Gliederung der Vorlesung

1. Einleitung
2. Power Management
3. Parallele I/O
4. Serielle Schnittstelle
   - I^2C
   - USART
5. Timer
6. Interrupt Handling
7. Signalverarbeitung
8. Regelungen
   1. Software Interrupt
9. Leitungscodes
10. Weiterführende Themen

Timer – Timer/Counter – zwei Komponenten

Timer sind Peripheriekomponenten, die

- als Zeitgeber
- als Zähler

verwendet werden können.

- Früher wurden Timer als extra Chips auf dem Board zugefügt.
- Heute sind sie bei praktisch jedem µC als On-Chip-Komponente enthalten, oft sogar mehrfach (Arduino DUE 9-fach).
- Ein Timer enthält neben Registern eine konfigurierbare Logik zur Prüfung auf Bedingungen als Hardware.
- Ist die konfigurierte Bedingung erfüllt, kann die Hardware Reaktionen umsetzen, ohne dass eine Software bzw. Maschinenbefehle in der CPU etwas veranlassen. Die Timer-Hardware gibt dem System somit ein paralleles Verhalten unabhängig vom laufenden Programm.

Timer – zwei grundlegende Funktionen

Als **Timer** kann die Komponente in zeitlich genau definierten Perioden ein Signal erzeugen.

- z.B. wechselt ein Betriebssystem (BS) alle n Millisekunden von einem aktiven Prozess zu einem anderen ausführbaren Prozess. (Timesharing, Multitasking) Das BS benötigt dafür alle n Millisekunden einen Interrupt.
- Ein Timer kann so konfiguriert werden, dass er dies erzeugt. Er arbeitet dann im sogenannten **Compare Mode**.
- Ein Timer kann statt internen Ereignissen wie Interrupts ein externes Signal erzeugen im sogenannten **Wave Mode**.
- Da die Modi **Compare** und **Wave** eine gleichartige Konfiguration der Bedingungen erfordern, werden sie hardwareseitig oft gleich behandelt.

Als **Counter** ist die Komponente in der Lage, die Zeit zwischen zwei Signalen zu bestimmen im sogenannten **Capture Mode.**

- Statt eines festen internen Zeitgebers kann ein Counter auch extern versorgt werden: er zählt dann externe Ereignisse in Hardware, auch wenn die CPU still steht oder etwas anderes macht

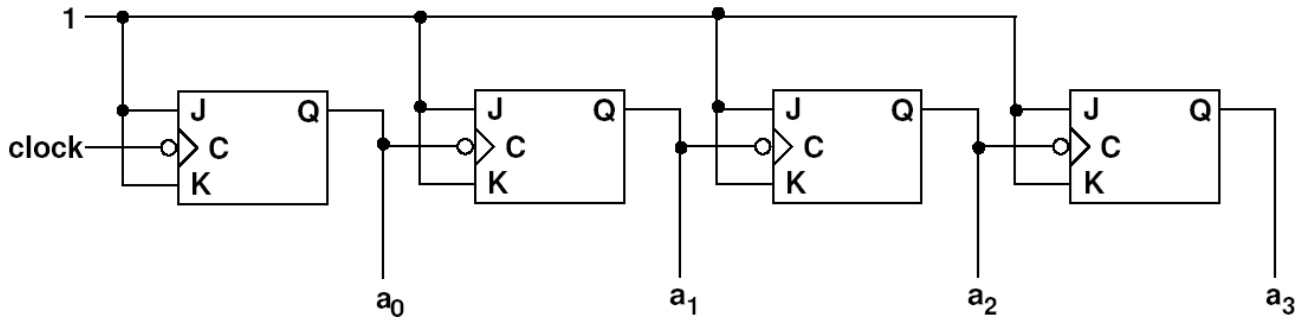
Timer - Anwendungen

- Erzeugung von periodischen Interrupts
  - Betriebssysteme, Uhr
- Messung von Zeiten
  - Pulsbreitenmessung, Periodendauermessung
  - Anwendung in der Messtechnik
- Messung von Frequenzen
- Zählen von Ereignissen
- Erzeugung von periodischen Signalen
  - Ansteuerung von Motoren, DA Wandlung, Signalerzeugung
- Timeout Behandlung, Watchdog-Timer
- Erzeugung von Pulsen

Timer – Komponenten – Counter/Zählregister

- Ein Counter bzw. Zählregister wird kontinuierlich inkrementiert.
- Die Inkrementierung erfolgt durch einen Takt, der üblicherweise vom Systemtakt durch einen konfigurierbaren Teiler abgeleitet ist.
- Das Zählregister hat eine vorgegebene Breite z.B. 8, 16 oder **32 Bits**.
- Ohne weitere Maßnahmen erfolgt nach dem Erreichen des maximalen Zählerwerts gibt es einen Overflow. Der Zähler beginnt wieder bei 0.

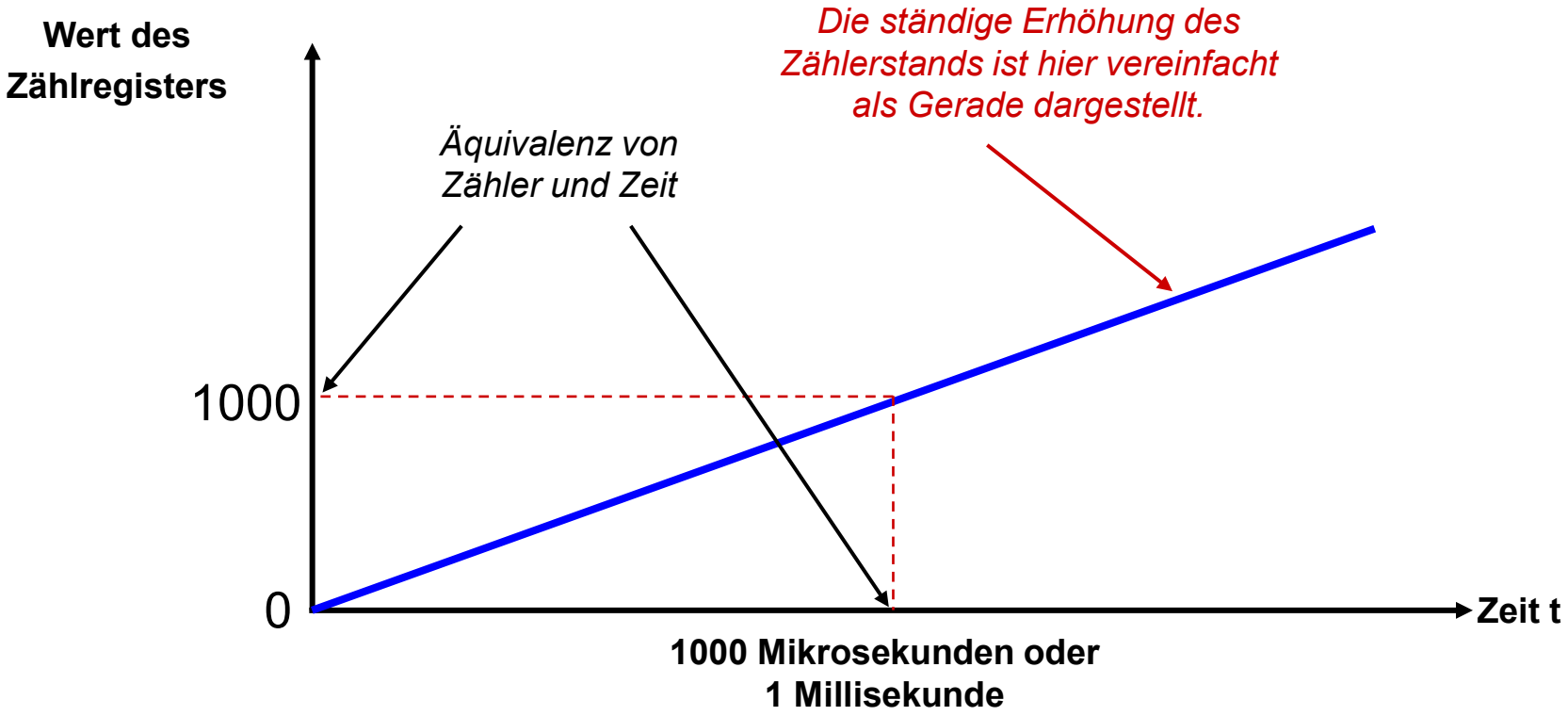
Vergleiche: Zählerschaltungen bestehend aus Flipflops aus den Technischen Grundlagen der Informatik (TGI)



Timer – Komponenten – Zähler und Zeit

- Der Zählerstand des Zählregisters kann als Zeit interpretiert werden.
- Die Periodendauer des gewählten Takts entspricht einem Zählschritt.

