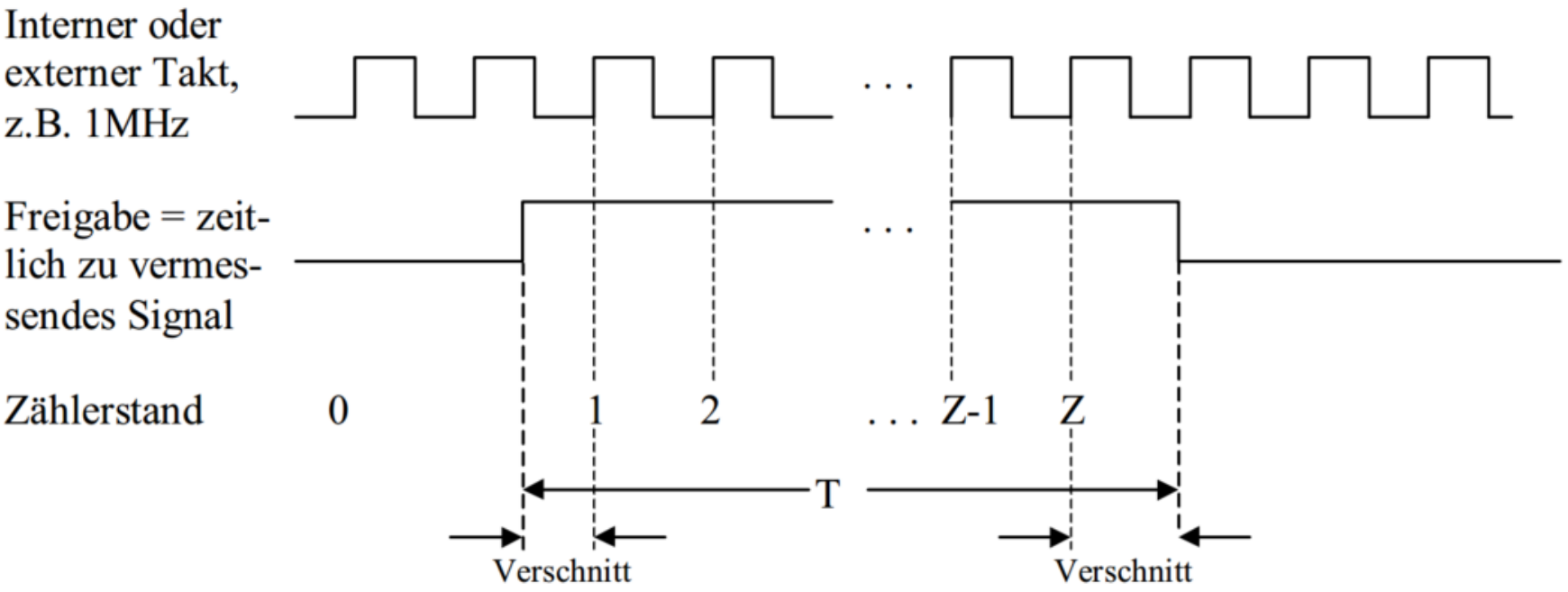
- Externer Eingang wird als "Takt" für Zähler verwendet.
- Jede steigende Taktflanke erhöht den Zählerstand um 1.
- Anwendungsbeispiel?



Quelle: [4]

