A5 Zeitschleifen

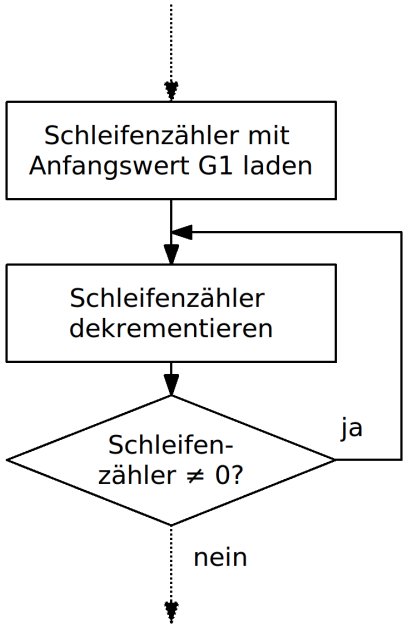
Als nächstes soll ein Blinklicht programmiert werden. Der Controller ist aber für diese Aufgabe viel zu schnell. Es muss also zwischen dem Ein- und Ausschalten der LED eine Verzögerung einbaut werden, so dass unser Auge dem Blinken folgen kann. Dies passiert mit einer so genannten Zeitschleife. Natürlich hat der ATmega-Chip auch interne Timer mit denen so etwas bewerkstelligt werden kann, ohne dass der Controller Zeit tot schlägt.

Für einfache Anwendungen werden dennoch immer wieder Zeitschleifen benötigt.

Zeitschleifen sind Programme, deren Aufgabe es ist Zeit zu verbrauchen.

8-Bit-Zeitschleife

Die erste einfachste Zeitschleife soll eine Zehntelsekunde verbrauchen. Um dies zu bewerkstelligen wird ein Schleifenzähler (zum Beispiel "**TCnt1**"[23]) mit seinem **Anfangswert G_1** initialisiert und dann auf Null abgezählt.



Für jeden Befehl in der Zeitschleife muss bestimmt werden, wie viele Takte erforderlich sind, um diesen Befehl auszuführen. Ist die Taktfrequenz des Controllers bekannt, so erhält man die Dauer der Zeitschleife.

Die folgenden Befehle sollen verwendet werden:

```
        ldi     TCnt1,0xFF      ;Schleifenzaehler TCnt1 = G1 (hier 255)
LOOP:   dec     TCnt1           ;Dekrementiere den Schleifenzaehler
        brne    LOOP            ;Verlasse die Schleife bei TCnt1=0
```

23 Zuweisung mittels Assembleranweisung (Bsp: **.DEF TCnt1 = r19**)

✎ **A500**

a) Im Anhang befindet sich der Befehlssatz des ATmega32A. In diesem ist unter "#Clocks" angegeben wie viele Takte jeder einzelne Befehl benötigt. Ermittle anhand des Befehlssatzes die Anzahl der benötigten Takte für die benutzen Befehle und trage sie in die Tabelle ein.
b) Wieso werden beim Sprungbefehl wohl zwei Werte angegeben?
c) Vervollständige die folgende Tabelle und stelle daraus ein Formel auf zur Berechnung der (totgeschlagenen) **Zeit t** in Funktion des **Anfangswertes G_1**. Benutze die Abkürzung t_T für die **Zeitdauer eines Taktes** (entspricht der Periodendauer T eines Rechtecksignals).
d) Stelle die Formel nach G_1 um.

Befehle	Takte	Durchläufe
`ldi TCnt1,255`		
`LOOP: dec TCnt1`		
`brne LOOP`		

_____

_____

Formeln zur Berechnung der Zeit **t** bzw. des Anfangwertes G_1 einer **8-Bit-Zeitschleife**:

$t =$

$G_1 =$

✎ **A501**

a) Ermittle die maximale Zeit die mit dieser Zeitschleife verbraucht werden kann einmal für die interne Frequenz von 1 MHz und einmal für eine externe Frequenz von 16 MHz. Die Zeitdauer eines Taktes t_T lässt sich aus der Frequenz errechnen mit $t_T = T = 1/f$.
b) Kann damit eine Zehntelsekunde abgedeckt werden?
c) Was passiert wenn man als Anfangswert Null in das Zählerregister setzt?

