Es gibt eine neuere Version des Kochbuchs.
Ihr findet sie unter
http://www.reaktivlicht.de/atmel.html

Reaktives Licht mit Atmel AVR

nach einem Thread im Forum www.geoclub.de

Zusammengeschrieben von Ralf
ralf.pongratz@gmx.de

Stand 11.07.09 18:21:10

Inhaltsverzeichnis

0 Warnhinweise

Dieses Kochbuch erhebt keinen Anspruch auf Vollständigkeit und Richtigkeit. Jeder, der die hier angegebenen Schaltungen nachbaut und programmiert, tut dies auf eigene Verantwortung. Was vielen wahrscheinlich nicht klar ist: Ihr bastelt an einem offenen PC rum. Eure Schaltung ist direkt mit der Hauptplatine verbunden. Von der Einkopplung von Störungen und unangepassten Leitungswiderständen abgesehen (ist bei diesen älteren Schnittstellen nicht so wild) kann ein Kurzschluss zwischen einer Signal- und Masseleitung der Parallelen Schnittstelle ernsthaftere Folgen für den Rechner haben. Also sollte jeder, der keine Erfahrungen im Gebiet der Elektrotechnik hat, im eigenen Interesse sich zuerst eine Tasse Tee und einen bequemen Sessel besorgen und das Kochbuch komplett durchlesen. Die meisten Probleme lösen sich dadurch automatisch und für den Rest sind eine Menge nette Leute im Forum zu finden, die jede Frage beantworten.

1 Prinzip der Lichtmessung mit Hilfe einer LED

Die Kapazität einer Leuchtdiode (LED) in Sperrrichtung ist abhängig von der Beleuchtung. Bei Beleuchtung ist sie klein, in dunklem Zustand groß.
Im Programm wird die Kapazität der LED regelmäßig aufgeladen und nach einer bestimmten Zeit nachgemessen, inwiefern sie sich schon entladen hat.
Durch den schmalen Abstrahlwinkel der LED ist gewährleistet, dass die Schaltung nur bei direkter Beleuchtung aktiviert wird. Als am Besten geeignete LED hat sich eine superhelle rote mit transparentem Gehäuse herausgestellt.

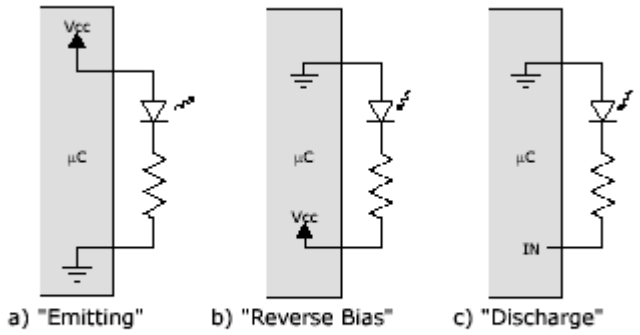

Fig. 5. Emitting and sensing light with an LED

Abbildung 1: Prinzip der Lichtmessung

2 Die Schaltung

2.1 Aufbau der Schaltung mit Lichtmessung mittels LED

In der Mitte befindet sich der Mikroprozessor ATtiny13(V). Wenn die Kerbe im Gehäuse nach links zeigt, ist links unten Pin 1. Weiter geht es gegen den Uhrzeigersinn bis zu Pin 8 links oben. Zwischen Pin 2 und 3 sind in Reihe eine Leuchtdiode LED_1 und ein passender Widerstand R_1 geschaltet. Dabei ist zu beachten, dass die Kathode der LED (kurzes Beinchen, flache Seite) zu Pin 3 zeigen muss. Der passende Wert des Vorwiderstandes R_1 kann mit dieser Formel berechnet werden:

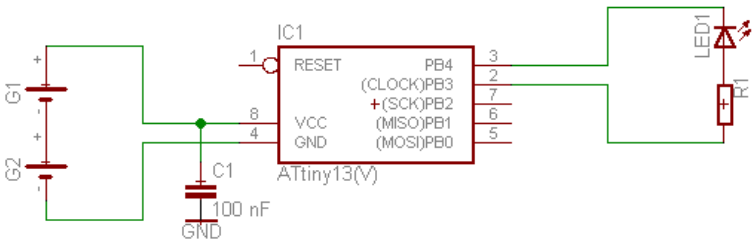

Abbildung 2: Aufbau der Schaltung

$$R_1 = \frac{U_B - U_D}{I_D}$$

Dabei ist U_B die Versorgungsspannung der Schaltung, U_D die Durchlassspannung der LED und I_D der Durchlassstrom der LED. Die Werte die LED können jedem Datenblatt entnommen werden, typisch sind hier U_D = 2 V und I_D = 20 mA. Daraus ergibt sich für U_B = 3 V für R_1 ein Wert von 50 Ohm, für U_B = 5 V ein Wert von 150 Ohm.
An Pin 8 wird die Versorgungsspannung U_B angelegt (Pluspol der Batterie), an Pin 4 die Masse (Minuspol der Batterie). Zwischen Pin 4 und 8 kommt noch ein Kondensator C_1 mit 100nF.

2.2 Aufbau mit Programmieradapter

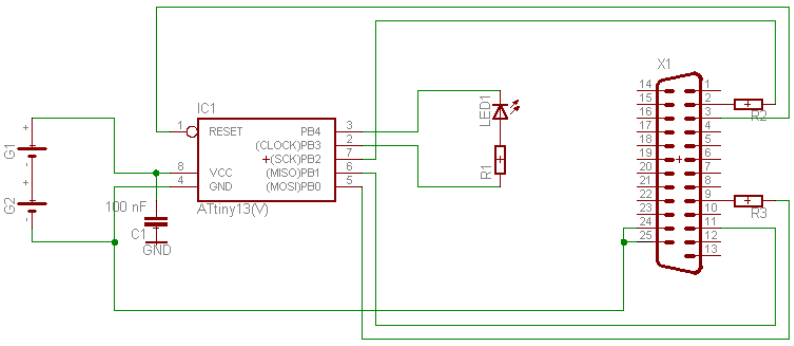

Abbildung 3: Aufbau mit Programmieradapter

Für den Programmieradapter wird ein 25-poliger SUB-D-Stecker benötigt (einfacher geht es mit der Hälfte eines alten Parallelkabels eines Druckers, die an den PC gesteckt wird). Pin 2 wird über einen Widerstand
R_2 = 220 Ohm an Pin 7 des ICs, Pin 3 an Pin 1 des ICs, Pin 9 über einen Widerstand R_3 = 220 Ohm an Pin 5 des ICs, Pin 11 an Pin 6 des ICs und Pin 24 und 25 an Pin 4 des ICs (Masse) angeschlossen.
In der hier dargestellten Schaltung ist die Grundschaltung nach Kapitel 2.1 (R_1 und LED_1) eingebaut. Dies hat den Vorteil, dass der Controller für das Programmieren und das Testen nicht jeweils umgesetzt werden muss.

2.3 Serieller Programmieradapter

Der Microcontroller kann auch über die serielle Schnittstelle programmiert werden. Abbildung 4 zeigt den dafür benötigten Programmieradapter. Programmiert werden kann mit der Software „PonyProg“. Erfahrungen zu „Bascom AVR“ liegen nicht vor. Die Baudrate wird automatisch ausgewählt. Man muss nur die COM-Schnittstelle wählen und „auto calibration“ aufrufen.

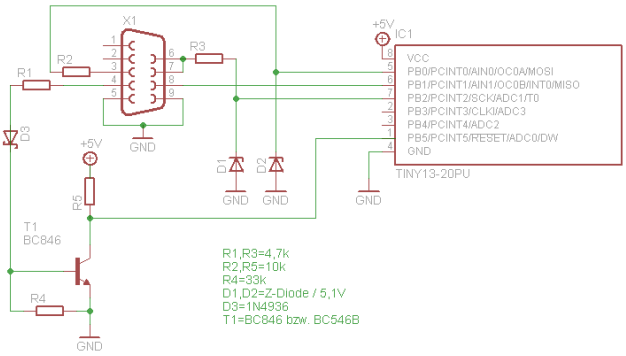

Abbildung 4: Serieller Programmieradapter

2.4 Versorgungsspannung

Als minimale Versorgungsspannung ist für den ATtiny13(V) 1,8V festgesetzt. Der ATtiny13 und der ATtiny12L benötigen mindestens 2,7V. Für Versuchszwecke reicht aber auch ein 5V-Netzteil.
Je geringer die Versorgungsspannung und je geringer die Taktfrequenz des Prozessors, desto weniger Strom wird verbraucht. So ist im normalen Betrieb ein Verbrauch im unteren µA-Bereich möglich, da viele interne Komponenten abgeschaltet werden können.

Figure 68. Active Supply Current vs. V_{CC} (Internal WDT Oscillator, 128 kHz)

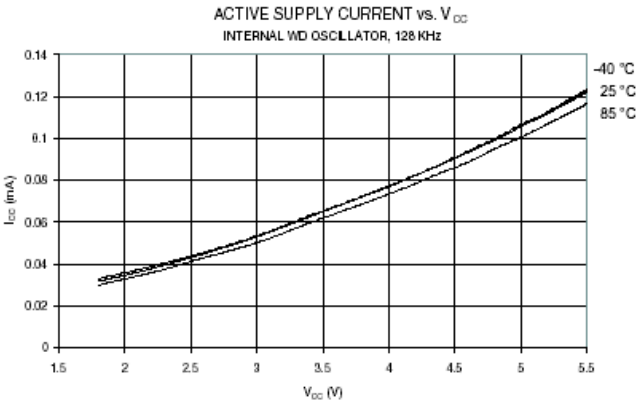

Abbildung 5: Versorgungsspannung/Stromverbrauch-Kennlinie bei 128 kHz

2.5 Variationen des Schaltplans

2.5.1 Aufbau der Schaltung mit Lichtmessung mittels eines Fotowiderstandes (LDR)

Der Einsatz eines LDR lässt eine zuverlässigere und empfindlichere Helligkeitsmessung zu, als bei der Referenzschaltung 2.1 ohne LDR. Die höhere Empfindlichkeit geht geringfügig zu lasten eines höheren Stromverbrauchs.
Der LDR und der Vorwiderstand bilden einen hochohmigen Spannungsteiler. Über PB0 wird der

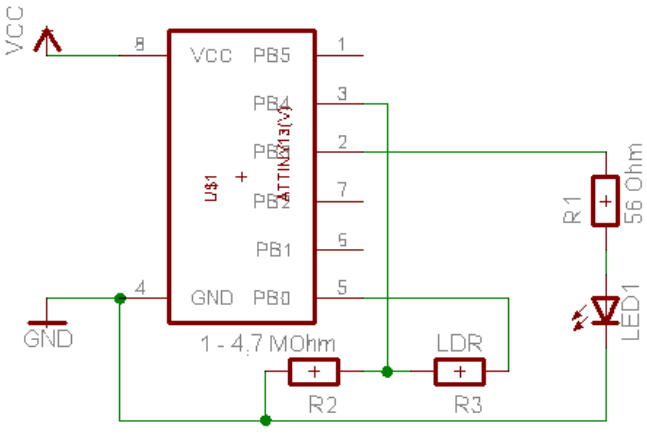

Abbildung 6: Schaltung mit LDR, Spannungsteiler abschaltbar (Programm 5.1)

Spannungsteiler immer nur dann mit Strom versorgt, wenn tatsächlich eine Messung stattfindet. Die Stromaufnahme des Spannungsteilers ist allerdings so gering, dass man eigentlich darauf verzichten kann.
Ist es dunkel, wird der LDR hochohmig und an PB4 liegt Low an. Fällt Licht auf den LDR, wird er niederohmig und es liegt High an. Die Grenze zwischen Low und High verschiebt sich proportional zur Versorgungsspannung.

