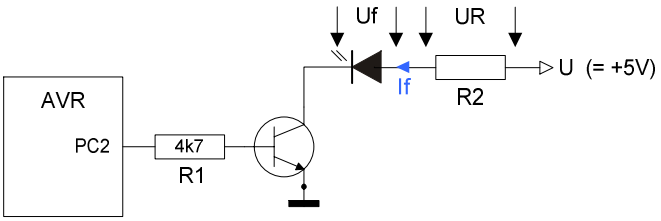
LED-Blitzlicht mit BASCOM

DL6GL, 12.05.2023

Ab und zu braucht man statt einer einfach nur leuchtenden LED ein etwas auffälligeres Blinklicht. Die primitivste Lösung sind Waits mit abwechselndem LED AN und LED Aus. Ein ernsthafterer Programmierer macht das dann doch lieber mit einem Counter in einer Interruptroutine und lässt den LED-Port toggeln. Schon besser, aber nicht auffällig genug. Wie wäre es mit einer aufblitzenden LED, wie etwa bei Flugzeugen? Ist nur unwesentlich aufwendiger als ein Toggeln.



Bei 5V Spannung wäre eine normale 20mA-LED mit R2 = 220R gut versorgt. Mit R2 = 100R oder ggf. weniger kann sie heller, aber nur für kürzere Zeit leuchten.

R2 berechnet sich aus

$$R2 = \frac{U - Uf}{If}, \quad If = \frac{U - Uf}{R2}$$

Für eine rote 20mA-LED mit einem Vorwärts-(Forward-)Spannungsabfall Uf = 1,7V an U = 5V wird R2 = 220Ω für Dauerbetrieb. If avg (average) wäre 15mA.

Richtwerte für Uf:	LED rot	~1,7V,
	LED gelb	~2,2V
	LED grün	~2,2V
	LED blau	~3,0V

Mit einer **Impulsansteuerung** kann die LED kurzzeitig höher mit einem If peak belastet werden. Der ist in manchen Datenblättern angegeben, Anhaltswert: ca. 100mA für 20mA-LED.

Mit einer Frequenz f (Hz = 1/sec) und einer Impulsbreite τ (sec) wird der Dutycycle

$$D = 100 * \frac{1}{f} * \tau \ (\%), \qquad f\ [Hz], \tau\ [\mathrm{sec}]$$

Beispiel: Für eine Blinkfrequenz 1Hz und Impulsdauer 50msec ist D = 5%.
Mit R2 = 100Ω wäre dann bei einer roten 20mA-LED If peak = 34mA.
Das Blitz-Blinken ist auffällig, die o.a. 100mA sind noch nicht erreicht.

Programmschnipsel hierzu für 16MHz Clock-Frequenz:

```
DDRC = &B0000_0100                              'PC0...PC7 input except PC2
PORTC = &B1111_1011                             'PC0...PC7 pullup except PC2
AL_Out Alias PORTC.2                            'Alarm output (LED)

Dim bitLED As Bit
Dim AL_Act As Bit
Dim bytLED_runs As Byte

'Configure Timer0-Interrupt (Timer0 = 8bit, 2^8=256 counts) -----------------
'Overflow time: Overflow-Counts * Prescale / crystal frequency
'Possible Prescales: 8, 64, 256, 1024, here 1024
'Timer0 overflow counts are reduced by PresetTimer0=100 to 156
'= 156 * 1,024 / 16,000,000 = 9.98 msec = ~ 100 Hz (crystal 16 MHz)
Config Timer0 = Timer , Prescale = 1024
Const Presettimer0 = 100                        'Gives 100 timer0 counts/sec
On Timer0 Timer0_isr                            'On Overflow go to Timer0_isr

bitAl_Act = 0                                   'Alarm OFF
AL_Out = 0                                      'Alarm LED OFF

Do

   'Do something usefull here
   'One result might be an alarm

   If bitAl_Act = 1 Then                        'Alarm is active
      AL_Out = bitLED                           'Alarm ON: LED flashing
   Else
      AL_Out = 0                                'Alarm LED OFF
   End If

Loop

'===============================================================================
Timer0_isr:
'Interrupt Service Routine for Timer0

   Timer0 = Presettimer0

   'LED Flashing (Timer0 rate ~ 10msec)
   If bytLED_runs < 101 Then                    '~1 sec interval
      Incr bytLED_runs
      If bytLED_runs < 6 Then                   '~50msec interval
         bitLED = 1
      Else
         bitLED = 0
      End If
   Else
      bytLED_runs = 0                           'Reset counter
   End If

Return
```