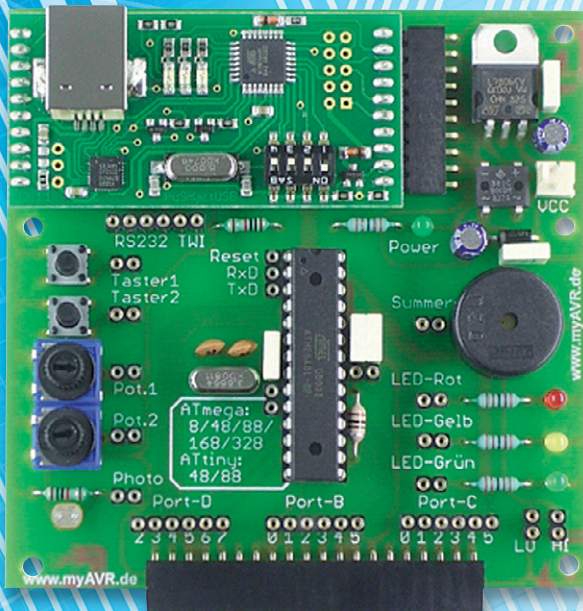


Mikrocontroller-Einstieg

Teil 5: Timer



```

Sub
  BASCOM-Programm
  Einfacher Blinker
  In: -
  Out: LED mit Vorwiderstand an Portb.4

$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 4
$swstack = 4
$sfrsize = 10

Config PORTB.4 = Output

Do
  PORTB.4 = 1
  Waitms 500
  PORTB.4 = 0
  Waitms 500
Loop
End
  
```

Verwendeter Chip
 Verwendete Frequenz
 Rücksprungadressen (je 2), Registersicherungen (32)
 Parameteruebergaben (je 2), LOCALs (je 2)
 Parameter (Daten-Laenge), Rechenbereich Funktionen
 B.4 als Ausgang definieren
 Schleifenbeginn
 B.4 auf 1
 Warteschleife 500 ms
 B.4 auf 0
 Warteschleife 500 ms
 Schleifenende
 Programmende

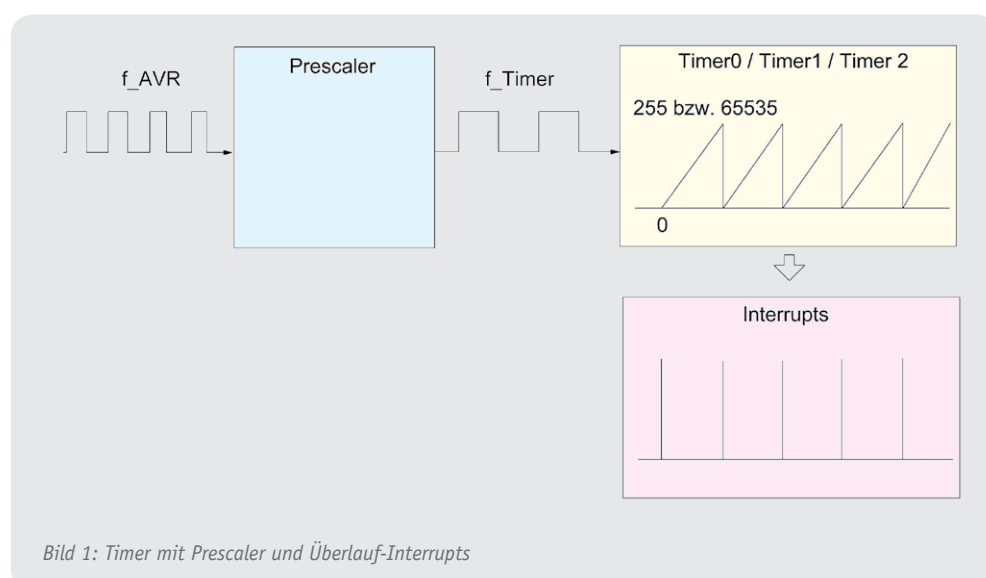
mit BASCOM-AVR

Timer

Timer sind Hardware-Bausteine in einem Mikrocontroller, die unabhängig vom sonstigen Programmablauf von 0 bis zu einem Maximalwert zählen können. In einem AVR-Mikrocontroller gibt es 8-Bit-Timer, die von 0 bis 255 zählen können, und 16-Bit Timer, die von 0 bis 65.535 zählen können. Bei einem ATmega88 gibt es Timer1 als sehr mächtigen 16-Bit-Timer sowie Timer0 und Timer2 als 8-Bit-Timer.

