Embedded Systems
Kapitel 5: Pulsweitenmodulation

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

Sommersemester 2020

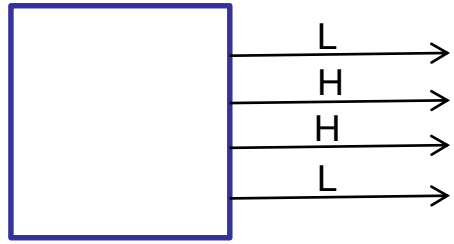
Inhalt

- **Grundlagen: Pulsweitenmodulation**
- ATmega2560: Konfiguration des PWM Modus
- Vorbereitung der Übung

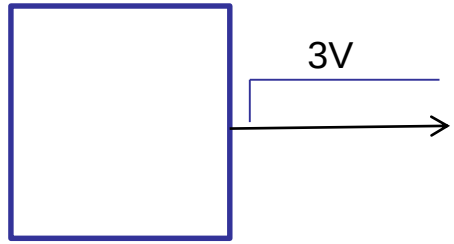
Motivation

- Wie kann Mikrocontroller analoge Information nach außen übertragen?
  - *Beispiel*: Wert 6 aus dem Bereich [0; 10]

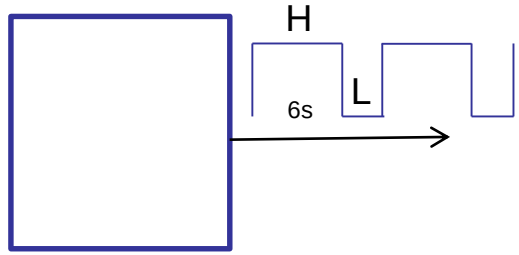
- > 1 Pin, parallele Leitungen
  - HIGH/LOW auf den Leitungen werden als Binärzahl interpretiert.
  - *Beispiel*: 0110



- 1 Pin, gibt analogen Spannungswert aus
  - *Beispiel*: 3V aus möglichem Bereich [0; 5V] für den Wert 6



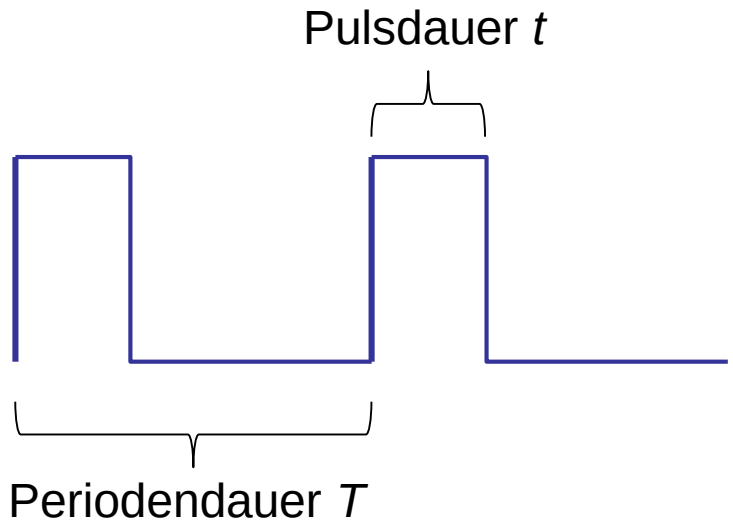
- 1 Pin, berücksichtige zeitliche Dimension
  - *Beispiel*: Alle 10s wird der Pin für 6s auf HIGH gesetzt
    → ***Pulsweitenmodulation!***



Pulsweitenmodulation (PWM)

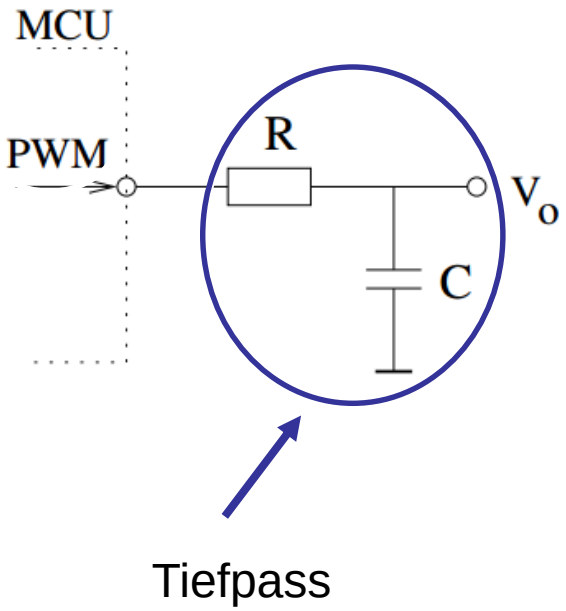
- **Definition: *Pulsweitenmodulation***
  - *Modulationsart* bei der ein Signal mit ***konstanter Periode***, aber ***variabler Pulsdauer***, erzeugt wird.
  - Information steckt im *Pulsdauer.*
  - *Pulsdauer*: HIGH-Anteil innerhalb einer Periode.