16-Bit-Zeitschleife

Leider reicht die Zeit die man mit einer 8-Bit-Zeitschleife erreichen kann nicht aus um das Blinken sichtbar zu machen.

sbiw Rdl,K

Subtrahiert die Konstante K (0-63) von einem Doppelregister (*subtract immediate from word*)

1	0	0	1	0	1	1	1	K	K	d	d	K	K	K	K

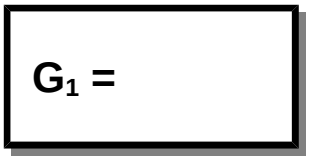
Resultat im Doppelregister. Neben den Doppelregistern X,Y,Z kann auch das Doppelregister r25:r24 verwendet werden. Obschon das Doppelregister adressiert wird ist im Befehl nur das niederwertige Register anzugeben Bsp.: adiw YL,10. (andere Schreibweisen sind je nach Assembler möglich). Die Konstante kann maximal 63 betragen (6 Bit).
Beeinflusste Flags: S, V, N, Z, C Taktzyklen: 2

✎ **A502**

a) Schreibe das obige Programm so um, dass statt einem 8-Bit-Register ein Doppelregister verwendet wird. Nutze hier das Doppelregister **X**. Dieses Doppelregister wird mit **XL** und **XH** angesprochen. Zum Dekrementieren kann der Befehl "**sbiw XL,1**" verwendet werden.
(!Achtung! Obschon in der Syntax **XL** angegeben wird bezieht sich die Subtraktion auf das gesamte Doppelregister **X**).
b) Stelle die Formel zur Berechnung der Zeit **t** in Funktion des Anfangswertes G_1 auf.
c) Die Formel lässt sich Vereinfachen. Wie groß ist der maximale Fehler, der mit der vereinfachten Formel in Kauf genommen wird?
d) Ist es zulässig die vereinfachte Formel zu verwenden?
e) Stelle die (vereinfachte) Formel nach G_1 um.
f) Ermittle die maximale Zeit die mit dieser Zeitschleife verbraucht werden kann einmal für die interne Frequenz von 1 MHz und einmal für die externe Frequenz von 16 MHz wenn G_1 auf dezimal **65535** (**0xFFFF**) gesetzt wird.
g) Ermittle die Anfangswerte des Doppelregisters damit bei beiden Frequenzen eine Zehntelsekunde erreicht wird.

Befehle	Takte	Durchläufe
LOOP:		

Formeln zur Berechnung der Zeit **t** bzw. des Anfangswertes G_1 einer **16-Bit-Zeitschleife**:



✎ **A503**

a) Ändere das Toggle-Programm aus dem Kapitel A3 so um, dass die LED blinkt (natürlich ohne Schalter). Dazu wird die vorhin erstellte 16-Bit-Zeitschleife eingebaut. Beginne mit dem maximalen Wert für den Schleifenzähler. Speichere das Programm als "**A503_16bit_loop.asm**".
b) Berechne die Blinkfrequenz und miss sie mit dem Oszilloskop.
c) Setze den Anfangswert so, dass die Schleife eine Zehntelsekunde benötigt. Wie hoch ist dann die Blinkfrequenz?
d) Ändere den Schleifenzähler so lange, bis das Blinken so eben noch sichtbar ist. Berechne die Blinkfrequenz.
e) Miss die Blinkfrequenz mit dem Oszilloskop und vergleiche sie mit dem errechneten Wert. Sind Pausendauer und Impulsdauer gleich?

