



Ralf Pongratz

# Das Kochbuch

Reaktivlichter mit dem ATtiny 13V

Ralf Pongratz

# Das Kochbuch

Reaktivlichter mit dem ATtiny 13V

6. August 2011

[www.reaktivlicht.de](http://www.reaktivlicht.de)

---

# Inhaltsverzeichnis

<b>1</b>	<b>Grundlagen</b> .....	5
1.1	Programmierung .....	5
1.1.1	Programmieradapter .....	5
1.1.2	Programmiersoftware .....	6
<b>2</b>	<b>Schaltungs- und Programmaufbau</b> .....	11
2.1	Sensorik .....	11
2.1.1	Spannungsteiler mit einem Fotowiderstand .....	11
2.1.2	Kapazitive Eigenschaft einer Leuchtdiode .....	13
2.2	Aktorik .....	15
2.2.1	Lichtsignale mit einer Leuchtdiode .....	15
2.2.2	Schalten von Lasten .....	16
2.3	Verarbeitung .....	18
2.3.1	Grundgerüst .....	18
2.3.2	Nachtaktives Programm .....	19
2.4	Spezialitäten .....	20
2.4.1	Watchdog-Abschaltung .....	20
2.4.2	Umschalter zwischen Betriebsmodi .....	22
2.4.3	Parametrierbare Tagschwelle .....	23
2.4.4	TeachIn-Modus .....	24
2.5	Verpackung .....	24
2.6	Beispiele .....	24
2.6.1	Nachtaktiver Blinker mit A/D-Wandler und Watchdog-Abschaltung .....	25
	<b>Literaturverzeichnis</b> .....	29

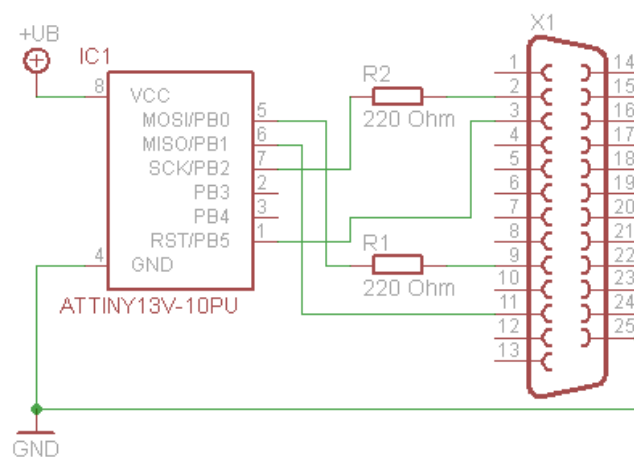


## Grundlagen

### 1.1 Programmierung

#### 1.1.1 Programmieradapter

Zum Überspielen des Programms auf den Microcontroller ist ein Anschluss an einen PC notwendig. Am einfachsten geschieht das Programmieren über die parallele Schnittstelle. Abbildung 1.1 zeigt einen einfachen Adapter hierfür.



**Abb. 1.1.** Programmieradapter für die parallele Schnittstelle.

X1 ist ein 25-poliger SUB-D-Stecker, der an die parallele Schnittstelle des PCs angeschlossen wird. IC1 ist der zu programmierende Microcontroller. Ferner

werden noch zwei Widerstände mit dem Wert 220 Ohm benötigt. Die Versorgungsspannung erhält der Microcontroller über eine externe Spannungsquelle. Das Kabel zum PC sollte möglichst kurz und abgeschirmt sein, damit keine Störungen in die Leitungen einkoppeln.

Sofern Pin 1 nicht genutzt wird und die Pins 5, 6 und 7 nicht auf feste Potentiale gezogen oder untereinander verbunden werden, kann der Programmieradapter in die Schaltung des Reaktivlichts integriert werden. So entfällt ein Umstecken des ICs bei der Programmierung.

### 1.1.2 Programmiersoftware

Für die Programmierung des Microcontrollers sind verschiedene Programme auf dem Markt. Hier wird das Programm „Bascom AVR“ mit dem parallelen Programmieradapter (Kap. 1.1.1) benutzt. Eine kostenlose Demoversion, bei der der Codeumfang auf 4 kB begrenzt ist, ist im Internet verfügbar<sup>1</sup>.

#### Voraussetzungen

Benötigt wird ein PC mit einer parallelen Schnittstelle (Druckerschnittstelle), der parallele Programmieradapter und die Programmierumgebung, mit der die Programme geschrieben und auf den Prozessor übertragen werden können.

Die parallele Schnittstelle muss im BIOS des Rechners auf ECP+EPP (In- und Output) eingestellt werden. Hier eine Beispieleinstellung:

Onboard Parallel Port: 378/IRQ7

Parallel Port Mode: ECP+EPP

ECP Mode Ust DMA: 3

Parallel Port EPP Type: EPP1.7

#### Bedienung von „Bascom AVR“

Alle Angaben basieren auf die Programmversion 1.11.8.3. Neuere Versionen können davon abweichen.

Zu Beginn wird „Bascom AVR“ gestartet und im Menü „File“ mittels „New“ ein neues Programmfenster erstellt. Danach werden der Programmieradapter und die Chip-Parameter eingestellt. Dazu wird im Menü „Options“ der Eintrag „Programmer“ ausgewählt.

<sup>1</sup> [http://www.mcselec.com/index.php?option=com\\_docman&task=cat\\_view&gid=99&Itemid=54](http://www.mcselec.com/index.php?option=com_docman&task=cat_view&gid=99&Itemid=54)

Nun wird der Reiter „Compiler“ und darin der Reiter „Chip“ (Abb. 1.2) aufgerufen. Im Drop-Down-Menü „Chip“ wird der Eintrag „attiny13.dat“ ausgewählt und in die darunterliegenden Felder folgende Werte eingetragen:

HW Stack = 2

Soft Stack = 8

Framesize = 24

Anschließend werden die Werte durch Klicken auf „Default“ gespeichert.