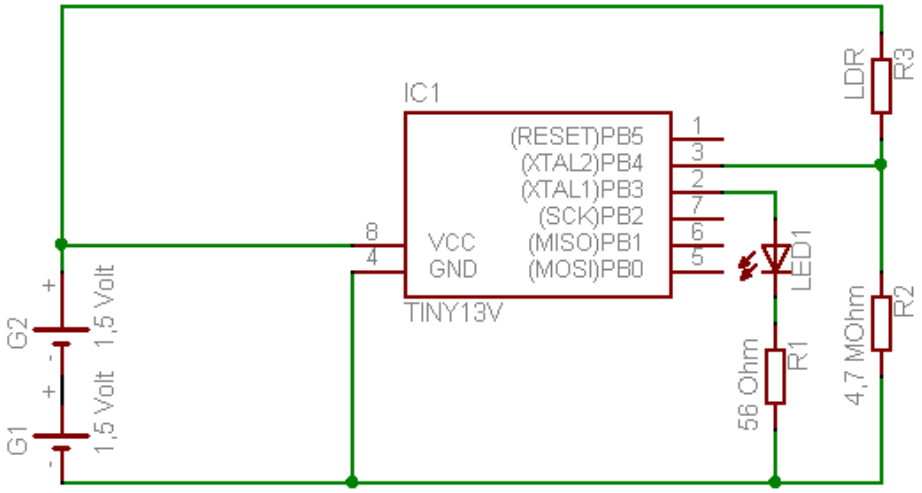

Abbildung 7: Schaltung mit LDR, Spannungsteiler nicht abschaltbar (Programm 5.2)

2.5.2 Weitere Variationen

Der Widerstand R_1 kann weggelassen werden, wenn man die LED im Leuchtbetrieb mit 100...1kHz moduliert. Allerdings kann es dann bei einer Fehlfunktion im Programm zu einer Zerstörung der LED kommen.
Durch Parallelschalten eines Kondensators zur LED kann die Zeit, die sie zur Entladung benötigt, verlängert werden, um evtl. den Prozessor zwischendurch schlafen zu lassen.

3 Programmierung

3.1 Benötigte Komponenten

Benötigt werden ein PC mit paralleler Schnittstelle (Druckerschnittstelle), der Programmieradapter mit der Schaltung und die Programmierumgebung. An Programmierumgebungen gibt es für BASIC das Bascom AVR (http://www.mcselec.com/bascom-avr.htm), für C http://winavr.sourceforge.net und für Pascal http://www.e-lab.de. Damit können dann die Programme geschrieben und auf den Mikrocontroller übertragen werden.

3.2 Einstellung der parallelen Schnittstelle

Die parallele Schnittstelle muss im BIOS auf ECP+EPP (In- und Output) eingestellt werden.
Hier eine Beispieleinstellung:
Onboard Parallel Port: 378/IRQ7
Parallel Port Mode: ECP+EPP
ECP Mode Use DMA: 3
Parallel Port EPP Type: EPP1.7

3.3 Benutzung von Bascom AVR

Hier wird beschrieben, wie der oben gezeigten Programmieradapter eingestellt und die „Fuses“, auch „Fuse-Bits“ genannt, des ATtiny13(V) gesetzt werden. Bascom sollte bereits installiert und die Schaltung angeschlossen sein. Die Screenshots basieren auf die Programmversion 1.11.8.3.

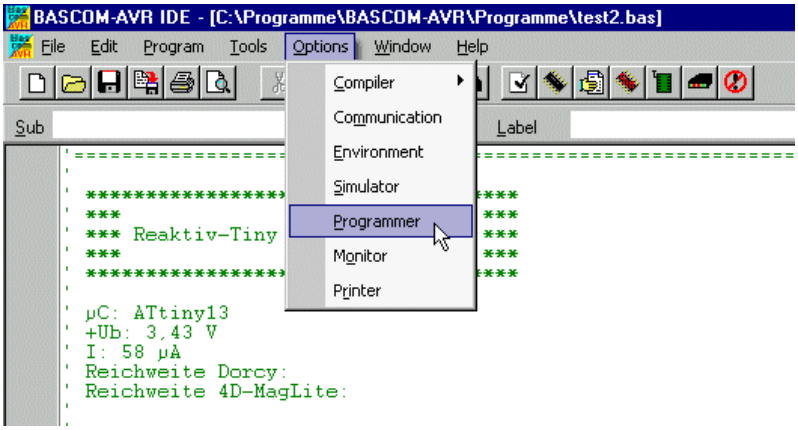

Abbildung 8: Einstellen des Programmers

Zu Beginn wird Bascom gestartet und im Hauptmenü „File“ mittels „New“ ein neues Programmfenster erstellt. Nun stellen wir den Programmieradapter ein. Dazu wählen wir zunächst im Hauptmenü unter „Options“ den Punkt „Programmer“ aus.

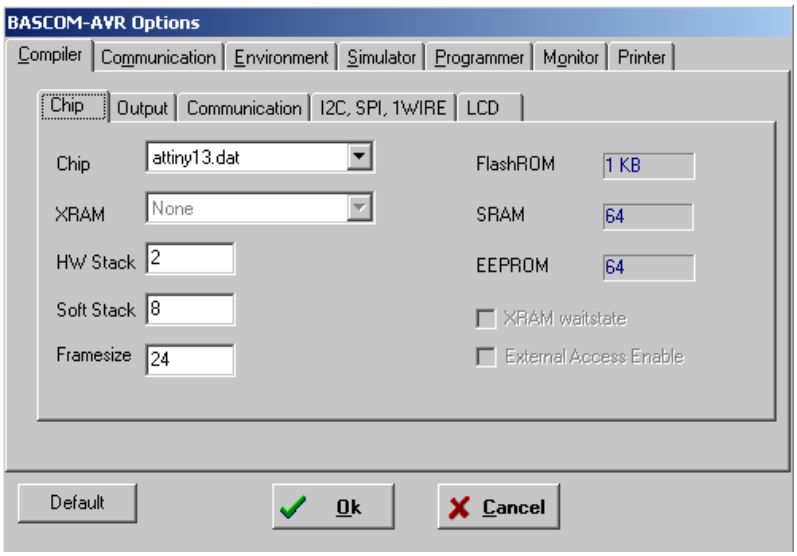

Abbildung 9: Einstellen des Chips

Nun im Hauptmenü unter Options wählen wir den Punkt „Compiler“den Eintrag „Chip“.
Bei dem Pulldownmenü „Chip“ wählen wir den attiny13.dat und ändern bei den darunter liegenden Eingabefeldern die Werte für
HW Stack = 2
Soft Stack = 8
Framesize = 24
und klicken nun einmal auf die Default Schaltfläche.

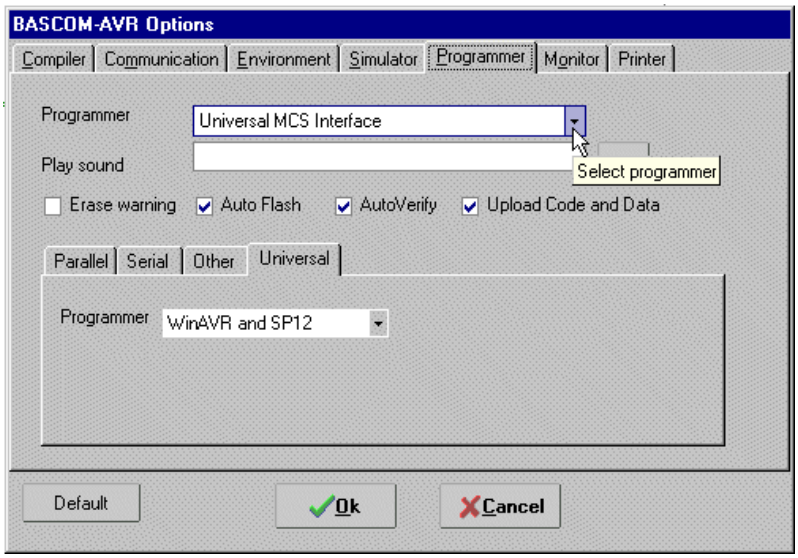

Abbildung 10: Einstellen des Programmers

Im folgenden Fenster wählen wir oben bei „Programmer“ das „Universal MCS Interface“ aus und klicken unten auf den Karteireiter „Universal“. Hier wählen wir dann als Programmer den „WinAVR and SP12“ aus.
Danach einfach auf „OK“ klicken und fertig.

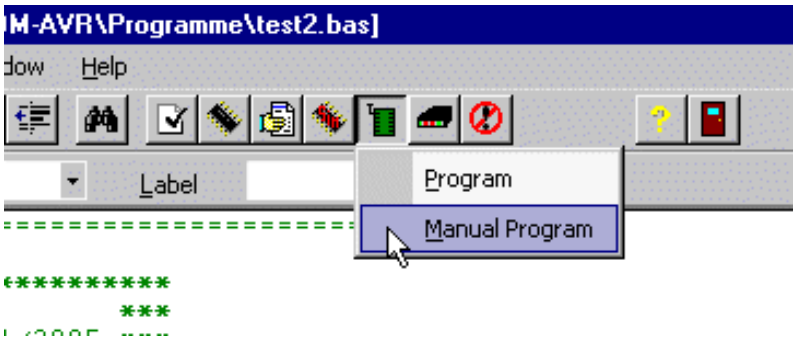

Abbildung 11: Einstellen der Fuse-Bits

Als nächstes stellen wir die Fuse-Bits ein. Die Fuse-Bits sind Speicherzellen, die das (Grund-)Verhalten des Tiny's festlegen (wir wollen ihn ja z. B. mit dem internen 128kHz-Oszillator laufen lassen). Dazu klickt ihr zunächst oben unter dem Hauptmenü auf den kleinen grünen IC-Sockel und wählt den Punkt „Manual Program“ aus.

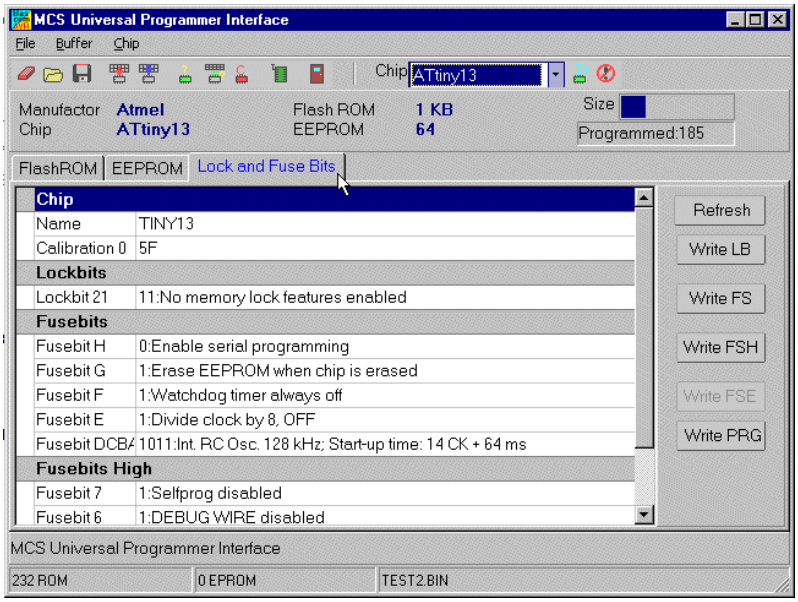

Abbildung 12: Einstellen der Fuse-Bits

Im nun folgenden Fenster klicken wir auf den Karteireiter „Lock and Fuse Bits“. Jetzt sollte Bascom die Fuse-Bits aus dem Tiny auslesen und anzeigen.

Um Veränderungen an den Fuse-Bits vorzunehmen, klicken wir einfach auf die entsprechende Zeile, in der sich das gewünschte bzw. zu ändernde Bit befindet. Wir können dann in dem sich öffnenden Auswahlmenü auswählen, welche Werte wir haben bzw. einstellen wollen. Wir stellen nun zwei Sachen sein: „Interner 128 kHz-Oszillatorbetrieb“ und „Taktverhältnis beim Starten 1:1“. Dazu klicken wir zunächst auf das „Fusebit DCBA“ und stellen es auf „1011:Int. RC Osc. 128 kHz; Start-up time: 14 CK + 64 ms“. Dann klicken wir auf das „Fusebit E“ und stellen dort „1:Divide clock by 8, OFF“ ein.

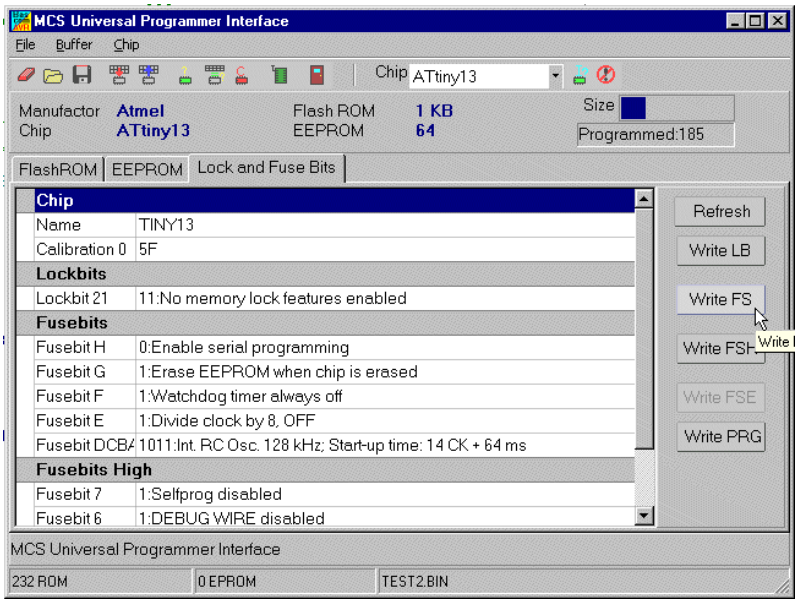

Abbildung 13: Einstellen der Fuse-Bits

Bitte kontrolliert diese Einstellungen jetzt alle nochmal sehr genau! Wenn wir an dieser Stelle nämlich einen Fehler machen, kann es sein, dass der Tiny nach dem nächsten Klick unwiderruflich hin ist. Ist alles so weit richtig, dann klicken wir jetzt rechts auf „Write FS“. Bascom schreibt nun die aktuellen, geänderten Fuse-Bit-Einstellungen in den Tiny.