Timer können dafür benutzt werden, regelmäßig Aktionen auszuführen, Zeiten exakt zu messen, unterschiedliche Arten von Signalen zu erzeugen oder externe Impulse zu zählen.

Im Datenblatt des jeweiligen Mikrocontrollers sind die Timer-Register und die Einstellungsmöglichkeiten im Detail beschrieben. BASCOM nimmt dem Programmierer die Arbeit der Timer-Konfiguration weitgehend ab. Anhand einiger Beispiele wird hier die Verwendung von Timern als normale Timer gezeigt, und im ELVjournal 5/2013 werden die CTC-, PWM- und Counter-Modi dargestellt.



Timer im normalen Timer-Modus

Im normalen Timer-Modus zählt ein Timer von 0 bis 255 (beim 8-Bit-Timer0/-Timer2) bzw. bis 65.535 (beim 16-Bit-Timer1). Wenn der Maximalwert erreicht ist, springt der Wert des Timers auf 0, und der Timer fängt von vorne an zu zählen.

Beim Übergang vom Maximalwert zu 0 kann man einen Timer-Interrupt auslösen und infolgedessen eine Interrupt-Service-Routine anspringen lassen. Da es nicht immer passend ist, den Timer im Takt des Systemtaktes zählen zu lassen, kann man durch Vorschalten eines Vorteilers (Prescaler) den Prozessortakt des AVR f_{AVR} in gewissen Stufen verlangsamen. Bei einem ATmega88 kann man den Systemtakt (= Prozessortakt) direkt oder um den Faktor 8, 64, 256, 1024 verlangsamt an den Timer0 bzw. Timer1 geben. Für den Timer2 gibt es zusätzlich noch die Vorteiler-Werte 32 und 128. In [Bild 1](#) ist dargestellt, wie der Systemtakt durch den Prescaler vorgeteilt wird, der Timer in diesem geteilten Takt f_{Timer} immer wieder von 0 bis zum Maximalwert zählt und beim Sprung auf 0 jeweils ein Timer-Interrupt ausgelöst wird.

Timer mit Interrupt

Im folgenden Programm wird – unabhängig vom normalen Programmablauf – durch den Timer ein Interrupt ausgelöst, in dessen Interrupt-Service-Routine eine LED getoggelt, also ein- und ausgeschaltet wird.

```
' BASCOM-Programm
,
' Timer Interrupt
' LED blinkt langsam
,
' In: -
' Out: LED mit Vorwiderstand an B.1

$regfile = "M88def.dat"           'Verwendeter Chip
$crystal = 1000000                'Verwendete Frequenz
$hwstack = 40                    'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                    'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                  'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Timer1 = Timer , Prescale = 64 'Timer1 konfigurieren
On Timer1 Timer1_isr               'Bei Überlauf zu Timer1_isr springen
Enable Timer1                      'Timer1-Interrupt einschalten/ermöglichen
Enable Interrupts                  'Alle Interrupts ermöeglichen

Config Portb.1 = Output            'Ausgang fuer blinkende LED
Led Alias Portb.1                  'Aliasnamen für PORTB.1 vergeben

Do
'hier irgendetwas tun
  Wait 100
Loop
End

Timer1_isr:
Toggle LED                         'Zustand von LED = Portb.1 umschalten
Return
```

Frequenz, Periodendauer; Hertz, ms/ μ s

Eine Frequenz gibt die Anzahl von Schwingungen pro Sekunde an und hat das Formelzeichen f und die Einheit Hz (Hertz). Die Periodendauer T einer Schwingung berechnet sich als Kehrwert der Frequenz mit $T = 1 / f$ und gibt an, wie lange eine Schwingung dauert. Die Einheit ist die Sekunde s . Zwei Hertz bedeutet z. B. zwei Schwingungen pro Sekunde. Die Periodendauer ist dann $T = 1 / 2 \text{ Hz} = 0,5 \text{ s}$. Taktfrequenzen von Mikrocontrollern befinden sich meistens im Bereich von einigen Millionen Schwingungen pro Sekunde. Standardmäßig werden AVR-Mikrocontroller mit 1 MHz interner Taktfrequenz ausgeliefert, führen in jeder Sekunde also eine Million Instruktionen aus. Bei 1 MHz ist die Dauer eines Taktes $T = 1 / 1 \text{ MHz} = 1 / 1.000.000 \text{ Hz} = 1 \mu\text{s}$ (1 Mikrosekunde = 1 Millionstelsekunde). Wenn ein Signal Schwingungen mit einer Periodendauer von 20 ms (Millisekunden = Tausendstelsekunden) hat, dann ist die Frequenz $f = 1 / T = 1 / 20 \text{ ms} = 50 \text{ Hz}$.

