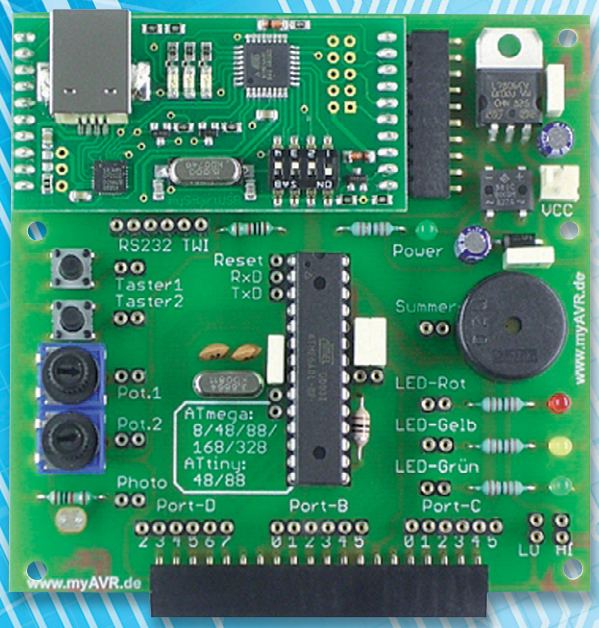
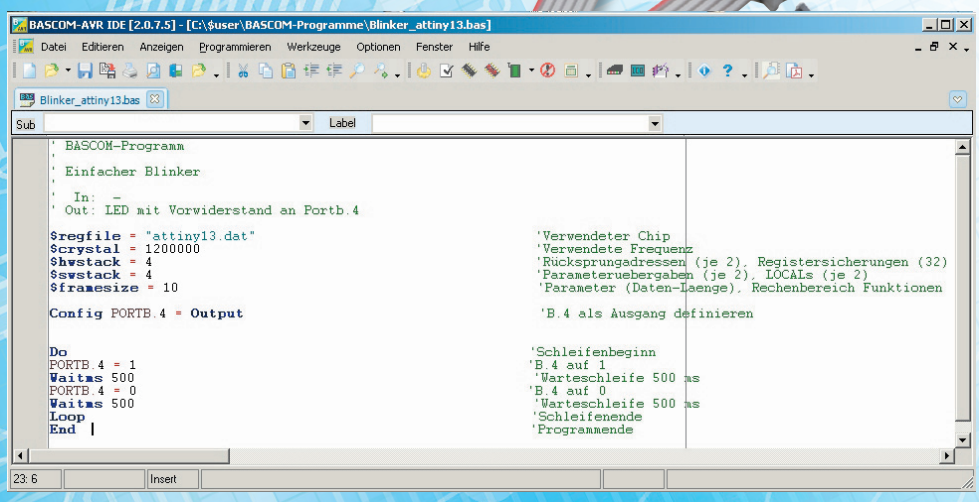
Mikrocontroller-Einstieg

Teil 6: Timer-Modi CTC, PWM und Counter





mit BASCOM-AVR

In Teil 5 der Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR" im ELVjournal 4/2013 wurden die Grundlagen der Timerprogrammierung vorgestellt. Durch einfache Konfigurationsparameter können Timer in speziellen Modi verwendet werden, die in diesem Artikel an Beispielen erläutert werden.