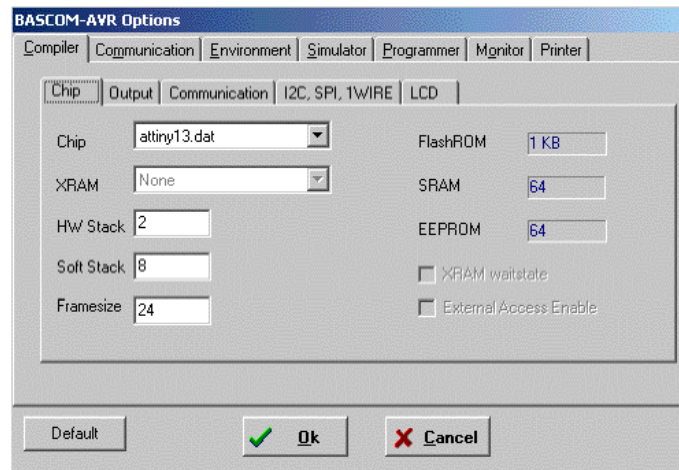


Abb. 1.2. Einstellen der Chip-Parameter.

Danach wird im Reiter „Programmer“ im Drop-Down-Menü „Programmer“ der Eintrag „Universal MCS Interface“ ausgewählt (Abb. 1.3). Im Reiter „Universal“ wird als Programmer „WinAVR and SP12“ ausgewählt. Anschließend wird der Dialog durch Klicken auf „OK“ geschlossen.

Als nächstes werden die Fuse-Bits eingestellt. Dies sind Speicherzellen, die das Grundverhalten des Microcontrollers festlegen. Dazu klickt man in der Icon-Leiste auf den kleinen grünen IC-Sockel und wählt den Punkt „Manual Program,“ aus (Abb. 1.4).

Im sich öffnenden Dialogfenster klickt man auf den Reiter „Lock and Fuse Bits“. Nun liest Bascom die Einstellungen der Fuse-Bits aus dem Controller aus und zeigt sie an (Abb. 1.5).

Um Veränderungen an den Fuse-Bits vorzunehmen, klickt man auf die entsprechende Zeile, in der sich das gewünschte bzw. zu ändernde Bit befindet. In dem sich öffnenden Drop-Down-Menü kann daraufhin der gewünschte Wert ausgewählt werden. Folgende Einstellungen müssen hier vorgenommen werden:

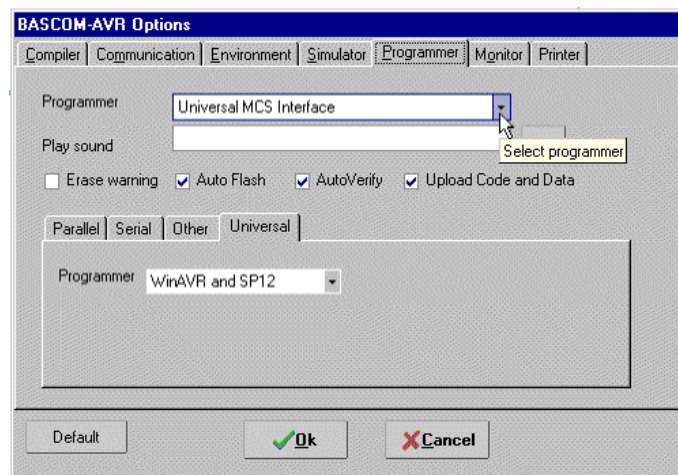


Abb. 1.3. Einstellen des Programmieradapters.

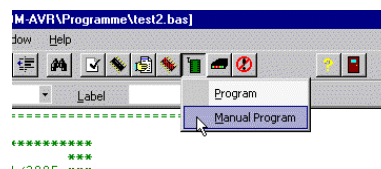


Abb. 1.4. Einstellen der Fuse-Bits.

Fusebit DCBA auf „1101:Int. Osc. 128 kHz; start-up time: 14 CK + 64 ms“  
 Fusebit E auf „1:Divide clock by 8, OFF“

Bitte die Einstellungen noch einmal sehr genau kontrollieren. Ist alles richtig, rechts auf „Write FS“ klicken. Die geänderten Einstellungen werden daraufhin in den Controller geschrieben.

Anschließend wird der Quellcode des Programms geschrieben. Ist dies erledigt, erfolgt die Kompilierung. Dazu im Menü „Program“ den Punkt „Compile“ auswählen (Abb. 1.6). Der Compiler startet nun. Enthält das Programm Fehler, wird eine entsprechende Meldung in der Fußzeile ausgegeben. Wurde das Programm erfolgreich kompiliert, kann es auf den Microcontroller übertragen werden. Dazu wird im Menü „Program“ auf den Punkt „Send to Chip“ geklickt. Im sich öffnenden Fenster muss dann aus dem Menü „Chip“ der Eintrag „Autoprogram“ aufgerufen werden (Abbl 1.7). Das zuvor kompilierte Programm wird nun in den Programmspeicher des Controllers geschrieben.

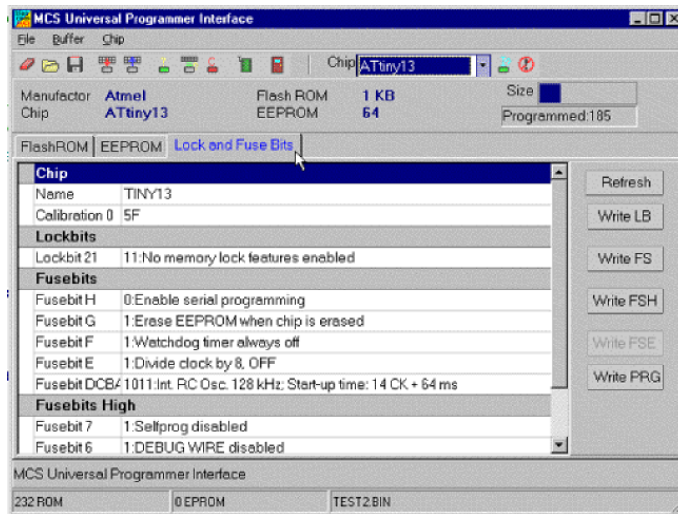


Abb. 1.5. Einstellen der Fuse-Bits.