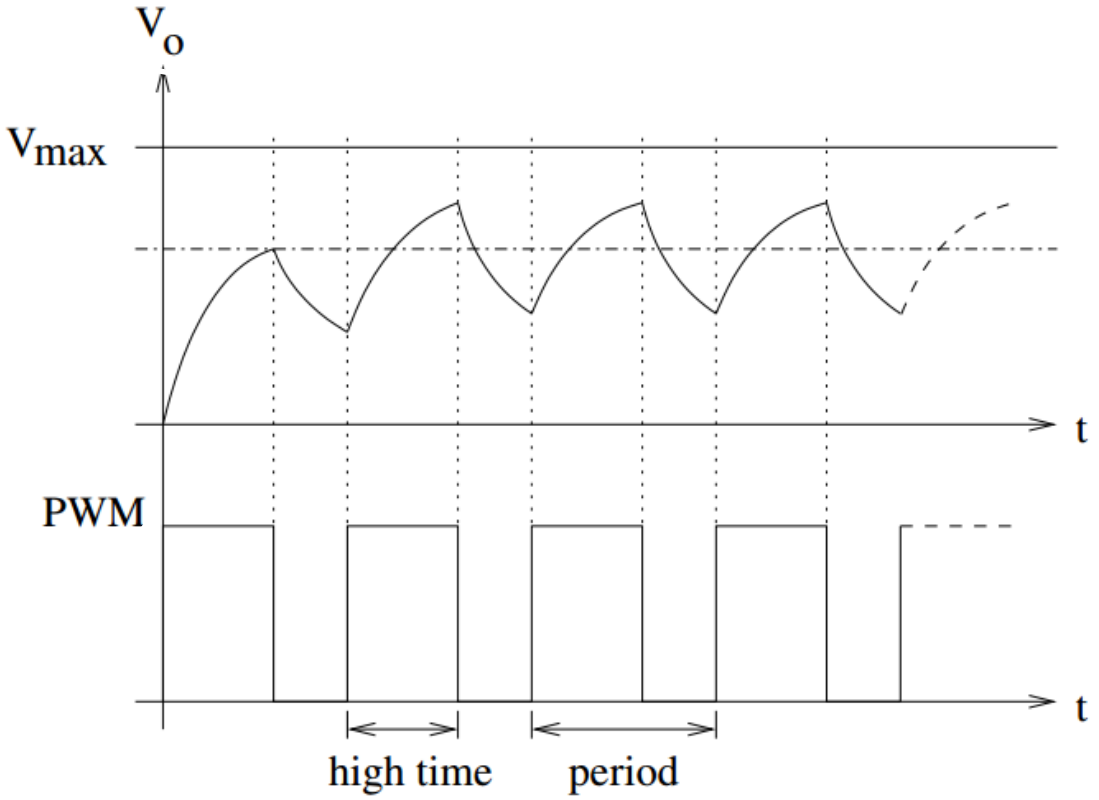
- **Begriffe:**
  - ***Period*** (dt. Periodendauer)
  - ***Pulse width*** (dt. Pulsdauer)
  - ***Duty Cycle*** (dt. Tastgrad)
    - Duty Cycle: $\frac{t}{T}$
    - Oft in Prozent angegeben!



Anwendungen der PWM (1)

- ***Emulation von analoger Ausgabe***
  - PWM Signal kann in analoges Signal umgewandelt werden. .
  - RC-Baugruppe ist ein Tiefpass und filtert hochfrequente Signale.
  - Annäherungsweise kommt der gewünschte analoge Wert V_0 heraus.

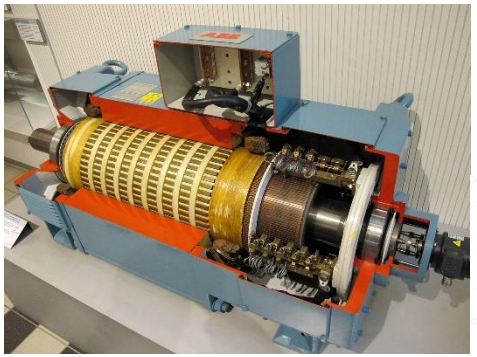




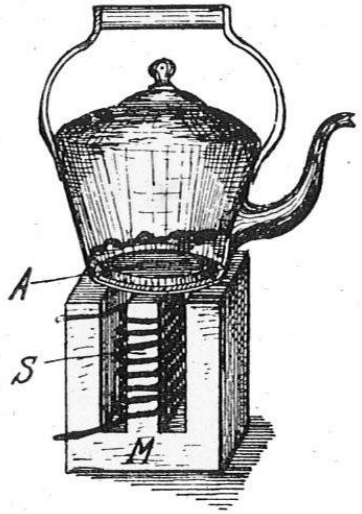
Quelle: [1], Seite 46

Anwendungen der PWM (2)

- Übertragen analoger Messwerte
- Ansteuern von Gleichstrommotoren
  - Variation des Duty Cycles verändert Leistung / Geschwindigkeit eines Motors stufenlos.
  - https://de.wikipedia.org/wiki/Gleichstrommaschine#/media/File:Animation_einer_Gleichstrommaschine_(Variante).gif

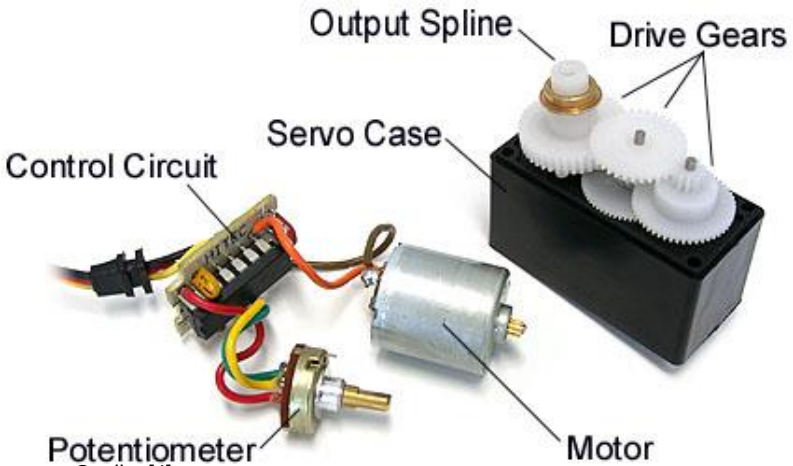


- Allgemeine Steuerungstechnik
  - Dimmen von LEDs
  - Drehzahländerung von Lüftern
- Leistungselektronik
  - Z.B. Ansteuern von Heizelementen