Timer im CTC-Modus

Man kann den Timer eines AVR-Mikrocontrollers so konfigurieren, dass er nicht bis zu seinem maximalen Wert zählt, sondern nur bis zu einem Vergleichswert, dem Compare-Wert (engl. CTC = clear timer on compare match). Dadurch wird die Zeit eines Durchlaufs entsprechend verkürzt, und es lassen sich unterschiedliche Frequenzen – z. B. zur Tonerzeugung – im Hintergrund erzeugen, ohne das Hauptprogramm durch Sprünge in eine Interrupt-Routine und Startwertzuweisungen unterbrechen zu müssen (Bild 1 und 2). Das Ausgabesignal am zugeordneten Ausgabepin behält seine Frequenz bei, bis ein neuer Compare-Wert zugewiesen wird.

```
' BASCOM-Programm
'
' Timer für CTC
' Tonerzeugung
'
' In:  -
' Out: Buzzer ohne Elektronik an OC1A = B.1 = Pin 15

$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 , Compare A = Toggle , Prescale = 1 , Clear Timer = 1   'Timer1 für CTC
                                                            'B.1 automatisch als Ausgang fuer Buzzer
Do
Pwm1a = 1136                                                'Hoher Ton (440 Hz) weil schnell wieder 0
Wait 1                                                      '1 Sekunde warten
Pwm1a = 2272                                                'niedriger Ton (220 Hz) weil spät auf Null
Wait 3                                                      '3 Sekunden warten
Loop
End
```