Bemerkung: Statt mit dem "**sbiw**"-Befehl kann eine 16-Bit Subtraktion auch in zwei Schritten mit den beiden Befehlen "**subi**" und "**sbci**" durchgeführt werden. Mit "**subi**" werden die **niederwertigen Register** subtrahiert. Der eventuell auftretende Übertrag (carry) durch Borgen wird vom "**sbci**"-Befehl der die beiden **hochwertigen Register** subtrahiert zusätzlich abgezogen. Die Formel zur Berechnung der Zeitschleife bleibt gleich, da beide Befehle zusammen auch nur zwei Taktzyklen benötigen.

```
LOOP:   subi    XL,1            ;Dekrementiere den Schleifenzaehler
        sbci    XH,0
        brne    LOOP            ;Verlasse die Schleife bei Schleifenzaehler X=0
```

subi Rd,K

Subtrahiere die Konstante K (8 Bit) vom Register Rd (*subtract immediate*).

0	1	0	1	K	K	K	K	d	d	d	d	K	K	K	K

Resultat in Rd. Nur r16-r31!
Beeinflusste Flags: H, S, V, N, Z, C Taktzyklen: 1

sbci Rd,K

Subtrahiere die Konstante K (8 Bit) und das Carry-Flag vom Register Rd (*subtract immediate with carry*).

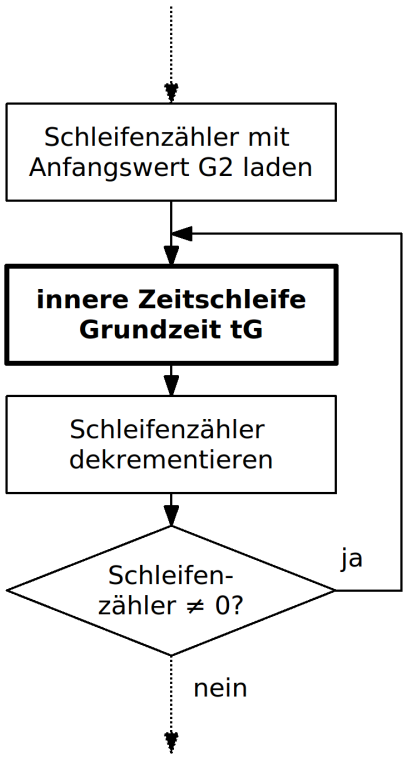
0	1	0	0	K	K	K	K	d	d	d	d	K	K	K	K

Resultat in Rd (Rd = Rd-K-C). Nur r16-r31!
Beeinflusste Flags: H, S, V, N, Z, C Taktzyklen: 1

Verschachtelte Zeitschleifen.

Um längere Zeiten zu erreichen, können die Zeitschleifen verschachtelt werden. In einer äußeren Zeitschleife wird dann eine innere Schleife G_2 mal ausgeführt. Die innere Schleife erhält eine feste Grundzeit t_G von zum Beispiel einer Millisekunde. Die Gesamtzeit ist dann ein Vielfaches dieser Grundzeit $t = G_2 \cdot t_G$.
(Bsp.: $t_G = 1ms$; $G_2 = 1000$; $t = 100 \cdot 1ms = 1\ s$).