Messen von Zeiten

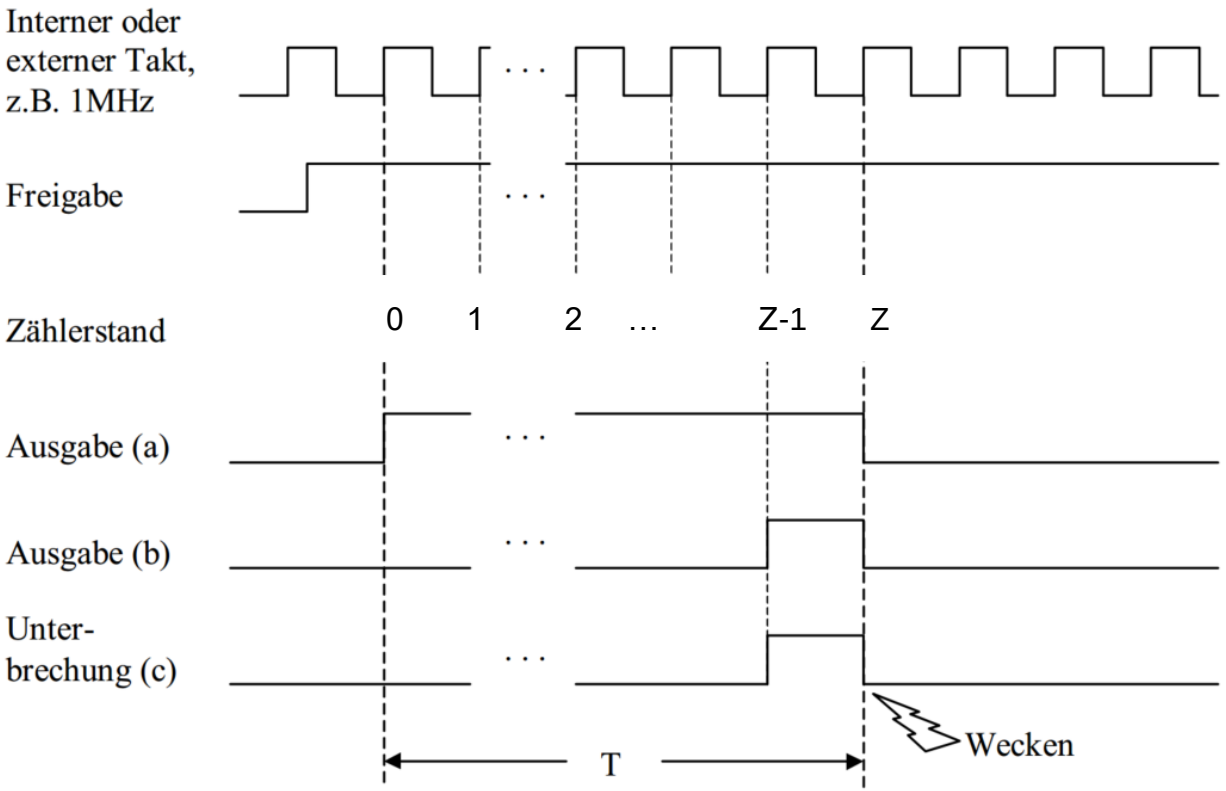
- Zu vermessendes Signal wird an "Freigabe" angeschlossen und aktiviert/deaktiviert somit den Zähler.
- *Zu messende Zeit = Zählerstand * Taktzykluszeit*
- Evtl. Problem: Messgenauigkeit



Quelle: [4]