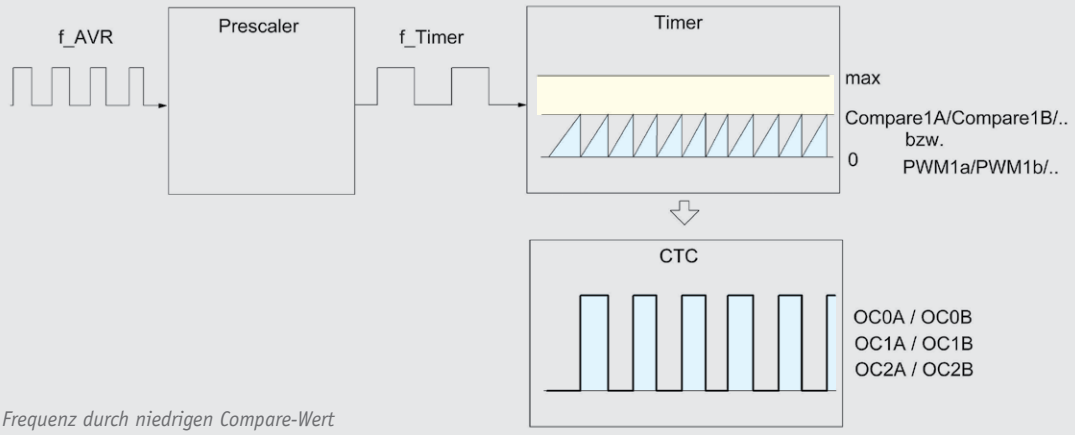


Bild 1: Timer für CTC mit hoher Frequenz durch niedrigen Compare-Wert

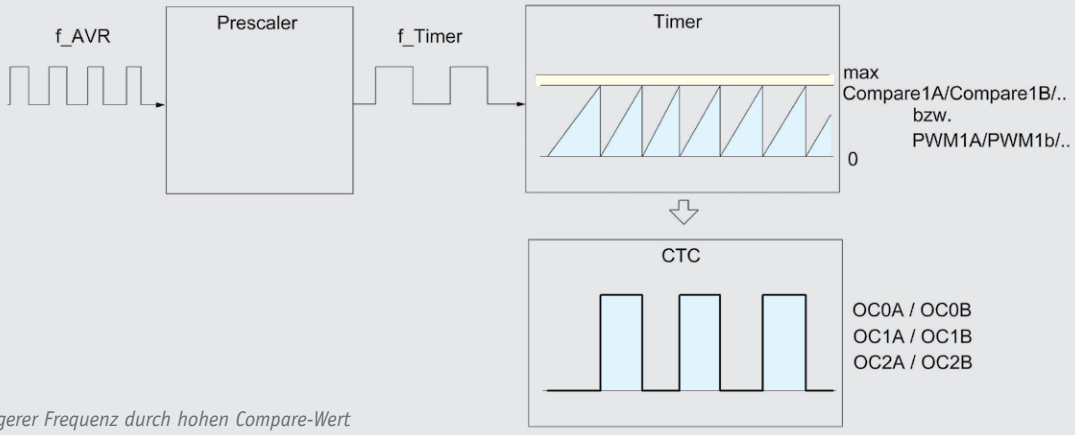


Bild 2: Timer für CTC mit niedrigerer Frequenz durch hohen Compare-Wert

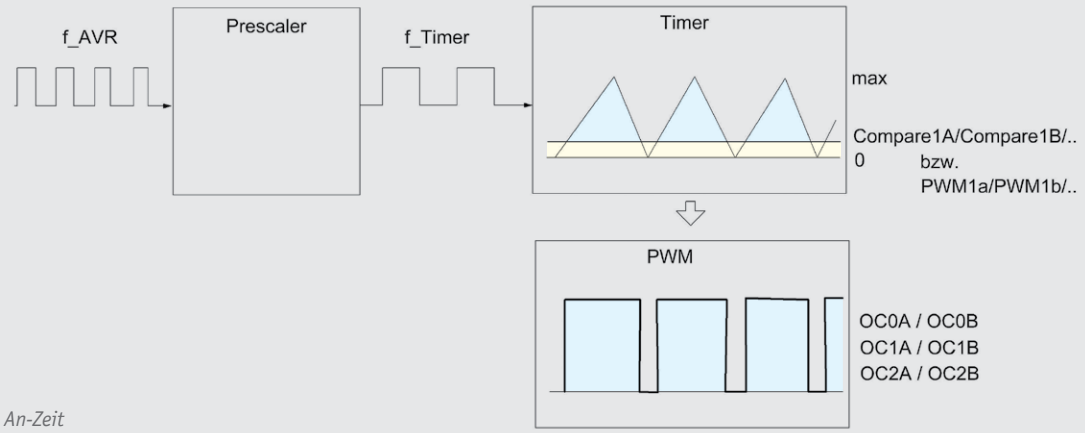


Bild 3: Timer für PWM – lange An-Zeit

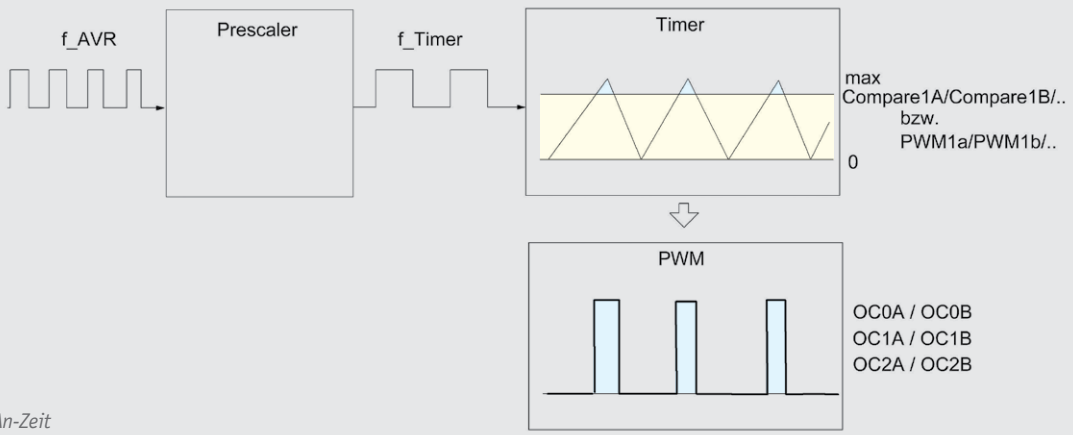


Bild 4: Timer für PWM – kurze An-Zeit

Erläuterungen:
Der Timer1 wird für die Betriebsart CTC konfiguriert, indem mit dem Befehl CONFIG Timer1 beschrieben wird, welche Aktion ausgeführt werden soll, wenn der Compare-Wert erreicht wird: Das Ausgangssignal soll getoggelt und der Zählerstand auf null gesetzt werden. Durch die Timerkonfiguration für den CTC-Modus wird der entsprechende Pin automatisch als Ausgang konfiguriert (Bild 5).
Ein Tick dauert bei 1 MHz Taktfrequenz und Prescaler 1 1 / 1 MHz / Prescaler = 1 µs.
Für 880 Hz benötigt man 1 / 880 Hz = 1,136 ms = 1136 µs. Also 1136 Ticks.
Für 440 Hz benötigt man 1 / 440 Hz = 2,272 ms = 2272 µs. Also 2272 Ticks.
Mit den Zuweisungen zu Pwm1a, dem Register für den Compare-Wert, wird jeweils der höchste Zählwert festgelegt, nach dem dann wieder auf null gesprungen wird.