- Periodendauer T = 1 / Frequenz F
- Bei einer Frequenz von 1 MHz ist die Periodendauer 1 Mikrosekunde.



Timer – Komponenten – Bedingungen und Aktionen

- Neben dem Zählregister können durch weitere Register (z.B. RA, RB, RC) Bedingungsprüfungen realisiert werden.

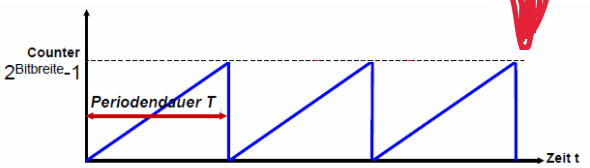
- Mögliche **Bedingungen** können sein:
  - Ein Überlauf des Zählregisters
  - Der Wert des Zählregisters entspricht einem der weiteren Register (z.B. Counter==RC als Compare)
  - Eine externe Leitung geschaltet als Eingang hat einen Pegelwechsel.

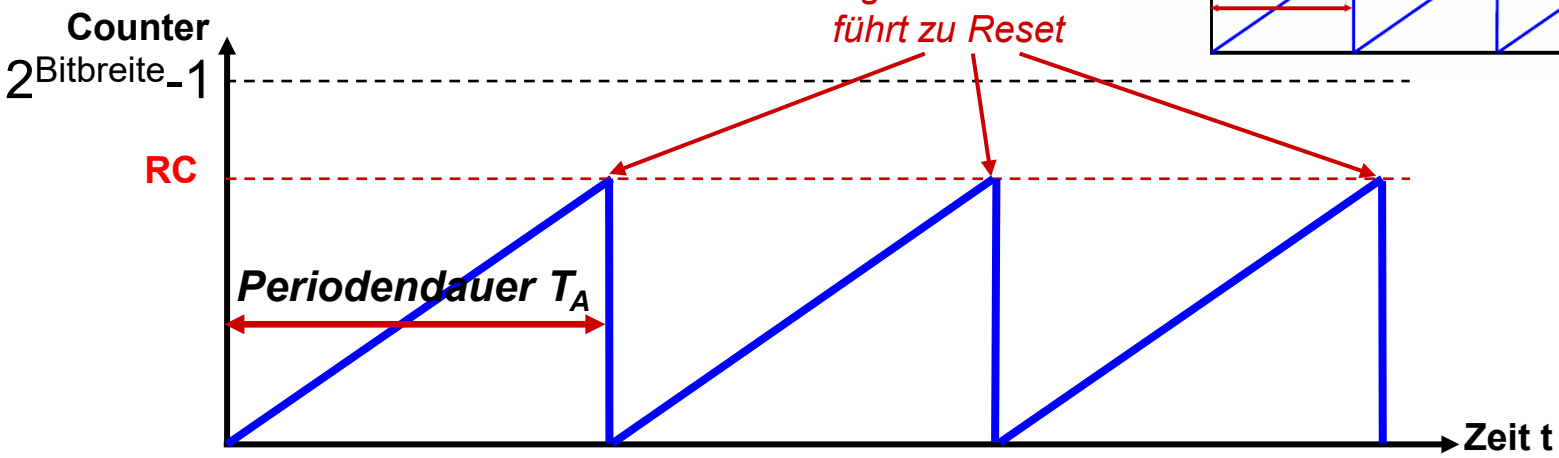
- Mögliche **Aktionen** können sein:
  - Das Zählregister wird zurückgesetzt auf 0
  - Ein weiteres Register wird mit dem Wert des Zählregisters geladen (z.B. RA=Counter als Capture).
  - Es erfolgt die Auslösung eines Interrupts.
  - Auf einer externen Leitung geschaltet als Ausgang erfolgt ein Pegelwechsel.

- Bedingungen mit daraus folgenden Aktionen werden als Bits oder Bitkombinationen in Konfigurationsregistern angegeben.
- Weiterhin können **bedingungslose Aktionen** wie das Starten, Stoppen oder Zurücksetzen des Zählregisters softwareseitig ausgelöst werden.

Timer – Komponenten – Flexibilisierung der Zeit – 1

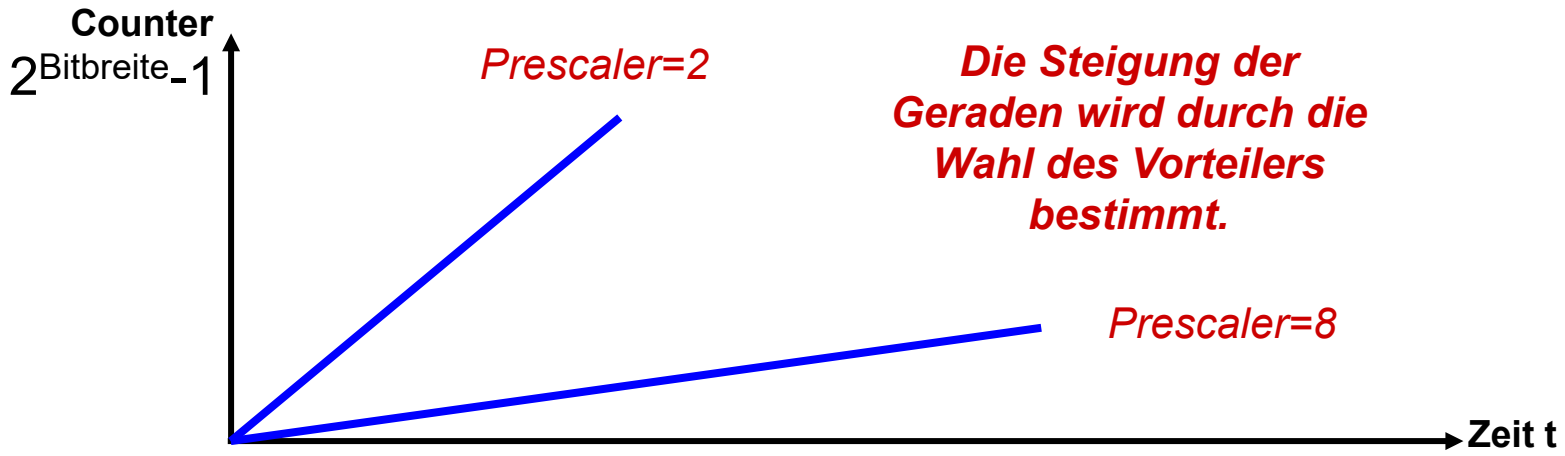
- Der Überlauf des Zählregisters nach dem Wert $2^{\text{Bitbreite}}$-1 führt zur Rücksetzung auf den Wert 0. Dies kann als Bedingung verwendet werden.
- Die resultierende Zeitdauer T ist dann direkt vom Systemtakt abhängig $T = 2^{\text{Bitbreite}}$ / Systemtakt [Hz].
- **Flexibilisierung 1:** Praktisch alle Timer besitzen ein **Reset-Register** (im Weiteren RC genannt). Wenn das Zählregister (Counter) den Wert des Reset-Registers (Counter==RC) erreicht, wird das Zählregister auf 0 zurückgesetzt.
- Durch Konfiguration des Reset-Registers RC kann eine Zeit bis zum Auslösen von Aktionen auf Grund der Resets eingestellt werden.
- Die resultierende **Periodendauer T_A** ist folglich vom Systemtakt und dem Wert von RC abhängig: T_A = RC / Systemtakt [Hz].





