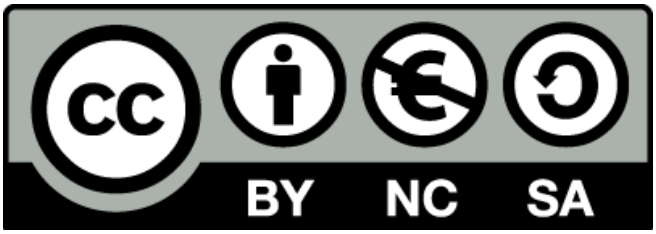
Mikrokontroller & I²C Bus

Attiny 841 – ein IC und
5 verschiedene Module
Teil 7 – Timer

I²C Bus und der Attiny 841

Copyright



Sicherheitshinweise

Lesen Sie diese Gebrauchsanleitung, bevor Sie diesen Bausatz in Betrieb nehmen und bewahren Sie diese an einem für alle Benutzer jederzeit zugänglichen Platz auf. Bei Schäden, die durch Nichtbe-achtung dieser Bedienungsanleitung verursacht werden, erlischt die Gewährleistung / Garantie. Für Folgeschäden übernehmen wir keine Haftung! Bei allen Geräten, die zu ihrem Betrieb eine elektrische Spannung benötigen, müssen die gültigen VDE-Vorschriften beachtet werden. Besonders relevant sind für diesen Bausatz die VDE-Richtlinien VDE 0100, VDE 0550/0551, VDE 0700, VDE 0711 und VDE 0860. Bitte beachten Sie auch nachfolgende Sicherheitshinweise:

- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige Person anschließen.
- In Schulen, Ausbildungseinrichtungen, Hobby- und Selbsthilfewerkstätten ist das Betreiben die-ser Baugruppe durch geschultes Personal verantwortlich zu überwachen.
- In einer Umgebung in der brennbare Gase, Dämpfe oder Stäube vorhanden sind oder vorhanden sein können, darf diese Baugruppe nicht betrieben werden.
- Im Falle eine Reparatur dieser Baugruppe, dürfen nur Original-Ersatzteile verwendet werden! Die Verwendung abweichender Ersatzteile kann zu ernsthaften Sach- und Personenschäden führen. Eine Reparatur des Gerätes darf nur von fachkundigen Personen durchgeführt werden.
- Spannungsführende Teile an dieser Baugruppe dürfen nur dann berührt werden (gilt auch für Werkzeuge, Messinstrumente o.ä.), wenn sichergestellt ist, dass die Baugruppe von der Versor-gungsspannung getrennt wurde und elektrische Ladungen, die in den in der Baugruppe befindlichen Bauteilen gespeichert sind, vorher entladen wurden.
- Sind Messungen bei geöffnetem Gehäuse unumgänglich, muss ein Trenntrafo zur Spannungsver-sorgung verwendet werden
- Spannungsführende Kabel oder Leitungen, mit denen die Baugruppe verbunden ist, müssen immer auf Isolationsfehler oder Bruchstellen kontrolliert werden. Bei einem Fehler muss das Gerät un-verzüglich ausser Betrieb genommen werden, bis die defekte Leitung ausgewechselt worden ist.
- Es ist auf die genaue Einhaltung der genannten Kenndaten der Baugruppe und der in der Baugruppe verwendeten Bauteile zu achten. Gehen diese aus der beiliegenden Beschreibung nicht hervor, so ist eine fachkundige Person hinzuzuziehen

Bestimmungsgemäße Verwendung

- Auf keinen Fall darf 230 V~ Netzspannung angeschlossen werden. Es besteht dann Lebensgefahr!
- Dieser Bausatz ist nur zum Einsatz unter Lern- und Laborbedingungen konzipiert worden. Er ist nicht geeignet, reale Steuerungsaufgaben jeglicher Art zu übernehmen. Ein anderer Einsatz als angegeben ist nicht zulässig!
- Der Bausatz ist nur für den Gebrauch in trockenen und sauberen Räumen bestimmt.
- Wird dieser Bausatz nicht bestimmungsgemäß eingesetzt kann er beschädigt werden, was mit Ge-fahren, wie z.B. Kurzschluss, Brand, elektrischer Schlag etc. verbunden ist. Der Bausatz darf nicht geändert bzw. umgebaut werden!
- Für alle Personen- und Sachschäden, die aus nicht bestimmungsgemäßer Verwendung entstehen, ist nicht der Hersteller, sondern der Betreiber verantwortlich. Bitte beachten Sie, dass Bedien- und /oder Anschlussfehler außerhalb unseres Einflussbereiches liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen.
- Der Autor dieses Tutorials übernimmt keine Haftung für Schäden. Die Nutzung der Hard- und Software erfolgt auf eigenes Risiko.