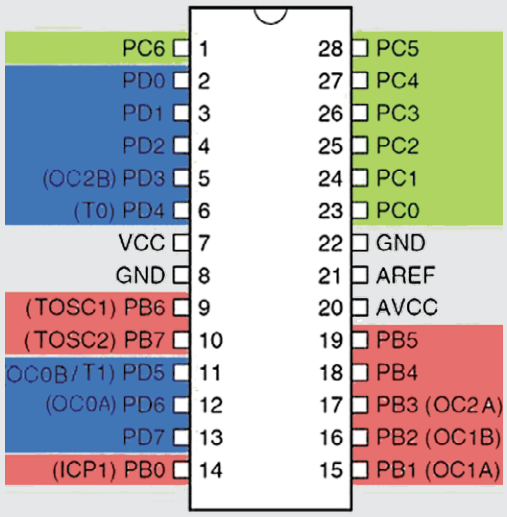


Bild 5: Pinbelegung ATmega88

Timer erzeugt Pulsweitenmodulation

Als Pulsweitenmodulation (PWM) bezeichnet man ein Verfahren, bei dem ein Signal eine konstante Frequenz hat, aber das High-Low-Verhältnis verändert wird. Ein Timer eines AVR-Mikrocontrollers kann im PWM-Modus betrieben werden. Der Timer zählt dann jeweils von null bis zu einem Maximalwert und von dort wieder bis null. Der Maximalwert wird durch die im Konfigurationsbefehl angegebene Bitgröße bestimmt. Bei einer 10-Bit-PWM wird von 0 bis 1023 gezählt und dann wieder hinunter bis 0. Die PWM-Frequenz ergibt sich dann zu: Systemtakt-Frequenz / Prescaler / (Timerauflösung * 2). Hier also 1 MHz / 1 / (1023 * 2) = 489 Hz.

Das gewünschte High-Low-Verhältnis wird durch Setzen eines Compare-Werts bestimmt, welcher angibt, bei welchem Zählerstand das Ausgangssignal ein- bzw. ausgeschaltet wird. In Bild 3 bewirkt ein niedriger Compare-Wert ein Schalten bei niedrigem Zählerstand des Timers. In Bild 4 sieht man das veränderte PWM-Ausgangssignal bei höherem Compare-Wert. Im PWM-Modus ist das Ausgangssignal einem bestimmten Pin zugeordnet, dessen Bezeichnung dem Datenblatt zu entnehmen ist. Für einen ATmega88 sieht man die Pinzuordnungen in Bild 5.

```
' BASCOM-Programm
'
' Timer für PWM
' LED dimmen
'
' In:  -
' Out: Buzzer ohne Elektronik an OC1A = B.1 = Pin 15

$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 Hell = 0
Const Mittel = 800
Const Dunkel = 1023
Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down , Prescale = 8  'Timer1 für PWM
                                                            'B.1 automatisch als Ausgang fuer LED
Helligkeit Alias Pwm1a

Do
Helligkeit = Dunkel                                         'LED dunkel weil sehr kurz an
Wait 1                                                      '1 Sekunde warten
Helligkeit = Mittel                                         'LED mittelhell weil länger an
Wait 4                                                      '4 Sekunden warten
Helligkeit = Hell                                           'LED hell weil lange an
Wait 8                                                      '8 Sekunden warten
Loop
End
```