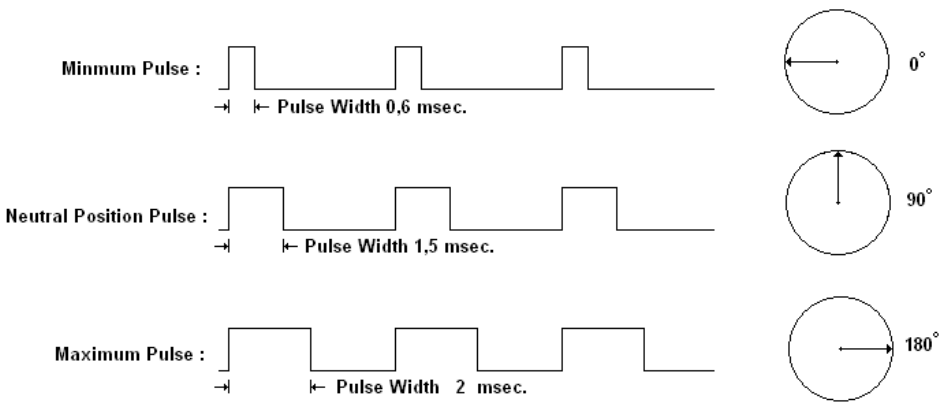


Servomotor

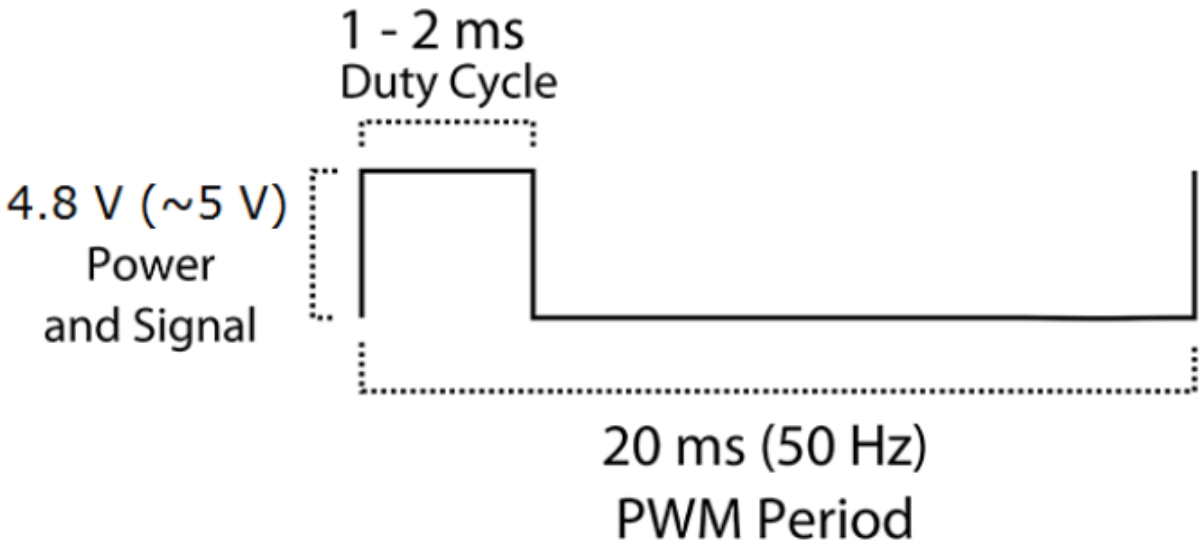
- Bei Servomotor wird beständig die Position oder Geschwindigkeit kontrolliert.
- ***Bestandteile***
  - Beliebiger (Elektro)motor
  - Messeinheit (z.B. Potentiometer)
  - Steuerung
- Integrierte ***Messeinheit*** misst permanent aktuelle Position/Geschwindigkeit.
- Integrierte ***Steuerung*** hält Motor in gewünschter Position/Geschwindigkeit.
- Pulsdauer der PWM teilt Servomotor die gewünschte Position/Geschwindigkeit mit.



Quelle: [4]



Übung: Servomotor SG90



Quelle: Datenblatt SG90

- Die Periodendauer wird nicht ausgewertet.
- Nur der Duty Cycle!
- Zu lange oder zu kurze Impulsdauern schaden dem Motor.
- "*Dead bandwidth*": Änderungen des Duty Cycles von 10 µs bewirken keine Positionsänderung des Servomotors.

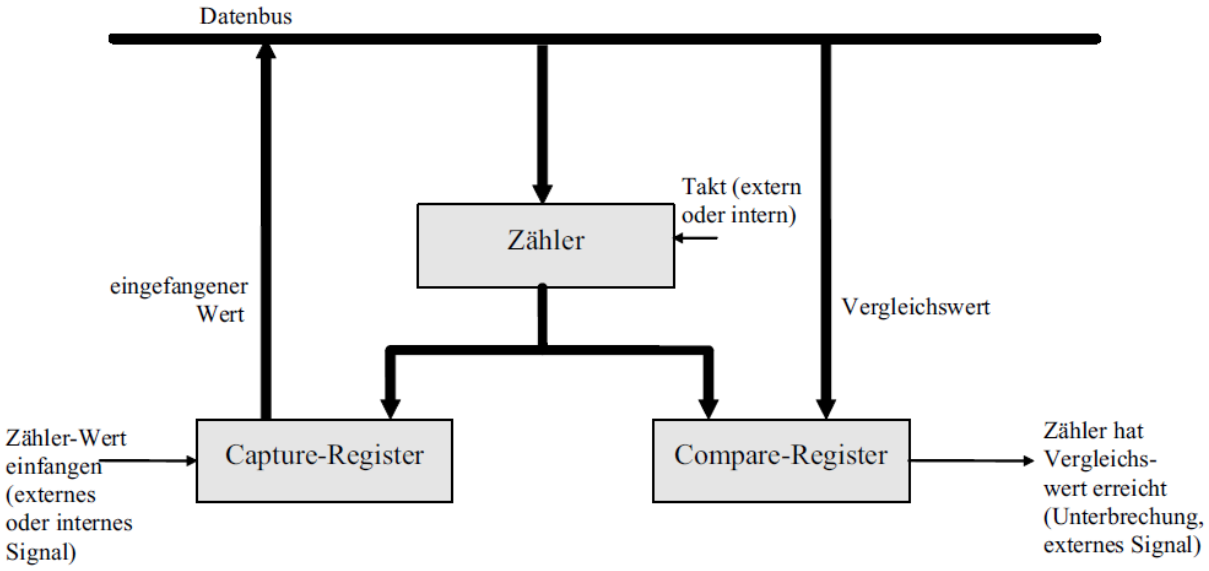
Inhalt

- Grundlagen: Pulsweitenmodulation
  - Analoger Ausgang, Impulsfolge
- **ATmega2560: Konfiguration des PWM Modus**
- Vorbereitung der Übung