Timer – Komponenten – Flexibilisierung der Zeit – 2

- **Flexibilisierung 2:**
  Ein weitere Flexibilisierung kann durch eine Änderung des Eingangstakts erfolgen.
- Mit einem langsameren Takt zählt das Zählregister entsprechend langsamer hoch.
- Der Systemtakt wird hierzu durch einen **Vorteiler (Prescaler)** reduziert.
- Dies lässt sich durch Binärzähler erreichen, deren Ausgang in Abhängigkeit ihrer Zählerbreite schalten.
- Dadurch entstehen typische Werte als Potenzen von 2, wobei ausgewählte Zählerbreiten zu Vorteilern wie 2, 8, 32, 128 und 1024 führen.
- $T_A = 2^{\text{Bitbreite}} * \text{Prescaler} / \text{Systemtakt [Hz]}$



Timer – Komponenten – Flexibilisierung der Zeit – 3

- Beide Flexibilisierungen durch Reset-Register RC und Vorteiler können kombiniert werden.
- Periodendauer T_A = RC * Prescaler / Systemtakt f_{CLK} [Hz]
- Hieraus resultiert die Frequenz $f_A = 1 / T_A$, mit der Aktionen ausgelöst werden können.
- Die Beziehungen werden oft in folgender Gleichung darstellt:

$$f_A = \frac{f_{CLK}}{\text{Prescaler} \cdot RC}$$

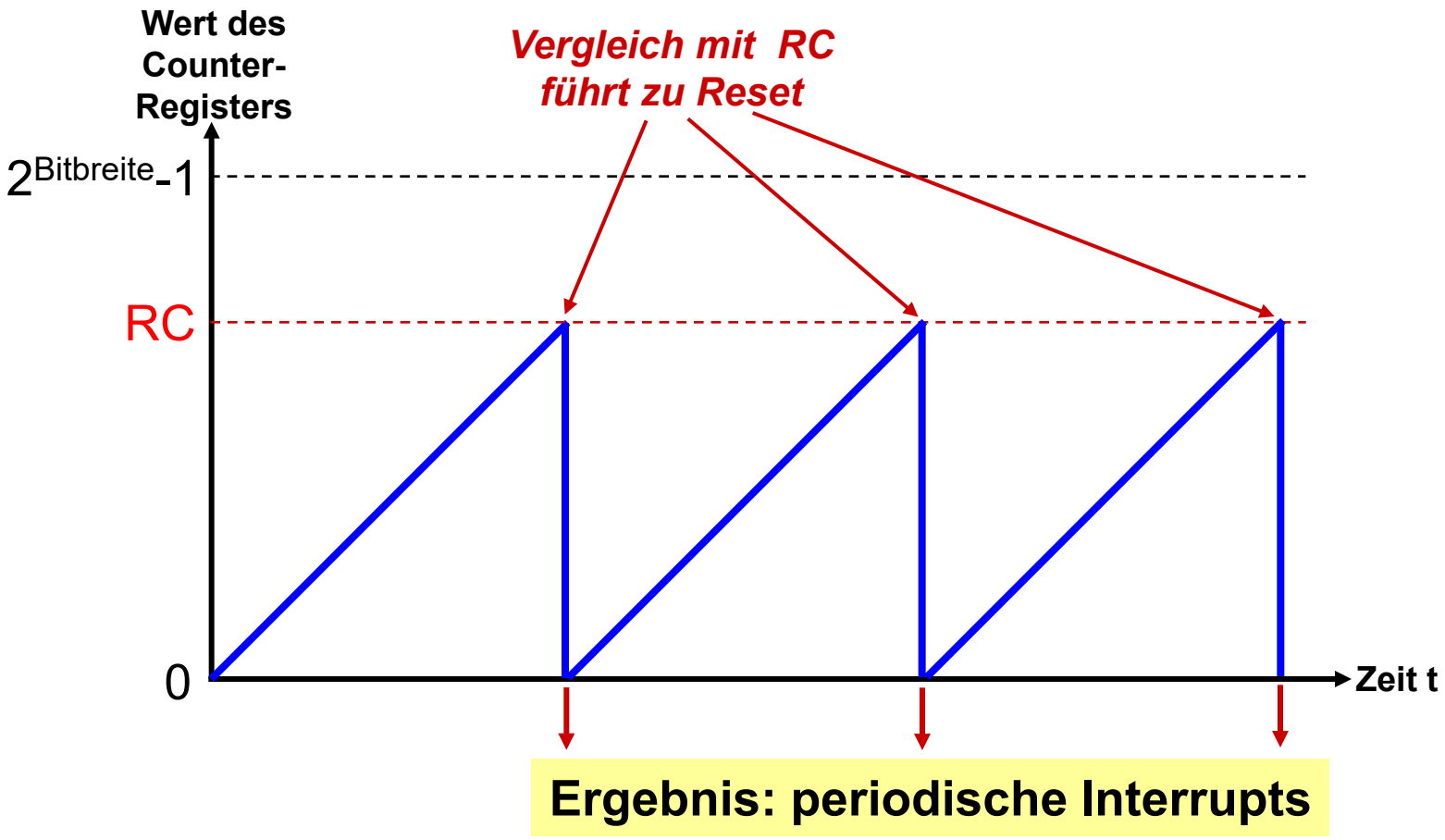
- Die Wahl von RC und Prescaler ist dabei oft aufgabenabhängig z.B. ein möglichst schnelles und genaues Zählen unter Ausnutzung des maximal möglichen Wertes für RC durch Wahl des passenden Vorteilers.

Timer – Periodische Interrupts – Konfiguration

- Für Aufgaben in gleichmäßigen Abständen können Timer periodisch Interrupts erzeugen.
  - Beispiel Betriebssysteme – ein Scheduler soll in periodischen Abständen den laufenden Prozess durch den nächsten Prozess im Round Robin Verfahren austauschen.
- **Modus:** Compare
- **Bedingung:** Es wird das Zählregister mit einem Wert in einem weiteren Register (RC) verglichen, der der Periodendauer entspricht.
  - Counter==RC ?
- **Aktionen:** Ist die Bedingung erfüllt, wird
  - ein Interrupt ausgelöst
  - Das Zählregister auf 0 zurückgesetzt