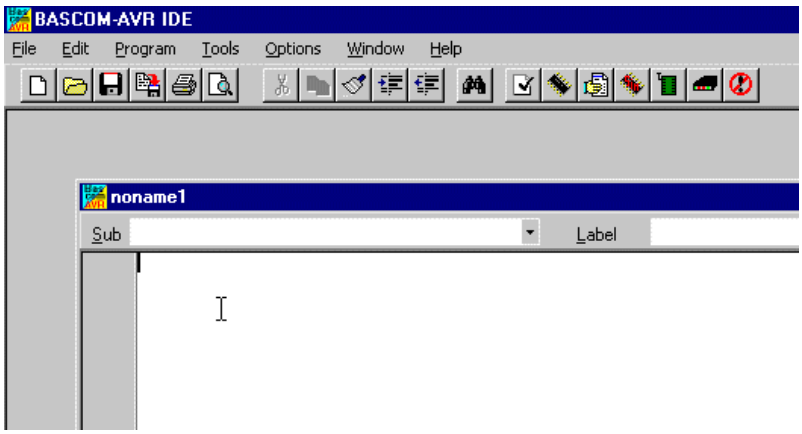

Abbildung 14: Programmierung

So, fertig. Jetzt geht’s ans eigentliche Programmieren. Als erstes wählen wir oben im Hauptmenü unter „File“ den Punkt „New“ aus.
Es öffnet sich ein Fenster, in dem das Programm geschrieben wird.

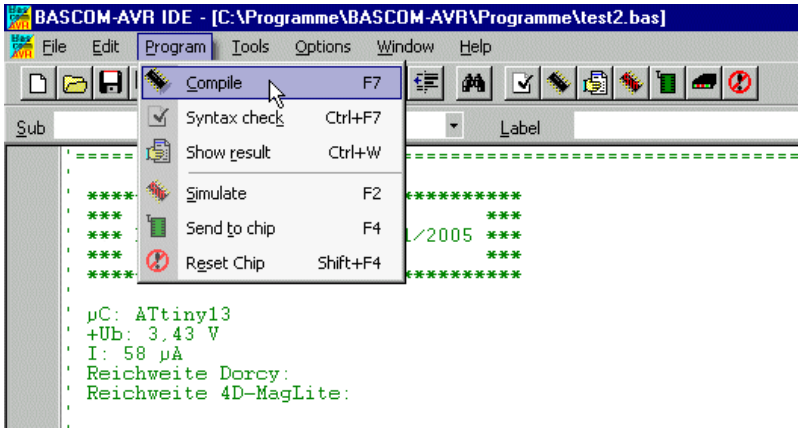

Abbildung 15: Programmierung

Wenn alles soweit fertig ist, muss das Programm zunächst kompiliert werden. Dazu klicken wir im Hauptmenü unter „Programm“ einfach auf „Compile“. Der Compiler startet nun und sollte im Idealfall nicht meckern.

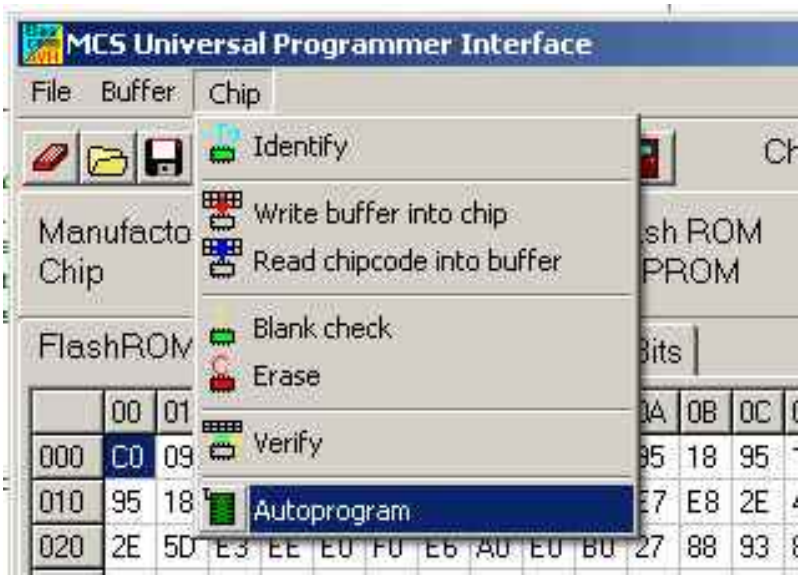

Abbildung 16: Programmierung

Ist das auch erledigt, starten wir jetzt die Programmübertragung zum Tiny. Dazu klicken wir, wieder im Hauptmenü unter „Programm“, auf den Punkt „Send to chip“. In dem sich dann öffnenden Fenster unter „Chip“ auf „Autoprogram“ klicken. Das zuvor kompilierte Programm wird nun in den Speicher unseres Tiny geladen.

4 Programme für Schaltung mit Helligkeitsmessung über LED

4.1 Grundprogramm

Dieses Programm gibt zehn kurze Lichtblitze zurück wenn die LED angeleuchtet wird.

```
'============================================================
'
' ***********************************
' ***                             ***
' *** Reaktiv-Tiny v0.1 20/11/2005 ***
' ***                             ***
' ***********************************
'
' µC: ATtiny13
' +Ub: 3,43 V
' I: 58 µA
' Reichweite Dorcy:
' Reichweite 4D-MagLite:
'
'============================================================
'
$regfile = "ATtiny13.DAT"
$crystal = 113000               'Reale Frequenz des internen 128kHz-Oszillators
'
Config Portb = &B00011000       'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111              'Pullups zuschalten, außer für Pinb.3 und .4
'
Stop Adc                        'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                         'Analog-Komparator abschalten, um Strom zu sparen
'
Dim A As Byte
Dim Hell_dunkel As Bit
'
Do
  Gosub Led_abfrage
  '
  If Hell_dunkel = 0 Then
    For A = 1 To 10
      Portb.3 = 1
      Waitms 50
      Portb.3 = 0
      Waitms 500
    Next A
  End If
'
Loop
'
Led_abfrage:
Portb.3 = 0                     'Portb.3 auf Masse schalten
Portb.4 = 1                     'Portb.4 auf +Ub schalten, um die LED zu 'laden'
Waitus 1                        'Ladezeit 1 µs, kann ggf. noch verkleinert werden
Config Portb.4 = Input          'Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
Portb.4 = 0                     'Pullup abschalten, sonst geht's nicht!
Waitus 1500                     'Entladezeit 1500 µs - je kleiner, je unempfindlicher
Hell_dunkel = Pinb.4            'Ladezustand einlesen
Config Portb.4 = Output         'Portb.4 wieder auf Ausgang schalten
Portb.4 = 0                     'Portb.4 auf Masse schalten
Return
'
End
```

4.2 Grundprogramm in C

```
#define F_CPU  8000000        // Quarzfrequenz 8MHz

#include <avr/io.h>
#include <avr/delay.h>


unsigned char f, i;
unsigned char led_abfrage(unsigned char zeit) {

  PORTB &= ~(1<<PB3);         //  Portb.3 auf Masse schalten
  PORTB |=  (1<<PB4);         //  Portb.4 auf +Ub schalten, um die LED zu 'laden'
  _delay_us(10);              // Ladezeit 10 (1) µs, kann ggf. noch verkleinert werden

  DDRB  &= ~(1<<PB4);         // Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
  PORTB &= ~(1<<PB4);         // Pullup abschalten, sonst geht's nicht!

  _delay_ms(zeit);            // Entladezeit zeit_ms (1500 µs) - je kleiner, je unempfindlicher

  i = (PINB & (1<<PINB4));    // Ladezustand einlesen

  DDRB  |=  (1<<PB4);         // Portb.4 wieder auf Ausgang schalten
  PORTB &= ~(1<<PB4);         // Portb.4 auf Masse schalten

  return i;
}

int main(void) {
  DDRB  = 0b00011000;         // Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
  PORTB = 0b11100111;         // Pullups zuschalten, außer für Pinb.3 und .4

  while (1) {
    if ( led_abfrage(6) == 0) {          // LED durch Licht entladen?
      for (f = 0; f < 10 ; f++) {        // 10x blinken
        PORTB |=  (1<<PB3);               // PB3 auf Vcc schalten
        _delay_ms(32);                    // 32 ms Blitz
        PORTB &= ~(1<<PB3);               // PB3 auf GND schalten
        for (i = 0; i < 10 ; i++)
          _delay_ms(32);      // ca. 320ms Pause (workaround wegen 8MHz Quarz)
      }
    }
  }
  return 0;
}
```

4.3 Nachtaktiver Blinker

Da die Grundschaltung unabhängig von der Tageszeit reagiert (also nur einen Helligkeitsdetektor darstellt), hier eine Verbesserung, die nur nachts reagiert.
Nur kurze Lichtimpulse auf die LED führen zum Blinken, langandauernde nicht.

```
'============================================================
'
' ***********************************
' ***                             ***
' *** Reaktiv-Tiny v0.1 24/11/2005 ***
' ***                             ***
' ***********************************
'
' µC: ATtiny13
' +Ub: 3,43 V
' I: 58 µA
' Reichweite Dorcy:
' Reichweite 4D-MagLite:
'
'============================================================
$regfile = "ATtiny13.DAT"
$crystal = 113000                     'Reale Frequenz des internen 128kHz-Oszillators

Config Portb = &B00011000             'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111                    'Pullups zuschalten, außer für Pinb.3 und .4
```

```
Stop Adc                                 'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                                  'Analog-Komparator abschalten, um Strom zu sparen

Dim A As Byte
Dim B As Byte
Dim Led_ladezustand As Bit
Dim Hell As Bit

Do
  Gosub Led_abfrage
  If Led_ladezustand = 0 Then Hell = 1   'Bei Licht Merker setzen
  If Hell = 1 And B < 255 Then B = B + 1 'Wenn Merker gesetzt wurde, Zähler für Licht-
                                         'dauer erhöhen (bis max. 255)
  If Led_ladezustand = 1 And Hell = 1 And B < 50 Then Gosub Blinken      'Wenn es wieder dunkel
                                         'ist und der Lichtimpuls nur kurz war, dann blinken
  If Led_ladezustand = 1 Then            'Bei Dunkelheit Merker und Zähler für Lichtdauer
    Hell = 0                             'löschen
    B = 0
  End If
Loop

Led_abfrage:
  Portb.3 = 0                            'Portb.3 auf Masse schalten
  Portb.4 = 1                            'Portb.4 auf +Ub schalten, um die LED zu 'laden'
  Waitus 1                               'Ladezeit 1 µs, kann ggf. noch verkleinert werden
  Config Portb.4 = Input                 'Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang
                                         'schalten
  Portb.4 = 0                            'Pullup abschalten, sonst geht's nicht!
  Waitms 20                              'Entladezeit 20 ms - je kleiner, je unempfindlicher
  Led_ladezustand = Pinb.4               'Ladezustand einlesen: '1' -> dunkel, '0' -> hell
  Config Portb.4 = Output                'Portb.4 wieder auf Ausgang schalten
  Portb.4 = 0                            'Portb.4 auf Masse schalten
Return

Blinken:
  For A = 1 To 10
    Portb.3 = 1
    Waitms 50
    Portb.3 = 0
    Waitms 500
  Next A
Return

End
```