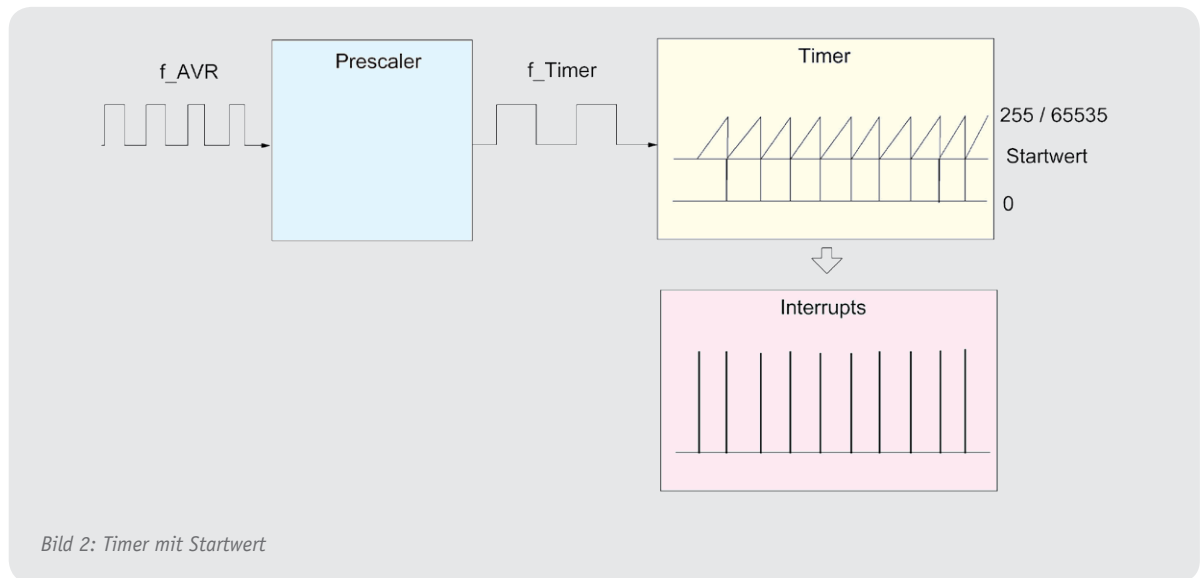


Bild 2: Timer mit Startwert

Erläuterung:

Mit CONFIG Timer1 = Timer, Prescale = 64 wird der Timer1 als Timer mit einem Vorteiler 64 konfiguriert. Der Timer wird also nicht mit dem Systemtakt von 1 MHz getaktet, sondern mit $1 \text{ MHz} / 64 = 15.625 \text{ Hz}$. Der Timer 1 zählt von 0 bis 65.535. Ein Interrupt erfolgt nach 65.536 Schritten mit einer Frequenz von $15.625 \text{ Hz} / 65.536 = 0,24 \text{ Hz}$. Die LED wird also ca. 4-mal pro Sekunde in der Interrupt-Routine umgeschaltet. Mit ON Timer1 wird angegeben, wohin bei einem Timer-Interrupt gesprungen werden soll. Der Name der Interrupt-Service-Routine (ISR) ist frei wählbar. Die ISR muss mit RETURN abgeschlossen werden. Mit ENABLE werden der Timer-Interrupt und auch generell alle Interrupts freigegeben.