Erläuterungen:
Der Timer1 wird für einen PWM-Modus mit 10 Bit Auflösung und einem Vorteiler von 8 konfiguriert. Das Compare-Register pwm1a erhält den Aliasnamen Helligkeit, um das Programm leichter lesbar zu gestalten. Es können nun im Programm Werte zwischen 0 und 1023 zugewiesen werden, was bewirkt, dass am Ausgangspin B.1 ein PWM-Signal mit konstanter Frequenz, aber je nach Wert anderem An-/Aus-Verhältnis ausgegeben wird. Eine am Pin angeschlossene LED wird dadurch pro Zeiteinheit länger oder kürzer angeschaltet, und aufgrund der Trägheit des menschlichen Auges, welches das schnelle Ein- und Ausschalten nicht wahrnehmen kann, erscheint die LED unterschiedlich hell. Statt eine LED zu dimmen, könnte man ebenso über einen Treiber die Drehzahl eines Motors steuern.
Der große Vorteil beim Betrieb des Timers im PWM-Modus ist, dass kein Programm- bzw. Zeit-Overhead erzeugt wird, indem bei jedem Überlauf eine Interrupt-Routine anspringt. Der Timer läuft im PWM-Modus unabhängig vom Programm im Hintergrund und schaltet den zugeordneten Pin ohne Sprünge in eine Interrupt-Routine.

Servo mit PWM
Ein Servo benötigt zur Ansteuerung jeweils ca. 20 ms lang 0 V und dann 1–2 ms lang 5 V. Das lässt sich mit einem Timer im PWM-Modus abbilden, sodass im Programm nur der Compare-Wert Pwm1a zugewiesen werden muss und dann die Steuerung im Hintergrund ohne Interrupt-Routine erfolgt.

```
' BASCOM-Programm
'
' Timer für PWM
' Servoansteuerung
'
' In:  -
' Out: Servo an OC1A = B.1 = Pin 15

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


Const Links = 898
Const Rechts = 960
Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down , Prescale = 8    'Timer1 für PWM
                                                                              'B.1 automatisch als Ausgang fuer Servo
Servo Alias Pwm1a

Do
Servo = Links                                               'Servo linker Anschlag
Wait 2                                                      '2 Sekunden warten
Servo = Rechts                                              'Servo rechter Anschlag
Wait 4                                                      '4 Sekunden warten
Loop
End
```

Erläuterungen:
Der Timer1 wird für einen PWM-Modus mit 10 Bit Auflösung und Vorteiler 8 konfiguriert. Die PWM-Frequenz ist $f_{PWM} = f_{AVR}$ / Prescaler / (PWM-Auflösung * 2) = 1 MHz / 8 / (1023 * 2) = 61 Hz. Die PWM-Periodendauer beträgt $1/f_{PWM}$ = 16,4 ms, was ausreichend genau an 20 ms liegt, um einen Servo ansteuern zu können. 1 Tick hat bei 1 MHz Systemtakt die Dauer T_{Tick} = 1 / 1 000 000 Hz * Prescaler = 8 µs.
Für 1 ms = 1000 µs benötigt man 1 000 µs / 8 µs = 125 Ticks. Für 2 ms benötigt man 250 Ticks.

Nun wird der Compare-Wert für 125 bzw. 250 Ticks berechnet. Davon jeweils die Hälfte Ticks VOR dem Maximalwert des Timers von 1023.
Für 1 ms (125/2 Ticks): 1023 - (125/2) = 960.
Für 2 ms (250/2 Ticks): 1023 - (250/2) = 898.

In der Hauptschleife kann man diese Compare-Werte dem Compare-Register pwm1a zuweisen, und der Timer erzeugt dann im Hintergrund das PWM-Signal, welches am zugeordneten Pin ausgegeben wird.