Einmaliges Wecken

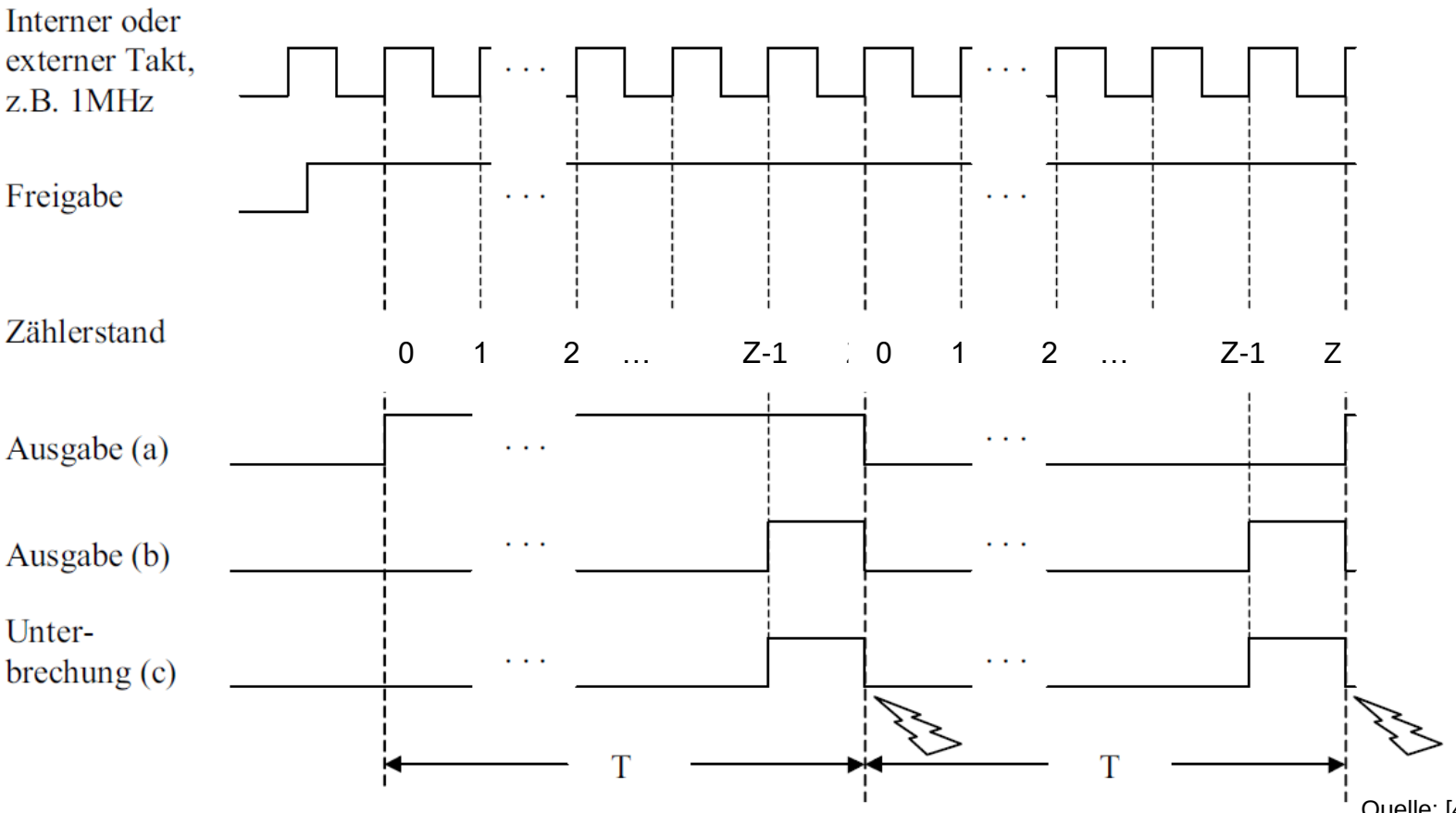
- Programmieren eines Schwellwertes (hier ***Z***) und Aufwärtszählen bis zu Schwellwert.
- Varianten: Erzeuge
  - (a) Impulsdauer der Länge ***Z***
  - (b) kurzen Impuls der Länge 1 wenn Schwellwert erreicht wird
  - (c) Erzeuge Interrupt bei Erreichen des Schwellwertes.



Quelle: [4]

Periodische Impulse

- Unterschied zu vorheriger Folie: Zähler setzt sich selbst nach Erreichen des Schwellwertes zurück.