Das Toggeln der LED dient hier der Visualisierung. Toggeln bedeutet Umschalten von 1 zu 0 bzw. von 0 zu 1. Man könnte in der ISR auch Flags setzen, Zähler hoch- oder runterzählen oder andere kurze Aktionen ausführen. Längere Aktionen, wie Berechnungen oder Ausgaben per PRINT oder LCD, sollten nicht in der ISR durchgeführt werden. Der Timer zählt unabhängig vom Programmablauf von 0 bis zu seinem Maximalwert und löst beim Übergang vom Maximalwert zu 0 den Interrupt aus. In der Hauptschleife wird hier durch WAIT 100 – stellvertretend für beliebige Befehle – demonstriert, dass der Timer parallel zum eigentlichen „Hauptprogramm“ arbeitet. Eigentlich ist das Programm durch das WAIT 100 in der Hauptschleife „gefangen“. Dadurch dass der Timer unabhängig im Hintergrund weiterläuft und beim Timer-Überlauf die Timer-Interrupt-Routine anspringt, kommt ein Blinken der LED dennoch zustande. Man könnte in der Hauptschleife auch ein in der ISR gesetztes Flag abfragen.

Timer0 und Timer2 können analog verwendet werden. Es wird bei diesen 8-Bit-Timern allerdings nur von 0 bis 255 gezählt.

Timer mit Startwert – Sekundentakt

Durch die Wahl des Timers (8 Bit oder 16 Bit) und des Prescalers kann man einen Timer grob konfigurieren. Möchte man genaue Frequenzen bzw. Zeiten einstellen, dann kann man dies erreichen, indem man den Timer nicht von 0, sondern von einem Startwert zählen lässt, wie in Bild 2 zu sehen. Die Zeit zwischen den Timer-Interrupts wird dadurch verkürzt bzw. die Frequenz vergrößert.

Zum Toggeln einer LED im Sekundenrhythmus stellt man folgende Überlegungen an:

Die Systemfrequenz f_{AVR} ist 1 MHz. Die Periodendauer T_{AVR} beträgt $1 / 1 \text{ MHz} = 1 \mu\text{s}$. Der Prescaler wird auf 64 gestellt, wodurch sich die Periodendauer, mit der der Timer getaktet wird, auf $64 \mu\text{s}$ verlängert. Alle $64 \mu\text{s}$ zählt also der Timer um 1 weiter. Anders gesagt: Ein „Tick“ des Timers dauert $64 \mu\text{s}$. Die gewünschte Zeit ist $1 \text{ s} = 1.000.000 \mu\text{s}$. Es werden also $1.000.000 \mu\text{s} / 64 \mu\text{s} = 15.625$ Ticks benötigt und nicht die vollen 65.536 Ticks, die der Timer insgesamt bis zum Überlauf machen kann. Deshalb wird mit dem Zählen nicht bei 0 angefangen, sondern bei $65.536 - 15.625 = 49.911$. Der Timer zählt von 49.911 bis 65.535 und springt dann auf 0, wobei der Timer-Interrupt ausgelöst wird.

```

' BASCOM-Programm
'
' Timer Interrupt
' LED blinkt im Sekundentakt
'
' In: -
' Out: LED mit Vorwiderstand an B.1

$regfile = "M88def.dat"           'Verwendeter Chip
$crystal = 1000000                'Verwendete Frequenz
$hwstack = 40                    'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                    'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                  'Parameter (Daten-Laenge), Rechenbereich

Config Timer1 = Timer , Prescale = 64 'Timer1 konfigurieren
Const Timer1_startwert = 49911      '..fuer 1 Sekunde

On Timer1 Timer1_isr
Enable Timer1
Enable Interrupts
Timer1 = Timer1_startwert

Config Portb.1 = Output           'Ausgang fuer blinkende LED
Led Alias Portb.1

Do
'hier irgendetwas tun
Wait 100
Loop
End

Timer1_isr:
Timer1 = Timer1_startwert        'Timerstartwert jedes Mal wieder zuwei
Toggle LED                      'Portb umschalten
Return

```



Timerberechnung

Das Thema Timerberechnung sorgt immer wieder für Verunsicherung. Dabei kann man die Werte in wenigen Schritten selbst berechnen, sich eine Tabellenkalkulation (siehe www.bascom-buch.de) aufbauen oder im Internet verfügbare Programme dafür benutzen.