Timer als Counter

Ein Timer des AVR-Mikrocontrollers kann als Zähler (= Counter) externer Signale verwendet werden. Es wird dann nicht der interne – eventuell vorgeteilte – Systemtakt als Takt für den Timer verwendet, sondern das extern am zugeordneten Pin angeschlossene Signal (Bild 6 und Bild 7). Ein Vorteiler kann im Counter-Modus nicht konfiguriert werden. Auch im Counter-Modus zählt der Timer aufwärts bis 255 bzw. 65535, und man kann beim Übergang zu null einen Interrupt auslösen lassen. Im folgenden Beispiel wird der Zählerstand durch angeschlossene LEDs dargestellt, und man kann beobachten, dass der Zählerstand sich bei jedem Tastendruck erhöht. Bei einem prellenden Taster wird mehrfach gezählt.


```
' BASCOM-Programm
'
' Timer als Counter
' Timerwert wird durch externes Signal hochgezählt
'
' In:    Entprellter (!!) Taster/Kontakt bzw. Signal an T0 = D.4 = Pin 6
' Out:   Leuchtdioden mit Vorwiderständen an Port C

$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 Portc = Output                     'Port C als Ausgang
Config Timer0 = Counter , Edge = Falling 'Timer 0 als Counter/Zähler konfigurieren

Config Portd.4 = Input                    'Eingang für Counter
Portd.4 = 1                               'Interner Pullup-Widerstand

Do
Portc = Timer0                            'Ausgabe des Timer/Zähler-Wertes in binärer Form an Port C
Loop
End
```

Erläuterungen:
In diesem Beispiel wird der 8-Bit Timer Timer0 als Zähler für fallende Flanken konfiguriert. Das Eingangssignal für den Counter muss an T0 (Pin 6 des Mikrocontrollers) angelegt werden (Bild 6). Für Timer1 im Counter-Modus wäre T1 an Pin 11 zu verwenden (Bild 5). Mit jeder fallenden Flanke (High-Low-Wechsel) wird Timer0 um 1 hochgezählt, bis der Zählerstand von 255 wieder auf 0 wechselt. Beim Überlauf könnte man einen Interrupt auslösen lassen, was hier nicht genutzt wird. In diesem Beispiel wird in der Endlosschleife der Inhalt von Timer0 ausgelesen und an das Portregister Portc zugewiesen. Dadurch zeigen an die Portpins angeschlossene LEDs den jeweiligen Zählerstand an. Im Schaltplan in Bild 6 sind drei Leuchtdioden an die niederwertigsten Bits von Portc angeschlossen, mit denen man die Zählerstände von 0 bis 7 anzeigen kann. Durch Anschließen von mehr Leuchtdioden an C3, C4 usw. können höhere Zählerstände angezeigt werden.