Quelle: [4]

Inhalt

- Aufbau und Grundanwendungen eines Timers
- **Prescaler**
- Input Capture und Output Compare
- Timer beim ATmega2560

Prescaler: Motivation

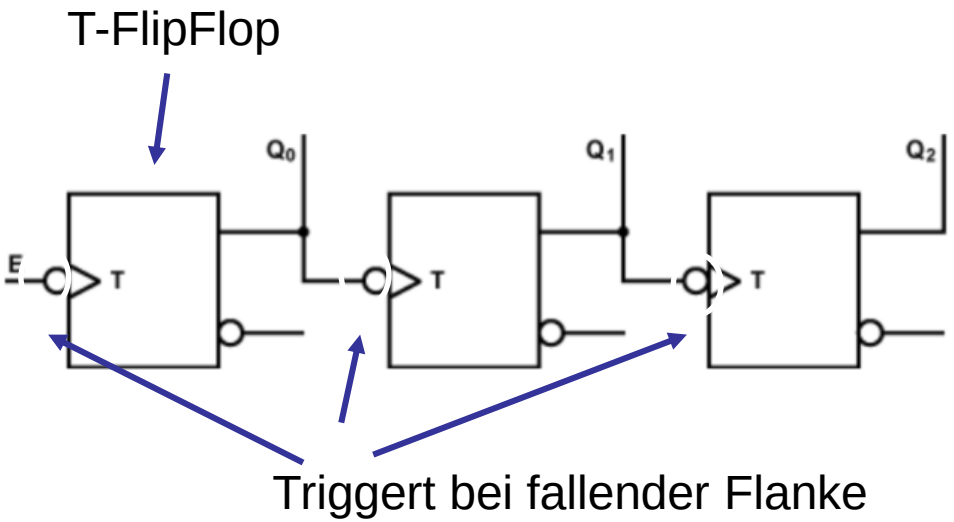
- Systemtakt des ATmega2560 beträgt $clk = 16Mhz$.
- Ein 16-Bit Timer wird durch den Systemtakt angesteuert
  - Startwert: 0
  - Zählt kontinuierlich hoch.

- ***Frage***
  - Wie lange dauert es bis dieser 16-Bit Timer überläuft?

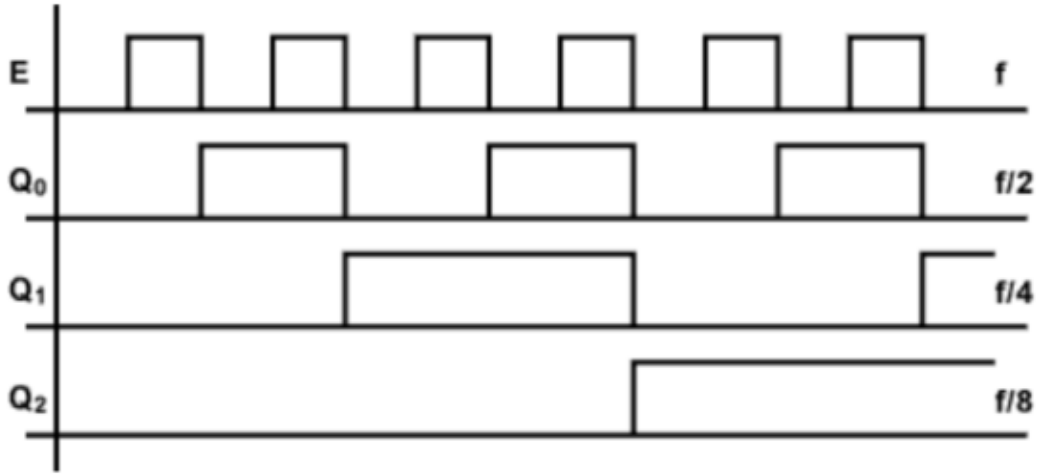
- ***Fazit***
  - Ohne weitere SW- oder HW-Maßnahmen können Mikrocontroller-Timer nicht zum Messen oder Abwarten von langen Zeitspannen verwendet werden.