Die wichtigsten Schritte mit Kurzbeispielen sind:

Gegeben: Taktfrequenz $f_{AVR} = 1 \text{ MHz}$ 16-Bit-Timer1 (0 bis 65.535)

Berechnet: Periodendauer $T_{AVR} = 1 / f_{AVR} = 1 / 1 \text{ MHz} = 1 \mu\text{s}$

Dauer eines Timer-Ticks berechnen:

Prescaler 1: $T_{\text{Tick}} = T_{AVR} * \text{Prescaler} = 1 \mu\text{s} * 1 = 1 \mu\text{s}$

Prescaler 8: $T_{\text{Tick}} = T_{AVR} * \text{Prescaler} = 1 \mu\text{s} * 8 = 8 \mu\text{s}$

A) Gesucht: Timerstartwert für 0,1-Sekunden-Interrupts

Also: Zeit gewünscht: $T_{\text{Wunsch}} = 0,1 \text{ s} = 100 \text{ ms} = 100.000 \mu\text{s}$

Benötigte Ticks berechnen: $\text{Ticks} = T_{\text{Wunsch}} / T_{\text{Tick}} = 100.000 \mu\text{s} / 8 \mu\text{s} = 12.500 \text{ Ticks}$

Startwert ausrechnen: $\text{Startwert} = \text{Timerüberlaufwert} - \text{Ticks} = 65.536 - 12.500 = 53.036$

Prescaler ist 8. Daher $T_{\text{Tick}} = 8 \mu\text{s}$.

B) Gesucht: Timerstartwert für 50-Hz-Interrupt mit Timer1 also $f_{\text{Wunsch}} = 50 \text{ Hz}$

Berechnet: $T_{\text{Wunsch}} = 1 / f_{\text{Wunsch}} = 1 / 50 \text{ Hz} = 0,02 \text{ s} = 20 \text{ ms} = 20.000 \mu\text{s}$

Benötigte Ticks berechnen: $\text{Ticks} = T_{\text{Wunsch}} / T_{\text{Tick}} = 20.000 \mu\text{s} / 1 \mu\text{s} = 20.000 \text{ Ticks}$

Startwert ausrechnen: $\text{Startwert} = \text{Timerüberlaufwert} - \text{Ticks} = 65.536 - 20.000 = 45.536$

Prescaler ist hier 1.

Erläuterung:

Timer1 wird als Timer mit Prescaler 64 konfiguriert und sowohl am Anfang des Programms als auch am Anfang der Interrupt-Routine wird der Timer auf den Startwert gesetzt, wodurch er nicht von 0 an, sondern vom Startwert aus zählt. Dadurch erfolgt jede Sekunde genau einmal ein Timer-Interrupt und die Timer-ISR mit dem (frei gewählten) Namen Timer1_isr springt an, wodurch der Ausgang für die LED umgeschaltet wird.

Timer für 440 Hz

Möchte man einen 440-Hz-Ton erzeugen, muss ein Piezo-Buzzer mit 880 Hz getoggelt werden – also alle $1 / 880 \text{ Hz} = 1,136 \text{ ms} = 1136 \mu\text{s}$.

Bei 1 MHz Systemtakt und Prescaler 1 dauert ein Timer-Tick $1 / 1 \text{ MHz} = 1 \mu\text{s}$. Bei Prescaler 8 wird jeder Tick auf $8 \mu\text{s}$ verlängert.

Für die gewünschten $1136 \mu\text{s}$ braucht man $1136 \mu\text{s} / 8 \mu\text{s} = 142$ Ticks.

Bei einem 8-Bit-Timer ist der Startwert deshalb $256 - 142 = 114$.

```
' BASCOM-Programm
'
' 440 Hz
'
' In:  -
' Out: Buzzer ohne Elektronik B.1

$regfile = "M88def.dat"           'Verwendeter Chip
$crystal = 1000000                'Verwendete Frequenz
$hwstack = 40                     'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                     'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                   'Parameter (Daten-Laenge), Rechenbereich Funktionen

Config Timer0 = Timer , Prescale = 8 'Timer 0 konfigurieren ..
Dim Timer0_startwert As Word
Timer0_startwert = 114            ' .. fuer 880 Hz
On Timer0 Timer0_isr
Enable Timer0
Enable Interrupts
Timer0 = Timer0_startwert

Config Portb.1 = Output           'Ausgang fuer Buzzer
Buzzer Alias Portb.1             'Aliasname für Portb.1

Do
'hier irgendetwas tun
Wait 100
Loop
End

Timer0_isr:
Timer0 = Timer0_startwert        'Sofort wieder Startwert für Timer zuweisen
Toggle Buzzer                    'Ausgangspegel umschalten 1 <-> 0
Return
```

*Erläuterung:*

Der 8-Bit-Timer Timer0 wird als Timer mit Prescaler 8 konfiguriert und dadurch, dass jeweils vom Timerstartwert statt von 0 an gezählt wird, springt die Interrupt-Routine alle $1,136 \text{ ms}$ an und der Buzzer wird dadurch mit der gewünschten Frequenz von 880 Hz umgeschaltet, was eine Tonfrequenz von 440 Hz ergibt.