4.4 Verbesserter nachtaktiver Blinker

Vermeidet, dass bei ungünstiger Beleuchtung dauernd geblinkt wird.
Nachdem ein Lichtwechsel Dunkel-Hell-Dunkel erkannt wurde wird während der abschließenden Dunkel-Phase mehrfach hintereinander der LED-Ladezustand geprüft. Nur wenn er bei den nächsten fünf Zyklen ebenfalls immer dunkel ist wird der Blinker ausgelöst. Gewählt wurde eine hohe Empfindlichkeit und ein Abtastzyklus von 100 ms. Deshalb wird nur auf fünf Zyklen getestet, sonst geht der Blinker zu spät an. Bei einem kürzeren Abtastzyklus (20 ms) kann man die Zyklusanzahl auf 25 stellen.

```
'============================================================
'
' ***********************************
' ***                             ***
' *** Reaktiv-Tiny v0.1 24/11/2005 ***
' ***                             ***
' ***********************************
'
' µC: ATtiny13
' +Ub: 3,43 V
' I: 58 µA
' Reichweite Dorcy:
' Reichweite 4D-MagLite:
'
'============================================================
```

```
$regfile = "ATtiny13.DAT"
$crystal = 113000                       'Reale Frequenz des internen 128kHz-Oszillators

Config Portb = &B00011000               'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111                      'Pullups zuschalten, außer für Pinb.3 und .4

Stop Adc                                'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                                 'Analog-Komparator abschalten, um Strom zu sparen

Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim Led_ladezustand As Bit
Dim Hell As Bit
Dim Dunkel As Bit

Do
  Gosub Led_abfrage
  If Led_ladezustand = 0 Then Hell = 1        'Bei Licht Merker setzen
  If Hell = 1 And B < 255 Then B = B + 1      'Wenn Merker gesetzt wurde, Zähler für Lichtdauer
                                              'erhöhen (bis max. 255)
  If Led_ladezustand = 1 And Hell = 1 And B < 30 Then    'Wenn es wieder dunkel ist dann
                                              'nachsehen für wie lange
    Dunkel = 1                                'Merker Setzen
    For C = 0 To 5                            'LED-Zustand mehrmals abfragen
      Gosub Led_abfrage
      If Led_ladezustand = 0 Then Dunkel = 0  'Wenn wieder Hell dann Dunkel-Merker löschen
    Next C
    If Dunkel = 1 Then Gosub Blinken           'Wenn Dunkel-Merker gesetzt blinken
  End If
  If Led_ladezustand = 1 Then        'Bei Dunkelheit Merker und Zähler für Lichtdauer löschen
    Hell = 0
    B = 0
  End If
Loop

Led_abfrage:
  Portb.3 = 0                   'Portb.3 auf Masse schalten
  Portb.4 = 1                   'Portb.4 auf +Ub schalten, um die LED zu 'laden'
  Waitus 1                      'Ladezeit 1 µs, kann ggf. noch verkleinert werden
  Config Portb.4 = Input        'Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
  Portb.4 = 0                   'Pullup abschalten, sonst geht's nicht!
  Waitms 100                    'Entladezeit 100 ms - je kleiner, je unempfindlicher
  Led_ladezustand = Pinb.4      'Ladezustand einlesen: '1' -> dunkel, '0' -> hell
  Config Portb.4 = Output       'Portb.4 wieder auf Ausgang schalten
  Portb.4 = 0                   'Portb.4 auf Masse schalten
Return
Blinken:
  For A = 1 To 10
    Portb.3 = 1
    Waitms 10
    Portb.3 = 0
    Waitms 100
  Next A
Return

End
```

4.5 Nachtaktiver Blinker in C

```
#define F_CPU  128000      // Oszillatorfrequenz in Hz

#include <avr/io.h>
#include <avr/delay.h>

#define LED_A PB4          // LED Pin Anode
#define LED_K PB3          // LED Pin Kathode

#define DISCHG 150         // Entladezeit der LED in ms. Je größer der Wert, desto empfindlicher.
#define DARKCNT 16         // Verhindert blinken bei Dauerbeleuchtung (Tag).
                           // Je größer der Wert, desto unempfindlicher gegen Taglicht.
```

```
unsigned char led_abfrage(void) {
  unsigned char i;

  PORTB &= ~(1<<LED_A);    // Port LED_A auf Masse schalten
  PORTB |=  (1<<LED_K);    // Port LED_K auf Vcc schalten, um die LED zu 'laden'

  _delay_ms(1);            // Ladezeit 1ms

  DDRB  &= ~(1<<LED_K);    // Port LED_K nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
  PORTB &= ~(1<<LED_K);    // Pullup abschalten

  _delay_ms(DISCHG);       // Entladezeit abwarten

  if (PINB & (1<<LED_K))   // Ladezustand einlesen
    i=0;
  else
    i=1;

  DDRB  |=  (1<<LED_K);    // Port LED_K wieder auf Ausgang schalten
  PORTB &= ~(1<<LED_K);    // Port LED_K auf Masse schalten
  return i;
}


int main(void) {
  unsigned char k;
  DDRB  = (1<<LED_A) | (1<<LED_K);         // LED_A und LED_K als Ausgang definieren
  PORTB = 0xFF ;                           // Pullups zuschalten
  PORTB &= ~((1<<LED_A)|(1<<LED_K));       // Pullups für LED abschalten
  OSCCAL = 0;                              // Oszillator auf geringste Taktfrequenz trimmen
  ACSR = (1<<ACD);                      // analogen Komperator ausschalten, spart Strom

  while (1) {                           // Main Loop

    k = 0;

    while (k < DARKCNT) {               // nur wenn k-mal hintereinander "dunkel" erkannt wurde
                                        // gehts weiter und die Schaltung ist "scharf"
      if (led_abfrage())
        k = 0;
      else
        k++;
    }

    while (!(led_abfrage())) {          // hier ist die Schaltung scharf und wartet auf Licht
    }

    for (k = 0; k < 30 ; k++) {         // ab hier beginnt das Blinkspektakel
      PORTB |=  (1<<LED_A);
      _delay_ms(60);
      PORTB &= ~(1<<LED_A);
      _delay_ms(80);
    }
  }
  return 0;
}
```

4.6 Verbesserter nachtaktiver Blinker mit Teach-In-Modus

Das Programm (siehe Anhang) tut nun, nach Anlegen der Betriebsspannung oder einem Reset, folgendes:

- 10 x kurz blinken, 1 x lang blinken (Teach-In-Anfang)
- Teach-In-Sequenz aufzeichnen (-> EEPROM)
- 3 x kurz blinken (Teach-In-Ende)
- Hauptschleife: Blinken (Standart- oder Teach-In-Sequenz) nach Auslösung durch kurze Lichtimpulse (Windi-Filter v1)

Die Teach-In-Sequenz (Aufzeichnung und Wiedergabe) dauert, wie die bisherige normale Blinksequenz, ungefähr fünf Sekunden. Das, was man während der Lernphase per Taschenlampe eingibt (Abtastinterval ca. 80 ms), wird später 1:1 wieder ausgegeben. Wird die LED während der gesamten Te-

ach-In-Phase jedoch nicht beleuchtet, dann wird zum Blinken einfach die Standard-Blinksequenz genommen. Wer also keine Lust auf Teach-In hat, läßt die LED während der Aufzeichnung einfach dunkel...
Für jeden einzelnen Abtastwert wird ein Byte zum Speichern genutzt. Mittels Bit-Rotation wäre es also möglich, die achtfache Zeit zu speichern.

```
'============================================================
'
' ***********************************
' ***                             ***
' *** Reaktiv-Tiny v0.1 28/11/2005 ***
' ***                             ***
' ***********************************
'
' µC: ATtiny13
' +Ub: 3,43 V
' I: 58 µA
' Reichweite Dorcy:
' Reichweite 4D-MagLite:
'
'============================================================

$regfile = "ATtiny13.DAT"
$crystal = 113000                       'Reale Frequenz des internen 128kHz-Oszillators

Config Portb = &B00011000               'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111                      'Pullups zuschalten, außer für Pinb.3 und .4

Stop Adc                                'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                                 'Analog-Komparator abschalten, um Strom zu sparen

Dim A As Byte
Dim B As Byte
Dim Led_ladezustand As Bit
Dim Led_lichtcode As Byte
Dim Lichtcode_valid As Bit
Dim Hell As Bit

Lichtcode_valid = 0             'Merker löschen (EEPROM-Inhalt wird beim Blinken nicht verwendet)
Gosub Blinken                           '10 x kurz blinken
Portb.3 = 1
Wait 1                                  '1 x lang blinken (Teach-In-Anfang)
Portb.3 = 0
For A = 1 To 63                         '63 Bytes des EEPROM's haben wir zur Verfügung
  Gosub Led_abfrage
  If Led_ladezustand = 0 Then           'Wenn LED beleuchtet wurde, dann
    Led_lichtcode = 255                 'Lichtcode-Byte setzen
    Lichtcode_valid = 1                 'Merker setzen (EEPROM-Inhalt wird beim Blinken verwendet)
  Else                                  'andernfalls
    Led_lichtcode = 0                   'Lichtcode-Byte löschen
  End If
  Writeeeprom Led_lichtcode , A         'Lichtcode-Byte in's EEPROM schreiben
  Waitms 65                             '65 ms warten
Next A
For A = 1 To 3
  Portb.3 = 1
  Waitms 50                             '3 x kurz blinken (Teach-In-Ende)
  Portb.3 = 0
  Waitms 50
Next A

Do
Gosub Led_abfrage
  If Led_ladezustand = 0 Then Hell = 1      'Bei Licht Merker setzen
  If Hell = 1 And B < 255 Then B = B + 1    'Wenn Merker gesetzt wurde, Zähler für Lichtdauer
                                            'erhöhen (bis max. 255)
  If Led_ladezustand = 1 And Hell = 1 And B < 50 Then Gosub Blinken  'Wenn es wieder dunkel ist
                                            'und der  Lichtimpuls nur kurz war, dann blinken
  If Led_ladezustand = 1 Then           'Bei Dunkelheit Merker und Zähler für Lichtdauer löschen
    Hell = 0
    B = 0
  End If
Loop
```

```
Led_abfrage:
  Portb.3 = 0                  'Portb.3 auf Masse schalten
  Portb.4 = 1                  'Portb.4 auf +Ub schalten, um die LED zu 'laden'
  Waitus 1                     'Ladezeit 1 µs, kann ggf. noch verkleinert werden
  Config Portb.4 = Input       'Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
  Portb.4 = 0                  'Pullup abschalten, sonst geht's nicht!
  Waitms 20                    'Entladezeit 20 ms - je kleiner, je unempfindlicher
  Led_ladezustand = Pinb.4     'Ladezustand einlesen: '1' -> dunkel, '0' -> hell
  Config Portb.4 = Output      'Portb.4 wieder auf Ausgang schalten
  Portb.4 = 0                  'Portb.4 auf Masse schalten
Return

Blinken:
  Portb.3 = 0
  Portb.4 = 0
  If Lichtcode_valid = 0 Then  'Standard-Blinksequenz (ohne EEPROM-Inhalt)
    For A = 1 To 10
      Portb.3 = 1
      Waitms 50
      Portb.3 = 0
      Waitms 500
    Next A
  Else                          'Blinksequenz mit EEPROM-Inhalt
    For A = 1 To 63
      Readeeprom Led_lichtcode , A
      If Led_lichtcode = 255 Then
        Portb.3 = 1
      Else
        Portb.3 = 0
      End If
      Waitms 90
    Next A
  End If
Return

End
```

4.7 Schreibschutz für Teach-In-Modus

Zur Programmierung werden die Ports B1 und B2 kurzgeschlossen. Achtung: Ungetestet!

```
Check_write_prot:
  Dim eeprom_write_prot as bit
  Config Portb.1 = Input           'Schalte Portb.1 als Eingang
  Portb.2 = 1                      'setze Portb.2 auf VCC
  eeprom_write_prot = Pinb.1       'eeprom_write_prot enthält bei aufgehobenen schreibschutz 1,
                                   'ansonsten 0
  Config Portb.1 = Output          'Portb.1 zurücksetzen
  Portb.2 = 0                      'Portb.2 auf Masse setzen
Return
```

4.8 Nachtaktiver Blinker mit Teach-In-Modus mit Lauflängenspeicherung

Das Programm ist sehr ähnlich zu 4,6, allerdings ohne Default-Blinksequenz.