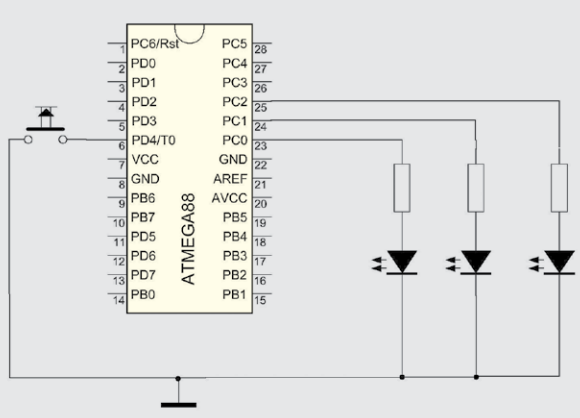


Bild 6: Timer als Counter

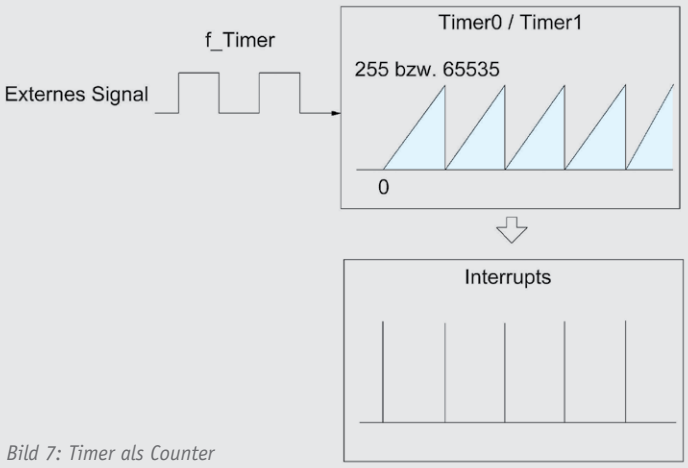
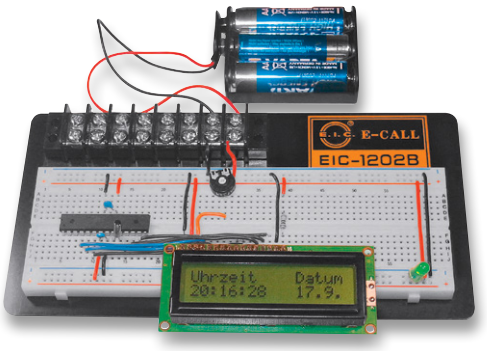


Bild 7: Timer als Counter

Quarzuhr mit Timer2

Der Timer2 bietet die Möglichkeit, sich asynchron mit einem Uhrenquarz mit einer Frequenz von 32,768 kHz takten und dadurch sehr leicht als Basis für eine Quarzuhr nutzen zu lassen. Der Uhrenquarz wird direkt – ohne Kondensatoren – zwischen TOSC1 und TOSC2 (Pin 9 und 10) angeschlossen (Bild 8).




```
' BASCOM-Programm
'
' Timer für Softclock
' Anzeige der Zeit und des Datums
'
' In:  Uhrenquarz (32.768 Hz) zwischen XTAL1 (Pin 9)
  und XTAL2 (Pin10)
' Out: LC-Display an D.2 bis D.7

$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 Clock = Soft                       'Verwendet Timer 2
'Automatisch dimensioniert werden die Variablen:
' _sec , _min , _hour, _day , _month , _year (Byte-Variablen)
' sowie TIME$ und DATE$  (String-Variablen)
' Die Variablen können gelesen und beschrieben werden
Enable Interrupts                         'Alle Interrupts ermöglichen

Config Date = Dmy , Separator = .         'Datumsformat tt.mm.jj  mit Punkt als Trennzeichen
Time$ = "20:15:00"                        'Startzeit zuweisen
Date$ = "17.09.13"                        'Datumswert zuweisen

'LCD-Konfiguration:
Config Lcdpin = Pin , Db4 = Portd.4 , Db5 = Portd.5 , Db6 = Portd.6 , Db7 = Portd.7 , E = Portd.3 , Rs = Portd.2
Config Lcd = 16 * 2
Cursor Off                                'Cursor ausschalten
Cls                                       'LC-Anzeige löschen

Lcd "Uhrzeit    Datum"                    'Ausgabe von statischem Text vor Schleife

Do
Locate 2 , 1                              'LCD-Schreibmarke in Position 1 der 2. Zeile
Lcd Time$                                 'Uhrzeit ausgeben
Lcd "    " ; _day ; "." ; _month ; "."    'Datum mit Einzelvariablen ausgeben
Loop
End
```

Erläuterungen:

Der Mikrocontroller wird mit dem internen Oszillator mit 1 MHz getaktet. Der Uhrenquarz liefert also den Takt für den Timer2, aber nicht für die CPU des Mikrocontrollers. Durch CONFIG Clock = Soft wird der Timer2 für den asynchronen Modus mit Taktung durch den Uhrenquarz eingestellt. Außerdem dimensioniert BASCOM automatisch einige Variablen, die für die Uhrzeit- und Datumsdarstellung benutzt werden können. Mit CONFIG Date wird das deutsche Datumsformat eingestellt. Das Einstellen von Uhrzeit und Datum erfolgt im Beispiel durch Zuweisung zu den entsprechenden String-Variablen Time$ und Date$. Man könnte auch Einstellroutinen programmieren und die Werte den numerischen Variablen _hour, _min, _day, _month usw. zuweisen. Die Uhrzeit- und Datumsvariablen können gelesen und geschrieben werden. Die Zeichenkettenvariablen und die numerischen Variablen werden intern synchron gehalten. In der Endlosschleife werden die Zeit und das Datum angezeigt, und man sieht sofort, wie die Uhrzeit weiterzählt. Man kann auf einfache Weise auch eine im Sekundentakt angesprungene Interrupt-Routine erhalten, indem man hinter CONFIG Clock = Soft noch „Gosub = Sectic“ schreibt und eine Interrupt-Routine mit dem Namen Sectic und abgeschlossen mit RETURN beschreibt. Dort kann man eine LED im Sekundentakt blinken lassen, ein Flag setzen oder Ähnliches.