Timer – periodische Interrupts / Vergleich

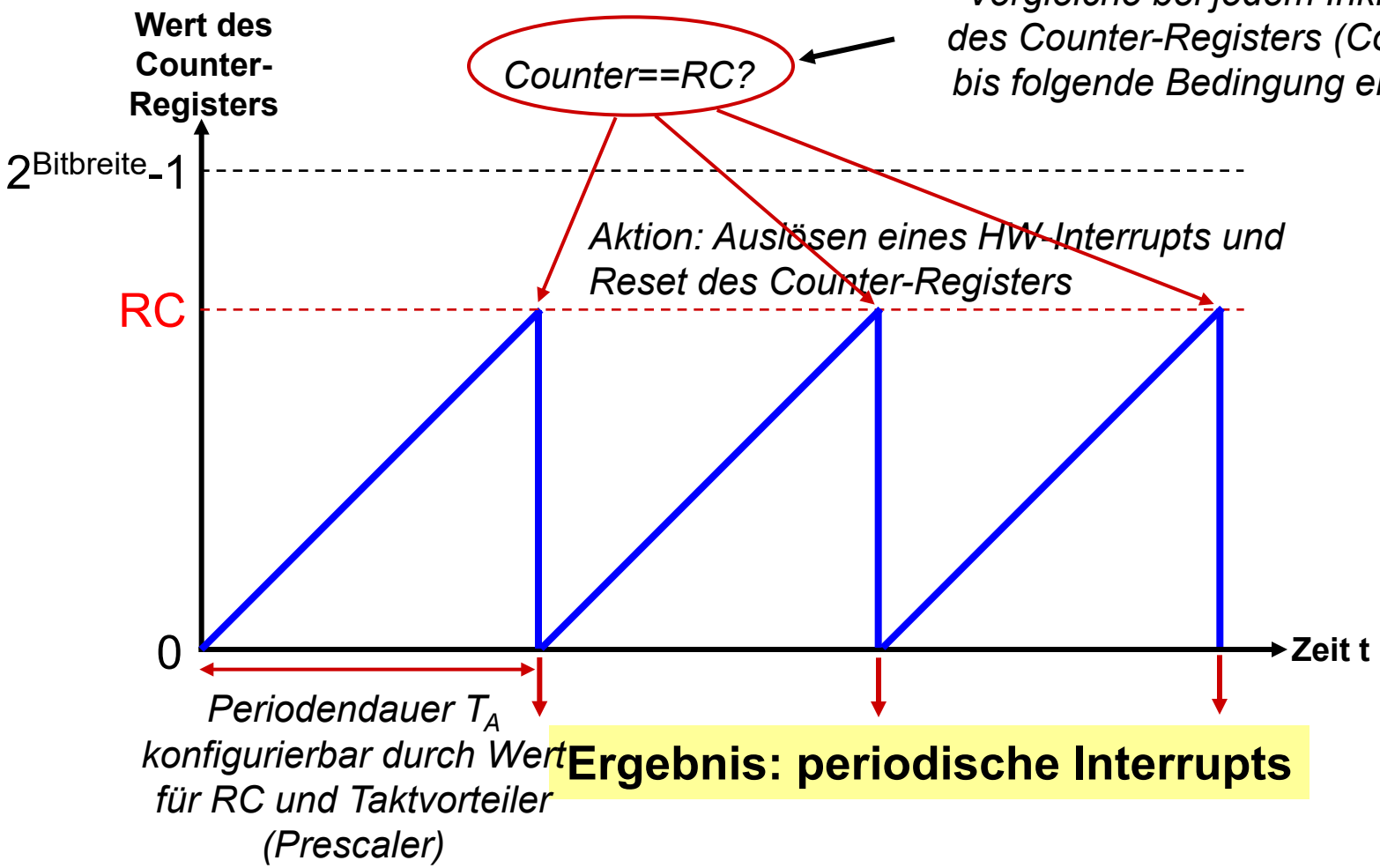
Vergleiche (compare) RC==Counter



Ergebnis: periodische Interrupts

Timer – periodische Interrupts / Reset RC

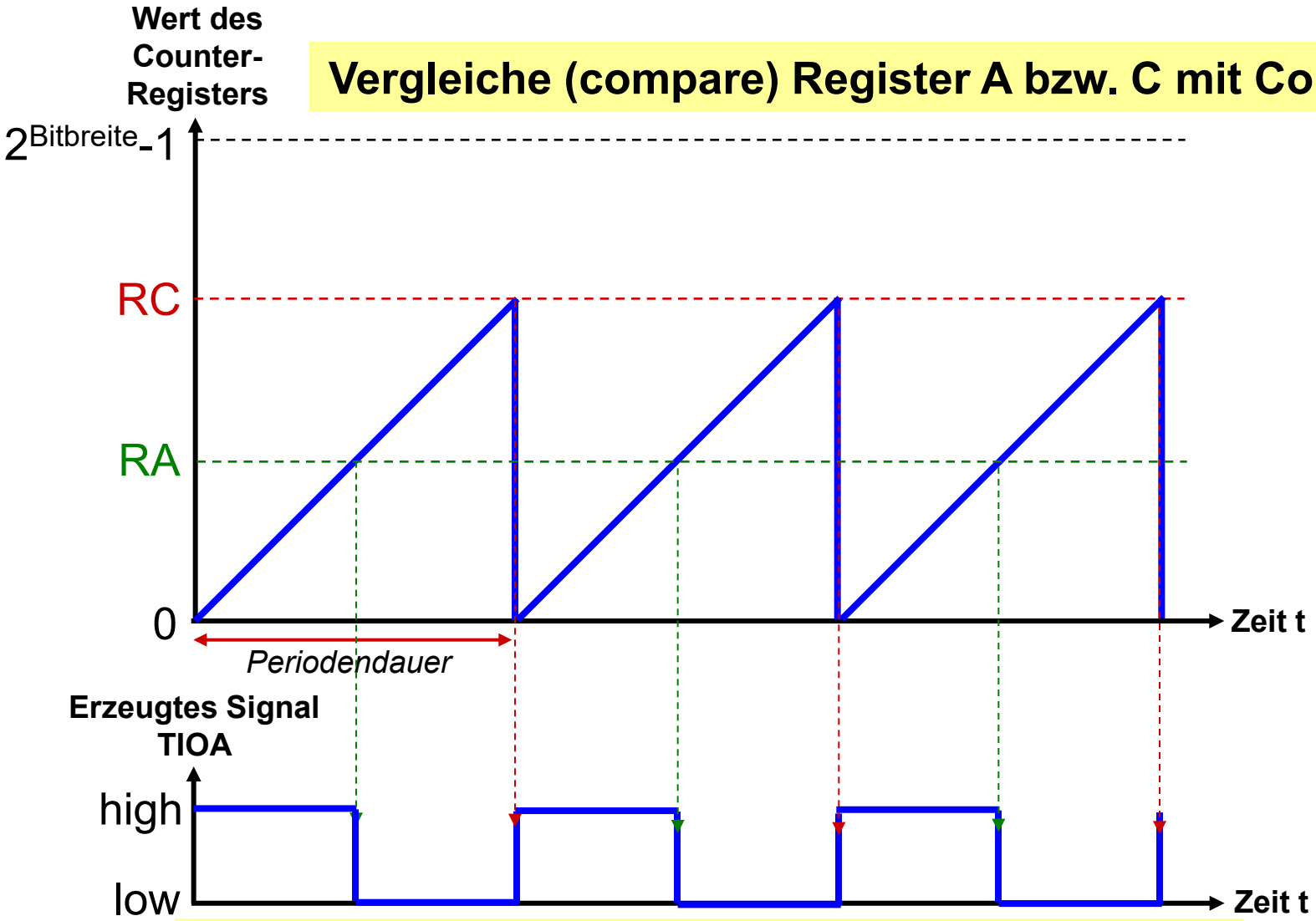
Vergleiche (compare) Register C mit Counter



Timer – Waveform – Konfiguration

- Für die Erzeugung von periodischen Signalen an Ausgängen des Mikrocontrollers können Timer periodisch den Signalausgang verändern.
  - Beispiel Motoransteuerung – ein Schrittmotor benötigt ein Signal mit vorgegebener Frequenz, um mit einer gegebenen Drehgeschwindigkeit zu drehen.
- **Modus:** Wave
- Oft wird zur Signalerzeugung eine zweite Bedingung benötigt, für die der Timer weitere Register wie RA (Beispiel im folgenden) oder RB bietet.
- **Bedingung 1:** Vergleiche das Zählregister mit dem Wert im Register RA
- **Bedingung 2:** Vergleiche das Zählregister mit dem Wert im Register RC
- **Aktionen:** Für die Signalerzeugung wird der Ausgang oft mit den Optionen Setzen/Set auf High, Löschen/Clear auf Low oder Wechseln des Zustands (Toggle) verändert.
- **Beispiel:**
  Bedingung 1 erfüllt, resultiert in Aktion Ausgang=Low
  Bedingung 2 erfüllt, resultiert in Aktionen Ausgang=High; RC=0

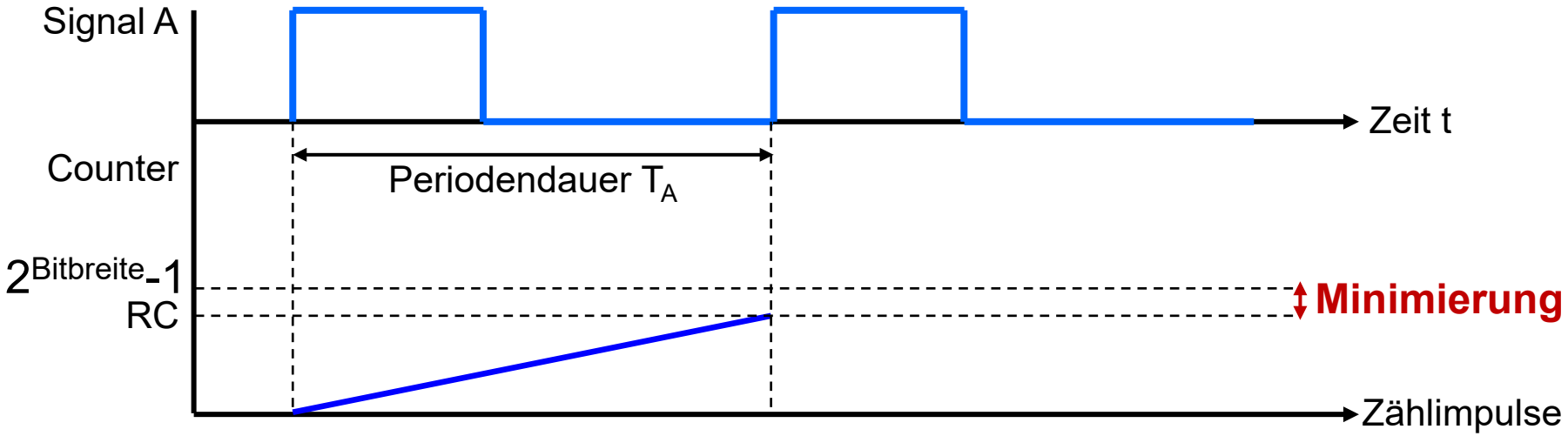
Timer – Waveform – Periodische Signale



Ergebnis: Erzeugung eines periodischen Signals

Timer – Periodische Signale – Wahl des Vorteilers

- Oft will man die Breite des Zählregisters möglichst weitgehend nutzen.
- Dazu berechnet man den **idealen Vorteiler**.