Attiny 841 – Timer/Counter

In diesem Teil möchte ich euch die Timer des Attiny 841 näher vorstellen.

- Was ist ein Timer, was ist PWM
- Welche Timer besitzt der Attiny 841
- Die Timer Modi (Normal, CTC, PWM)
- Die Register der Timer

Was ist ein Timer, was ist PWM?

Ein Timer ist im Grunde nichts anderes als ein bestimmtes Register im Mikrocontroller, das hardwaregesteuert fortlaufend um 1 erhöht (oder verringert) wird (statt ***um 1 erhöhen*** sagt man auch **inkrementieren**, und das Gegenstück, **dekrementieren**, bedeutet ***um 1 verringern***). Anstatt also Befehle im Programm vorzusehen, die regelmäßig ausgeführt werden und ein Register inkrementieren, erledigt dies der Mikrocontroller ganz von alleine. Dazu ist es möglich, den Timer mit dem Systemtakt zu verbinden und so die Genauigkeit des Quarzes auszunutzen, um ein Register regelmäßig und vor allen Dingen unabhängig vom restlichen Programmfluss (!) hochzählen zu lassen.
Davon alleine hätte man aber noch keinen großen Gewinn. Nützlich wird das Ganze erst dann, wenn man bei bestimmten Zählerständen eine Aktion ausführen lassen kann. Einer der **'bestimmten Zählerstände'** ist zum Beispiel der **Overflow**.
Das Zählregister eines Timers kann natürlich nicht beliebig lange inkrementiert werden – z. B. ist der höchste Zählerstand, den ein **8**-Bit-Timer erreichen kann, $2^8 - 1 = 255$. Beim nächsten **Inkrementierschritt** tritt ein **Überlauf** (engl. **Overflow**) auf, der den Timerstand wieder zu **0** werden lässt. Und hier liegt der springende Punkt. Wir können uns nämlich an diesen **Overflow** "**anhängen**" und den Controller so konfigurieren, dass beim Auftreten des Timer-**Overflows** ein **Interrupt** ausgelöst wird.
Unter einem **Interrupt** versteht man eine vorübergehende Unterbrechung des laufenden Programms, um einen anderen in der Regel zeitkritischen und meist kurzen Vorgang abzuarbeiten. Das auslösende Ereignis wird **IRQ** (**Interrupt Request**) genannt. Nach dieser Anforderung wird die **ISR** (**Interrupt Service Routine**) ausgeführt. Anschließend wird das unterbrochene Programm dort fortgeführt, wo es unterbrochen wurde.
PWM ist einfach ausgedrückt die Technik, die digitalen Ausgänge periodisch HIGH und LOW (ein-aus) zu schalten, also Rechtecksignale mit einem bestimmten Muster zu erzeugen. Da die Erzeugung von PWM sehr schnell und im Hintergrund erfolgen soll, verwendet man dafür Timer.

Welche Timer besitzt der Attiny 841?

Die Mikrocontroller der AVR (Attiny) - Familie besitzen je nach Typ eine unterschiedliche Anzahl an programmierbaren Timer. Beim Attiny 841 sind es 3 Timer.

- **Timer 0 – 8 Bit (0 - 255 Schritte sind 256)**
- **Timer 1 – 16 Bit (0 – 65536 Schritte sind 65537)**
- **Timer 2 – 16 Bit**

Die Timer werden immer mit Timer x benannt, wobei x für die Timernummer steht (also 0, 1 oder 2). Die Konfigurationsmöglichkeiten sind von Timer zu Timer unterschiedlich.

Betriebsarten

Die Betriebsart, d.h. das Verhalten der **Timer/Counter** und der **Output Compare Pins (Ausgangs Pins)**, wird durch die Kombination einzelner Bits in den Registern bestimmt. Die Namen der

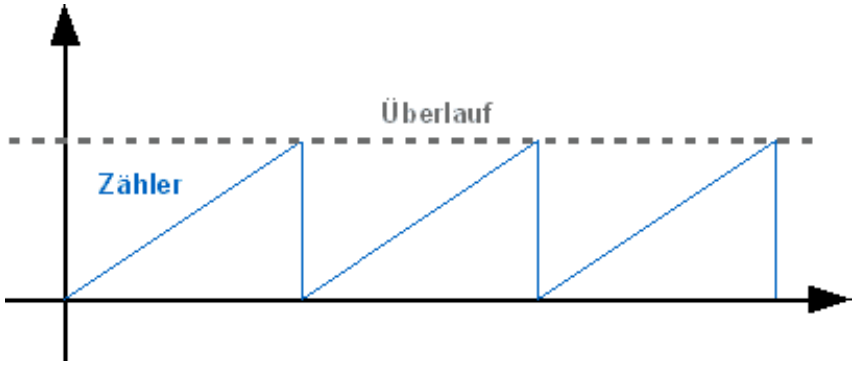
notwendigen Register muss dem Datenblatt des Prozessors entnommen werden.