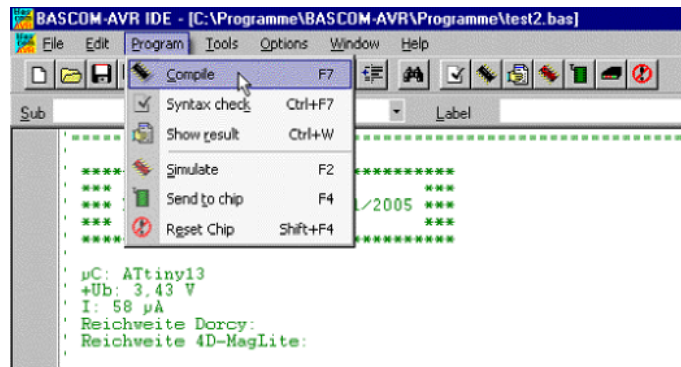


Abb. 1.6. Kompilieren des Programms.

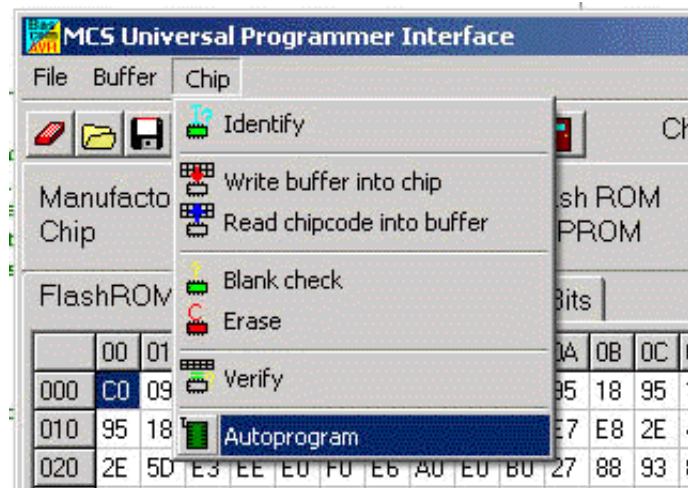
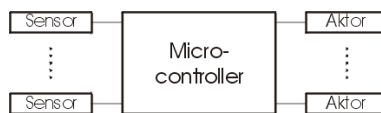


Abb. 1.7. Übertragen des Programms.

---

## Schaltungs- und Programmaufbau

Ein Reaktivlicht besteht üblicherweise aus drei verschiedenen Arten von Elementen. Im Zentrum steht der Microcontroller, der die Eingangssignale von einem oder mehreren Sensoren verarbeitet und entsprechend die Aktoren ansteuert (Abb. 2.1). In seltenen Fällen kann es auch vorkommen, dass mehrere Microcontroller genutzt werden, die untereinander kommunizieren. Dies ist allerdings eher die Ausnahme. Die einzelnen Elemente werden in den folgenden Abschnitten besprochen.



**Abb. 2.1.** Schematischer Aufbau eines Reaktivlichts.

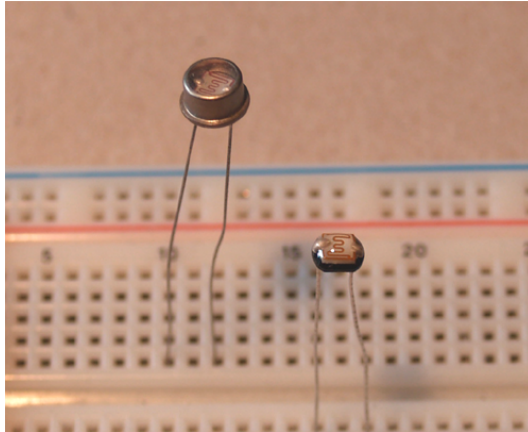
### 2.1 Sensorik

Die Aufgabe der Sensorik ist es, Daten zu sammeln, die das Programm im Microcontroller verarbeitet. Diese Daten können zum einen Ereignisse auslösen, zum anderen Zustände des Programms ändern (Umschaltung zwischen Tag- und Nachtmodus, Aktivierung während einer bestimmten Zeit).

#### 2.1.1 Spannungsteiler mit einem Fotowiderstand

Ein Fotowiderstand (Abb. 2.2) ist die einfachste Möglichkeit, die Helligkeit abzufragen. Fällt viel Licht auf ihn, so ist sein Widerstand gering, bei Dunkelheit ist er hoch. Über einen Spannungsteiler kann diese Widerstandsänderung in

eine Spannung umgesetzt werden, die durch den Analog-Digital-Wandler im Microcontroller eingelesen wird. Während sich der Widerstand im beleuchteten Zustand im Kiloohm-Bereich befindet, steigt er im unbeleuchteten Zustand auf mehrere Megaohm<sup>1</sup>. Dabei ist auf die Trägheit zu achten. In den Datenblättern findet man üblicherweise den Widerstandswert nach einer Minute Dunkelheit ( $R_{01}$ ) und nach fünf Minuten ( $R_{05}$ ).



**Abb. 2.2.** Fotowiderstände. Links der Typ A 1060, rechts A 9050 14.

In Verbindung mit einem zweiten (festen) Widerstand  $R_2$  kann ein Spannungsteiler aufgebaut werden (Abb. 2.3). Mit einer Versorgungsspannung  $U_B$  ergibt sich eine Ausgangsspannung von

$$U_S = U_B \cdot \frac{R_2}{R_1 + R_2}. \quad (2.1)$$

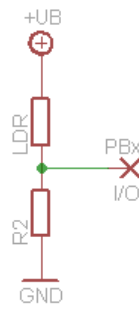
Somit ergibt sich im beleuchteten Zustand eine hohe Ausgangsspannung, im unbeleuchteten eine niedrige. Um eine hohe Empfindlichkeit zu erreichen, sollte der Wert von  $R_2$  sich etwa in der Mitte zwischen dem Hell- und dem Dunkelwiderstand von  $R_1$  befinden.

```

1 Config Adc = Single , Prescaler = Auto
2 Config Portb = &B00x0xxxx
3 Portb = &Bxxx0xxxx
4
5 Dim Ldr As Integer

```

<sup>1</sup> Das Datenblatt des in diesem Buch eingesetzten Fotowiderstandes A 9050 14 kann unter [http://www.perkinelmer.de/CMSResources/Images/44-3571DTS\\_PhotoCellsA9050.pdf](http://www.perkinelmer.de/CMSResources/Images/44-3571DTS_PhotoCellsA9050.pdf) heruntergeladen werden.