- 10 x kurz blinken (Teach-In-Anfang)
- Teach-In-Sequenz aufzeichnen (-> EEPROM)
- 3 x kurz blinken (Teach-In-Ende)
- Hauptschleife: Blinken der Teach-In-Sequenz nach Auslösung durch kurze Lichtimpulse (Windi-Filter v1)

Das Besondere ist, dass die Blinksequenz lauflängencodiert gespeichert wird. In Bit 1 des EEPROMs wird die LED-Ladezustand zu Beginn der Sequenz hinterlegt. In den nachfolgenden Bits wird nun die Länge der folgenden Intervalle gespeichert. Nachdem ein gespeichertes Intervall abgearbeitet wurde, wird der Zustand der LED invertiert. Bei einer Abtastfrequenz von 12,5 Hz ist also eine Gesamtspeicherdauer von 20 Minuten möglich. Allerdings ist durch die Lauflängencodierung die Speicherung von nur maximal 62 Helligkeitszuständen möglich, wodurch die Speicherzeit deutlich reduziert wird. Zustände, die länger als 254 Abtastzyklen dauern, belegen dabei mehrere Bytes, die maximale Anzahl

verringert sich entsprechend (je 254 Zyklen ein Zustand).

```
'============================================================
'
' ***********************************
' ***
' *** Reaktiv-Tiny
' *** Teach-In mit Lauflängencodierung
' *** Ralf Pongratz
' *** type_joti@yahoo.de
' *** Version 060401 1545
' ***
' ***********************************
'
' µC: ATtiny13
'
'============================================================
$regfile = "ATtiny13.DAT"
$crystal = 113000 'Reale Frequenz des internen 128kHz-Oszillators
$hwstack = 6
Config Portb = &B00011000 'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111 'Pullups zuschalten, außer für Pinb.3 und .4
Stop Adc 'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                          'Analog-Komparator abschalten, um Strom zu sparen

Dim A As Byte
Dim B As Byte
Dim Led_ladezustand As Bit
Dim Led_lichtcode As Byte
Dim Hell As Bit
Dim Last As Bit                  'Speichern des letzten Helligkeitswertes
Dim Count As Byte                'Speichern der Lauflänge
Dim I As Byte                    'Zählvariable

For I = 1 To 10
  Portb.3 = 1
  Waitms 50                      '10 x kurz blinken (Teach-In-Anfang)
  Portb.3 = 0
  Waitms 250
Next I

Gosub Led_abfrage
Last = Led_ladezustand
B = 0
B = Bits(last)
Writeeeprom B , 1                'Speichern der Starthelligkeit an Byte 1 im EEPROM

Count = 1
Waitms 65
Gosub Led_abfrage
For A = 2 To 63                  '62 Bytes des EEPROM's haben wir zur Verfügung für die Lauflängen
   While Last = Led_ladezustand And Count < 254             'Längen werden gezählt
      Count = Count + 1
      Waitms 65
      Gosub Led_abfrage
   Wend
   If Count = 254 Then    '254 Längen erreicht -> Abspeichern 255 -> weiter mit gleichem LED-Zustand
      Count = 255
   Else                   'LED-Zustand.Wechsel -> Abspeichern der Länge, Invertierung von Last
      Last = Not Last
   End If
   Writeeeprom Count , A
   Count = 0
Next A

For A = 1 To 3
   Portb.3 = 1
   Waitms 150                     '3 x kurz blinken (Teach-In-Ende)
   Portb.3 = 0
   Waitms 150
Next A
```

```
Do
   Gosub Led_abfrage
   If Led_ladezustand = 0 Then Hell = 1     'Bei Licht Merker setzen
   If Hell = 1 And B < 255 Then B = B + 1   'Wenn Merker gesetzt wurde, Zähler für Lichtdauer
                                            'erhöhen (bis max. 255)
   If Led_ladezustand = 1 And Hell = 1 And B < 50 Then Gosub Blinken      'Wenn es wieder dunkel ist
                                            'und der Lichtimpuls nur kurz war, dann blinken
   If Led_ladezustand = 1 Then              'Bei Dunkelheit Merker und Zähler für Lichtdauer löschen
      Hell = 0
      B = 0
   End If
Loop

Led_abfrage:
  Portb.3 = 0 'Portb.3 auf Masse schalten
  Portb.4 = 1 'Portb.4 auf +Ub schalten, um die LED zu 'laden'
  Waitus 1 'Ladezeit 1 µs, kann ggf. noch verkleinert werden
  Config Portb.4 = Input 'Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
  Portb.4 = 0 'Pullup abschalten, sonst geht's nicht!
  Waitms 20                                 'Entladezeit 20 ms - je kleiner, je unempfindlicher
  Led_ladezustand = Pinb.4 'Ladezustand einlesen: '1' -> dunkel, '0' -> hell
  Config Portb.4 = Output 'Portb.4 wieder auf Ausgang schalten
  Portb.4 = 0 'Portb.4 auf Masse schalten
Return

Blinken:
  Portb.3 = 0
  Portb.4 = 0
  Readeeprom Led_lichtcode , 1     'Blinksequenz mit EEPROM-Inhalt
  If Led_lichtcode = 1 Then        'Auslesen des LED-Startzustandes
    Last = 1
  Else
    Last = 0
  End If
  For A = 2 To 63                  'Auslesen der gespeicherten Bytes
    Readeeprom Led_lichtcode , A
    If Led_lichtcode = 255 Then
      Portb.3 = Last
      For I = 1 To 254
        Waitms 90
      Next I
    Else
      Portb.3 = Last
      For I = 1 To Led_lichtcode
        Waitms 90
      Next I
      Last = Not Last
    End If
  Next A
Return

End
```

4.9 Blinker mit Morsezeichen

Es werden zehn Takte gemorst. Der String in dem Morseunterprogramm kann vor dem Compilen angepaßt werden.
Wenn mehr als zehn Signale gegeben werden sollen, muß am Programmanfang die Stringlänge verlängert werden, sie steht zur Zeit auf 10. Die Schaltung könnte interessant sein, wenn der Blinkrythmus z.B. ein Teil der Geocache-Aufgabenstellung ist. Im Hauptprogramm habe ich den Sprung in das "Blinken"-Unterprogramm auskommentiert und stattdessen einen Sprung in das Unterprogramm "Morsen" eingebaut.

```
'============================================================
'
' **********************************
' ***                            ***
' *** Reaktiv-Tiny v0.1 24/11/2005 ***
' *** Version v0.2 13.02.2006 Morse***
' ***                        (FU) ***
' **********************************
```

```
'
' µC: ATtiny13
' +Ub: 3,43 V
' I: 58 µA
' Reichweite Dorcy:
' Reichweite 4D-MagLite:
'
'============================================================

$regfile = "Attiny13.DAT"
$crystal = 113000                       'Reale Frequenz des internen 128kHz-Oszillators
$hwstack = 6                            'sram Compilerfehler vermeiden

Config Portb = &B00011000               'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111                      'Pullups zuschalten, außer für Pinb.3 und .4

Stop Adc                                'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                                 'Analog-Komparator abschalten, um Strom zu sparen

Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim Led_ladezustand As Bit
Dim Hell As Bit
Dim Dunkel As Bit
Dim Morse$ As String * 10                                '10 Takte, die gemorst werden sollen
Dim S$ As String * 1                                     'Merken zum Morsen (aktueller Takt)

' *********************************************************
' Start des Hauptprogramms
Do
  Gosub Led_abfrage
  If Led_ladezustand = 0 Then Hell = 1                  'Bei Licht Merker setzen
  If Hell = 1 And B < 255 Then B = B + 1                'Wenn Merker gesetzt wurde, Zähler für
                                                        'Lichtdauer erhöhen (bis max. 255)

  If Led_ladezustand = 1 And Hell = 1 And B < 30 Then   'Wenn es wieder dunkel ist dann
                                                        'nachsehen für wie lange
    Dunkel = 1                                          'Merker Setzen
    For C = 0 To 5                                      'LED-Zustand mehrmals abfragen
      Gosub Led_abfrage
      If Led_ladezustand = 0 Then Dunkel = 0            'Wenn wieder Hell dann Dunkel-Merker löschen
    Next B
    Rem If dunkel = 1 Then Gosub Blinken                'Wenn Dunkel-Merker gesetzt Blinken
    If Dunkel = 1 Then Gosub Morsen                     'Wenn Dunkel-Merker gesetzt Morsen
  End If

  If Led_ladezustand = 1 Then            'Bei Dunkelheit Merker und Zähler für Lichtdauer löschen
    Hell = 0
    B = 0
  End If
Loop
' *********************************************************


' *********************************************************
' Abfrage, ob LED beleuchtet wird (Ladezustand 0 = Licht, 1 = kein Licht)
Led_abfrage:
  Portb.3 = 0                'Portb.3 auf Masse schalten
  Portb.4 = 1                'Portb.4 auf +Ub schalten, um die LED zu 'laden'
  Waitus 1                   'Ladezeit 1 µs, kann ggf. noch verkleinert werden
  Config Portb.4 = Input     'Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
  Portb.4 = 0                'Pullup abschalten, sonst geht's nicht!
  Waitms 20                  'Entladezeit  20ms (100 ms) - je kleiner, je unempfindlicher
  Led_ladezustand = Pinb.4   'Ladezustand einlesen: '1' -> dunkel, '0' -> hell
  Config Portb.4 = Output    'Portb.4 wieder auf Ausgang schalten
  Portb.4 = 0                'Portb.4 auf Masse schalten
Return
' ************************************************************************
```

```
' ********************************************************************************
' Mit LED kurz blinken
Blinken:
  For A = 1 To 10
    Portb.3 = 1
    Waitms 10
    Portb.3 = 0
    Waitms 100
  Next A
Return
' ********************************************************************************



' ********************************************************************************
' Mit LED Zeichen morsen (10 Impulse)
Morsen:
  Morse$ = "--***--***"                              'String aus 10 Zeichen * = kurz - = lang
  For A = 1 To 10                                    '10 Zeichen auslesen und auswerten
  S$ = Mid(morse$ , A , 1)                           'Teilstring auslesen
    If S$ = "*" Then                                 'wenn kurz Blinken
      Portb.3 = 1                                    ' LED AN
      Waitms 30                                      ' kurz warten
      Portb.3 = 0                                    ' LED AUS
      Waitms 1000                                    ' lang warten
    Else                                             'sonst lang blinken
      Portb.3 = 1                                    ' LED AN
      Waitms 300                                     ' lange warten
      Portb.3 = 0                                    ' LED AUS
      Waitms 1000                                    ' lang warten
    End If
  Next A
Return
'********************************************************************************
End
```