Prescaler (dt. "Vorteiler")

- Mehrere T-Flip-Flops werden in Reihe vor Timer geschaltet.
  - Beispiel: "Toggelt" Zustand bei jeder fallenden Flanke am Eingang
- Prescaler ähnelt selbst einem Zähler.
- T-Flip-Flop wechselt mit jedem Taktimpuls seinen Ausgangszustand:



Hardwareschaltung für einen Prescaler



Signalverlauf

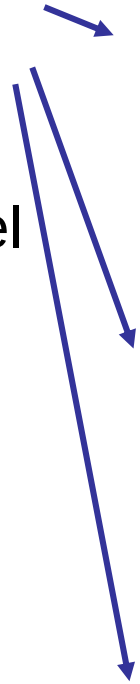
Prescaler: Funktionsweise

- Timer ist mit bestimmten Bit Q_n des Prescalers getaktet.

  Fallende Flanke am vordersten Bit ist Eingang des eigentlichen Timers

- *n* ist variabel und in der Regel konfigurierbar.

- Beispiel: Linkes Bit des 2-Bit Prescalers viertelt die Frequenz.



Prescaler (2 Bit)	Timer (8 Bit)
00	00000000
01	00000000
10	00000000
11	00000000
00	**00000001**
01	00000001
10	00000001
11	00000001
00	**00000010**
01	00000010
10	00000010
11	00000010
00	**00000011**

Prescaler: Diskussion

- Konfigurierbar welches Bit des Prescalers den Timer-Zähler triggert.
  - Bei 8-Bit Prescaler minimal mögliche Frequenz $\rightarrow f : 256$
- ***Vorteil*** großer Prescaler
  - Messen langer Zeiten möglich, längere Dauer bis zu Overflow.
- ***Nachteil*** großer Prescaler.
  - Die Auflösung nimmt ab.
  - **Frage**: Kleinstes Zeitintervall, das mit f/1024-Prescaler bei 16 MHz noch messbar ist?
- ***Richtlinie***
  - Verwende kleinstmöglichen Prescaler!

Prescaler: Weitere Überlegung

- Ein Mikrocontroller arbeitet mit 1 MHz.
- Es gibt einen 16-Bit Timer.
- Es stehen die folgenden Prescaler zur Verfügung
  - 8, 64, 256 und 1024.
- Es sollen Zeitintervalle der Größenordnung [0*s*; 3*s*] gemessen werden.
  - Forderung: Kein Overflow für diese Zeitintervalle!

- ***Frage***
  - Welchen Prescaler würden Sie nehmen?

Inhalt

- Motivation
- Aufbau und Grundanwendungen eines Timers
- Prescaler
- **Input Capture und Output Compare**
- Timer beim ATmega2560

8-Bit und 16-Bit-Timer

- **Vorteile**
  - 8- Bit: Einfache HW-Schaltung
  - 16-Bit: Größerer Zählbereich.