Beispiel: Systemtakt 32 MHz, Breite des Zählers 16 Bit sowie eine Periodendauer T_A 10ms bzw. Frequenz f_A = 100 Hz

$$Prescaler = \frac{32\ MHz}{100Hz \cdot 2^{16}} = \frac{320000}{65536} \approx 4{,}88$$

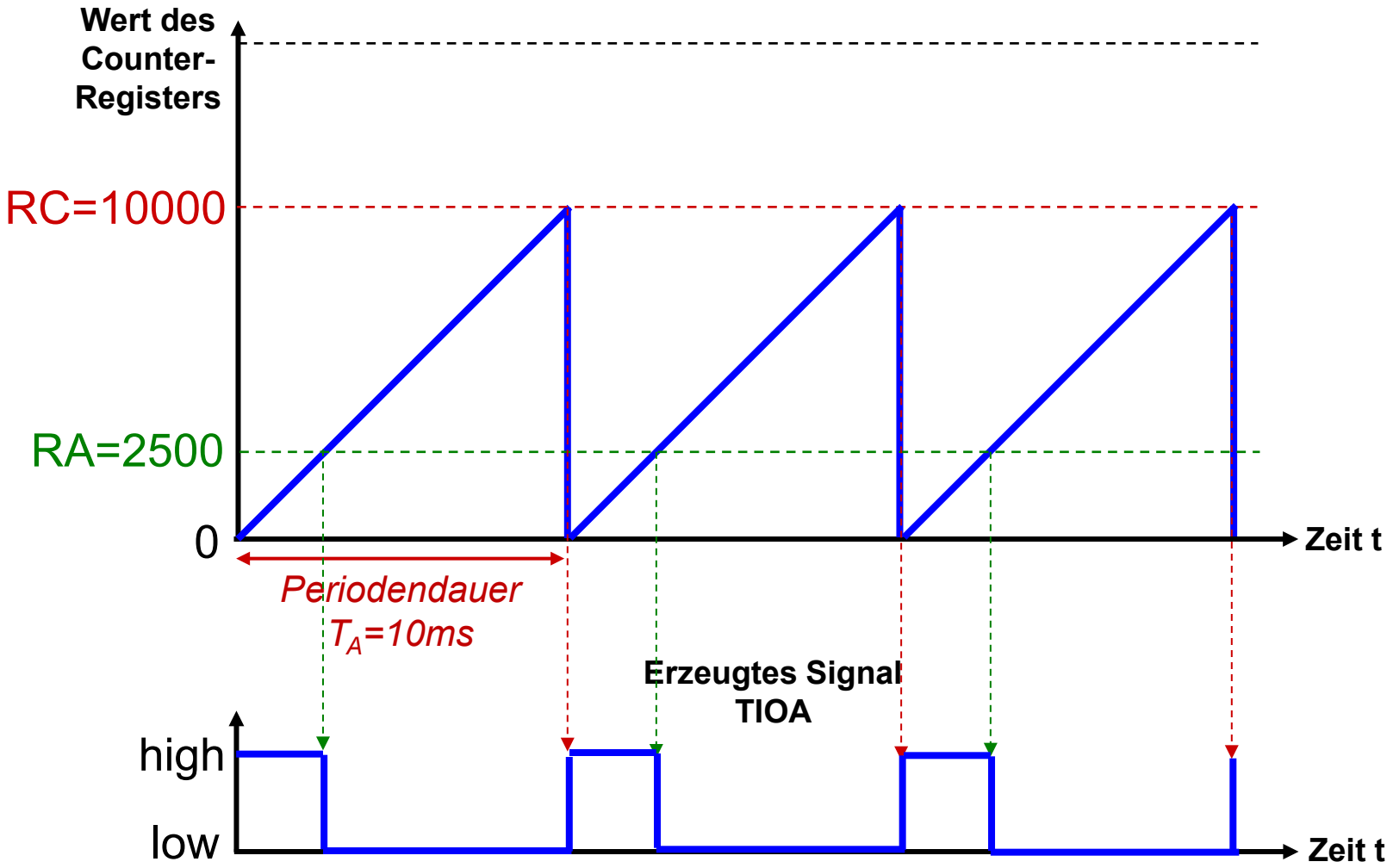
- Von der verfügbaren Frequenzteilern wird der Nächstgrößere gewählt.
- Beispiel 2, 8, 32, 128, 1024: Es wird Prescaler=8 statt 32 gewählt.
- RA und RC ändern sich im Verhältnis des Vorteiler 32 zu 8: Faktor 4

Timer – Waveform – Pulsweitenmodulation PWM

- Für die Erzeugung von periodischen Signalen mit Pulsweitenmodulation lässt sich das Verhältnis von High- und Low-Anteil konfigurieren.
- **Modus:** Wave
- **Bedingung 1:** Vergleiche das Zählregister mit dem Wert im Register RA
- **Bedingung 2:** Vergleiche das Zählregister mit dem Wert im Register RC
- **Aktion bei 1:** Löschen/Clear auf Low
- **Aktionen bei 2:** Setzen/Set auf High; Zurücksetzen des Zählers
- Das Tastverhältnis ist definiert als Zeitanteil des High-Pegels im Verhältnis zur Periodendauer also RA / RC
- **Beispiel:** Systemtakt f_{CLK} = 32 MHz, Periodendauer T=10ms, Prescaler=32 sowie ein Tastverhältnis von 25%
- Dauer eines Zählimpulses $T_{prescaler}$:
  $T_{prescaler}$ = Prescaler / f_{CLK} = 32 / 32 MHz = $1 \cdot 10^{-6}$ s = 1 µs

- Vorgabe Periodendauer T_A: RC = T_A / $T_{prescaler}$ = 10ms / 1µs = 10.000
- Vorgabe High-Anteil: RA = 0,25 * 10000 = 2.500
- Da Zähler bei 0 starten: RC = 9.999, RA = 2.499

Timer – Waveform – Beispiel PWM



Timer – Waveform – Anwendungen PWM

- Dimmen einer LED – Die LED leuchtet nur bei einem Pegel.
  - Der High-Anteil im PWM-Signal gibt die Helligkeit vor.
  - Bei genügend hoher Frequenz ist das Auge zu träge, um ein Flackern zu bemerken.
- Ein Gleichstrommotor kann mittels eines PWM-Signals unterschiedlich kräftig beschleunigen.
  - Der High-Anteil im Verhältnis zur Gesamtzeit gibt das Drehmoment des Motors (Kraft des drehenden Ankers) vor.
  - Die Frequenz sollte so gewählt sein, dass der Motor gleichmäßig läuft. Die Induktivität der Wicklung sowie die Schwungmasse sorgen dann für einen gleichmäßigen Lauf.
- Aus PWM-Signalen lassen sich analoge Signale durch eine Tiefpassfilterung erzeugen, die das Signal glättet.
- Oft werden zwei Ausgänge von nur einem Zähler angesteuert. Das Tastverhältnis kann unabhängig durch zwei Vergleichsregister RA und RB gewählt werden.

Timer – Waveform – Kaskadierung von Zählern

- Die durch die Bitbreite des Zählregisters begrenzte Periodendauer kann durch die Verwendung weiterer Zähler erweitert werden.
- Die Zählregister werden nacheinander kaskadiert geschaltet.
- Im einfachsten Fall erzeugt ein Überlauf des niederwertigsten Zählregisters ein Zählimpuls für den nächsten Zähler in der Zählkaskade.
- Statt eines Überlaufs lassen sich durch RC wählbare Zeitdauern verwenden, um ein Zählimpuls an das nächste Zählregister zu geben.
- Beispiel:
  - Das erste Zählregister hat eine Periodendauer von einer Sekunde. Danach geht ein Zählimpuls an das zweite Zählregister.
  - Das zweite Zählregister zählt bis RC=3600 also eine Stunde lang. Danach geht ein Zählimpuls an das dritte Zählregister.
  - Das dritte Zählregister zählt Stunden.