4.10 Morsecodeausgabe- und Abfrage

Was ausgegeben wird, wird aus dem EEPROM gelesen. Dieses sollte also gefüllt sein, was direkt übers BASCOM geht. Das wird aber per Code gemacht (siehe 4.11). Nicht vergessen, das Fusebit G auf 0 zu stellen. Es spielt keine Rolle, wie lang die Dunkelpausen sind. Das erste dit entscheidet die Länge der Zeichen. Ein dah sollte mindestens doppelt so lang wie das erste dit sein.

```
'============================================================
'
' ***********************************************************
' ***                                                     ***
' *** Reaktiv-Tiny v0.5 2006-02-25                  (AP) ***
' ***                                                     ***
' *** Morsecodeabfrage eines Morsezeichens mit            ***
' *** anschließender Ausgabe von bis zu 64 Morsezeichen.  ***
' *** Es können Koordinaten oder beliebiger Text          ***
' *** ausgegeben werden.                                  ***
' *** Die Eingabegeschwindigkeit wird mit der ersten      ***
' *** empfangenen Zeichenlänge bestimmt                   ***
' ***                                                     ***
' ***                                                     ***
' *** --------------- Einstellungen --------------------- ***
' ***                                                     ***
' *** Fusebit H:    0                                     ***
' *** Fusebit G:    0                                     ***
' *** Fusebit F:    1                                     ***
' *** Fusebit E:    1                                     ***
' *** Fusebit DCBA: 1011                                  ***
' ***                                                     ***
' *** Beachtet das Fusebit G. Es muss auf 0 stehen!       ***
' ***                                                     ***
' *** Unter 'Options' ' Compiler' 'Chip'                  ***
' *** den "HW Stack" runterstellen. z.B. auf 6            ***
' ***                                                     ***
' ***********************************************************
'
' µC: ATtiny13V
```

```
'
'============================================================
'
$regfile = "ATtiny13.DAT"
$crystal = 113000               'Reale Frequenz des internen 128kHz-Oszillators
$hwstack = 6
'
Config Portb = &B00011000       'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111              'Pullups zuschalten, außer für Pinb.3 und .4
'
Stop Adc                        'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                         'Analog-Komparator abschalten, um Strom zu sparen
'
'
'Anmerkung zur Deklaration: bin sparsam mit den Variablen umgegangen,
'd.h. diese werden zum Teil im Programm mehrfach für verschiedene Aufgaben verwendet.
'Grund: ich kenne das "Drosseln" des HW Stack erst seit es hier im Fred stand. Dank an Gonzoarts.
Dim Hell As Bit
Dim Vhell As Bit
Dim Ok As Bit
Dim A As Byte
Dim B As Byte
Dim Ein As Integer
Dim Bx As Byte
Dim Z As Byte
Dim D1 As Byte
Dim D2 As Integer
'
'
'
'
' Es können alle Buchstaben und Zahlen gesendet werden, da diese maximal 5 Zeichen lang sind.
' Ausgabe der Zeichen aus dem EEPROM mit Länge
'
'
'
' dit = kurz = . = 0
' dah = lang = - = 1
'
'

'            765 43210
'            \_/ \___/
'             |    |
'           Länge  |
'                  |
'            Morsezeichen v.rechts (rückwärts)
'
'     Morse   Bit v.r.
'1 = .---- = 100|11110
'2 = ..--- = 100|11100
'3 = ...-- = 100|11000
'4 = ....- = 100|10000
'5 = ..... = 100|00000
'6 = -.... = 100|00001
'7 = --... = 100|00011
'8 = ---.. = 100|00111
'9 = ----. = 100|01111
'0 = ----- = 100|11111
'
'N = -.    = 001|00010
'E = .     = 000|00000
'
'KA (Spruch Anfang) = -.-.- = 100|10101
'AR (Spruch Ende)   = .-.-. = 100|01010
'
'Wortabstand einfügen mit 111|11111  (FF)
'
'
' Übertragungsrate
' 1 WPM: 1 dit = 1200 ms
' 5 WPM: 1 dit =  240 ms
'10 WPM: 1 dit =  120 ms
'20 WPM: 1 dit =   60 ms
'
'
'==> Übertragungsrate festlegen, hier 5 Wörtern pro Minute (WPM)
```

```
D1 = 240                                    'D1 entspricht der Länge eines dit in Millisekunden
D2 = 480                                    'D2 entspricht der doppelten Länge eines dit
'
'==> Abgefragtes Zeichen festlegen
Ein = &B0000000000111000                    'Beispiel: SOS  ...---...
'
'
Ok = 1
Gosub Morsen                                'beim Start das ganze mal testweise Morsen
'
' Morsezeichen - Abfrage:
Do
 Gosub Led_abfrage
 If Hell = 1 Then Vhell = 1                'Bei Licht Merker setzen
 If Vhell = 1 And B < 255 Then B = B + 1   'Wenn Merker gesetzt wurde, Zähler für Lichtdauer
                                           'erhöhen (bis max. 255)
 If Hell = 0 Then
  If Vhell = 1 Or B = 255 Then             'Übergang zu Dunkel
    If Z = 0 Then
      Ok = 1                               'Neues Spiel, neues Glück
                                           'Festlegen der Grenzdauer. Diese entscheidet, ob
                                           'ein dit oder dah eingegeben wurde.
                                           '==> Leider muss man hier "von Hand" eintragen,
                                           'ob das erste Zeichen dit oder dah ist. Bei dit B),
                                           'bei dah A) auskommentieren
                                           'Die benötigte If-Abfrage hat im 1k Programmspeicher
                                           'des ATtiny13V keinen Platz mehr.
                                           'A) erstes abgefragtes Zeichen ist ein dit
      If B < 127 Then                      'A) dies sollte nicht zu lange sein
        Bx = B * 1.7                       'A) Grenzdauer Bx durch Länge des ersten dit festlegen
      Else                                 'A)
        Ok = 0                             'A) das war zu lange fürs erste dit
      End If                               'A)
                                           'B) erstes abgefragtes Zeichen ist ein dah
      'Bx = B / 2                          'B) Grenzdauer Bx durch Länge des ersten dah festlegen
      End If
      If B < Bx Then
        If Ein.z = 1 Then Ok = 0       'dit empfangen, wenn Ein.z = 0, dann richtig
      Else
        If Ein.z = 0 Then Ok = 0       'dah empfangen, wenn Ein.z = 1, dann richtig
      End If
      Z = Z + 1
      If Z = 9 Or Ok = 0 Then          'Auswerten wenn Falsch ODER Anzahl Zeichen erreicht
        Gosub Morsen                   '==> Z auf die Anzahl abgefragter Zeichen einstellen
      End If
    End If
  End If
  If Hell = 0 Then                     'Bei Dunkelheit Merker und Zähler für Lichtdauer löschen
    Vhell = 0
    B = 0
  End If
Loop


Morsen:
  Waitms 500                           'Pause, sonst ist das erste Zeichen schwer zu lesen
  If Ok = 1 Then                       'Morsezeichen senden
    For Bx = 0 To 36                   '=> Länge anpassen!
      Readeeprom A , Bx                'in A steht der zu morsende Code (Bit 0 bis 4 dit und
                                       'dah, Bit 5 bis 7 die Anzahl
      Z = 0
      Z.0 = A.5                        'Zeichenlänge steht in den vorderen 3 Bit (A.5 bis 7)
      Z.1 = A.6
      Z.2 = A.7
      If Z = 7 Then                    'bei FF einen Wortabstand einfügen (4dit = 7dit - 3dit)
        Waitms D2
        Waitms D2
      Else                             'Zeichenlänge: 1 bis 5  (0 bis 4)
        Gosub Licht
      End If
    Next Bx
  Else                                 ' IRRUNG senden
    If B < 255 Then                    ' aber nicht, wenn die Dämmerung eintritt bzw. ein
                                       'langes Lichtsignal anlag
      Z = 7
      A = 0                            ' Irrung  =  ........ =  0
      Gosub Licht
```

```
    End If
  End If
  'A = 0
  B = 0
  Z = 0
Return


Licht:
  For B = 0 To Z
    Portb.3 = 1                       'LED an
    Waitms D1                         '1  . dit
    If A.b = 1 Then
      Waitms D2                       '3  - dah
    End If
    Portb.3 = 0                       'LED aus
    Waitms D1                         '1 dit  Pause
  Next B
  Waitms D2                           '+2 dit Pause nach Ziffer
Return


Led_abfrage:
  Portb.3 = 0                         'Portb.3 auf Masse schalten
  Portb.4 = 1                         'Portb.4 auf +Ub schalten, um die LED zu 'laden'
  Waitus 1                            'Ladezeit 1 µs, kann ggf. noch verkleinert werden
  Config Portb.4 = Input              'Portb.4 nun zwecks Abfrage der LED-Ladung
                                      'auf 'Eingang' schalten
  Portb.4 = 0                         'Pullup abschalten, sonst geht's nicht!
  Waitms 20                           'Entladezeit 20 ms - je größer, desto weniger Licht genügt
                                      '_(je kleiner, je unempfindlicher)
  If Pinb.4 = 0 Then                  'Ladezustand einlesen: '1' -> dunkel, '0' -> hell
    Hell = 1
  Else
    Hell = 0
  End If
  Config Portb.4 = Output             'Portb.4 wieder auf Ausgang schalten
  Portb.4 = 0                         'Portb.4 auf Masse schalten
Return

End
```

4.11 Morsezeichen in EEPROM schreiben

- das Fusebit G = 0 setzen (sonst überschreibt man sich dieses beim eigentlichen Programmieren wieder.)
- die gewünschte Ausgabe in diesem Code anpassen
- hiermit das EEPROM beschreiben (F7, dann F4)
- das eigentliche Programm in den ATtiny schreiben

```
'============================================================
'
' ************************************************************
' ***                                                      ***
' *** Reaktiv-Tiny EEPROM schreiben  2006-02-25      (AP) ***
' ***                                                      ***
' ***                                                      ***
' *** Beachtet das Fusebit G. Es muss auf 0 stehen!        ***
' ***                                                      ***
' *** Fusebit H:    0                                      ***
' *** Fusebit G:    0                                      ***
' *** Fusebit F:    1                                      ***
' *** Fusebit E:    1                                      ***
' *** Fusebit DCBA: 1011                                   ***
' ***                                                      ***
' ***                                                      ***
' ************************************************************
'
' µC: ATtiny13V
'
```

```
'============================================================
'
$regfile = "ATtiny13.DAT"
$crystal = 113000               'Reale Frequenz des internen 128kHz-Oszillators
$hwstack = 6
'
Config Portb = &B00011000       'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111              'Pullups zuschalten, außer für Pinb.3 und .4
'
'Stop Adc                       'A/D-Wandler abschalten, um Strom zu sparen
'Stop Ac                        'Analog-Komparator abschalten, um Strom zu sparen
'
'
Dim A As Byte
'
'
'-------------------------------------------------------------
' Bitwert 0=dit, 1=dah
' das Byte von rechts lesen, da Schleife von 0 bis z läuft
' z ist die Zeichenlänge minus eins, die in den oberen 3 Bit festegelgt wird
'
'     Morse  Bit v.r.       Hex   Dec
'1 = .---- = 100|11110   1  9E    158
'2 = ..--- = 100|11100   2  9C    156
'3 = ...-- = 100|11000   3  98    152
'4 = ....- = 100|10000   4  90    144
'5 = ..... = 100|00000   5  80    128
'6 = -.... = 100|00001   6  81    129
'7 = --... = 100|00011   7  83    131
'8 = ---.. = 100|00111   8  87    135
'9 = ----. = 100|01111   9  8F    143
'0 = ----- = 100|11111   0  9F    159
'
'N = -.    = 001|00001   N  21     33
'E = .     = 000|00000   E  00      0
'
' Pause    = 111|11111   0  FF    255
'
' KA (Spruch Anfang) = -.-.- = 100|10101   95    149
' AR (Spruch Ende)   = .-.-. = 100|01010   8A    138
'
'
'
'Wortabstand ==             FF
'H = ....  = 011|00000   H  60     96
'A = .-    = 001|00010   A  22     34
'P = .--.  = 011|00110   P  66    102
'P   .--.  = 011|00110   P  66    102
'Y = -.--  = 011|01101   Y  6D    109
'Wortabstand ==             FF
'H = ....  = 011|00000   H  60     96
'U = ..-   = 010|00100   U  44     68
'N = -.    = 001|00001   N  21     33
'T = -     = 000|00001   T  01      1
'I = ..    = 001|00000   I  20     32
'N = -.    = 001|00001   N  21     33
'G = --.   = 010|00011   G  43     67
'Wortabstand ==             FF
'Ende                       8A
'
'
'
'===========================================
' Beispiel Cache: N49.12.345 E008.56.789
'===========================================
'
'
A = &B11111111           ' Pause
Writeeeprom A , 0
A = &B10010101           ' KA (Spruch Anfang) = -.-.-
Writeeeprom A , 1
A = &B11111111           ' Pause
Writeeeprom A , 2
A = &B00100001           ' N = -.
Writeeeprom A , 3
A = &B10010000           ' 4 = ....-
Writeeeprom A , 4
```

```
A = &B10001111          ' 9 = ----.
Writeeeprom A , 5
A = &B10011110          ' 1 = .----
Writeeeprom A , 6
A = &B10011100          ' 2 = ..---
Writeeeprom A , 7
A = &B10011000          ' 3 = ...--
Writeeeprom A , 8
A = &B10010000          ' 4 = ....-
Writeeeprom A , 9
A = &B10000000          ' 5 = .....
Writeeeprom A , 10
A = &B11111111          ' Pause
Writeeeprom A , 11
A = &B00000000          ' E = .
Writeeeprom A , 12
A = &B10011111          ' 0 = -----
Writeeeprom A , 13
A = &B10011111          ' 0 = -----
Writeeeprom A , 14
A = &B10000111          ' 8 = ---..
Writeeeprom A , 15
A = &B10000000          ' 5 = .....
Writeeeprom A , 16
A = &B10000001          ' 6 = -....
Writeeeprom A , 17
A = &B10000011          ' 7 = --...
Writeeeprom A , 18
A = &B10000111          ' 8 = ---..
Writeeeprom A , 19
A = &B10001111          ' 9 = ----.
Writeeeprom A , 20
A = &B11111111          ' Pause
Writeeeprom A , 21
A = &B01100000          ' H = ....
Writeeeprom A , 22
A = &B00100010          ' A = .-
Writeeeprom A , 23
A = &B01100110          ' P = .--.
Writeeeprom A , 24
A = &B01100110          ' P = .--.
Writeeeprom A , 25
A = &B01101101          ' Y = -.--
Writeeeprom A , 26
A = &B11111111          ' Pause
Writeeeprom A , 27
A = &B01100000          ' H = ....
Writeeeprom A , 28
A = &B01000100          ' U = ..-
Writeeeprom A , 29
A = &B00100001          ' N = -.
Writeeeprom A , 30
A = &B00000001          ' T = -
Writeeeprom A , 31
A = &B00100000          ' I = ..
Writeeeprom A , 32
A = &B00100001          ' N = -.
Writeeeprom A , 33
A = &B01000011          ' G = --.
Writeeeprom A , 34
A = &B11111111          ' Pause
Writeeeprom A , 35
A = &B10001010          ' AR (Spruch Ende)   = .-.-. =
Writeeeprom A , 36
```