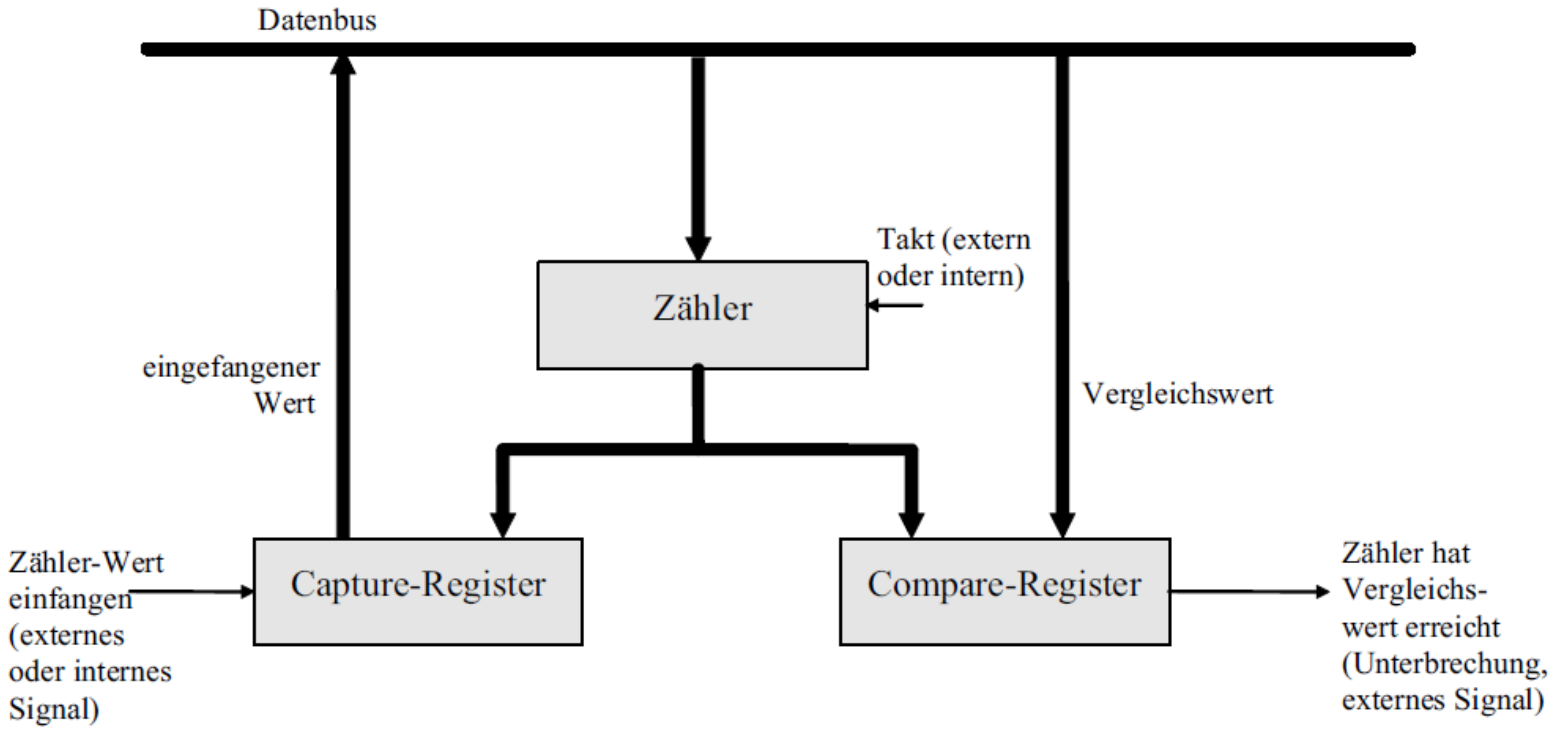
- **Problematik:** 16-Bit Timer auf 8-Bit Mikrocontroller
  - Low und High Byte müssen zeitlich hintereinander gelesen werden.
  - Es könnten inkonsistente Werte gelesen werden, wenn sich während Lesen der Zählerstand ändert.
  - Abhilfe: Puffer-Register, der C-Compiler „richtet das“.

- **ATmega2560**
  - 2 * 8-Bit Timer
  - 4 * 16-Bit Timer

- **Diskussion:**
  - Wie macht man aus zwei 8-Bit Timer einen 16-Bit Timer?

Input Capture und Output Compare

- Häufiges bei Mikrocontroller-Timer vorhanden.
- **Input Capture**: Bei externen oder internen Signalen / EEreignissen wird aktueller Zählerstand in *Input Capture Register* gespeichert (***Snapshot***).
- **Compare Register**: Bei Erreichen eines konfigurierten Zählerstandes wird Interrupt ausgelöst oder bestimmtes Signal erzeugt.