Betriebsarten des Attiny 841: (je nach Timer)

- **Normaler Modus**
- **Vergleichs Modus (CTC)**
- **PWM**

Normaler Modus

Der einfachste Betriebsmodus ist der normale Modus. Die Zählrichtung des Timers ist immer aufsteigend, bis zum Überlauf - dann fängt der Zähler wieder bei 0 an. Der Überlauf kann einen Interrupt (**Timer-Overflow**) auslösen.

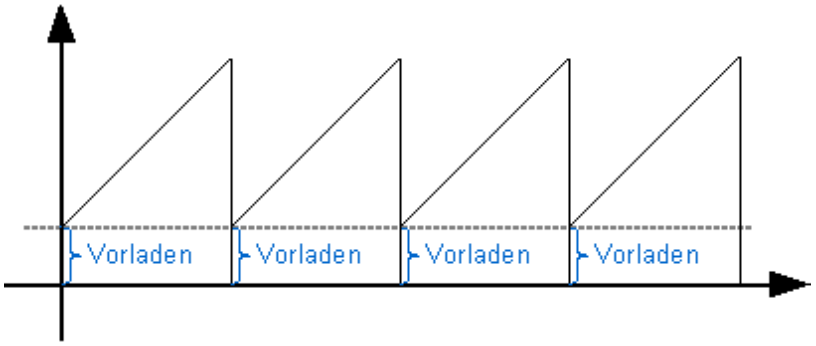


Im einfachsten Fall kann dieser Modus im Diagramm dargestellt werden:

Der Zähler des Timers ist in dem Register **TCNTx** gespeichert, wobei x für eine Zahl steht. Soll z.B. auf den **Timer0** des Controllers zugegriffen werden, so ist an **TCNT** eine **0** anzuhängen, also **TCNT0**. Wie lange es braucht, bis der Zähler einen Overflow auslöst, ist von der Taktfrequenz des Controllers, dem eingestellten Prescaler und von der Timerauflösung abhängig.

CTC Modus (Clear Timer on Compare Match)

Im CTC Modus des Timers ist es möglich, anstelle der durch die Hardware bedingten Obergrenze des Timers, einen anderen Wert zu benutzen, an dem der Timer einen Interrupt auslöst und wieder bei 0 zu zählen anfängt.
Neben dem Aktivieren des CTC Modus genügt es dazu, einfach den gewünschten Endwert in ein spezielles Register, das **OCR0A**, zu laden. Und natürlich hat auch die ISR dann einen anderen Namen.
Dazu wird der Zähler vorgeladen, bevor dieser wieder vom eigentlichen Timer hochgezählt wird.