4.12 Testprogramm

Wenn es dunkel ist, blinkt die LED alle drei Sekunden. Wenn es hell ist, bleibt sie dunkel.

```
'============================================================
'
' ***********************************
' ***                               ***
' ***   Blinklicht 3 Sekunden       ***
' ***                               ***
```

```
' ***********************************
'
' µC: ATtiny13
' +Ub: 3,00 V
'
'
'============================================================

$regfile = "ATtiny13.DAT"
$crystal = 128000                       'Frequenz des internen 128kHz-Oszillators

Config Portb = &B00011000               'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111                      'Pullups zuschalten, außer für Pinb.3 und .4
Stop Adc                                'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                                 'Analog-Komparator abschalten, um Strom zu sparen

Dim Led_ladezustand As Bit

Do
  Wait 3
  Gosub Led_abfrage
  If Led_ladezustand = 1 Then Gosub Leuchten
Loop


Led_abfrage:
  Portb.4 = 1                           'Portb.4 auf +Ub schalten, um die LED zu 'laden'
  Waitus 1                              'Ladezeit 1 µs, kann ggf. noch verkleinert werden
  Config Portb.4 = Input       'Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
  Portb.4 = 0                           'Pullup abschalten, sonst geht's nicht!
  Waitms 100                            'Entladezeit 100 ms - je kleiner, je unempfindlicher
  Led_ladezustand = Pinb.4              'Ladezustand einlesen: '1' -> dunkel, '0' -> hell
  Config Portb.4 = Output               'Portb.4 wieder auf Ausgang schalten
Return


Leuchten:
  Portb.3 = 1
  Waitms 50
  Portb.3 = 0
Return

End
```

4.13 Nachtaktiver Blinker mit Watchdog-Abschaltung

Eine durch die Watchdog-Abschaltung stromsparende Variante. Stromverbrauch im Leerlauf nur 5 µA.
Die Schwierigkeit beim Programmieren ist, dass der WD verschiedene Betriebszustände hat (Aus, Reset, Interrupt, Reset und Interrupt) die von Bascom aber nicht unterstützt werden. Mit den normalen Basic-Befehlen kann man immer nur den Reset-Modus aktivieren. Abhilfe schafft das händische Setzen des Watchdog-Registers. Erklärt ist alles auf den Seiten 39 und 40 des Datenblattes. Die Watchdog-Zeit wird über die Bits WDP0-WDP3 eingestellt.
Stromaufnahme mit WAIT-Befehl: 50 Mikroampere
Stromaufnehme mit IDLE-Befehl: 15 Mikroampere
Stromaufnehme mit POWERDOWN-Befehl: 5 Mikroampere
Gemessen jeweils bei einer Versorgungsspannung von 3 Volt

```
'============================================================
'
' ******************************************
' ***                                    ***
' *** Reaktiv-Tiny mit Watchdog-Abschaltung  ***
' ***      von Sir Vivor und Windi           ***
' ***                                    ***
' ******************************************
'
' µC: ATtiny13
```

```
' +Ub: 3,00 V
' I: 5 µA im Leerlauf
'
'============================================================

$regfile = "ATtiny13.DAT"
$crystal = 16000            'Frequenz des internen Oszillators

Config Portb = &B00011000   'Pinb.3 und .4 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = &B11100111          'Pullups zuschalten, außer für Pinb.3 und .4
Stop Adc                    'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                     'Analog-Komparator abschalten, um Strom zu sparen

Wdtcr = &B11010011          'Watchdog definieren: 0.125 Sekunden, Interrupt auslösen, kein Reset

'Interrupts freigeben
Enable Interrupts

Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim Led_ladezustand As Bit
Dim Hell As Bit
Dim Hell_2 As Byte

Do
  Gosub Led_abfrage
  If Led_ladezustand = 0 Then Hell = 1                    'Bei Licht Merker setzen
  If Hell = 1 And B < 255 Then B = B + 1                  'Wenn Merker gesetzt wurde, Zähler für
                                                          'Lichtdauer erhöhen (bis max. 255)
    If Led_ladezustand = 1 And Hell = 1 And B < 30 Then   'Wenn es wieder dunkel ist und der
                                                          'Lichtimpuls nur kurz war
      Hell_2 = 0                                          'zweiten Hell-Merker setzen
      For C = 0 To 5                                      'und 5 mal afragen
        Gosub Led_abfrage
        If Led_ladezustand = 0 Then Hell_2 = Hell_2 + 1   'ob es auch wieder dunkel ist
      Next C
      If Hell_2 = 0 Then Gosub Blinken                    'erst dann blinken
      Hell = 0
      B = 0
    End If
    If Led_ladezustand = 1 Then                           'Bei Dunkelheit Merker und Zähler
                                                          'für Lichtdauer löschen
      Hell = 0                'damit sich das Programm nicht aufhängt
      B = 0
   End If
Loop

Led_abfrage:
  Portb.4 = 1               'Portb.4 auf +Ub schalten, um die LED zu 'laden'
  Waitus 1                  'Ladezeit 1 µs, kann ggf. noch verkleinert werden
  Config Portb.4 = Input    'Portb.4 nun zwecks Abfrage der LED-Ladung auf 'Eingang' schalten
  Portb.4 = 0               'Pullup abschalten, sonst geht's nicht!
  Reset Watchdog            'Watchdog zurücksetzen dass µC rechtzeizig aufwacht
  Powerdown                 'Während des Entladens den µC Schlafen schicken
  Led_ladezustand = Pinb.4  'Ladezustand einlesen: '1' -> dunkel, '0' -> hell
  Config Portb.4 = Output   'Portb.4 wieder auf Ausgang schalten
Return


Blinken:
  For A = 1 To 10
    Portb.3 = 1
    Reset Watchdog
    Powerdown
    Portb.3 = 0
    Reset Watchdog
    Powerdown
  Next A
Return

End
```

11.07.09 18:21:11

Hier noch ein Programm zum Watchdog-Experimentieren. Die LED blinkt 100 mal, danach geht der Prozessor für 8 Sekunden schlafen.

```
' Prozessor in den Schlaf schicken und per Watchdog aufwecken
' µC: ATtiny13
'
' =================================================

$regfile = "ATtiny13.DAT"
$crystal = 128000
Config Portb = &B00011000
Portb = &B11100111
Stop Adc
Stop Ac

'Watchdog definieren: 8 Sekunden, Interrupt auslösen, kein Reset
Wdtcr = &B11110001

'Interrupts freigeben
Enable Interrupts

Dim A As Byte

Do
  Gosub Blinken
Loop

Blinken:
  For A = 1 To 100
    Portb.3 = 1
    Waitms 20
    Portb.3 = 0
    Waitms 20
  Next A

  'Watchdog-Timer zurücksetzen
  Reset Watchdog

  'Prozessor schlafen schicken
  Powerdown
  'Das Programm läuft nach Ablauf der WD-Zeit hier weiter
Return

End
```

5 Programme für Schaltung mit Helligkeitsmessung über LRD

5.1 Grundprogramm (Schaltung nach Abb. 6)

Das Programm ist im Vergleich zur LED-only-Variante nahezu gleich geblieben. Lediglich ein Schlafmodus wurde hinzugefügt, den man aber auch der LED-Variante spendieren könnte. Wird 200 Zyklen lang „hell“ gemessen, geht der Prozessor für 8 Sekunden in den Schlafmodus. Das ist die längste Zeit, die mit dem Watchdog-Timer realisierbar ist. Nach diesen 8 Sekunden wacht er kurz auf und misst erneut. Ist es immer noch hell, schläft er weitere 8 Sekunden. Ist es hingegen dunkel, geht er in den normalen Betriebszustand über (eine Messung alle 125 ms).

```
'============================================================
'
' ************************************************
' ***                                          ***
' *** Reaktiv-Tiny mit Watchdog-Abschaltung und LDR  ***
' ***      von Sir Vivor und Windi             ***
' ***                                          ***
' ************************************************
'
' µC: ATtiny13
' +Ub: 3,00 V
' I: 5 µA im Leerlauf
'
'============================================================

$regfile = "ATtiny13.DAT"
$crystal = 16000                'Frequenz des internen Oszillators

Config Portb = &B00001001       'Pinb.0 und 3 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = 0
Stop Adc                        'A/D-Wandler abschalten, um Strom zu sparen
Stop Ac                         'Analog-Komparator abschalten, um Strom zu sparen

Wdtcr = &B11010011              'Watchdog definieren: 0.125 Sekunden, Interrupt auslösen, kein Reset
Enable Interrupts               'Interrupts freigeben

Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim Ldr As Bit
Dim Hell As Bit
Dim Hell_2 As Byte

Do
  Gosub Ldr_abfrage
  If Ldr = 1 Then Hell = 1                    'Bei Licht Merker setzen
  If Hell = 1 And B < 255 Then B = B + 1       'Wenn Merker gesetzt wurde, Zähler für Lichtdauer
                                               'erhöhen (bis max. 255)
  If B > 200 Then Gosub Abschalten            'Schlafmodus wenn es zu lange am Stück hell ist
  If Ldr = 0 And Hell = 1 And B < 30 Then      'Wenn es wieder dunkel ist und der Lichtimpuls nur
                                               'kurz war
     Hell_2 = 0                               'zweiten Hell-Merker setzen
     For C = 0 To 5                           'und 5 mal abfragen
       Gosub Ldr_abfrage
       If Ldr = 1 Then Hell_2 = Hell_2 + 1     'ob es auch wieder dunkel ist
     Next C
     If Hell_2 = 0 Then Gosub Blinken          'erst dann blinken
     Hell = 0
     B = 0
  End If
```

```
  If Ldr = 0 Then                         'Bei Dunkelheit Merker und Zähler für Lichtdauer löschen
    Hell = 0                              'damit sich das Programm nicht aufhängt
    B = 0
  End If
Loop

Ldr_abfrage:
  Portb.0 = 1                             'Spannung auf LDR geben
  Reset Watchdog                          ' Kurz warten bis sich alles eingeschwungen hat
  Powerdown
  Ldr = Pinb.4                            'LDR abfragen
  Portb.0 = 0                             'Spannung von LDR wieder wegnehmen
  Return

Blinken:                                    'LED blinken lassen
  For A = 1 To 10
    Portb.3 = 1
    Reset Watchdog
    Powerdown
    Portb.3 = 0
    Reset Watchdog
    Powerdown
    Next A
  Return

Abschalten:                                  'Prozessor für 8 Sekunden schlafen lassen
  Wdtcr = &B11110001
  Reset Watchdog
  Powerdown
  Wdtcr = &B11010011
  Return

End
```

5.2 Einlesen der Helligkeit über A/D-Wandler (Schaltung nach Abb. 7)

Das Programm erfasst ständig die Umgebungshelligkeit und reagiert nur, wenn es schlagartig heller wird.

Leider erkauft man sich den Luxus von 650 m Reichweite mit einer Maglite 5D mit einer erhöhten Stromaufnahme von ca. 17 µA. Das Programm ist mit einer Tagschaltung versehen, die nur etwa 5 µA verbraucht. Das Programm gibt im Tagbetrieb alle 60 sekunden einen kurzen Impuls an die Leuchtdiode. Dies dient der Funktionskontrolle oder dass die Schaltung bei Fremdlicht noch als Blinker funktioniert. Man kann dies natürlich auch weglassen. Die Schaltung ist minimal verändert. Der Spannungsteiler ist direkt angeschlossen und wird nicht mehr getaktet. Er ist so hochohmig, dass die Umschaltvorgänge des IC-Ausgangs mehr Strom verbrauchen als ein dauerversorgter Spannungsteiler.

```
'============================================================
'
' ********************************************************
' ***                                                  ***
' ***    Tiny-Reaktivlicht mit LDR und A/D-Wandler     ***
' *** mit Watchdog-Energiesparmodus und Tagabschaltung ***
' ***                                                  ***
' ***      erstellt von Windi für www.geoclub.de       ***
' ***                    15.10.2006                    ***
' ***                                                  ***
' ********************************************************
'
' µC: ATtiny13V
' +Ub: 3,00 V
' I: 17 µA im Leerlauf (Nachtbetrieb)
' I: 5 µA bei Tagabschaltung
' Reichweite: 650 m mit Maglite 5D
'
'============================================================
```

```
$regfile = "ATtiny13.DAT"
$crystal = 16000            'Frequenz des internen Oszillators

$hwstack = 28               'hardwarestack herabsetzen damit genügend variablen zur verfügung stehen


Config Adc = Single , Prescaler = Auto
Config Portb = &B00001000   'Pinb.3 auf 'Ausgang', Rest auf 'Eingang' schalten
Portb = 0                   'Ausgänge auf Low setzen
Stop Ac                     'Analog-Komparator abschalten, um Strom zu sparen

Wdtcr = &B11010011          'Watchdog definieren: 0.125 Sekunden, Interrupt auslösen, kein Reset
Enable Interrupts           'Interrupts freigeben

Const Schwelle = 50         'je größer der Schwellwert, desto unempfindlicher
Const Tagschwelle = 800     'Schwellwert für Schlafmodus
Const Zwangsimpuls = 8      'LED-Impuls tagsüber alle X Schlafyklen (á ca. 8 Sekunden)

Dim A As Byte               'Variablen definieren
Dim Tagzaehler As Byte
Dim Schlafzaehler As Byte
Dim Ldr As Integer          '0 = Dunkel, 1023 = Hell
Dim Alt As Integer
Dim Merker As Integer

Do
Reset Watchdog
Powerdown                   'prozessor bremsen da sonst lichtänderung nicht erkannt wird
Start Adc                   'A/D-Wandler starten
Ldr = Getadc(2)             'Helligkeitswert einlesen
Stop Adc                    'A/D-Wandler zum Stromsparen wieder stoppen
Merker = Ldr - Alt          'Unterschied zwischen letzter und aktueller Messung ermitteln
Alt = Ldr                   'letzten LDR-Wert sichern
If Merker > Schwelle Then Gosub Blinken             'Bei großer Änderung Dunkel->Hell: Blinken
If Ldr > Tagschwelle Then    'prüfen ob helligkeit über tagschwelle liegt
   If Tagzaehler < 255 Then  'int-variable geht nur bis 255
      Tagzaehler = Tagzaehler + 1
   End If
Else
   Tagzaehler = 0            'wenn wieder dunkel tagzähler löschen
End If

If Tagzaehler > 200 Then Gosub Pause                 'wenn mehr als x zyklen hell dann schlafmodus

Loop


Blinken:                     'LED blinken lassen
For A = 0 To 10
  Portb.3 = 1
  Reset Watchdog
  Powerdown
  Portb.3 = 0
  Reset Watchdog
  Powerdown
Next A
Alt = 1023                    'Doppelauslösung verhindern
Return

Pause:
Wdtcr = &B11110001            'Watchdog auf 8 Sekunden stellen
Reset Watchdog
Powerdown
Wdtcr = &B11010011             'Watchdog wieder auf 0,125 Sekunden zurückstellen
Schlafzaehler = Schlafzaehler + 1       'merken wie oft Schlafmodus durchlaufen wurde
If Schlafzaehler = Zwangsimpuls Then    'als Funktionskontrolle tagsüber LED auslösen
   Portb.3 = 1
   Reset Watchdog
   Powerdown
   Portb.3 = 0
   Schlafzaehler = 0
End If
Return

End
```