Wiederholung: Output Compare

- Konfiguration eines Schwellwertes im *Output Compare Register*
  - **`OCRnX`**: Der ATmega2560 hat für jeden `Timer n` drei solche Schwellwerte **`OCRnA, OCRnB, OCRnC`**.

- Bei Erreichen des Schwellwertes
  - Auslösen eines Interrupts oder
  - Ändern des Pegels an speziellem ***Output Compare Pins*** durch Hardware.
    - **`OCnX`** gehört zum Schwellwert in **`OCRnX`**.

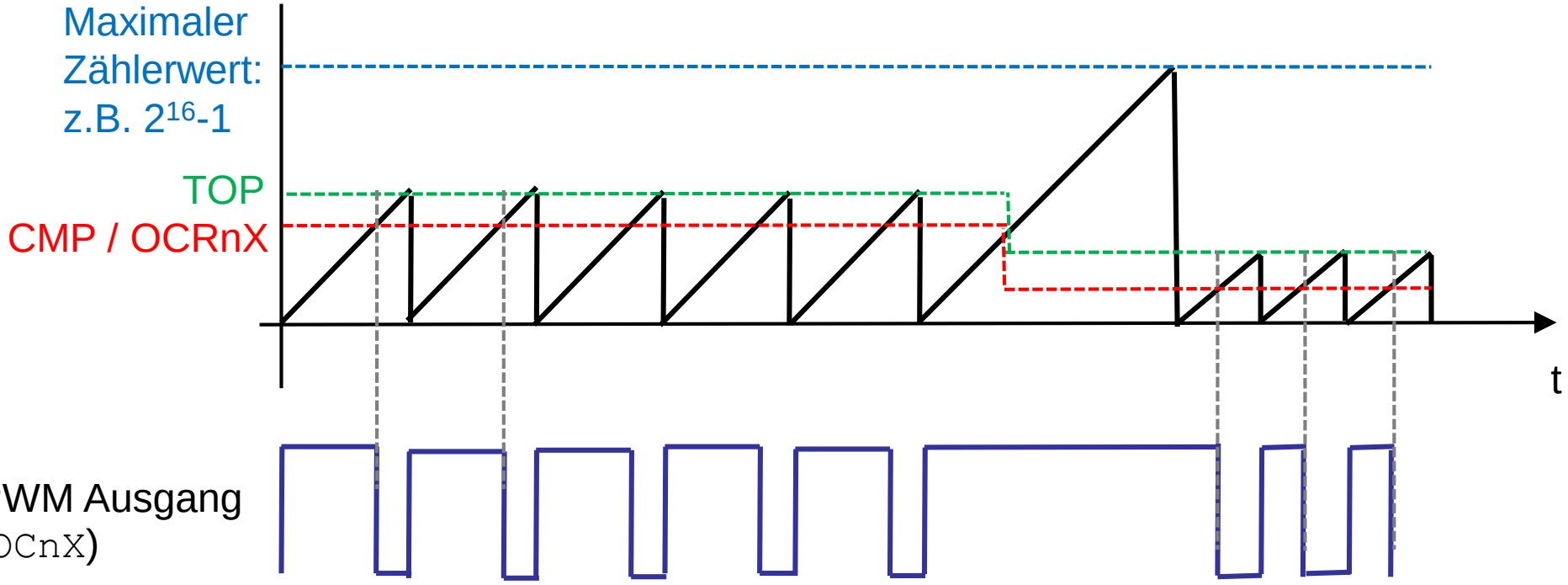


Quelle: [4]

Details: siehe Datenblatt [2]
Abbildung 17-3, Seite 140
Abbildung 17-4, Seite 142

Erzeugung von PWM Signalen

- ***"Zutaten":*** Timer, Output Compare und Interrupts
- **CMP / `OCRnX`**:
  - Zähler läuft unverändert weiter.
  - Output Compare Match setzt PWM-Ausgang `OCnX` automatisch auf LOW.
- ***TOP***:
  - Umkehrwert des Zählers
  - Gleichzeitig wird PWM-Ausgang automatisch auf HIGH gesetzt.



Fazit: Erzeugung von PWM Signalen

- Mikrocontroller generieren PWM Signal meist ***durch Hardware***
  - Keine Programmierung, aber Konfiguration was passieren soll.
  - Software Alternative: Digitalen Ausgangspin manuell zwischen LOW und HIGH umschalten (z.B. mittels `delay()`).