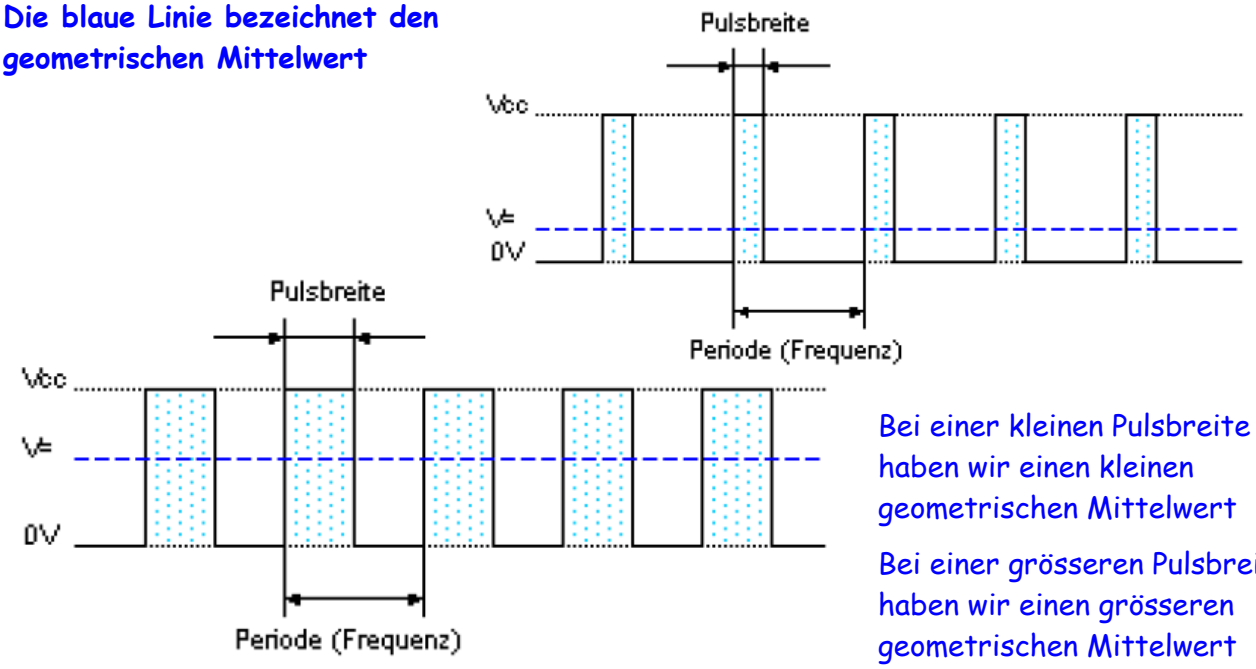


PWM

PWM – **Puls-Weiten-Modulation** (Puls-Breiten-Modulation)
Viele sprechen über das Thema, doch wissen scheinbar nicht alle wie das funktioniert.
Ein Mikrocontroller ist ein rein digitales Bauteil. Er schaltet einen Pin entweder ein- oder aus. Definieren wir einen Pin als Ausgang, dann können wir diesen Ausgang entweder auf **HIGH** setzen worauf am Ausgang die Versorgungsspannung **Vcc** anliegt, oder aber wir setzen den Ausgang auf **LOW** wonach dann **0V** am Ausgang liegt.
Was passiert aber nun, wenn wir periodisch mit einer festen Frequenz zwischen HIGH und LOW umschalten?

Wir erhalten eine Rechteckspannung. Diese Rechteckspannung hat nun einen geometrischen Mittelwert, der je nach Pulsbreite kleiner oder grösser ist.

Die blaue Linie bezeichnet den geometrischen Mittelwert



Bei einer kleinen Pulsbreite haben wir einen kleinen geometrischen Mittelwert

Bei einer grösseren Pulsbreite haben wir einen grösseren geometrischen Mittelwert

Wiederholen wir den Vorgang fortlaufend, haben wir ein PWM an einem Pin anliegen.
Mit den AVR's können wir direkt PWM-Signale erzeugen. Dazu dient der 16-Bit Zähler, welcher im sogenannten PWM-Modus betrieben werden kann.

Dabei gibt es noch verschiedene Arten der PWM:

- **Fast PWM**
- **Phasen-korrekte PWM**
- **Phasen- und frequenzkorrekte PWM**

Auf die genaue Funktion gehe ich im Bereich PWM ein.
Sehen wir uns als nächstes die Grundlegenden Funktionen des Timers an.

Taktquelle

Der Timer/Counter kann von einer internen oder externen Taktquelle getaktet werden und wird durch einen Prescaler (Vorteiler) die gewünschte Frequenz eingestellt.
Da wir in den Beispielen meistens mit einer Quarzfrequenz von 16MHz arbeiten, ist diese Frequenz unsere Bezugsgrösse.

Der Vorteiler (Prescaler) für 8 und 16-Bit Timer

Wenn also der Quarzoszillator mit 16 MHz schwingt, dann würde auch der Timer 16 Millionen mal in der Sekunde erhöht werden. Da der Timer jedes Mal von 0 bis 255 zählt (bei 8 Bit), bevor ein Overflow (Überlauf) auftritt, heißt das auch, dass in einer Sekunde 16000000 / 256 = 62500 Overflows vorkommen. Ganz schön schnell!
Oft ist das nicht sinnvoll. Um diese Raten zu verzögern, gibt es den **Vorteiler**, oder auf Englisch, **Prescaler**. Er kann z.B. auf die Werte 1, 8, 64, 256 oder 1024 eingestellt werden, je nach Timer (Bitte dem Datenblatt entnehmen). Seine Aufgabe ist es, den Systemtakt um den angegebenen Faktor zu teilen. Steht der Vorteiler also auf 1024, so wird nur bei jedem 1024-ten Impuls vom Systemtakt das Register des Timers 0 um 1 erhöht. Entsprechend weniger häufig kommen dann natürlich die Overflows.