Tabelle 1 fasst die vier Haupt-Modi der Timer-Benutzung zusammen. Dabei wird von den üblichen Verwendungen ausgegangen.

Tabelle 1

Hauptmodi der Timer-Benutzung

Timer-Modus	Normalmodus	CTC	PWM	Counter
Taktquelle	Systemtakt	Systemtakt	Systemtakt	extern
Prescaler	ja	ja	ja	nein
Ausgabepin fest	nein	OCnx	OCnx	nein
Eingabepin	nein	nein	nein	Tn
Hauptaktion	ISR/Timerstand	Signalerzeugung	Signalerzeugung	Zähler/ISR
Läuft im Hintergrund	ja	ja	ja	ja
Verwendung	- Aktion in festen Intervallen (1s, 1/10s ..) - Zeitmesser	Signal/Töne erzeugen	- LED dimmen - Motordrehzahl - Servo - DAC	Ereigniszähler

Im Rahmen dieses Artikels wurde ein kleiner Einblick in die sehr vielfältigen Möglichkeiten mit Timern gegeben. Wer mehr Details kennenlernen möchte – z. B. unterschiedliche PWM-Modi –, findet in der BASCOM-Hilfe und dem Datenblatt des jeweiligen Mikrocontrollers alle benötigten Informationen.

Ausblick

Im nächsten Teil der Artikelserie „Mikrocontroller-Einstieg mit BASCOM-AVR" werden die Grundlagen der seriellen (UART) Kommunikation zwischen einem AVR-Mikrocontroller und PCs, anderen Mikrocontrollern, Sensoren und Aktoren beschrieben. Dabei werden die Kommunikation mit diversen ELV-Modulen und deren Ansteuerung mit BASCOM-AVR gezeigt.

ELV

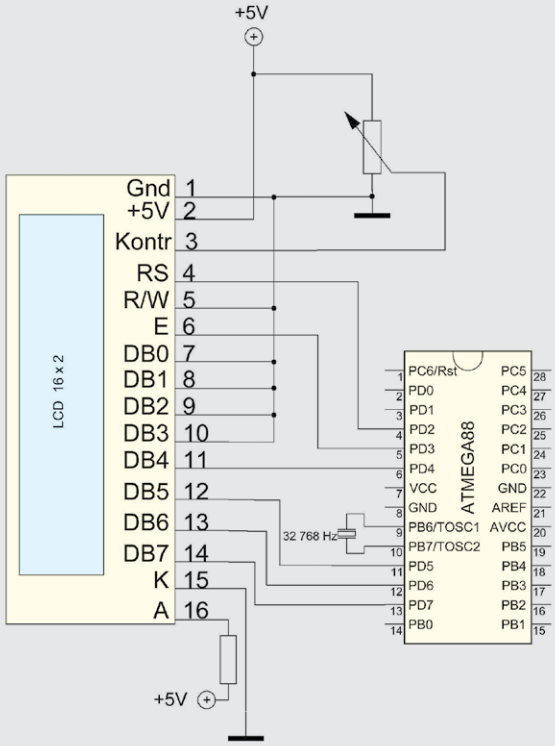


Bild 8: Schaltplan Quarzuhr

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

Preisstellung August 2013 – aktuelle Preise im Web-Shop