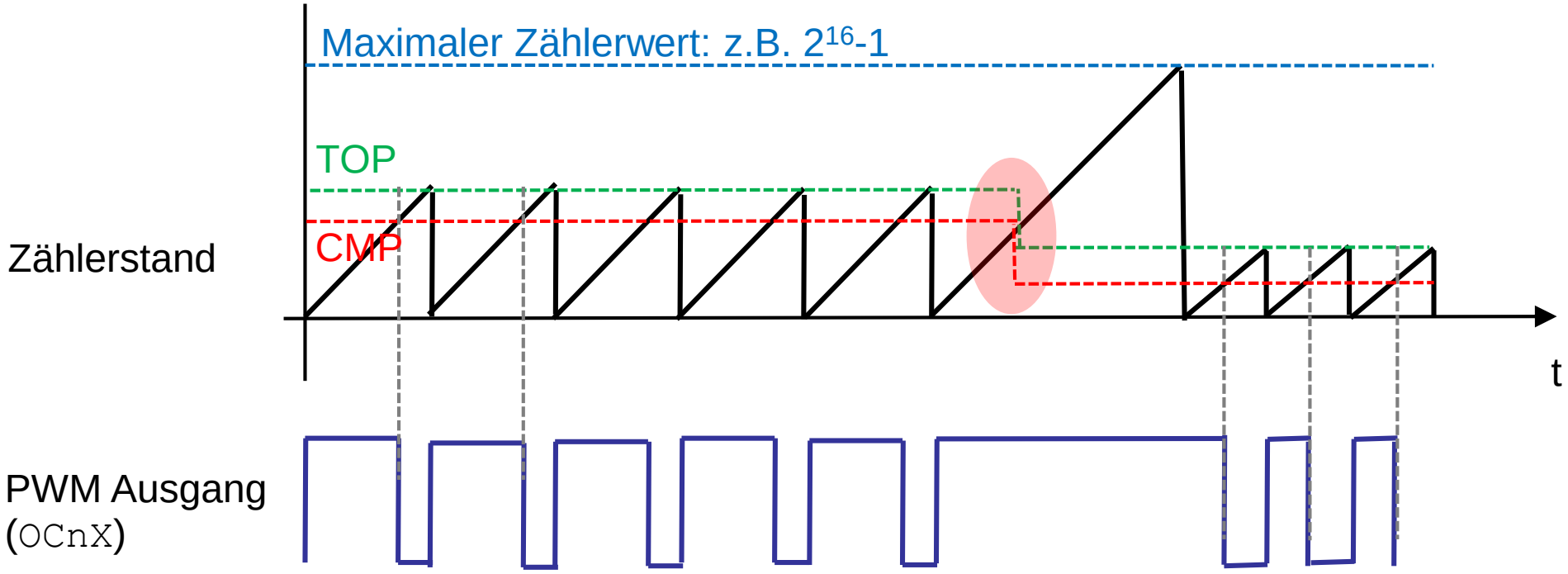
- PWM ist somit ein Timer-Feature.

- Automatische Erzeugung des PWM Signals nur auf speziellen ***Output Compare Pins*** **`OCnX`** möglich.
  - Pro Timer beim Atmega2560 3 PWM Ausgänge `A`, `B` und `C`.
  - `n` ist Nummer des Timers an, `X` ist entweder `A`, `B` oder `C`.
  - Im Register `OCRnX` wird jeweils der Schwellwert gesetzt, der den PWM-Ausgang `OCnX` beinflusst (z.B. `OCRnB` bestimmt `OCnB`).

Änderung der PWM Parameter zur Laufzeit

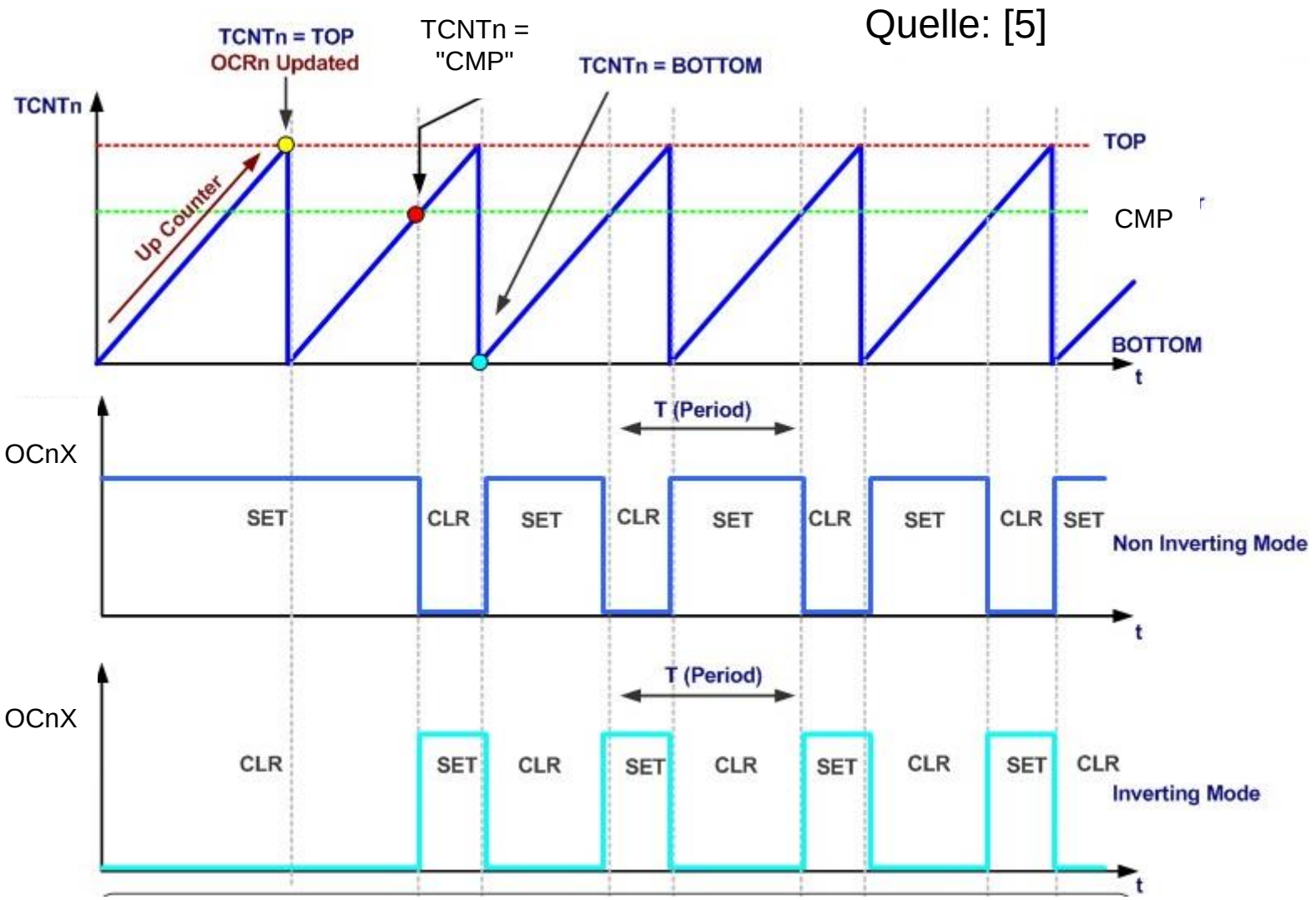
- Änderungen des Duty Cycles (`CMP`) oder der Periodendauer (`TOP`) können einmalig zu Artifakten führen, z.B. sehr langes HIGH oder LOW-Signal.