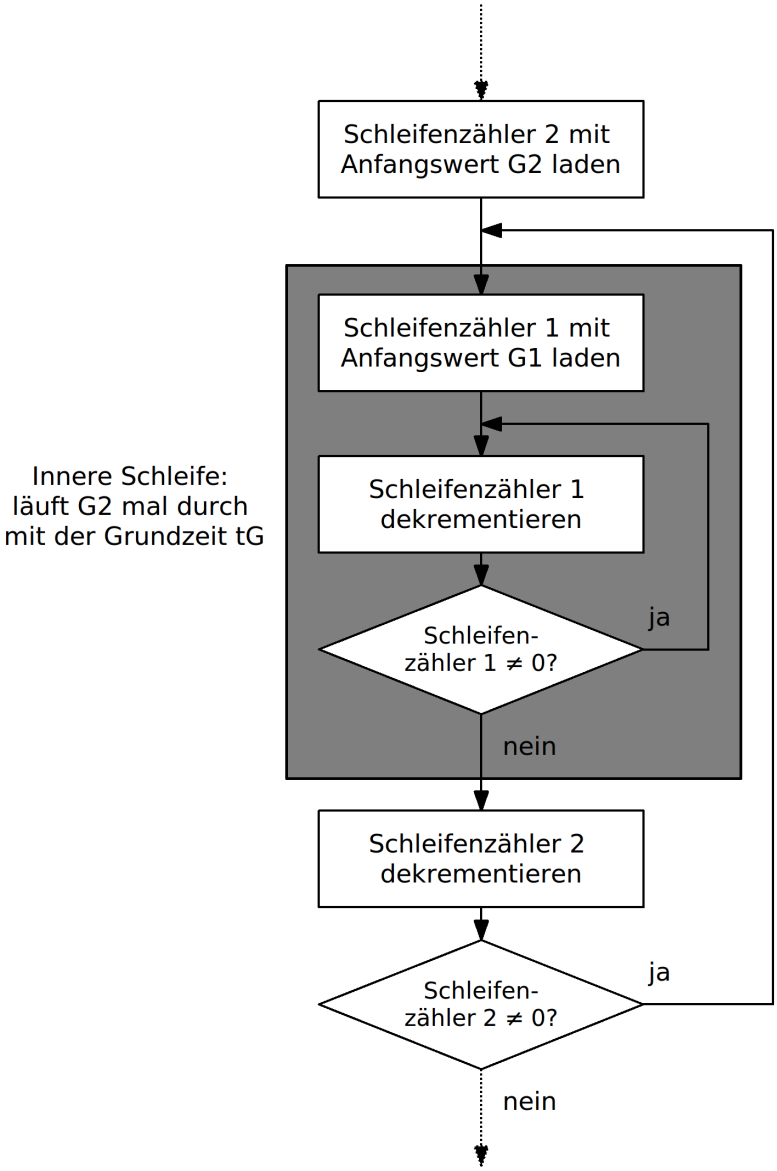
Prinzip der verschachtelten Zeitschleife:



Wenn die Grundzeit t_G der inneren Schleife viel größer ist als die Zeit, die für die Befehle der äußeren Schleife benötigt werden, (hier "**sbiw**" und "**brne**", max. 4 Takte) kann die Gesamtzeit mit $t = G_2 \cdot t_G$ angenommen werden:

$$t = G_2 \cdot t_G$$

Gesamtes Flussdiagramm:



✎ **A504** Ersetze die einfache 16-Bit-Zeitschleife im vorigen Programm durch eine verschachtelte 16-Bit-Zeitschleife. Verwende das Doppelregister **`r25:r24`** als Schleifenzähler 2. Dieses Doppelregister soll mit **`WL`** und **`WH`** angesprochen werden (siehe **.DEF**-Anweisungen in der Vorlage). Zum Dekrementieren kann der Befehl "**`sbiw W,1`**"[24] verwendet werden.
Die Grundzeit soll eine Millisekunde betragen. Die LED soll mit einer Frequenz von 1 Hz aufleuchten. Speichere das Programm als "**`A504_16bit_nested_loop_1.asm`**".

24 Beim Befehl **`sbiw r24,1`** wird das niederwertigste Register (LBYTE) angegeben obschon sich die Subtraktion auf das gesamte Doppelregister (**`r25:r24`**) bezieht.

Bemerkung: Eine Zeitschleife kann natürlich auch mehrfach verschachtelt werden.