Timer – Capture – Messungen

- Für die Messung von Zeitdauern können Timer Wechsel des Pegels an Signaleingängen als Bedingungen erfassen.
  - Beispiel Messung der Zeitdauer eines High-Signals
- **Modus:** Capture
- Oft werden zur Signalmessung zwei Bedingungen benötigt, für die der Timer die weiteren Register RA und RB mit Messwerten laden kann.
- **Bedingung 1:** Änderung des Signalpegels z.B. positive Flanke
- **Bedingung 2:** Änderung des Signalpegels z.B. negative Flanke
- **Aktionen:** Es lassen sich die Register RA und RB mit dem Zählerstand zu den Zeitpunkten füllen, zu denen die Bedingungen eintreten.
- **Beispiel:**
  Bedingung 1 erfüllt, resultiert in Aktion RA=Counter
  Bedingung 2 erfüllt, resultiert in Aktionen RB=Counter

Timer – Capture – Konfiguration

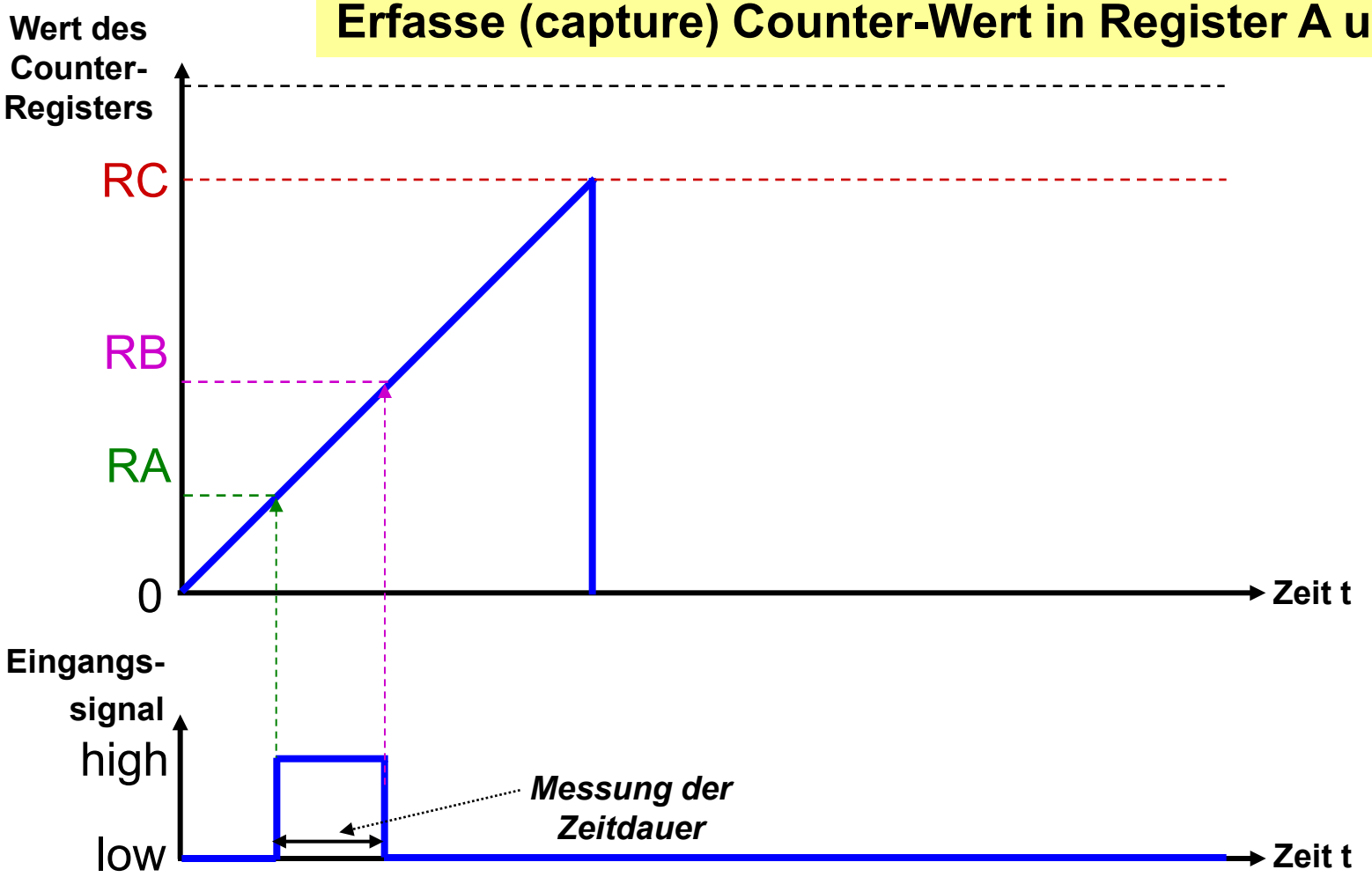
- Die gemessene Zeitdauer resultiert aus der Differenz der Register RA und RB.
- Zeitdauer = (RB-RA) / Systemtakt f_{CLK} [Hz]
- Durch Wahl eines Vorteilers lässt sich die mögliche Messdauer bis zum Zählerüberlauf erweitern.
- Dann gilt:

$$T = \frac{RB - RA}{Prescaler \cdot f_{CLK}}$$

- Es können weitere Aktionen wie das Stoppen des Zählers bei Bedingung 2 konfiguriert werden, so dass die Messwerte RA/RB zur weiteren Bearbeitung erhalten bleiben.
- Das Register RC lässt sich zur Beendigung einer Messung nutzen, wenn die Bedingungen in dem vorgegebenen Zeitraum nicht erfüllt wurden.
- Es kann als Aktion ein Interrupt beim Laden von RB oder Erreichen von RC ausgelöst werden, der die Messwerte RA/RB verarbeitet.