Servo mit Timer

Ein Modellbauservo lässt sich ansteuern, indem in endloser Wiederholung 20 ms lang 0 V und dann 1 ms bis 2 ms lang 5 V ausgegeben werden.

Mit dem folgenden BASCOM-Programm wird ein Servo in einer Schleife jeweils 2 s lang in seine linke Position und dann 4 s lang in seine rechte Position gesteuert.

```

' BASCOM-Programm
'
' Timer Interrupt
' Servoansteuerung
'
' In:  -
' Out: Servo an B.1

$regfile = "M88def.dat"           'Verwendeter Chip
$crystal = 1000000                'Verwendete Frequenz
$hwstack = 40                     'Rücksprungadressen (je 2), Registersicherungen (32)
$swstack = 40                     'Parameteruebergaben (je 2), LOCALs (je 2)
$framesize = 60                   'Parameter (Daten-Laenge), Rechenbereich Funktionen

Const 20ms = 45536                '20 ms
Const Links = 63536               '2 ms
Const Rechts = 64536              '1 ms
Config Timer1 = Timer , Prescale = 1 'Timer 1 als Timer konfigurieren
Dim Servo As Word

On Timer1 Timer1_isr
Enable Timer1
Enable Interrupts
Timer1 = 20ms

Config Portb.1 = Output           'Ausgang fuer Servo. Könnte auch anderer Pin sein.
Servoausgang Alias Portb.1       'Aliasname für Portb.1

Do
Servo = Links                     Servo linker Anschlag.
Wait 2                            2 Sekunden warten
Servo = Rechts                   Servo rechter Anschla
Wait 4                            4 Sekunden warten
Loop
End

Timer1_isr:
If Servoausgang = 1 Then
    Servoausgang = 0              '0 für 20 ms
    Timer1 = 20ms
Else
    Servoausgang = 1              '1 für 1 bzw. 2 ms
    Timer1 = Servo
End If
Return

```



Erläuterung:

Taktfrequenz ist 1 MHz. Bei Prescaler 1 dauert jeder Timer-Tick $(1 / 1 \text{ MHz}) \cdot \text{Prescaler} = 1 \mu\text{s}$.
Für 20 ms benötigt man $20.000 \mu\text{s} / 1 \mu\text{s} = 20.000$ Ticks -> Startwert = $65.536 - 20.000 = 45.536$
Für 1 ms benötigt man $1000 \mu\text{s} / 1 \mu\text{s} = 1000$ Ticks -> Startwert = $65.536 - 1000 = 64.536$
Für 2 ms benötigt man $2000 \mu\text{s} / 1 \mu\text{s} = 2000$ Ticks -> Startwert = $65.536 - 2000 = 63.536$
Wenn vorher der Ausgang logisch 1 war, dann wird in der Interrupt-Routine der Ausgang auf 0 geschaltet und dann der Timerstartwert für 20 ms gesetzt. Dadurch ist der Ausgang 20 ms lang logisch 0.
Wenn der Ausgang logisch 0 war, dann wird der Ausgang auf 1 geschaltet und der Timerstartwert für 1 ms bzw. 2 ms zugewiesen. Dadurch erhält man das typische Servosignal von 20 ms lang 0 V und 1 ms bzw. 2 ms lang 5 V.

Ausblick

Durch die beschriebene Verwendung eines Timers im Timer-Modus ergeben sich vielfältige Einsatzmöglichkeiten: Es können genaue Takte oder Töne erzeugt werden. Es können Servos angesteuert werden. Und durch Auslesen der jeweiligen Timerwerte können auch Zeiten präzise erfasst werden. Wichtiger Vorteil eines Timers ist immer, dass der Timerwert unabhängig vom sonstigen Programm im Hintergrund gezählt wird. Ein mögliches Layout für eine Tabellenkalkulation von Timerstartwerten zeigt [Bild 3](#).