- `TOP/CMP` lässt sich in manchen Modi nur ändern, wenn Zähler gerade auf `BOTTOM/TOP`.



Inverting und Non-Inverting Mode

Quelle: [5]



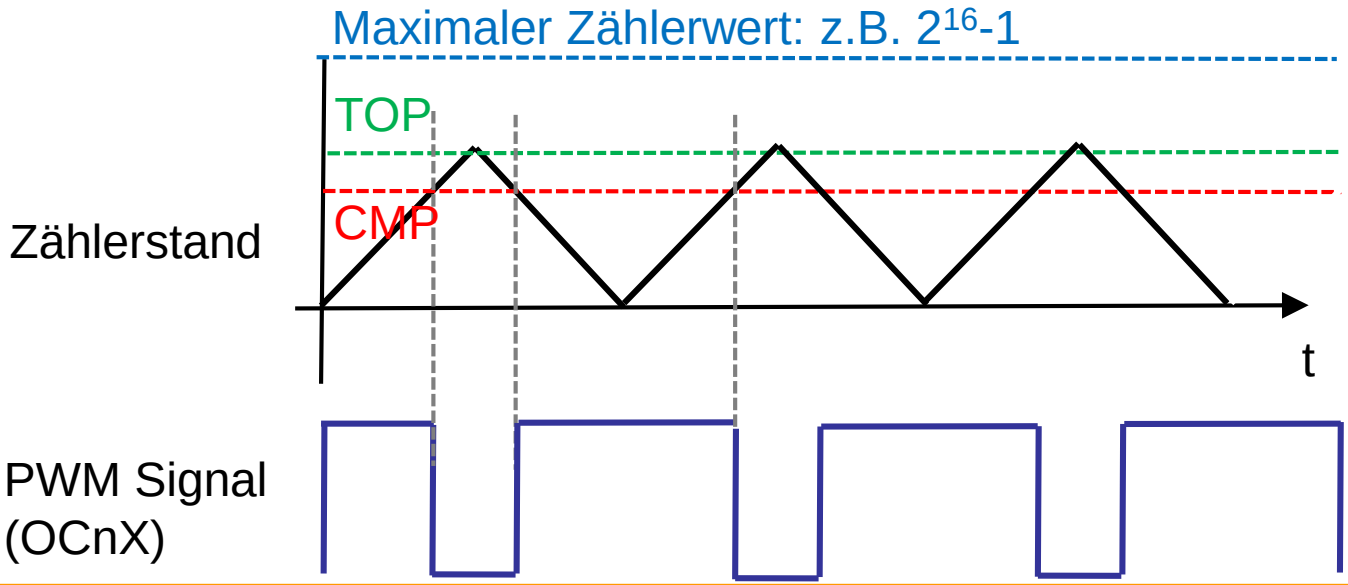
Non-Inverting Mode
`OCnX` wird bei BOTTOM auf HIGH und bei CMP auf LOW gesetzt.

Inverting Mode
`OCnX` wird bei BOTTOM auf LOW und bei CMP auf HIGH gesetzt.

Up-Down-Counter

- ***Funktionsweise***
  - Zähler zählt hoch bis TOP, dann runter bis BOTTOM.
  - Bei Hochzählen: Setze PWM-Ausgang bei CMP auf LOW.
  - Bei Herunterzählen: Setze PWM-Ausgang bei CMP auf HIGH.

- ***Unterschiede*** zu normalem Up-Counter
  - Doppelte Periodendauer, geringere Auflösung.
  - Symmetrisch: BOTTOM und TOP Zählerstand liegen zeitlich immer genau in der Mitte des LOW und HIGH Pegels → http://aquaticus.info/pwm-modes



Programmierung von PWM Signalen

- ***Arduino Library***
  - `analogWrite(<pin>, <duty cycle>)`
    - Erzeugt beliebiges PWM-Signal.
    - Verwendet PWM-Hardwareunterstützung des ATmega2560 Mikrocontollers. https://www.arduino.cc/en/Reference/AnalogWrite
  - *Servo Library*
    - Unterstützt Ansteuerung von bis zu 48 Servomotoren, darunter Ansteuerungen von Servomotoren, darunter der SG90.
    - https://www.arduino.cc/en/Reference/Servo

- **AVR-Libc**
  - Konfiguration über (zahlreiche) Register.
  - Äußerst mächtig.
  - 16 verschiedene Betriebsmodi, siehe nächste Folie.

PWM Betriebsmodi beim ATmega2560

Table 17-2. Waveform Generation Mode Bit Description[(1)]

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Quelle [1, S. 145]:

- Die PWM Modi sind farbig hinterlegt.
- Grün: Up-Counter
- Blau: Up-Down Counter.
- Weiteres Unterscheidungs-merkmal ist: Zeitpunkt an dem Änderungen des Duty Cycles angenommen werden.