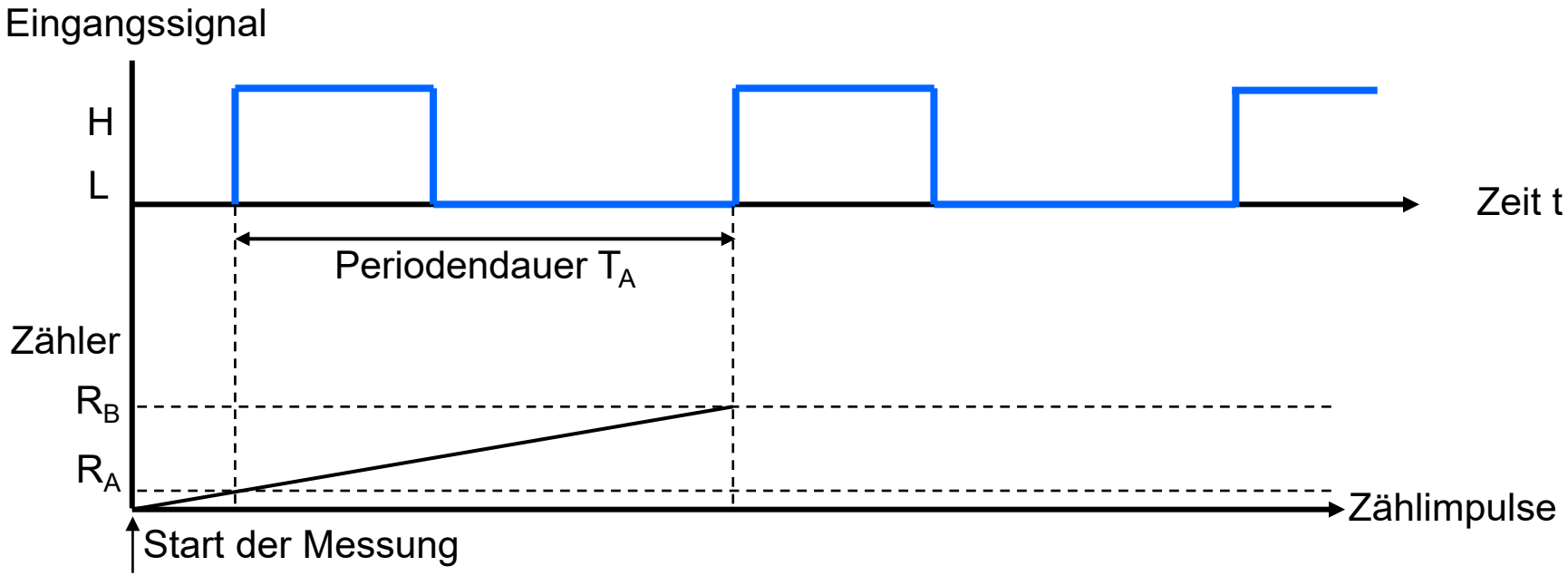
Timer – Capture – Messung Impulsdauer

Erfasse (capture) Counter-Wert in Register A und B



Ergebnis: Messung einer Zeitdauer

Timer – Capture – Messung einer Periodendauer T_A

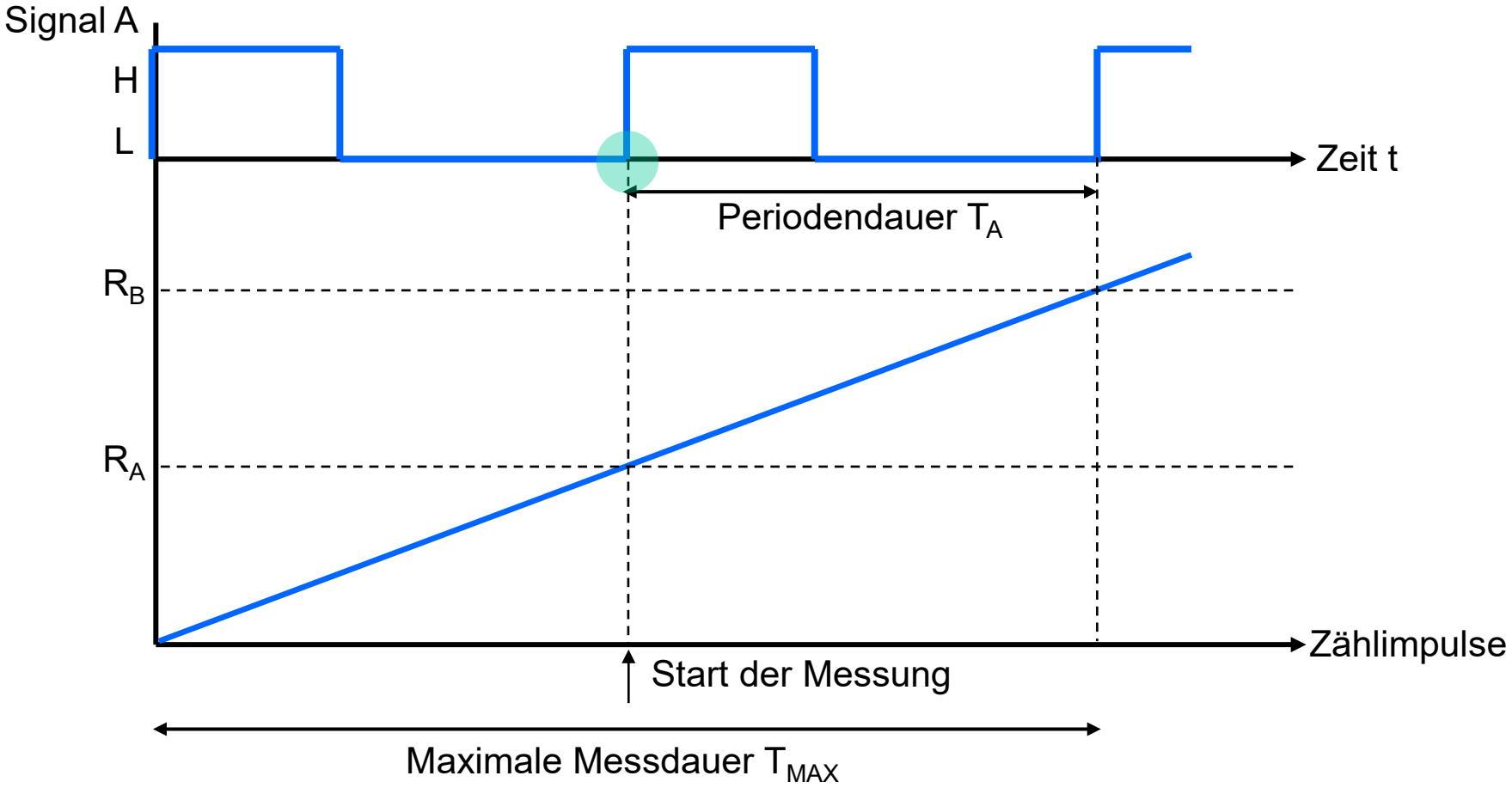


Beispiel einer Messung

- Prescaler=8; Systemtakt=25MHz; Taktdauer Prozessor=40ns
- Messung der Periodendauer T bei den positiven Flanken
  Bedingung 1: RA = 125; Bedingung 2: RB = 3250
- $T_A = (3250\text{-}125) \cdot (8 \cdot 40\ ns) = 3125 \cdot 320\ ns = 10^{-3}s = 1ms$

Wie lang muss die Messdauer mindestens sein, damit eine Periodendauer garantiert erfasst wird?

Timer – Capture – Messdauer



Wenn die Messung im „worst case“ direkt nach einer steigender Flanke beginnt, können fast zwei Periodendauern in ein Messintervall fallen.

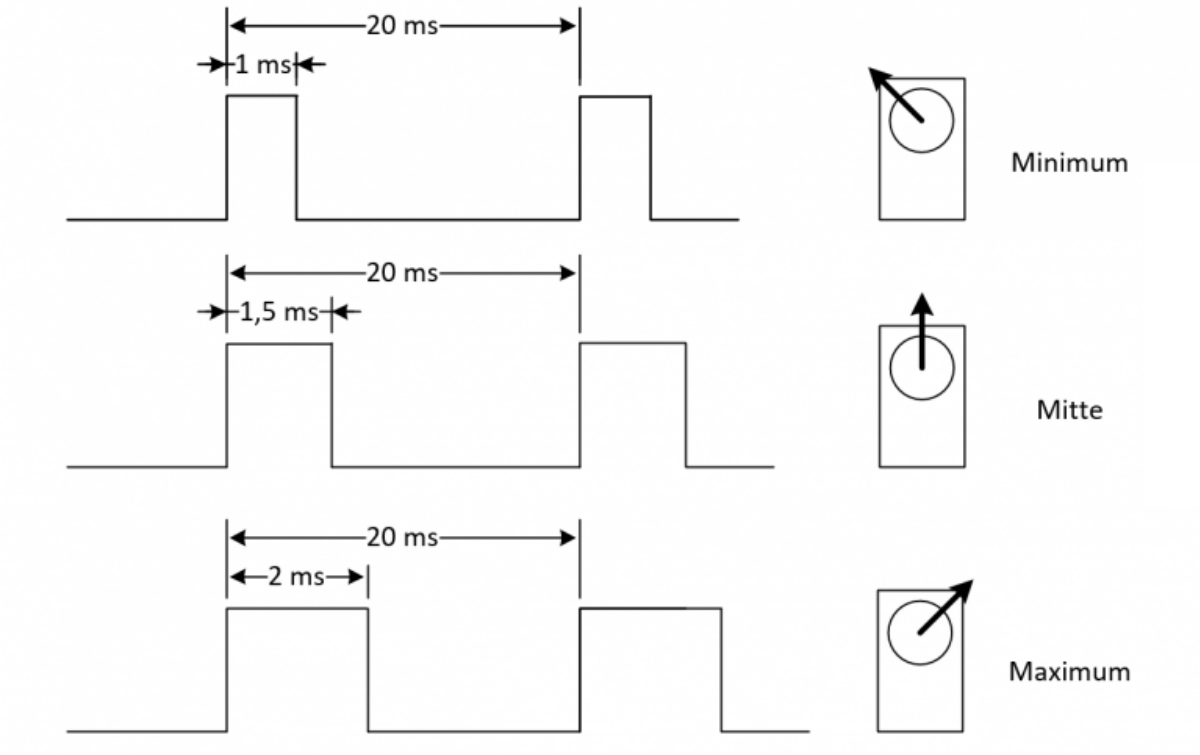
- Im Beispiel sollten mindestens 2ms Messdauer konfiguriert werden.

Timer – Zählen von Ereignissen

- In der Regel kann der Takteingang des Zählregisters auch direkt auf einen Eingang gelegt werden.
- Bei jedem positiven (oder bei jedem negativen) Flankenwechsel des Eingangs wird der Zähler erhöht.
- Zählereignisse (auch oft Impulse oder Pulse (engl.) genannt) können in Hardware sehr viel schneller als in Software erfasst werden.
- In der Regel müssen diese Eingangssignale nicht periodisch sein.
  Es werden also auch sporadisch auftretende Flankenwechsel gezählt.