A.) Timerstartwert für Wunsch-Periodendauer:

f_AVR: 1000000 Hz -> T_AVR: 0,001 ms

T_Wunsch: 100 ms

		--- Maximalzeit (ms) ---		Benötigte Ticks	--- Timerstartwert ---	
		--- 8-Bit Timer --- (0..255)	--- 16-Bit Timer --- (0..65535)		--- 8-Bit Timer --- (0..255)	--- 16-Bit Timer --- (0..65535)
Prescale	1 Tick (ms)					
	1	0,001	0,256	100000	--	--
	8	0,008	2,048	12500	--	53.036
Nur Timer2:	32	0,032	8,192	3125	--	
	64	0,064	16,384	1563	--	63.974
Nur Timer2:	128	0,128	32,768	781	--	
	256	0,256	65,536	391	--	65.145
	1024	1,024	262,144	98	158	65.438

Wähle Prescaler für möglichst niedrigen Wert.

B.) Timerstartwert für Wunschfrequenz:

f_AVR: 1000000 Hz -> T_AVR: 0,001 ms

f_Wunsch: 50 Hz -> T_Wunsch: 20 ms

		--- Maximalzeit (ms) ---		Benötigte Ticks	--- Timerstartwert ---	
		--- 8-Bit Timer --- (0..255)	--- 16-Bit Timer --- (0..65535)		--- 8-Bit Timer --- (0..255)	--- 16-Bit Timer --- (0..65535)
Prescale	1 Tick (ms)					
	1	0,001	0,256	20000	--	45.536
	8	0,008	2,048	2500	--	63.036
Nur Timer2:	32	0,032	8,192	625	--	
	64	0,064	16,384	313	--	65.224
Nur Timer2:	128	0,128	32,768	156	100	
	256	0,256	65,536	78	178	65.458
	1024	1,024	262,144	20	236	65.516

Wähle Prescaler für möglichst niedrigen Wert.

Bild 3: Tabellenkalkulation zur Berechnung von Timerstartwerten

Im nächsten ELVjournal werden weitere Timer-Modi (CTC, PWM, Counter) vorgestellt, mit denen sich noch mehr Möglichkeiten eröffnen. **ELV**

**Weitere Infos:**

- Stefan Hoffmann: Einfacher Einstieg in die Elektronik mit AVR-Mikrocontroller und BASCOM. Systematische Einführung und Nachschlagewerk mit vielen Anregungen. ISBN 978-3-8391-8430-1
- www.bascom-buch.de
- www.mcselec.com
- www.atmel.com

Alle Infos zu den Produkten/Bauteilen finden Sie im Web-Shop.

Empfohlene Produkte/Bauteile:

Best.-Nr.	Preis
BASCOM-(Demo-)Lizenz von MCS Electronics	–
Programmer Atmel AVRISP mkII	JX-10 03 55 € 39,95
oder myAVR-Board MK2	JX-10 90 00 € 49,–
Netzteil für myAVR-Board MK2	JX-10 90 01 € 6,95
ATtiny13	JX-10 03 39 € 1,95
ATmega8	JX-05 29 71 € 3,20
ATmega88	JX-10 07 62 € 3,95
100-nF-Kondensator	JX-10 03 17 € 0,08
Batteriehalter für 3x Mignon	JX-08 15 30 € 0,75
Batterieclip für 9-V-Block-Batterie	JX-08 01 28 € 0,30
BASCOM-Buch	JX-10 90 02 € 54,–
Experimentier-Board 1202B	JX-07 72 89 € 12,95
Schalt draht-Sortiment	JX-05 47 68 € 5,95
LED-Set	JX-10 63 56 € 3,95
oder Leuchtdioden	JX-10 66 60 € 1,95
und Widerstände	JX-10 66 57 € 1,85
Piezo-Signalgeber	JX-00 73 87 € 0,95
Mikroschalter und -taster	JX-10 66 67 € 2,80
LC-Display, 2 x 16 Zeichen	JX-05 41 84 € 9,95
oder myAVR-LCD-Add-on	
Pin-Ausrichter	JX-00 84 63 € 4,95

Preisstellung Juni 2013 –
aktuelle Preise im Web-Shop