**Abb. 2.3.** Spannungsteiler mit einem Fotowiderstand. PBx ist ein Eingang des Microcontrollers.

```

6
7 Start Adc
8 Ldr = Getadc(2)
9 Stop Adc

```

Zeile 1 konfiguriert den Analog-Digital-Wandler, mit dem die Ausgangsspannung des Spannungsteilers gemessen wird. In diesem Beispiel ist sie an PB4 (Pin 3) angeschlossen. Daher wird in Zeile 2 und 3 PB4 als Eingang definiert. Die mit 'x' gekennzeichneten Bits können, je nach weiterer Beschaltung, beliebige Werte annehmen. In Zeile 5 wird die Integer-Variable definiert, in der der eingelesene Helligkeitswert gespeichert wird.

Das Einlesen des Spannungswertes geschieht in Zeile 8. `Ldr` nimmt dabei je nach Beleuchtungszustand Werte zwischen 0 und 1023 an. In den Zeilen 7 und 9 wird der Analog-Digital-Wandler ein- und wieder ausgeschaltet. Spielt der Stromverbrauch der Schaltung keine Rolle, können diese Befehle entfallen.

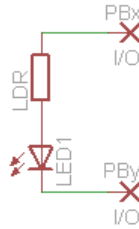
### 2.1.2 Kapazitive Eigenschaft einer Leuchtdiode

Eine Leuchtdiode (LED), die in Sperrrichtung betrieben wird, wirkt wie ein kleiner Kondensator, dessen Kapazität abhängig von der Helligkeit ist. Im beleuchteten Zustand ist sie kleiner als im unbeleuchteten [1]. Diese Kapazität unterliegt einer Selbstentladung.

Für die Messung der Helligkeit wird diese Kapazität aufgeladen und danach die Stromzufuhr unterbrochen. Nach einer bestimmten Zeit wird die verbleibende Spannung nachgemessen. Aufgrund der Selbstentladung wird sie kleiner als die ursprünglich aufgeladene Spannung sein. Je größer die Kapazität (also je weniger Licht auf die LED fällt), desto höher ist die verbliebene Spannung.

Durch den schmalen Abstrahlwinkel der LED ist gewährleistet, dass der Sensor nur bei direkter Beleuchtung sein Signal ändert. Es hat sich herausgestellt,

dass superhelle rote LEDs mit transparentem Gehäuse am Besten geeignet sind.



**Abb. 2.4.** Helligkeitsmessung mit einer Leuchtdiode über die kapazitive Eigenschaft. PBx und PBy sind zwei Ein- und Ausgänge des Microcontrollers.

```

1 Config Portb = &B00x11xxx
2 Portb = &Bxxx00xxx
3
4 Dim Led As Bit
5
6 Portb.3 = 0
7 Portb.4 = 1
8 Waitus 1
9 Config Portb.4 = Input
10 Portb.4 = 0
11 Waitus 1500
12 Led = Pinb.4
13 Config Portb.4 = Output
14 Portb.4 = 0

```

Zeile 1 und 2 definieren PB3 und PB4 (Pin 2 und 3) als Ausgänge und setzt sie auf low. Die Variable `Led` in Zeile 4 enthält während des Programmablaufs die Information, ob 'hell' oder 'dunkel' detektiert wurde.

Die Zeilen 6 bis 8 laden die Kapazität der Leuchtdiode auf: PB3 wird auf Masse gelegt, PB4 auf  $+U_B$ . Nach einer Wartezeit ist die Kapazität geladen. Nun wird PB4 als Eingang definiert und mit 0 beschrieben, um das Laden zu beenden. Nach einer Wartezeit (Zeile 11) wird der Zustand des Eingangs abgefragt und in `Led` gespeichert. Hinterher wird PB4 wieder als Ausgang geschaltet und auf Masse gelegt. So kann während des Rest des Programms die Leuchtdiode durch Setzen von PB3 geschaltet werden. Eine Veränderung der Wartezeit in Zeile 11 verändert die Schaltschwelle.

## 2.2 Aktorik

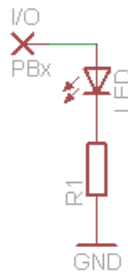
Mit Hilfe von Aktoren gibt ein Reaktivlicht Signale aus. Der weitaus gebräuchteste ist eine Leuchtdiode, aber auch die Ansteuerung einer Infrarotdiode oder das Aktivieren einer externen Schaltung sind denkbar.

### 2.2.1 Lichtsignale mit einer Leuchtdiode

Die einfachste Möglichkeit für den Microcontroller, Signale auszugeben, ist eine Leuchtdiode. Zusammen mit einem Vorwiderstand wird sie an einen der Ausgänge des Microcontrollers angeschlossen (Abb. 2.5). Der Wert des Vorwiderstandes  $R_1$  beträgt dabei

$$R_1 = \frac{U_B - U_F}{I_F}. \quad (2.2)$$

Dabei ist  $U_B$  die Betriebsspannung des Microcontrollers,  $U_F$  die Spannung, die über der Leuchtdiode abfällt, und  $I_F$  der maximal zulässige Strom<sup>2</sup>. Maximal liefert ein Ausgang 40 mA Strom.



**Abb. 2.5.** Ausgabe von Lichtsignalen mit Hilfe einer Leuchtdiode. PBx ist ein Ausgang des Microcontrollers.

```

1 Config Portb = &B00xx1xxx
2 Portb = &Bxxxx0xxx
3
4 Portb.3 = 1
5 Portb.3 = 0

```

In den ersten beiden Zeilen wird PB3 (Pin 2) als Ausgang konfiguriert. Durch ein Setzen des Ausgangsbits auf 1 (Zeile 4) wird die Leuchtdiode angeschaltet. Ein Setzen auf 0 (Zeile 5) schaltet sie wieder aus.

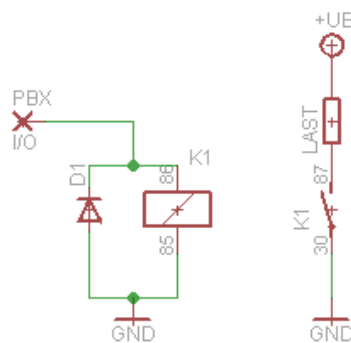
<sup>2</sup>  $U_F$  und  $I_F$  findet man in den Datenblättern zu der Leuchtdiode. Typischerweise beträgt  $I_F$  20 mA,  $U_F$  ist abhängig von der Leuchtdiode.