6 Problemlösungen

6.1 Ist es normal, dass der Tiny13V beim Anlegen von 3V relativ schnell heiß wird?

- Nein, dies ist ein Mikroprozessor und kein Intel-PC mit Herdplattentechnologie ;-).
- Wenn die Schaltung auf einem Steckbrett ist, alles bis auf die Stromversorgung abstecken. Wird diese heiß, ist sie defekt. Wenn sie kalt bleibt, langsam an die volle Schaltung herantasten.
- Ist der Prozessor richtig herum eingesteckt? Pin 1 ist derjenige mit der Kerbe im Gehäuse. Danach wird gegen den Uhrzeigersinn gezählt.
- Sind mehrere Beinchen, die nicht zusammen gehören, innerhalb einer durchkontaktierten Reihe im Steckbrett oder ist das Steckbrett defekt?

6.2 Der Mikroprozessor ist kaputt

- Schon kurzes Verpolen der Versorgungsspannung bereitet dem Prozessor ein schnelles Ende. Da hilft nur ein neuer.

6.3 Fehlermeldung „The HW-Stack, SW-Stack and frame space may not exceed the chip memory“

- Bei einem frisch installierten Programm muss man zuerst auf „Neue Datei“ klicken, dann funktioniert es.

6.4 Fehlermeldung „Could not Identify chip with IDE: ...“ beim Fuse-Bits setzen

- Compilieren des Quelltextes hat nicht funktioniert. Fehler entfernen und neu kompilieren, dann erkennt er ihn wieder.
- Ein weiterer Grund für die Fehlermeldung kann bei der externen Beschaltung des Chips liegen. Bei mir hat ein ausgeschaltetes Multimeter, dass aber noch an der Schaltung angeklemmt war, diese Fehlermeldung hervorgerufen.

6.5 Fehlermeldung „Out of SRAM space“

- In Bascom sind die Stacks standardmäßig zu hoch eingestellt. Unter „Options“ „Programmer“ im Register den „HW Stack“ runterstellen. Diese Stack Size ist von Haus aus auf 32 Byte eingestellt, pro Gosub werden allerdings nur 2 Byte benötigt. Alternativ könne auch folgende Zeilen eingefügt werden:

```
$regfile = "ATtiny13.DAT"
$crystal = 113000              'Reale Frequenz des internen 128kHz-Oszillators
$hwstack = 6
```

6.6 Fuse-Bits

- Nicht die Fuse-Bits vergessen! CKSEL0, CKSEL1 und BODLEVEL1 nicht aktivieren und BOD-

LEVEL0 aktivieren (aktivieren meint auf 0 brennen).

6.7 LED leuchtet nicht

- Ist die LED richtig herum gepolt?
- Manchmal muss das Programmierkabel abgemacht werden.

6.8 Wenn man den Tiny vom Strom nimmt und später wieder anschließt, läuft das Programm nicht automatisch an.

- Unter http://www.mikrocontroller.net/tutorial/equipment kann man eine Erweiterung für einen Reset zu Beginn sehen. An dem Reset-Pin liegt ein 10kΩ Widerstand gegen die Versorgungsspannung und ein 47nF Kondensator gegen Masse. Die Ladezeit des Kondensators beträgt damit 5*R*C=2,35µs, was einen sicheren Reset gewährleisten sollte.
- Der Programmieradapter zieht zwischendurch manchmal den Reset-Pin auf Masse (evtl. Problem mit der Leitungslänge -> kürzere Leitungen) bzw. lässt ihn nach dem Programmieren nicht wieder los. Dagegen hilft ein 10 bis 15 kΩ-Widerstand von Reset an Versorgungsspannung (Pin 1 – Pin 8). Den braucht man dann aber später bei der autarken Schaltung nicht mehr.

6.9 Nach dem Setzen des Fuse-Bits 3 auf „External Reset Disable“ hat man keinen Zugriff mehr auf den Chip. Der Programmer kann den Chip nicht mehr identifizieren.

Der Chip ist nicht kaputt, bekommt aber wegen des umprogrammierten Fuse-Bits nicht mehr das Reset des Programmieradapters mit. Im High-Voltage-Modus kann die Fuse wieder zurückprogrammiert werden. Dabei werden 12V an den Reset-Pin gelegt.

6.10 Ist es möglich, den Tiny voll beschaltet zu programmieren?

Ja. Das Programmierinterface schimpft sich ISP (In System Programmable), d. h. Du kannst den Tiny programmieren, während er in der Schaltung steckt. Das funktioniert aber natürlich nur dann, wenn die Ports, an denen der Tiny programmiert wird, hardwaremäßig korrekt verschaltet sind. Hängen z. B. LEDs dran, blinken diese während des Programmiervorgangs. Wenn Du diese Ports nromalerweise als Eingangsports benutzt und womöglich permanent ein Low- oder High-Signal abliegen hast, geht es natürlich nicht. Dann muss man den Tiny aus der Schaltung nehmen, um ih zu programmieren.

7 Sonstige Dinge und Ideen

7.1 LC-Oszillator

LED als Varicap in einem LC-Oszillator

7.2 Zweite LED als Tag-/Nachtsensor

Vielleicht hilft es, eine zweite LED in der gleichen Art und Weise, aber senkrecht dazu, einzubauen und ebenfalls abzufragen. Melden beide LEDs „Licht“, ist es Tag, weil diffuses Licht von überall her kommt. Ist nur eine von beiden hell, ist es Nacht und die Schaltung tut wie es programmiert ist.

7.3 Man könnte Takte zählen und die LED-Messung somit über Tag abschalten

Das dürfte schwierig werden ohne Quarz (z.B. 32768 kHz), da der interne RC-Oszillator doch recht stark von der Betriebsspannung abhängig ist. Die quarzlose Version könnte aber z.B. die Umgebungshelligkeit über einen gewissen Zeitraum 'messen', um dann erst nach längerer Dunkelheit aktiv zu werden. Ein Quarzversion zwecks Zeitsteuerung wäre natürlich auch toll, Stichwort 'mitternächtliches Geblinke' und/oder '- Gepiepe'. Wer's dann ganz luxuriös haben möchte, bastelt noch eine kleine Hardware-RTC (z.B. DS1337) mit zwei Alarmen dazu...

7.4 Was kann man noch alles abschalten?

- A/D-Wandler
- Analog-Komparator
- Brown-Out-Detection
- Interne Spannungsreferenz
- Watchdog Timer
- Digit. Port-Pin-Treiber

Problem ist nur, dass sich z. B. die interne Spannungsreferenz automatisch selbst abschaltet, wenn A/D-Wandler, Analog-Komparator und BOD abgeschaltet sind. Die BOD zu deaktibieren bedeutet aber, dass das µC-Verhalten bei Unterspannung nicht mehr definierbar ist.

7.5 Aufwecken über Interrupt

Wenn es einen Timer gibt, der während sleep weiterläuft, dann könnte es so gehen: Wenn die Spannung an der LED soweit gesunken ist, dass sie als low ausgewertet wird, löst das einen Interrupt aus, der den Atmel wieder weckt. Dann kann man den Timer auslesen und je nachdem die LED wieder laden und den Atmel schlafen schicken oder blinken. So schläft der Atmel die meiste Zeit.

7.6 Relative / absolute Helligkeit

Über die Zeit, die der Prozessor bis zum Auslesen der LED wartet, wird die relative Empfindlichkeit bzw. Helligkeit eingestellt. Am Anfang der Subroutine wird die LED „geladen" und mit der Verzöge-

rung Zeit gegeben, diese Ladung durch Einwirkung von Licht abzubauen. Wartet man nur kurz, muss der Impuls bzw. das Licht stark sein. Wartet man lange, reicht ein schwaches Licht über längere Zeit um den Zustand "hell" zu erzeugen. Welcher Wert richtig ist, hängt auch von der verwendeten LED ab.

9 Bestelldaten

Wer der Meinung ist, dass er trotz der Liste Hilfe benötigt, darf mich kontaktieren. Gegen Übernahme der Portokosten bin ich bereit, Euch die benötigten Bauteile zum Umkostenpreis zu überlassen.

Bauteil	Reichelt
ATtiny 13V-10PU ATtiny 13V-10SU (für SMD-Bestückung) Sockel DIL 8-polig	ATTINY 13V-10PU 0,91 EUR ATTINY 13V-10SU 0,84 EUR GS6 0,06 EUR
Widerstand 56 Ω Widerstand 220 Ω Widerstand 1 MΩ	¼ W 56 Ω 0,033 EUR (>= 10 Stück) ¼ W 220 Ω 0,033 EUR (>= 10 Stück) ¼ W 1,0 M 0,033 EUR (>= 10 Stück)
Kondensator 100 nF	MKS-2 100N 0,10 EUR
Leuchtdiode grün 13000 mcd Leuchtdiode weiß 18000 mcd	LED 5-13000 GN 0,48 EUR LED 5-18000 WS 0,40 EUR
SUB-D Stecker 25 polig Kappe für Stecker	D-SUB ST 25 0,13 EUR KAPPE CG25G 0,19 EUR
Fotowiderstand	A 905014 0,57 EUR A 906014 0,59 EUR

8 Interessante Links

http://www.mikrocontroller.net/
http://www.avr-asm-tutorial.net/index.html
ftp://ftp.rcs.ei.tum.de/pub/courses/rtproz.pdf Skript über Mikrocontroller, Vergleich zwischen Atmel und PIC
http://www.merl.com/reports/docs/TR2003-35.pdf Das Paper, in dem die Idee der LED als Lichtsensor beschrieben wird
http://www.csd-electronics.de/ günstige Bezugsquelle für den ATtiny13V
http://www.mcselec.com/bascom-avr.htm Bascom AVR
http://winavr.sourceforge.net/ Entwicklungsumgebung für C
http://ponyprog.sourdeforge.net Programmierumgebung
http://www.roboternetz.de/
http://www.atmel.com/dyn/resources/prod_documents/doc2535.pdf Datenblatt vom ATtiny13

9 Dank und ähnliches

Auch, wenn ich das hier zusammengeschrieben habe, habe ich mir das nicht ausgedacht. Wem die Ehre für die einzelnen „Erfindungen“ gebühren, könnt ihr nachschauen im Forum unter http://www.-geocache-forum.de/viewtopic.php?t=5753&sid=b11a21f7aa642152d4f31aa216bcf2a4.
Sicherlich werde ich mich beim Zusammenschreiben an der einen oder anderen Stelle vertan haben. Wer Fehler findet, darf sie behalten. Über eine kurze Information würde ich mich aber trotzdem freuen. Erreichen tut ihr mich über ralf.pongratz@gmx.de.

Ralf