Der Systemtakt sei wieder 16 000 000 Hz. Dann wird der Timer in 1 Sekunde 16000000 / 1024 = 15625 mal erhöht. Da der Timer wieder jedes Mal bis 255 zählen muss bis ein Overflow auftritt, bedeutet dies, dass in 1 Sekunde 15625 / 256 = 61,035 Overflows auftreten.

Systemtakt: 16 MHz

Vorteiler	Frequenz(/256)	Overflows	Zeit zwischen 2 Overflows [s]
1	16000000 Hz	62500	0.000016 = 16 µs
8	2000000 Hz	7812,5	0.000128 = 128 µs
64	250000 Hz	976,56	0.001024 ≈ 1,1 ms
256	62500 Hz	244,14	0.004096 ≈ 4,1 ms
1024	15625 Hz	61,035	0.016384 ≈ 16,4 ms

Beispiel für Berechnung des Overflows ohne Vorteiler mit 8-Bit Timer

Diese Tabelle wurde dem Datenblatt für den 8-Bit Timer auf Seite 88 entnommen.

Table 11-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Und die deutsche Übersetzung:

No clock (Timer/Counter stopped) – Stopp, der Timer/Counter wird angehalten
$clk_{I/O}$ / (No prescaling) - CPU Takt
$clk_{I/O}$/ 8 (From prescaler) - CPU Takt / **8**
$clk_{I/O}$/ 64 (From prescaler) - CPU Takt / **64**
$clk_{I/O}$/ 256 (From prescaler) - CPU Takt / **256**
$clk_{I/O}$/ 1024 (From prescaler) - CPU Takt / **1024**
External Clock source on T0 pin. Clock on falling edge - Externer PIN T0, fallende Flanke
External Clock source on T0 pin. Clock on rising edge - Externer PIN T0, steigende Flanke

Es besteht die Möglichkeit die benötigte Frequenz im Internet ausrechnen zu lassen oder es selber zu mache. Dazu hat der der Hersteller in seinem Datenblatt im Kapitel 11.7.2 eine Formel angegeben um den benötigten OCRnAes selber zu berechnen.

$$f_{OCnx} = \frac{f_{clk\ I/O}}{2\ x\ N\ x\ (\ 1 +\ OCRnA\ )}$$

N – Prescaler - 64
f_{OCRnA} - Frequenz bei der der ISR auslösen soll – 1000Hz – 1ms - 1000µs
$f_{clk\ I/O}$ – Taktgeschwindigkeit – 8000 000 Hz
OCRnA - Wert mit der Timer geladen werden muss

Für eine Berechnung müssen wir die Formel umstellen

$$(\ 1 + OCRnA\ ) = \frac{f_{clk\ I/O}}{2\ x\ N\ x\ f_{OCRnA}}$$

Weiter mit

$$OCRnA = \left(\frac{f_{clk\ I/O}}{2\ x\ N\ x\ f_{OCRnA}}\right) - 1 = \left(\frac{8\ 000\ 000\ Hz}{64\ x\ 1000\ \mu s}\right) - 1 = 124$$

In der Formel habe ich den Wert 2 weggelassen. Im Datenblatt des Herstellers wird darauf verwiesen das dieser Wert genommen wird in Abhängigkeit der Wellenform. Bei den 1000Hz handelt es sich um eine reine Frequenz. (1ms entspricht 1000µs)

Die Register des Attiny 841 – 8 Bit Timer

Timer/Counter 0 ist ein universelles 8-Bit Timer/Counter Modul, mit zwei unabhängigen Output Compare Units (Vergleichern) mit den Registern OCR0A und OCR0B.

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Die Auswahl des Prescalers erfolgt im Register TCCR0B durch die Bits CS02, CS01 und CS00. Die notwendigen Einstellungen für CS02, CS01 und CS00 bitte der Tabelle 11-9 entnehmen.

Table 11-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Mit WGM02 kann im Register TCCR0B zusammen mit Register TCCR0A die Betriebsart

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

eingestellt werden.

In der Tabelle 11-8 können die verschiedenen Betriebsarten für den Timer eingestellt werden.
Die Einstellungen erfolgen durch WGM02, WGM01 und WGM00.

Table 11-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Note: 1. MAX = 0xFF
BOTTOM = 0x00

Die wichtigsten Angaben in Deutsch:

Modus	WGM02	WGM01	WGM00	Betriebsart
0	0	0	0	Normal Modus
1	0	0	1	PWM Phase korrekt
2	0	1	0	CTC - Modus
3	0	1	1	Fast PWM
5	1	0	1	PWM Phase korrekt
7	1	1	1	Fast PWM

Den normal Modus und den CTC-Modus habe ich bereits weiter oben erläutert. Auf PWM werde ich in einem anderen Tut näher eingehen.