### 2.2.2 Schalten von Lasten

Der maximale Strom, den ein eingeschalteter Ausgang liefern kann, ist auf 40 mA begrenzt. Sollen Lasten geschaltet werden, die höhere Ströme benötigen, sind Bauteile zur Verstärkung des Stromes zwischen zu schalten.

#### Schalten per Relais

Die einfachste Möglichkeit ist es, die Last über ein Relais zu schalten. Dabei sind Reedrelais herkömmlichen Relais vorzuziehen, da sie einen geringeren Schaltstrom benötigen. Allerdings ist der Strom, den die Last verbrauchen darf, ebenfalls geringer. In Abbildung 2.6 ist ein Schaltplan hierfür gezeichnet. Links ist der Schaltkreis mit der Spule des Relais K1 zu sehen, rechts die Last und der Schließer des Relais.



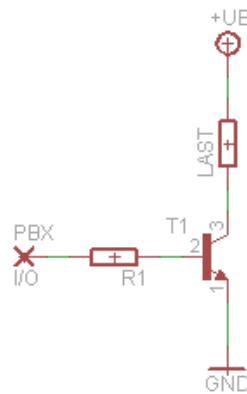
**Abb. 2.6.** Schalten von Lasten mit Hilfe eines Relais. PBx ist ein Ausgang des Microcontrollers.

Parallel zum Relais muss eine Diode in Sperrrichtung geschaltet werden. Wenn das Relais ausgeschaltet wird, wird durch das Zusammenbrechen des Magnetfeldes in der Relaisspule eine Spannung induziert, die den Ausgang des Microcontrollers zerstören kann. Über die Diode wird diese Spannung gefahrlos abgebaut. Üblicherweise reicht hierfür eine gewöhnliche Diode (z. B. 1N4148).

Beachtet werden muss, dass der Strom, der zum Schalten des Relais nötig ist, die 40 mA des Ausgangs nicht überschreitet. Darüber hinaus muss der Strom, den das Relais schalten kann, kleiner sein als der Laststrom des Verbrauchers.

### Schalten per Transistor

Eine kompliziertere, aber platzsparendere Möglichkeit ist es, die Last über einen Transistor zu schalten. Abbildung 2.7 zeigt den Schaltplan. Zum Einsatz kann ein beliebiger NPN-Transistor kommen, der die zur Last passenden Parameter bietet.



**Abb. 2.7.** Schalten von Lasten mit Hilfe eines Transistors. PBx ist ein Ausgang des Microcontrollers.

Der Strom, der zur Basis des Transistors reinfließen muss, ergibt sich aus

$$I_B = \frac{I_{Last}}{h_{FE}}, \quad (2.3)$$

wobei  $I_{Last}$  der Laststrom,  $h_{FE}$  der Verstärkungsfaktor des Transistors ist. Der Verstärkungsfaktor ist aus dem Datenblatt ersichtlich. Zur Sicherheit sollte der minimal angegebene Wert benutzt werden. Wichtig ist, dass der Basisstrom  $I_B$  kleiner als 40 mA ist. Andernfalls ist ein Transistor mit einer größeren Verstärkung zu wählen.

Zum Einstellen des Basisstromes ist ein Vorwiderstand  $R_1$  nötig. Er berechnet sich aus

$$R_1 = \frac{U_B - U_{BE}}{I_B}. \quad (2.4)$$

$U_B$  ist dabei die Betriebsspannung des Microcontrollers und  $U_{BE}$  die Sättigungsspannung zwischen Basis und Emitter des Transistors<sup>3</sup>.

<sup>3</sup> Bei einem Silizium-Transistor ist  $U_{BE}$  normalerweise 0,7 V.

Bei der Auswahl des Transistors muss darauf geachtet werden, dass der maximale Kollektorstrom  $I_C$  mindestens dem Laststrom entspricht. Ferner hat jeder Transistor einen Leckstrom  $I_{CES}$ , der permanent zwischen Kollektor und Emittter fließt. Dieser sollte, wenn die Schaltung längere Zeit über Batterie betrieben werden soll, möglichst klein sein.

## 2.3 Verarbeitung

Um die Signale der Sensoren mit den Ausgängen zu verknüpfen, muss ein Programm für den Microcontroller geschrieben werden.

### 2.3.1 Grundgerüst

Die einfachste Möglichkeit besteht aus einer Endlosschleife, in der eine Fallunterscheidung steht. Dieses Grundgerüst stellt schon ein lauffähiges Programm dar, das auch in jeder umfangreicheren Variante wiederzufinden ist.

```

1 Dim hell As Bit
2
3 Do
4   Gosub Sensorabfrage
5
6   If hell = 1 Then
7     ' Hier kann z. B. eine LED aufblinken
8   End If
9 Loop
10
11 Sensorabfrage:
12   ' Hier wird der Beleuchtungszustand des
13   ' Sensors abgefragt. Ist er beleuchtet,
14   ' wird die Variable hell auf true gesetzt,
15   ' andernfalls auf false.
16   return

```

In Zeile 1 wird eine Variable definiert, in der der Zustand des Sensors gespeichert wird. Die Abfrage findet in der Unteroutine `Sensorabfrage` statt. Je nach Sensor ist hier eine andere Routine zu wählen. Welche Routine zu welchem Sensor passt, ist dem Kapitel 2.1 zu entnehmen.

Zeile 3 und 9 stellen die Endlosschleife dar. Der dazwischenstehende Code wird kontinuierlich ausgeführt. Zuerst wird in Zeile 4 der Zustand des Sensors abgefragt. Wenn er beleuchtet ist, leitet die if-Abfrage (Zeile 6) weiter zur Aktion. Verschiedene Möglichkeiten sind in Kapitel 2.2 aufgelistet.