Externer Quarz mit 16 MHz

Bis jetzt wurde mit dem internen RC-Oszillator von 1 MHz gearbeitet, für den der ATmega32A bei der Auslieferung programmiert wurde. Es soll für alle weiteren Versuche jedoch ein präziserer externer Quarz mit 16 Mhz verwendet werden. Hierzu müssen die Fuse-Bits des ATmega32A umprogrammiert werden. Das Programmieren der Fuse-Bits wird im Anhang beschrieben und sollte jetzt vorgenommen werden.

✎ **A505** Ändere das vorige Programm so um, dass es trotz der veränderten Quarzfrequenz genau das gleiche tut wie in der vorigen Aufgabe. Speichere das Programm als "**A505_16bit_nested_loop_2.asm**".

✎ **A506** Zeichne ein Flussdiagramm und schreibe ein Programm, welches ein Rechtecksignal mit einer Frequenz von 100 Hz ausgeben kann. Das Tastverhältnis (Tastgrad, engl. duty cycle, Verhältnis der Länge des eingeschalteten Zustands (Impulsdauer) zur Periodendauer bei einem Rechtecksignal) soll 33 % betragen. Kontrolliere das Ergebnis mit dem Oszilloskop und kommentiere die Genauigkeit.
Speichere das Programm als "**A506_frequency_generator_1.asm**".

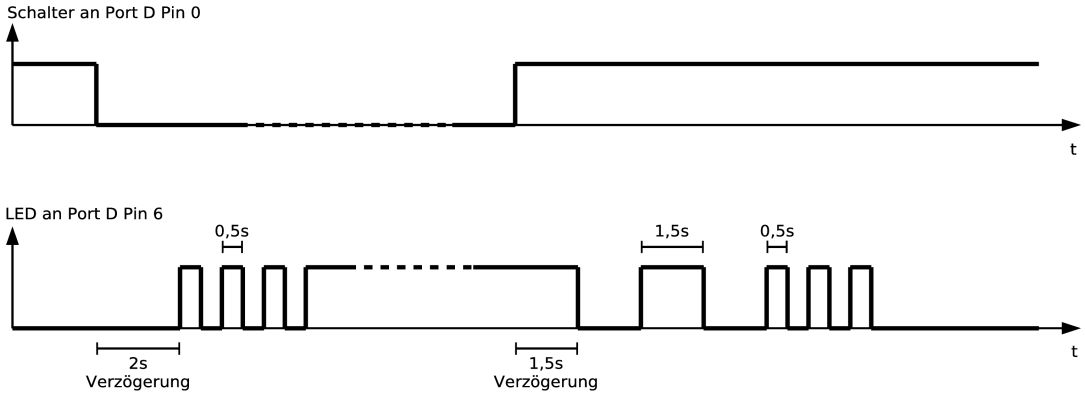
✎ **A507** a) Zeichne ein Flussdiagramm und schreibe ein Programm, welches 4 verschiedene Rechtecksignale ausgeben kann. Die Frequenz wird über zwei Schalter nach folgender Vorgabe eingestellt:

0b00 → 1 Hz
0b01 → 10 Hz,
0b10 → 100 Hz,
0b11 → 500 Hz

Die Grundzeit der Zeitschleife soll 0,2 ms betragen. Kontrolliere das Ergebnis mit dem Oszilloskop und kommentiere die Genauigkeit. Speichere das Programm als "**A507_frequency_generator_2.asm**".

b) Für Fleißige:
Bei hohen Frequenzen kann die zur Schalterabfrage benötigte Zeit die Genauigkeit der Frequenz beeinflussen. Ändere das Programm so um, dass die Schalter nur ein mal pro Sekunde abgefragt werden. Speichere das Programm als "**A507_frequency_generator_3.asm**".

✎ **A508** Schreibe ein Assemblerprogramm, das eine LED nach folgendem Muster ein- bzw. ausschaltet. Speichere das Programm als "**A508_pulse_delay_1.asm**"


Schalter an Port D Pin 0
t
LED an Port D Pin 6
0,5s
1,5s
0,5s
2s
Verzögerung
1,5s
Verzögerung
t