Der Vorteiler (Prescaler) für 16-Bit Timer 1 und 2

Wenn also der Quarzoszillator mit 16 MHz schwingt, dann würde auch der Timer 16 Millionen mal in der Sekunde erhöht werden. Da der Timer jedes Mal von 0 bis 65537 zählt, bevor ein Overflow auftritt, heißt das auch, dass in einer Sekunde 16000000 / 65537 = 244,14 Overflows vorkommen.
Um diese Raten zu verzögern, kann wieder ein Vorteiler (Prescaler) verwendet werden. Er kann z.B. auf die Werte 1, 8, 64, 256 oder 1024 eingestellt werden. Seine Aufgabe ist es, den Systemtakt um den angegebenen Faktor zu teilen. Steht der Vorteiler also auf 1024, so wird nur bei jedem 1024-ten Impuls vom Systemtakt das Register des Timer 1 oder 2 um 1 erhöht. Entsprechend weniger häufig kommen dann natürlich die Overflows.
Der Systemtakt sei wieder 16000000 Hz. Dann wird der Timer in 1 Sekunde 16000000 / 1024 = 15625 mal erhöht. Da der Timer wieder jedes Mal bis 65537 zählen muss bis ein Overflow auftritt, bedeutet dies, dass in 1 Sekunde 15625 / 65537 = 0,2384 Overflows auftreten.
Wenn wir die gleiche Rechnung machen wie oben aber mit veränderten Angaben bekommen wir ein anderes Ergebnis. Alle Ergebnisse müssen eine glatte Zahl sein ohne Rest oder Abweichung.

$$OCRnA = \left(\frac{f_{clk\ I/O}}{2\ x\ N\ x\ f_{OCRnA}}\right) - 1 = \left(\frac{16\ 000\ 000\ Hz}{64\ x\ 1000\ \mu s}\right) - 1 = 249$$

Bei einem 16 Bit Timer und einem Prescaler von 64 ergibt sich ein OCRnA von 249. Mit anderen Prescaler (128-124, 32-499, 16-999) sind verschiedene Ergebnisse möglich.
Aus diesem 1ms Takt können weiter verschiedene Takte hergeleitet werden. Ein Beispiel dazu kommt weiter hinten im Tut.

Die Register des Attiny 841 – 16 Bit Timer

Der ATtiny441/841 verfügt über zwei 16-Bit Timer/Zähler; Timer/Counter1 und Timer/Counter2. Die 16-Bit Timer/Counter sind funktionell identisch und haben daher die gleiche Beschreibung. Die meisten verwendeten Registernamen und Bit-Referenzen enthalten ein kleingeschriebenes "n", wobei "n" für einen Timer/Counter Nummer steht. Als Betriebsart stehen wieder der Normale Betrieb, CTC und PWM zur Verfügung.
Auf Grund der unterschiedlichen Betriebsarten und Einstellmöglichkeiten kann ich im Rahmen dieses Tuts nur auf die Grundlagen eingehen. Bitte für genaue Informationen das Datenblatt lesen und im Internet informieren.
Auch die 16 Bit Timer verfügen über je 2x unabhängigen Output Compare Units (Vergleichern) mit den Registern OCRnA und OCRnB.

Auf Grund der unterschiedlichen Betriebsarten und Einstellmöglichkeiten kann ich im Rahmen dieses Tuts nur auf die Grundlagen eingehen. Bitte für genaue Informationen das Datenblatt lesen und im Internet informieren.

TCCRnA - Zeitgeber-/Zählersteuerregister A

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xCA)	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bits 7:6 - COMnA[1:0] : Vergleich der Ausgangsart für Kanal A
Bits 5:4 - COMnB[1:0] : Vergleich der Ausgangsart für Kanal B
Bits 1:0 - WGMn[1:0]: Wellenform-Generierungsmodus

Zusammen mit den WGMn[3:2]-Bits im TCCRnB-Register steuern diese Bits die Auswahl der Betriebsart wie in der Tabelle 12-5 angegeben.

TCCRnB - Zeitschaltuhr/Zählersteuerung Register B

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xC9)	ICNC2	ICES2	–	WGM23	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 - ICNCn: Input Capture Rauschunterdrückung
Bit 6 - ICESn: Input Capture Flankenauswahl
Bits 4:3 - WGMn[3:2]: Wellenform-Generierungsmodus
Bits 2:0 - CSn[2:0]: Clock Select

Table 12-6. Clock Select Bit Description

CSn2	CSn1	CSn0	Description
0	0	0	No clock source (timer/counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge.
1	1	1	External clock source on Tn pin. Clock on rising edge.