### 2.3.2 Nachtaktives Programm

Damit das Reaktivlicht nur nachts ausgelöst werden kann und tagsüber nicht dauerhaft blinkt, soll Tageslicht erkannt werden. Dies geschieht, indem die Beleuchtungsdauer gemessen wird. Überschreitet sie eine festgelegte Dauer, so wird angenommen, dass Tag ist und das Licht blinkt nicht.

```

1 Dim cntH As Byte
2 Dim cntD As Byte
3 Dim lst As Bit
4
5 Do
6   Gosub Sensorabfrage
7
8   If hell = 1 And cntH < 255 Then
9     cntH = cntH + 1
10  End If
11
12  If hell = 0 And cntD < 255 Then
13    cntD = cntD + 1
14  End If
15
16  If hell = 0 Then
17    If lst = 1 And cntH < 50 And cntD > 5 Then
18      ' Hier kann z. B. eine LED
19      ' aufblinken.
20      cntH = 0
21      lst = 0
22    End If
23  Else
24    cntD = 0
25    lst = 1
26  End If
27 Loop

```

Es werden drei weitere Variablen `cntH`, `cntD` und `lst` eingeführt. In der ersten werden die Zyklen gezählt, in der der Sensor den Zustand hell detektiert. Sie wird in den Zeilen 8-10 inkrementiert, wenn der Sensor hell detektiert hat und die Variable noch nicht an ihrer Überlaufgrenze<sup>4</sup> angekommen ist. Die zweite Variable zählt die Zyklen, in denen der Sensor den Zustand dunkel detektiert. Sie wird in den Zeilen 12-14 inkrementiert. Die dritte Variable speichert den Beleuchtungszustand des vorangegangenen Durchlaufes, sofern er hell war. Ein dunkler Durchlauf wird nur nach einer Aktion gespeichert.

<sup>4</sup> Eine Bytevariable kann Werte zwischen 0 und 255 annehmen.

Geht der Zustand des Sensors auf dunkel (Zeile 16), wird geschaut, ob im vorangegangenen Durchlauf noch der Zustand hell erkannt wurde (Zeile 12). Außerdem wird überprüft, ob die Länge der Hellphase kürzer als ein Schwellwert (hier 50) und die Länge der Dunkelphase größer als ein Schwellwert (hier 5) war. Die letzte Bedingung sorgt dafür, dass in der Dämmerung das Licht nicht dauerhaft angetriggert wird. Falls alle Bedingungen zutreffen, wird eine Aktion eingeleitet. Außerdem wird der Hellzähler zurückgesetzt.

Aus diesem Programm ergibt sich, dass die Aktion erst eintritt, wenn die Beleuchtung des Sensors wieder aufhört.

## 2.4 Spezialitäten

In diesem Abschnitt werden einige Besonderheiten vorgestellt, mit denen das Programm verfeinert werden kann.

### 2.4.1 Watchdog-Abschaltung

Anstatt den Microcontroller mit dem Befehl `waitms` eine Wartezeit einlegen zu lassen, ist es möglich, ihn während dieser Zeit in einen Betriebszustand zu versetzen, in dem nur sehr wenig Strom verbraucht wird. Realisiert wird dies über den Watchdog-Timer. Im Ruhezustand wird der Stromverbrauch so auf wenige  $\mu\text{A}$  senken.

### Funktionsweise des Watchdog-Timers

Der Watchdog-Timer ist ein im Microcontroller integrierter Baustein und wird über Register (Abb. 2.8) konfiguriert. Er ist ein Zähler, der die Zyklen eines eingebauten 128 kHz-Oszillators zählt. Ist der eingestellte Zielwert erreicht, wird ein Interrupt und/oder ein Reset ausgelöst. In diesem Fall wird der Interrupt genutzt, um den Microcontroller aus dem Schlafmodus zu wecken. Die Interrupts müssen dafür aktiviert sein.

Bit	7	6	5	4	3	2	1	0	
	WDTIF	WDTIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

**Abb. 2.8.** Konfigurationsregister des Watchdog-Timers.

WDP3	WDP2	WDP1	WDP0	Anzahl der Oszillator-Zyklen	Typische Timeout-Zeit bei $V_{CC} = 5,0V$
0	0	0	0	2K (2048) Zyklen	16 ms
0	0	0	1	4K (4096) Zyklen	32 ms
0	0	1	0	8K (8192) Zyklen	64 ms
0	0	1	1	16K (16384) Zyklen	0,125 s
0	1	0	0	32K (32768) Zyklen	0,25 s
0	1	0	1	64K (65536) Zyklen	0,5 s
0	1	1	0	128K (131072) Zyklen	1 s
0	1	1	1	256K (262144) Zyklen	2 s
1	0	0	0	512K (524288) Zyklen	4 s
1	0	0	1	1024K (1048576) Zyklen	8 s

Tabelle 2.1. Timeout-Zeit des Watchdog-Timers

WDTON	WDE	WDTIE	Modus	Aktion nach Timeout
0	0	0	Gestoppt	keine
0	0	1	Interrupt-Modus	Interrpt
0	1	0	System-Reset-Modus	Reset
0	1	1	Interrupt- und System-Reset-Modus	Interrupt, danach Reset
1	x	x	System-Reset-Modus	Reset

Tabelle 2.2. Modus des Watchdog-Timers

### Benutzung des Watchdog-Timers

```

1 Wdtcr = &B11010011
2 Enable Interrupts
3
4 Reset Watchdog
5 Powerdown

```

In der ersten Zeile wird der Watchdog-Timer konfiguriert. Der Timeout-Wert liegt bei 0,125 s. Beim Erreichen dieses Wertes wird ein Interrupt ausgelöst. In Zeile 2 werden die Interrupts aktiviert. Diese Zeilen müssen im Programm nur einmal eingefügt werden unabhängig davon, wie häufig der Watchdog-Timer genutzt wird. Nur wenn im Programm die Wartezeit verändert werden soll, muss die Zeile 1 entsprechend angepasst erneut eingefügt werden.

Im Programm selber wird an der Stelle, an der die Wartezeit eingelegt wird, die Zeilen 4 und 5 eingefügt. Zuerst wird der Timer zurückgesetzt, sodass er beginnt zu zählen. Danach geht der Microcontroller in den Schlafmodus. Ist der Timer abgelaufen, wird der Controller durch den Interrupt wieder geweckt.

### 2.4.2 Umschalter zwischen Betriebsmodi

Gerade für Serienfertigungen oder Wartungszwecke ist es schön, wenn man zwischen verschiedenen Betriebsmodi wechseln kann. Aufgrund der Wetterbeständigkeit der Verpackung scheiden normale Schalter aber aus. Besser geeignet sind Reedkontakte. Sie bestehen aus zwei Metallzungen, die mit einem kleinen Abstand parallel in einem Glasröhrchen befestigt sind. Wird der Reedkontakt in ein Magnetfeld gebracht, schließen sich diese Kontakte. Somit ist es möglich, durch die Gießharzummhüllung der Schaltung hindurch mittels eines kleinen Magneten den Schalter zu betätigen.