- Konfiguration über `WGMn0:WGMn3` Bits in den beiden(!) Registern `TCCRnA` und `TCCRnB`

ATmega2560: Konfiguration des PWM Moduls

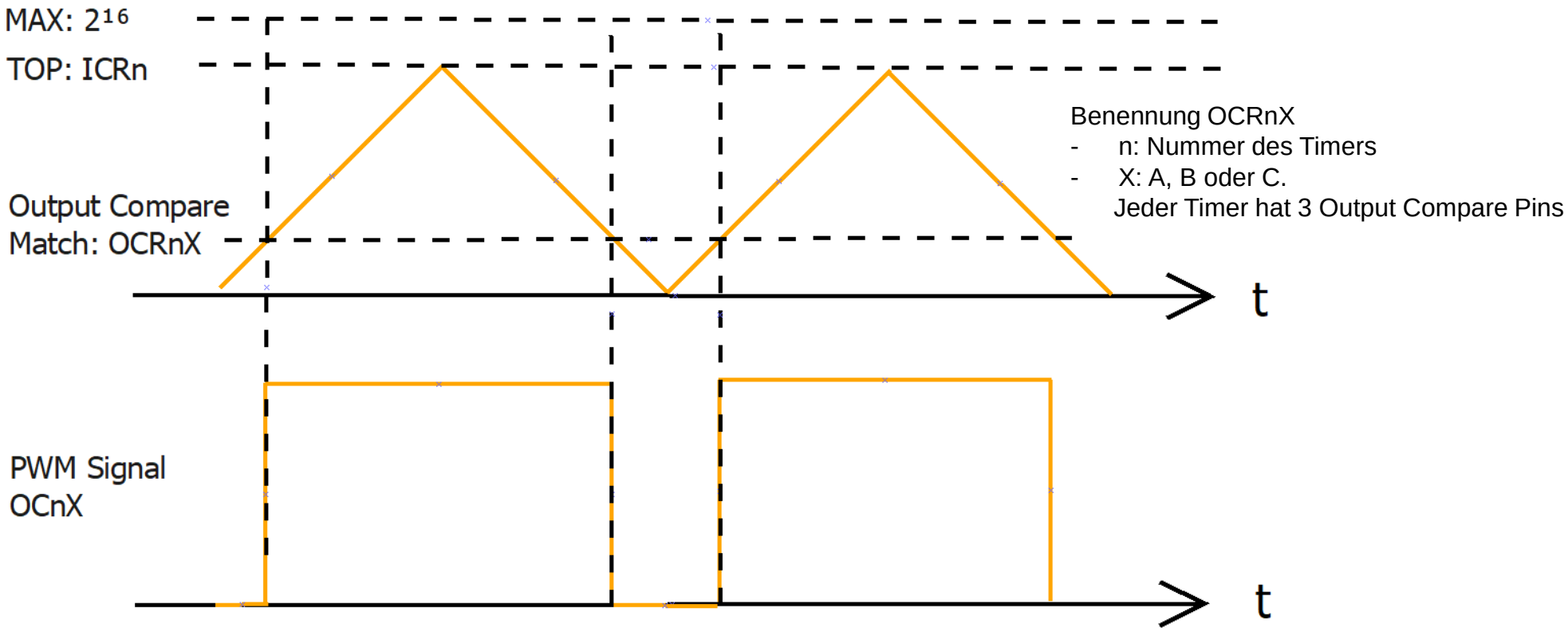
- **Betriebsmodi**
  - Fast PWM, Phase-Correct PWM, Phase and Frequency Correct PWM
  - `WGMn3`, `WGMn2`, `WGMn1`, `WGMn0` (Datenblatt Seite 145)

- **Output Compare Pins OCnX**
  - 3 `OCnX` pins per timer: `OCnA`, `OCnB` und `OCnC` (Datenblatt Seite 155)
  - Konfiguration notwendig, ob im *Inverting* oder *Non-Inverting* Mode.
  - Bei Verwendung eines PWM Modes werden `OCnX` Pins je nach gewähltem Modus (Table 17-2) automatisch gesetzt.
  - Wichtig: Output Compare Pins müssen als Ausgang im DDR Register konfiguriert sein.

- **Output Compare Register OCRnX**
  - Vergleichswert muss gesetzt werden!
  - Falls OC**n**X Output Compare Pin verwendet wird, muss auch das ***entsprechende*** Output Compare Register OCR**n**X verwendet werden!

Spielbeispiel / Live Coding

- Anforderung 1: 16-Bit `Timer3` soll als Up-Down Counter arbeiten
- Anforderung 2: Schwellwert OCR3B steuert LED an.
  - Beim Hochzählen: Setze Output Compare Pin `OC3B` auf 1
  - Beim Runterzählen: Setze Output Compare Pin `OC3B` auf 0.



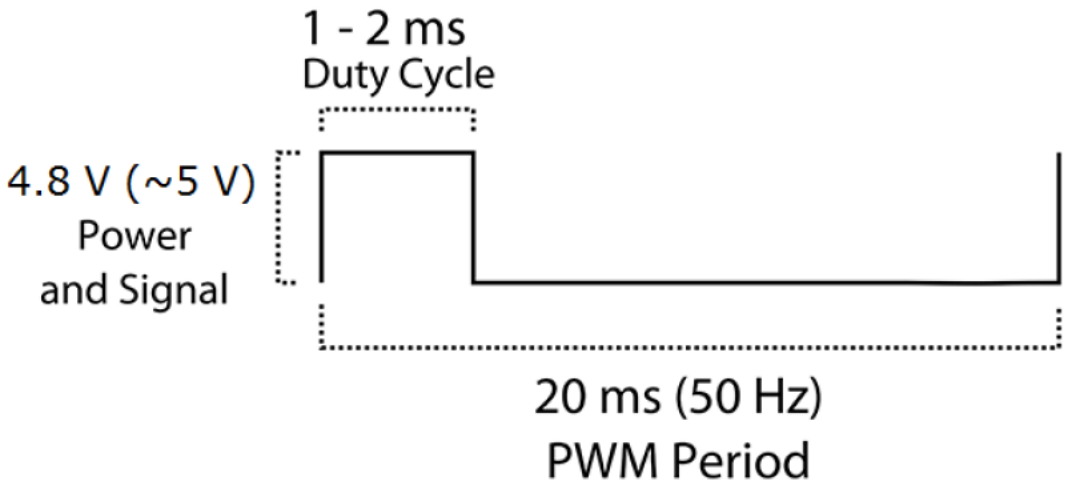
Inhalt

- Grundlagen: Pulsweitenmodulation
- ATmega2560: Konfiguration des PWM Modus
- **Vorbereitung der Übung**