Auswahl des Prescaler und der Flanken

TCCRnC - Zeitgeber-/Zählersteuerregister C

Bit	7	6	5	4	3	2	1	0	
0x22 (0x42)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	W	W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xC8)	FOC2A	FOC2B	–	–	–	–	–	–	TCCR2C
Read/Write	W	W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 - FOCnA: Force Output Vergleich für Kanal A
Bit 6 - FOCnB: Force Output Compare für Kanal B

TOCPMSA1 und TOCPMSA0 - Timer/Counter Output Vergleich der Pin Mux-Auswahlregister

Bit	7	6	5	4	3	2	1	0	
(0x68)	TOCC7S1	TOCC7S0	TOCC6S1	TOCC6S0	TOCC5S1	TOCC5S0	TOCC4S1	TOCC4S0	TOCPMSA1
(0x67)	TOCC3S1	TOCC3S0	TOCC2S1	TOCC2S0	TOCC1S1	TOCC1S0	TOCC0S1	TOCC0S0	TOCPMSA0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bits 7:0 - TOCCnS1 und TOCCnS0: Timer/Zähler-Ausgang Kanalauswahl vergleichen

Mit den TOCCnS1 und TOCCnS Bits wählt man aus, welcher Timer/Counter Compare Ausgang zum entsprechenden TOCCn geleitet wird. Die drei Timer/Zähler bieten sechs mögliche Vergleichsausgänge, die auf die Ausgangspins geroutet werden können, wie in derTabelle 12-7 angegeben. Beachten Sie, dass jeder zweite TOCCn-Pin auf den Ausgang Vergleichskanal A und jeder zweite TOCCn-Pin auf den Ausgang Vergleichskanal A geführt werden kann

Table 12-7. Selecting Timer/Counter Compare Output for TOCCn Pins

TOCCn Output	TOCCnS1:0		
	00	01	1X
TOCC0	OC0B	OC1B	OC2B
TOCC1	OC0A	OC1A	OC2A
TOCC2	OC0B	OC1B	OC2B
TOCC3	OC0A	OC1A	OC2A
TOCC4	OC0B	OC1B	OC2B
TOCC5	OC0A	OC1A	OC2A
TOCC6	OC0B	OC1B	OC2B
TOCC7	OC0A	OC1A	OC2A

TOCPMCOE - Timer/Counter Output Compare Pin Mux Kanal Ausgang Freigabe

Bit	7	6	5	4	3	2	1	0	
(0x66)	TOCC7OE	TOCC6OE	TOCC5OE	TOCC4OE	TOCC3OE	TOCC2OE	TOCC1OE	TOCC0OE	TOCPMCOE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bits 7:0 - TOCCnOE: Timer-/Zählerausgang Vergleich Kanalausgang Freigabe

Diese Bits aktivieren den ausgewählten Ausgangsvergleichskanal am entsprechenden TOCCn-Pin, unabhängig davon, ob der Ausgangsvergleichsmodus Vergleichsmodus ausgewählt ist oder nicht.

TCNTnH und TCNTnL - Zeitgeber/Zähler

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	TCNT1[15:8]								TCNT1H
0x2C (0x4C)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xC7)	TCNT2[15:8]								TCNT2H
(0xC6)	TCNT2[7:0]								TCNT2L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Die beiden Timer/Counter-I/O-Stellen (TCNTnH und TCNTnL, kombiniert TCNTn) ermöglichen den direkten Zugriff, sowohl zum Lesen als auch zum Schreiboperationen, in die Timer/ Zählereinheit 16-Bit-Zähler.

OCRnAH und OCRnAL - Ausgang Vergleichsregister n A

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	OCR1A[15:8]								OCR1AH
0x2A (0x4A)	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xC5)	OCR2A[15:8]								OCR2AH
(0xC4)	OCR2A[7:0]								OCR2AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

OCRnBH und OCRnBL - Ausgangsvergleichsregister n B

Bit	7	6	5	4	3	2	1	0	
0x29 (0x49)	OCR1B[15:8]								OCR1BH
0x28 (0x48)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xC3)	OCR2B[15:8]								OCR2BH
(0xC2)	OCR2B[7:0]								OCR2BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Die Output-Compare-Register enthalten einen 16-Bit-Wert, der kontinuierlich mit dem Zählerwert (TCNTn) verglichen wird.