Der Schaltungsaufbau ist identisch mit dem des Spannungsteilers mit einem Fotowiderstand (Abb. 2.3). Allerdings wird anstelle des Fotowiderstandes der Reedkontakt eingefügt.

```

1 Config Portb = &B00XX10XX
2 Portb = &BXXXX00XX
3
4 Wdtcr = &B11101110
5 Enable Interrupts
6
7 Const Maxmodus = 4
8 Dim Modus As Integer
9 Dim Old As Bit
10
11 Old = 0
12
13 Do
14   If Portb.2 = 1 And Old = 0 Then
15     Portb.3 = 1
16     Modus = Modus + 1
17     Modus = Modus Mod Maxmodus
18     Reset Watchdog
19     Powerdown
20     Portb.3 = 0
21   End If
22   Old = Portb.2
23
24   If Modus = 0 Then
25     ' Aktion in Modus 0
26   ElseIf Modus = 1 Then
27     ' Aktion in Modus 1
28   ElseIf Modus = 2 Then
29     ' Aktion in Modus 2
30   ElseIf Modus = 3 Then
31     ' Aktion in Modus 3

```

```

32   End If
33 Loop

```

Für die Abfrage des Schaltzustandes muss lediglich der Eingangswert ausgelesen werden. In dem Beispielcode ist der Spannungsteiler mit dem Reedkontakt an Port 2 (Pin 7) gehängt. Dieser Port wird in den ersten beiden Zeilen als Eingang definiert. Die Zeilen 4 und 5 konfigurieren den Watchdog-Timer (Kapitel 2.4.1), der zur Entprellung des Signals benötigt wird. Die Anzahl der Betriebsmodi wird in der Konstanten `maxModus` festgelegt. Die Variable `modus` dient der Speicherung des aktuellen Modus. In Zeile 9 wird eine Variable definiert, die den Schaltzustand aus dem letzten Scheifendurchlauf speichert. Sie wird vor dem Beginn der Schleife auf 0 initialisiert.

Die Zeilen 14 bis 21 werden ausgeführt, wenn der Reedkontakt geschlossen ist nachdem er im vorherigen Durchlauf noch offen war. Zur optischen Ausgabe, dass ein Moduswechsel stattfindet, wird eine an Port 3 angeschlossene Leuchtdiode in Zeile 15 eingeschaltet. Sie wird am Ende des Wechsels in Zeile 20 wieder ausgeschaltet. In Zeile 16 wird der Modus gewechselt. Ist der höchste Modus erreicht, wird der Zähler automatisch auf 0 und damit den Modus 0 zurückgesetzt. Danach wird der Controller für 1 s in den Schlafmodus versetzt. Ohne dies würden mehrere Moduswechsel in Folge durchgeführt werden. Der Grund liegt darin, dass der Reedkontakt prellt. Er schaltet nicht nur einmal ein, sondern während des Schaltens springt er mehrfach zwischen offen und geschlossen hin und her. Dies wird durch die Wartezeit ausgeblendet. Nach dem Moduswechsel wird der aktuelle Zustand des Reedkontaktes in der Variablen `old` gespeichert.

In den Zeilen 24 bis 32 wird der Modus ausgewertet. Je nach aktuellem Stand werden unterschiedliche Blöcke der `if`-Schleife ausgeführt.

Beim Eingießen der Schaltung in Kunstharz ist auf die Zerbrechlichkeit der Reedkontakte zu achten. Sie dürfen niemals nur zum Teil im Gießharz sein, da sie die Spannungen beim Aushärten nicht aushalten, sondern müssen immer komplett bedeckt werden.

### 2.4.3 Parametrierbare Tagschwelle

Viele Programmvarianten enthalten einen Tagmodus, bei dem der Microcontroller in einen stromsparenden Schlafmodus verfällt, wenn dauerhaft ein Helligkeitswert über einer bestimmten Schwelle liegt. Dieser Schwellwert ist normalerweise fest einprogrammiert. Mit Hilfe eines Reedkontaktes (Kapitel 2.4.2) lässt er sich als frei parametrierbarer Parameter realisieren.

```

1 Dim Tagschwelle As Integer
2 Tagschwelle = 800
3

```

```

4 Do
5
6   ' Hier steht der normale Code des Reaktivlichts
7
8   If Portb.2 = 1 Then
9     Portb.3 = 1
10    Tagschwelle = Getadc(2)
11    Wdtcr = &B11010101
12    Reset Watchdog
13    Powerdown
14    Portb.3 = 0
15  End If
16
17 Loop

```

Zeile 1 und 2 definieren eine Variable, die anstelle der Konstanten als Tagschwelle genutzt wird. Diese Zeilen müssen in den Kopfbereich des Programms geschrieben werden. Innerhalb der Schleife steht dann in den Zeilen 8 bis 15 der eigentliche Programmcode. An Port 2 (Pin 7) hängt der Spannungsteiler mit dem Reedkontakt. Er muss entsprechend als Eingang parametrieren werden. Wenn der Kontakt geschlossen ist, wird dieses Codestück ausgeführt. Zuerst schaltet es die LED an Port 3 an. Dies geschieht als Rückmeldung an den Benutzer. Danach wird der aktuelle Helligkeitswert als neue Tagschwelle gespeichert. Anschließend wird eine halbe Sekunde gewartet, bevor die LED wieder ausgeschaltet wird. Der Parametrierungsvorgang ist damit abgeschlossen.

Falls die Schaltung sich im Tagmodus befindet, ist darauf zu achten, dass der Reedkontakt lange genug geschlossen bleibt, da das Parametrieren nur funktioniert, wenn die Schaltung aus dem Schlafmodus erwacht. Dies geschieht in der Regel nur alle 8 Sekunden.

#### 2.4.4 TeachIn-Modus

## 2.5 Verpackung

## 2.6 Beispiele

Hier sind einige Beispiele von Reaktivlicht-Programmen aufgeführt, die aus den Bausteinen, die in diesem Kapitel beschrieben wurden, zusammengesetzt werden können.

