AVR-Mikrocontroller in BASCOM programmieren, Teil 3

Alle Beispiele in diesem Kapitel beziehen sich auf den Mega8.
Andere Controller können unterschiedliche Timer haben.

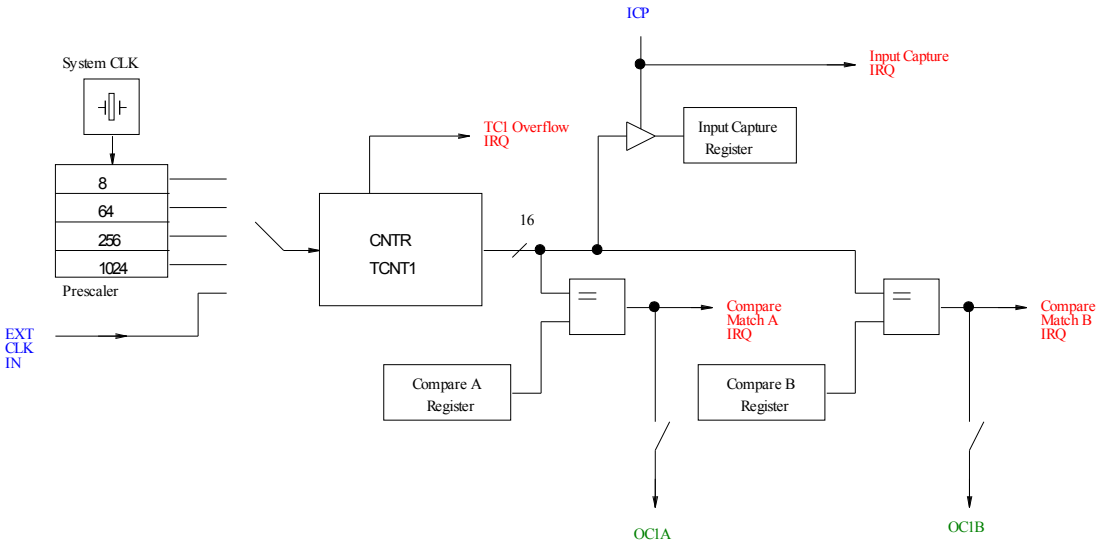
14. Timer

14.1 Allgemeines

- Beim Mega 8 gibt es drei Timer:
  Timer0: 8 bit (0...255)
  Timer1: 16 bit (0...65535)
  Timer2: 8 bit (0...255)

- Timer1 bietet die meisten Möglichkeiten:
  - Erzeugung von präzisen Impulsen, ohne dass dafür Rechenzeit benötigt wird
  - Regelmäßiges Ausführen von Unterprogrammen
  - Uhr
  - Pulsweitenmodulation (PWM)
  - Erfassen von Impulslängen und Zeiten

Grob gesehen hat der Timer1 folgendes Blockschaltbild:



Rot: Interrupt Requests (Interrupt-Anforderungen durch Timer1)

Blau: Eingangspins

Grün: Ausgangspins

- Der Takt des Timers kann gleich dem Takt des Controllers sein, oder um einen Faktor 8,

64, 128 oder 1024 heruntergeteilt werden.

Dies erledigt der **Prescaler**.

Für einen 4MHz-Quarz ergibt sich z.B. folgendes:

Prescaler	**1**	**8**	**64**	**256**	**1024**
Auflösung in µs (T_{CLK})	0.25	2	16	64	256
Tmax Timer0 (8 bit)	64µs	512µs	4096µs	16384µs	65 536µs
Tmax Timer1 (16 bit)	16.384ms	131.072ms	1.048s	4.194s	16.777s

14.2 Timer1 mit Overflow-Interrupt

Der Timer zählt vor sich hin, bei jedem Overflow wird ein Interrupt ausgelöst.

Diese Betriebsart ist nur selten von Nutzen, da sie als Zeitintervall nur wenige Werte zuläßt, nämlich $T = 65536 \cdot T_{CLK}$, wobei T_{CLK} die durch den Prescaler heruntergeteilte Taktfrequenz des Controllers ist.

Eine Anwendung wäre das regelmäßige Ausführen einer Aktion, wenn die Zeitspanne frei gewählt werden kann.

Im folgenden Beispiel blinkt eine LED, ohne dass das Hauptprogramm etwas dafür tun muß:

```
$crystal = 8000000
$regfile = "m8def.dat"

Config Pind.5 = Output                          'LED

Config Timer1 = Timer , Prescale = 64
On Timer1 Timer1serv
Enable Timer1
Enable Interrupts

'-----------------------------------------------
'Hauptprogramm:
Do
   'Dreh Däumchen
   'oder tu sonstwas....
Loop
'-----------------------------------------------
Timer1serv:
  Toggle Portd.5
Return
```

Mit dem Prescaler von 64 wird der 8MHz-Takt heruntergeteilt, dies entspricht einer Takt-Periodendauer von 0.125µs · 64 = 8µs.
Ein Overflow tritt also alle 65536 · 8µs = 524ms auf.
Die Blinkfrequenz ist also ca. 1Hz.

Zuerst müssen Timer-Interrupt und Interrupts allgemein freigeschaltet werden, damit ein Timer-Interrupt möglich ist! Dies geschieht mit:

```
On Timer1 Timer1serv
Enable Timer1
Enable Interrupts
```

14.3 Timer1 mit Output Compare

Dies ist die geeignete Betriebsart, um zu definierten Zeitpunkten eine Aktion auszuführen oder ein Signal mit definierter Frequenz zu erzeugen.