Quelle: [4]

Input Capture

- ***Verwendung***
  - Messen eines Zeitstempels für Ereignisse

- ***Funktionsweise***
  - Timer getaktet mit Systemuhr des Mikrocontrollers
  - Bei Eintreten eines auslösendes Ereignisses:
    - Kopieren des aktuellen Timer-Wertes in spezielles Register
    - Setzen eines Flags, optional: Auslösen eines Interrupts
  - Auslösendes Ereignis
    - Extern: z.B. steigende Flanke an einem Eingang
    - Intern: z.B. Ausgang eines Komparator ändert sich (nicht behandelt)

- ***Achtung:***
  - Genauigkeit des Prescalers beeinflusst Messgenauigkeit!
  - 2 kurz hintereinander auftretende Ereignisse können dazu führen, dass nur der Zeitstempel des 2. Ereignisses gemessen wird.

Output Compare

- ***Verwendung***
  - Zu einem bestimmten Zeitpunkt einen Ausgang schalten

- ***Funktionsweise***
  - *Output Compare Register*: Konfiguration des Timer Wertes, bei dem Ereignis ausgelöst werden soll.
  - Mögliche Reaktionen, wenn konfigurierter Wert des Timers erreicht wird:
    - Auslösen eines Interrupts
    - Automatisches Aktivieren oder Deaktivieren eines Ausgangs
    - Zustand eines Ausgangs ändern (Toggle)
  - Manchmal *Reset* Option: Bei Erreichen des Timer-Wertes erfolgt Zurücksetzen auf Startwert.

- Output Compare wird für Pulsweitenmodulation (PWM) verwendet
  - Siehe nächstes Kapitel!

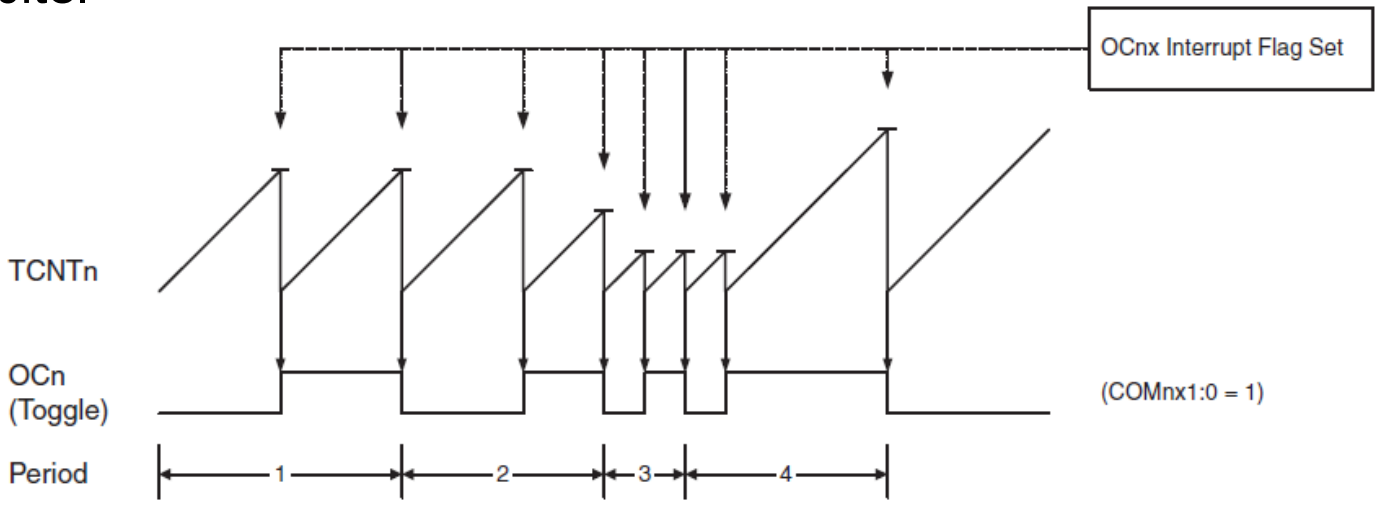
Inhalt

- Aufbau und Grundanwendungen eines Timers
- Prescaler
- Input Capture und Output Compare
- **Timer beim ATmega2560**