TIMSKn - Timer/Zähler-Unterbrechungsmaskenregister

Bit	7	6	5	4	3	2	1	0	
0x0F (0x2F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
0x11 (0x31)	–	–	ICIE2	–	–	OCIE2B	OCIE2A	TOIE2	TIMSK2
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 2 – OCIEnA/B: Timer/Zähler, Ausgangsvergleich B Match Interrupt Enable

Wenn dieses Bit auf eins geschrieben wird und das I-Flag im Statusregister gesetzt ist

(Interrupts global freigegeben), wird das Timer/Counter Output Compare B Match Interrupt ist aktiviert.

Alle Timer/Counter teilen sich das gleiche Prescaler-Modul, aber jeder Timer/Counter kann unterschiedliche Prescaler-Einstellungen haben.

Auf den letzten Seiten habe ich versucht nur das nötigste der möglichen Einstellungen zu zeigen bzw. zu erklären. Es sind noch weitere Einstellungen der einzelnen Register möglich, auf die ich im Rahmen dieses Tuts nicht eingehe. Weitergehende Informationen bitte dem Datenblatt entnehmen.

Genug mit der trockenen Theorie, kommen wir zu Beispielen für die einzelnen Timer. Alle Beispiele zeigen die Erzeugung eines 1ms Taktes.

Diese Angabe ist unbedingt in deinem Programm notwendig: #include "avr/interrupt.h"

Beispiel für Timer 0 mit 8 MHz und 1 ms:

```
void timer0_init()                  // Timer 0, 8 Bit, 8MHz, 1ms
 {                                  // Timer 0 konfigurieren
   TCCR0A = (1<<WGM01);             // Auswahl CTC Modus
   TCCR0B = (1<<CS01)|(1<<CS00);    // Prescaler auf 64 setzen
   OCR0A=124;                       // Wert für 1ms
   TIMSK0|=(1<<OCIE0A);             // Interrupt erlauben
 }

ISR (TIMER0_COMPA_vect)             // ISR für Timer 0
 {
   // ISR Programm für Timer 0
 }
```

Beispiel für Timer 1 mit 8 MHz und 1 ms:

```
void timer1_init()                  // Timer 1, 16 Bit, 8MHz, 1ms
 {                                  // Timer 1 konfigurieren
   TCCR1A = (1<<WGM01);             // Auswahl CTC Modus
   TCCR1B = (1<<CS01)|(1<<CS00);    // Prescaler auf 64 setzen
   OCR1A=124;                       // Wert für 1ms
   TIMSK1|=(1<<OCIE1A);             // Interrupt erlauben
 }

ISR (TIMER1_COMPA_vect)             // ISR für Timer 1
 {
   // ISR Programm für Timer 1
 }
```

Beispiel für Timer 2 mit 16 MHz und 1 ms:

```
void timer2_init()                  // Timer 2, 16 Bit, 16MHz, 1ms
 {                                  // Timer 2 konfigurieren
   TCCR2A = (1<<WGM01);             // Auswahl CTC Modus
   TCCR2B = (1<<CS01)|(1<<CS00);    // Prescaler auf 64 setzen
   OCR2A=249;                       // Wert für 1ms
```

```
  TIMSK2|=(1<<OCIE2A);              // Interrupt erlauben
 }
ISR (TIMER2_COMPA_vect)             // ISR für Timer 2
 {
  // ISR Programm für Timer 2
 }
```

Mit den gezeigten Beispielen kann sich jeder seine Frequenz ausrechnen. Im Internet benutze ich dazu das Programm „AVR Timer Calculator". Bei der Verwendung von „krummen" Frequenzen ist besondere Vorsicht geboten.
Falls jemand andere Frequenzen braucht oder eine Uhrzeit programmieren möchte kann man es wie in diesem Beispiel machen:

```
ISR (TIMER0_COMPA_vect)
 {
  millisekunden++;
  if(millisekunden == 1000)
   {
    sekunde++;
    millisekunden = 0;
    if(sekunde == 60)
     {
      minute++;
      sekunde = 0;
     }
    if(minute == 60)
     {
      stunde++;
      minute = 0;
     }
    if(stunde == 24)
     {
      stunde = 0;
     }
   }
 }
```

Bitte die korrekte Fuse Einstellung kontrollieren.

Fuse Einstellung ohne Quarz

Ex - 0xFF
Hi - 0XDF
Lo - 0xC2

Einige Teile des Textes wurden zur besseren Übersicht **farblich** gestaltet.
Die Nutzung erfolgt auf eigenes Risiko.
Ich wünsche viel Spaß beim Bauen und programmieren
Achim

myroboter@web.de

Quellenangaben:

https://www.mikrocontroller.net/articles/AVR-Tutorial:_Timer#Der_Vorteiler_.28Prescaler.29