Im Prinzip wird der Zustand des Zählregisters mit dem Zustand des Compare-Registers verglichen und bei Übereinstimmung ein Interrupt ausgelöst oder der Zustand des OC1A bzw. OC1B-Pins geändert.

a) Regelmäßig auszuführender Interrupt

Beispiel: alle 100ms soll im Programm etwas geschen, hier der Einfachheit halber eine LED blinken.

Wenn wir einen 8MHz-Quarz benutzen haben wir eine Taktperiodendauer T_{CLK} = 0.125µs.

Wir müssen den Prescaler so bemessen, dass sich einerseits eine möglichst hohe Auflösung ergibt (kleiner Wert), andererseits aber auch der Zähler nicht vor dem Erreichen des Endwertes überläuft großer Wert).

Wir probieren ein wenig herum:

Prescaler	Auflösung	Endwert
1 8 64 ...	 8·0.125µs = 1µs 64·0.125µs = 8µs	 65536 · 1µs = 65.536ms (zu klein!) 65536 · 8µs = 524.288ms (reicht)

Also wählen wir Prescaler = 64.

Dabei wird alle 8µs um eins weitergezählt.

Der Interrupt soll alle 100ms ausgeführt werden, dies entspricht $\frac{100000\,\mu s}{8\,\mu s} = 12500$ Zählschritten.

Diese Zahl muß also ins Compare Register geschrieben werden.

```
$crystal = 8000000
$regfile = "m8def.dat"

Config Pind.5 = Output                          'LED

Ocr1a = 12500
Config Timer1 = Timer , Prescale = 64 , Clear Timer = 1 , Compare A = Disconnect
On Oc1a Timer1serv
Enable Oc1a
Enable Interrupts

'-----------------------------------------------
'Hauptprogramm:
Do
   'Dreh Däumchen
   'oder tu sonstwas....
Loop
'-----------------------------------------------
Timer1serv:
  'statt eine LED zum Blinken zu bringen kann hier
  'regelmässig etwas Wichtiges getan werden
  Toggle Portd.5
```

```
Return
```

In der Konfiguration des Timers

```
Config Timer1 = Timer , Prescale = 64 , Clear Timer = 1 , Compare A = Disconnect
```

wird **Clear Timer = 1** gesetzt. Dies bewirkt, dass bei jedem Compare Match (Übereinstimmung zwischen Zähl- und Vergleichsregister) der Zähler zurückgesetzt wird und somit wieder von null mit zählen beginnt.
Das **Disconnect** bewirkt, dass am OC1A-Pin kein Signal erscheint.

Vorsicht!
Bei hohen Frequenzen ist die Einsprungzeit für den Interrupt nicht mehr zu vernachlässigen.
(Siehe Kapitel Interrupts)

b) Generierung eines Signals am OC1A-Pin

Wenn ein Signal mit fester Frequenz an einem OC1-Pin erzeugt werden soll, ist noch weniger Software-Aufwand nötig, da sich die Hardware des Controllers um alles kümmert.

Nur das Compare-Register muß initialisiert werden.

Beispiel: es soll ein **1kHz-Signal** erzeugt werden, bei einer Quarzfrequenz von 4MHz.

Die Taktperiodendauer beträgt $T_{CLK} = 0.25\mu s$.

Bei Prescale = 1 ist dies auch die Auflösung des Zählers.

Das Umschalten des Pins muß alle 500µs ausgeführt werden, dies entspricht $\frac{500\,\mu s}{0.25\,\mu s} = 2000$ Zählschritten.

Dies entspricht einem Comparewert von 1999, da der Zähler dann von 0...1999 zählt und dabei 2000 Schritte macht.

```
$crystal = 4000000
$regfile = "m8def.dat"

Ocr1a = 1999
Config Timer1 = Timer , Prescale = 1 , Clear Timer = 1 , Compare A = Toggle

Do
Loop
```

Mit **Compare A = Toggle** wird dafür gesorgt, dass der OC1A-Pin bei jedem Erreichen des Vergleichswerts umgeschaltet wird.

Wichtig ist auch **Clear Timer = 1** , sodass der Timer bei jedem Compare Match rückgesetzt wird.

Der Vorteil dieser Methode im Gegensatz zum Arbeiten mit Interrupts ist, daß die Frequenz völlig unabhängig davon ist, was der Controller sonst noch alles tun muß.

Soll die Frequenz geändert werden, genügt es, der Wert im Register OCR1A zu ändern.

c) Generierung von phasenverschobenen Signalen an OC1A und OC1B

Obschon der Controller zwei Compare-Register hat, kann man keine unterschiedlichen Frequenzen oder Signale mit einem Tastverhältnis ungleich 0.5 mit einem Timer erzeugen.

Mit dem Compare B – Register kann aber ein phasenverschobenes Signal erzeugt werden.

Leider gibt es dazu keine vernünftige Dokumentation, oder ich habe sie noch nicht gefunden.

Dieser Teil ist aus Zeitgründen (na, warum sonst?) leider noch nicht fertig geworden.

Hoffentlich demnächst mehr zum Thema Timer!

Kritik und Rückmeldungen bitte an jean-claude.feltes@education.lu