Anforderung

- Der Servomotor SG90 muss laut Handbuch wie folgt angesteuert werden.
  - Periodendauer: 20 ms
  - Duty Cycle: 1 ms (0°), 2 ms (180°)

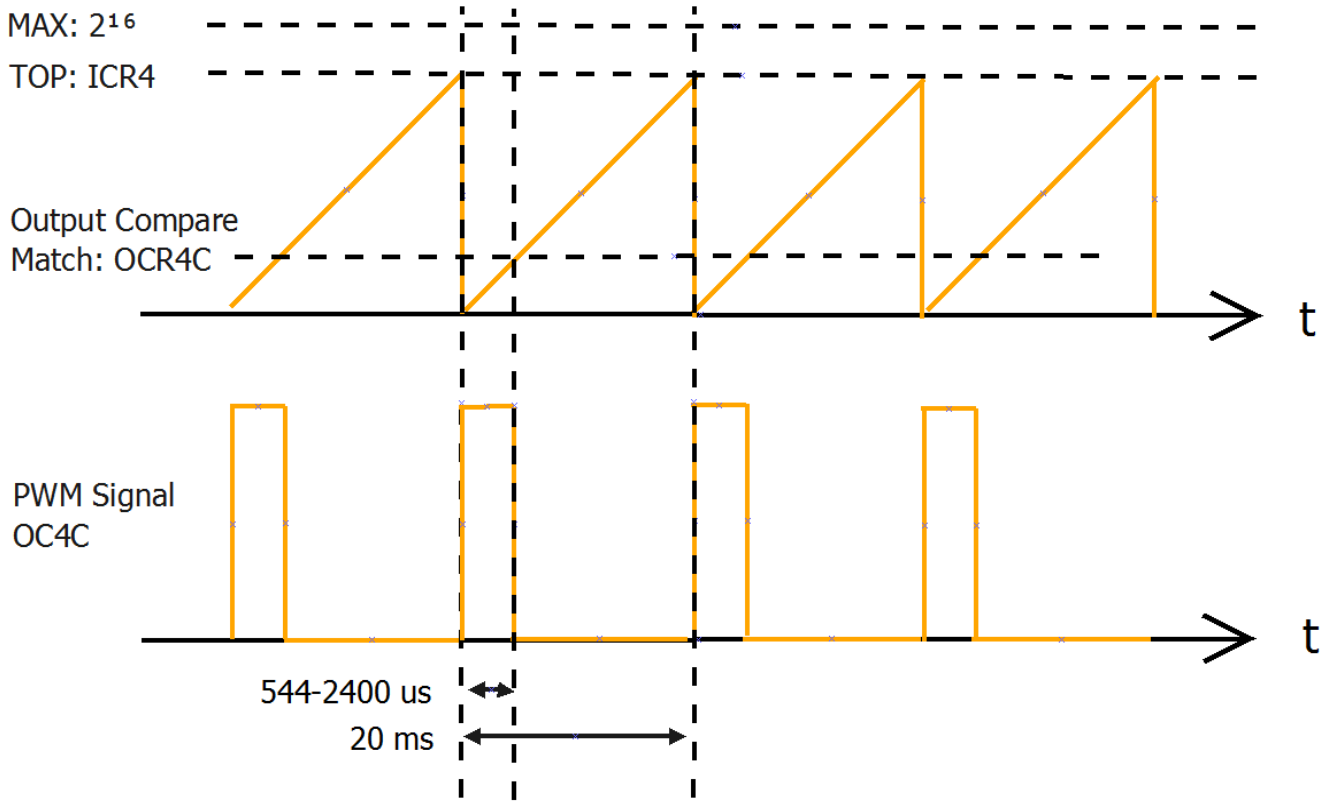
- Tatsächlich funktionieren aber folgende Werte für den Duty Cycle
  - Linker Vollausschlag: 544 µs (0°),
  - Rechter Vollausschlag: 2500 µs (180°)



Aus Datenblatt SG90 [6]

Vorbereitung der Übung

- Welche Register müssen mit welchen Werten gesetzt werden?



Vorüberlegung: Periodendauer

- Auswahl des PWM Modus: **Fast PWM, Inverting Mode**

- Konfigurieren der PWM Periodendauer / TOP Wert
  - Systemtakt 16 MHz: Wie oft würde ein 16-Bit Timer ohne Prescaler im Zeitraum von 20 ms (50 Hz) erhöht werden?
    - 320000
  - Welcher Prescaler würde ausreichen? Wie oft würde der Timer dann innerhalb von 20 ms erhöht werden?
    - 40000
  - Der TOP-Wert soll im ICR4 Register konfiguriert sein. Welche WGMn3:0 Bits müssen gesetzt werden?
    - 1110
  - Es soll der Non-Inverting Mode konfiguriert werden → COMnX1und COMnX2 setzen (siehe Übung)
    - 10

Vorüberlegung: Duty Cycle

- Wie muss OCR4C gesetzt werden für
  - Pulsdauer: 1.0 ms
    - 2000
  - Pulsdauer: 1.5 ms
    - 3000
  - Pulsdauer: 2.0 ms
    - 4000

- TOP-Wert durch ICR4-Register festgelegt.

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] http://www.powershow.com/view/3c9955-YmRhN/SERVO_MOTOR_CONTROL_powerpoint_ppt_presentation (abgerufen am 18.04.2017)

[5] http://www.errmicro.com/blog (abgerufen am 18.04.2017)

[6] Datenblatt SG90. http://www.micropik.com/PDF/SG90Servo.pdf