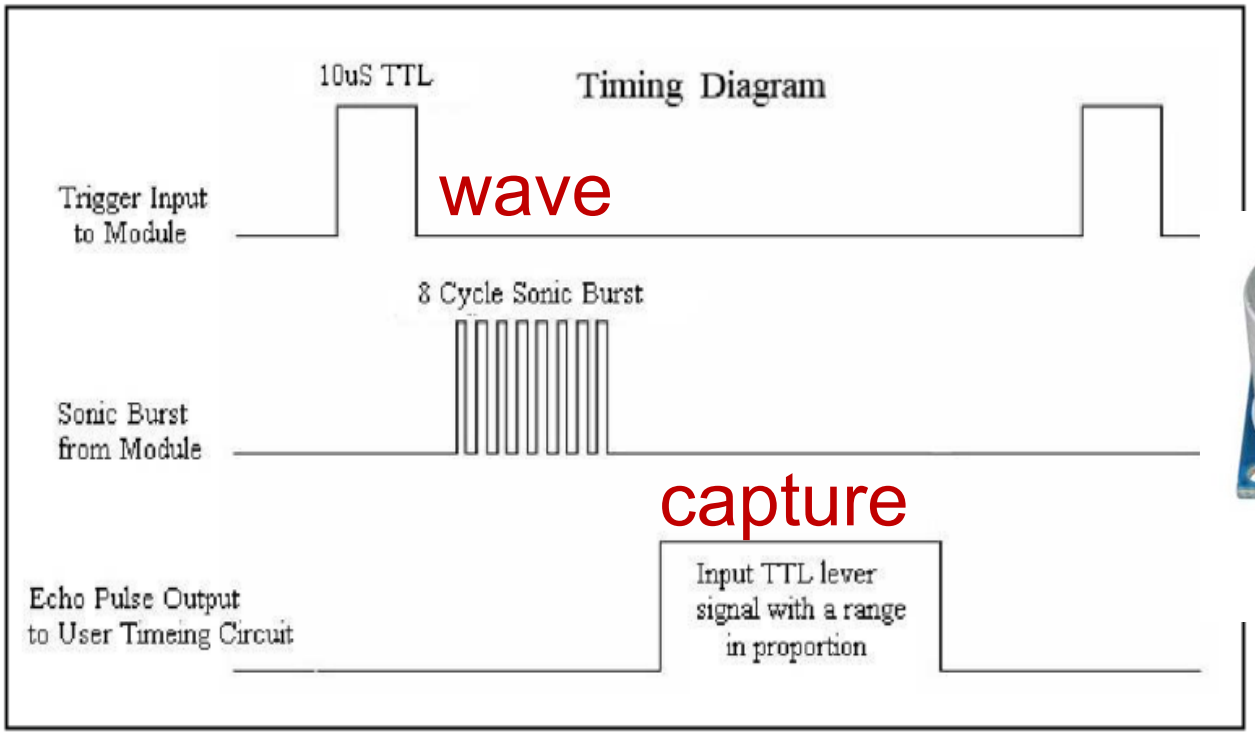
- Anwendungen:
  - Drehzahlmessung
  - Durchflussmessung
  - Frequenzmessung
  - Stückzahlmessung mit Optischer Schranke
  - Oft mit Unterstützung von Quadraturencodern in Hardware
    (Vorwärts-Rückwärtszähler wie bei einer Rollmaus)

Timer – Ansteuerung eines Modellbau-Servos



- Ansteuerung mit PWM
- Die Länge des High-Anteils definiert die Servo-Position
- 20ms Periodendauer entspricht 50 Hz Frequenz.
- 1.5ms: Servo in Mittelstellung

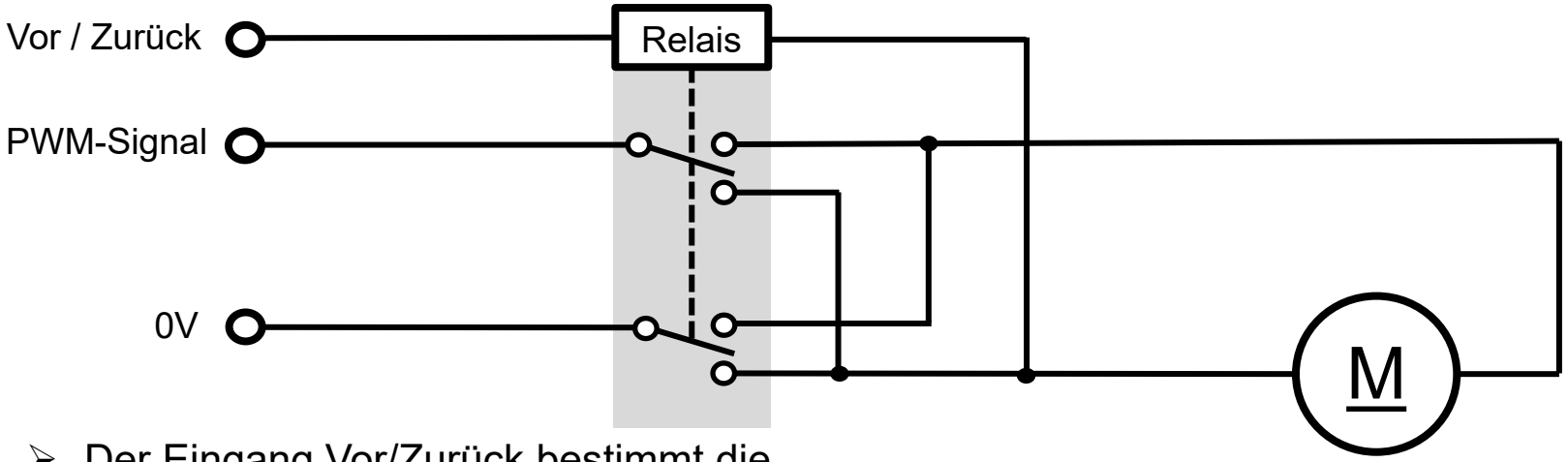
Timer – Ansteuerung eines Ultraschall Sensors



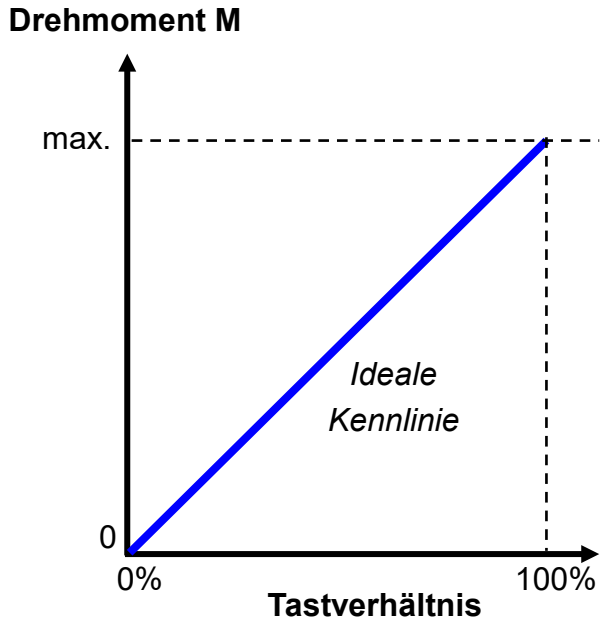
Beispiel: HC-SR04
Messbereich 2-300cm
Genauigkeit ca. 3mm

- Trigger: Ein max. 10 Mikrosekunden langer Impuls
- Messperiode: 20 ms (50 Mal pro Sekunde) oder länger
- Die Dauer des Echosignals ist proportional zur Entfernung (hin und zurück!).
- Maximale Distanz messbar bei 340 m/s Schallgeschwindigkeit (20°C):
  Distanz = (340m/s) / (10 · 10^{-3}s) = 3,4m

Timer – Gleichstrommotor mit Richtungswechsel



- Der Eingang Vor/Zurück bestimmt die Drehrichtung des Motors durch Umpolung mittels eines Relais.
- Das Tastverhältnis bestimmt das Drehmoment des Gleichstrommotors
- Die Frequenz des PWM-Signals sollte nicht zu langsam (stotterrn) oder zu hoch (mechanische Grenzen des Motors) sein.
- Das Drehmoment [Newtonmeter Nm] ist die Kraft auf die Drehbewegung eines Körpers (Anker des Motors).
- Das Drehmoment ist idealerweise linear abhängig vom Tastverhältnis des PWM-Signals.



Timer – Laborhardware

Mikrocontroller	Fähigkeiten für Timer/Counter
AT91	• **Zwei Timer/Counter-Blöcke** mit jeweils drei identischen **16-Bit** Timer-/Counter-Channels • Jeder Channel kann unabhängig programmiert werden für Anwendungen wie Frequenz-/ oder Intervallmessung, Ereigniszählung, Pulsgenerierung auch mit Zeitverzögerung oder Pulsweitenmodulation. • Die Channels in einem Timer-/Counterblock lassen sich auch kaskadiert miteinander zur Erweiterung des Zählbereichs nutzen. • Takteingang wählbar als Systemtakt mit Prescaler 2, 8, 32, 128, 1024 oder externer Takt • Konfigurierbar zur Nutzung mit Interrupt Controller • Ein Watchdog-Timer
SAM3X	• Zwei Timer-/Counterblöcke gleichartig zu der AT91-Funktionalität mit **32-Bit** Timer-/Counter-Channels • Ein dritter Timer-/Counterblock für chipinterne Anwendung • Takteingang wählbar als Systemtakt mit Prescaler 2, 8, 32, 128 oder externem Takt • Ein 32-Bit Real Time Timer, eine Real Time Clock in 12/24h-Format und ein Watchdog-Timer
RP2040	• 8 PWM-Slices mit je 16 Bit Breite und jeweils 2 Ausgängen • 1 x Timer als globaler 1µs-Zeitgeber mit 64 Bit und 4 x 32 Bit Alarm • Programmable IO kann für Timer-Anwendungen genutzt werden • Jeder ARM-Core hat jeweils einen eigenen 24-Bit SysTick-Counter