ATmega2560: Betriebsmodi des Zählers

- Zähler zählt immer nach oben.
- ***Normal Mode***
  - Bei Overflow fängt Zähler automatisch wieder bei 0 an.
- ***Clear Timer on Compare Match Mode (CTC)***
  - Konfiguriere Maximalwert (OCRxA oder ICRx Register).
  - Zähler setzt sich bei Erreichen dieses Wertes selbst automatisch auf 0.
- ***PWM Modes***
  - Siehe nächstes Kapitel

Betriebsmodus CTC



[2], Seite 147

Überblick: Wichtige Timer- Register beim ATmega2560

- **TCCRnA:** Timer/Counter n Control Register A
  - Nächste Vorlesung, Pulsweitenmodulation.
  - Vorerst einfach auf 0x00 setzen.
- **TCCRnB**: Timer/Counter n Control Register B
  - Einstellen des Prescalers + Starten des Timers
  - Input Capture
- **TCNTn**: Timer Counter n (16 Bit)
  - Aktueller Zählerstand
- **OCRnA** und **OCRnB**: Output Compare Register (16 Bit)
  - Wert gegen den Zählerstand verglichen werden kann.
- **ICRn** (16 Bit)
  - Hier wird bei *Input Capture* der erfasste Zeitwert gespeichert.
- **TIMSKn**
  - Aktivieren und Deaktivieren der *Timer Interrupts*
- **TIFRn**
  - Timer-bezogene Interrupt Flags

Das „n“ bezieht sich immer auf die konkrete Timerinstanz. Atmega: `Timer0, Timer1, …, Timer 4`

Timer und Interrupts

- Beispiel:
  - Bei Overflow des `Timer0` soll Interruptroutine isr() aufgerufen werden.
- Wichtige Referenz:
  - http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html
- Programmiermuster:

```
void setup() {
          // timer setup
          // activate interrupt for timer overflow
}


ISR(TIMER0_OVF_vect)        {
          // ISR called when timer overflows

}

void loop() {
          // do some other work
}
```

Programmierung: Wartefunktion

- **`delay(x)`** (Arduino Library)
  - Funktioniert nicht in Interrupt Service Routine.
  - Blockiert!
  - Basiert intern auf `Timer0`
  - https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/wiring.c

- **`millis()`** (Arduino Library)
  - Wird innerhalb von Interrupt Service Routine nicht hochgezählt.
  - Blockiert nicht.
  - Speichere Zeitstempel: `timestamp =millis()`
  - Teste danach mit `if (millis() - timestamp > xxx`), ob genügend Zeit verbangen ist.

- Mit **Interrupts** (AVR Libc)
  - Siehe Übung und vorherige Folie.
  - Beispiel: Nach Overflow wird Programm unterbrochen.
  - Blockiert nicht.

Zusammenfassung

- Timer-Anwendungen, z.B.
  - Zählen von (schnellen) externen Ereignissen
  - Messen von Zeitabständen
  - Erzeugen von festen Delays
  - Periodisches Ausführen von Programmteilen

- Timer und Interrupts sind **elementar** für die Umsetzung eines Echtzeitsystems.
  - Echte Parallelität durch Timer- und Interrupt-Modul.

Quellenverzeichnis

[1] G. Gridling und B. Weiss. *Introduction to Microcontrollers*, Version 1.4, 26. Februar 2007, Kapitel 2.5, verfügbar online: https://ti.tuwien.ac.at/ecs/teaching/courses/mclu/theory-material/Microcontroller.pdf (abgerufen am 08.03.2017)

[2] Datenblatt ATmega2560, http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf, (abgerufen am 19.03.2017)

[3] M. Jimenez, R. Palomera und I.Couvertier. Introduction to Embedded Systems, Springer Verlag, 2014

[4] U. Brinkschulte und T. Ungerer. *Mikrocontroller und Mikroprozessoren*, 3. Auflage, Springer Verlag, 2017.