### 2.6.1 Nachtaktiver Blinker mit A/D-Wandler und Watchdog-Abschaltung

Dieses Reaktivlicht besitzt als Sensor einen Fotowiderstand, dessen Wert über den A/D-Wandler eingelesen wird (Kapitel 2.1.1). Der Spannungsteiler ist an Port 4 (Pin 3) angeschlossen. Als Ausgabe dient eine einfache Leuchtdiode (Kapitel 2.2.1), die an Port 3 (Pin 2) angeschlossen ist.

```

1 $regfile = "Attiny13.DAT"
2 $crystal = 16000
3 $hwstack = 2
4
5 Config Adc = Single , Prescaler = Auto
6 Config Portb = &B00001000
7 Portb = 0
8 Stop Ac
9 Wdtcr = &B11010011
10 Enable Interrupts
11
12 Const Schwelle = 50
13 Const Tagschwelle = 800
14 Const Zwangsimpuls = 8
15 Dim A As Byte
16 Dim Tagzaehler As Byte
17 Dim Schlafzaehler As Byte
18 Dim Ldr As Integer
19 Dim Alt As Integer
20 Dim Merker As Integer
21
22 Do
23   Reset Watchdog
24   Powerdown
25   Start Adc
26   Ldr = Getadc(2)
27   Stop Adc
28   Merker = Ldr - Alt
29   Alt = Ldr
30   If Merker > Schwelle Then
31     Gosub Blinken
32   End If
33   If Ldr > Tagschwelle Then
34     If Tagzaehler < 255 Then
35       Tagzaehler = Tagzaehler + 1
36     End If
37   Else

```

```

38     Tagzaehler = 0
39     End If
40     If Tagzaehler > 200 Then
41         Gosub Pause
42     End If
43 Loop
44
45 Blinken:
46     For A = 0 To 10
47         Portb.3 = 1
48         Reset Watchdog
49         Powerdown
50         Portb.3 = 0
51         Reset Watchdog
52         Powerdown
53     Next A
54     Alt = 1023
55 Return
56
57 Pause:
58     Wdtcr = &B11110001
59     Reset Watchdog
60     Powerdown
61     Wdtcr = &B11010011
62     Schlafzaehler = Schlafzaehler + 1
63     If Schlafzaehler = Zwangsimpuls Then
64         Portb.3 = 1
65         Reset Watchdog
66         Powerdown
67         Portb.3 = 0
68         Schlafzaehler = 0
69     End If
70 Return
71
72 End

```

In den Zeilen 1 bis 3 werden allgemeine Einstellungen an der Hardware vorgenommen. Zuerst wird dem Compiler mitgeteilt, um welchen Prozessortyp es sich handelt. Danach wird die Frequenz des internen Oszillators gesetzt. Zum Schluss wird der Stack auf 2 gesetzt, damit für die Variablen des Programms genügend Platz zur Verfügung steht. Dies hat allerdings zur Folge, dass die Verschachtelungstiefe von Funktionsaufrufen maximal zwei betragen darf, was für dieses Programm jedoch ausreichend ist.

Danach wird der Microcontroller über die Register konfiguriert. Zuerst wird der Analog-Digital-Wandler eingestellt, mit dem der Helligkeitsgrad eingestellt wird. Danach wird der Port 3 als Ausgang konfiguriert und auf low gesetzt. Die restlichen Port sind Eingänge. In Zeile 8 wird der Analog-Komparator abgestellt, um Strom zu sparen. Er wird in diesem Programm nicht benötigt. Zeile 9 und 10 stellen den Watchdog-Timer, über den die Wartezeiten realisiert werden, auf 0,125 s und in den Interrupt-Modus.

Nun werden die Konstanten und Variablen definiert, die im Programm benötigt werden. **Schwelle** gibt die Mindestgröße der Helligkeitsänderung zwischen zwei Durchläufen an, damit das Reaktivlicht ausgelöst wird. Hiermit kann die Empfindlichkeit der Schaltung eingestellt werden. **Tagschwelle** gibt den Wert vor, oberhalb dessen die Schaltung in den Tagmodus wechselt. Um den Tagmodus anzuzeigen, kann mit der Konstante **Zwangsimpuls** festgelegt werden, nach wieviel Schlafzyklen (hier etwa 8 s) ein Impuls auf der LED ausgegeben werden soll. **A** ist eine Zählvariable, die in der Unterprozedur **Blinken** genutzt wird. **Tagzaehler** zählt während des Programmablaufs die aufeinanderfolgenden Zyklen, in denen der Helligkeitswert über der Tagschwelle liegt. **Schlafzaehler** zählt die Schlafzyklen für die Ausgabe des Kontrollimpulses auf der LED. Der aktuelle Helligkeitswert wird in der Variablen **Ldr**, derjenige des vorangegangenen Durchlauf in **Alt** gespeichert. **Merker** ist eine Variable, in der die Differenz zwischen diesen beiden Werten gespeichert wird.

Die Zeilen 22 bis 43 enthalten das Hauptprogramm. Zu Beginn wird der Controller für 0,125 s in den Schlafmodus versetzt. Dadurch wird die Wahrscheinlichkeit einer Fehlauflösung verringert, da beim Wechsel zwischen unbeleuchtetem und beleuchtetem Sensor ein größerer Unterschied ist als bei kontinuierlicher Abfrage. Danach wird in Zeile 25 bis 27 der Analog-Digital-Wandler gestartet, der Helligkeitswert eingelesen und der Wandler wieder ausgeschaltet. Es wird die Differenz zum vorangegangenen Zyklus berechnet und anschließend der jetzige Helligkeitswert für den nächsten Durchlauf in der Variablen **Alt** gespeichert. Ist der Helligkeitsunterschied größer als die eingestellte Schwelle, wird die Unteroutine **Blinken** aufgerufen. Ab Zeile 33 werden die Bedingungen für den Tagmodus überprüft. Ist der Helligkeitswert über der Tagschwelle, wird der Tagzaehler hochgezählt, ansonsten zurück auf 0 gesetzt. Sollte der Tagzähler den Wert 200 erreichen, was bedeutet, dass über 200 Zyklen a 0,125 s Tag detektiert wurde, wird die Unterprozedur **Pause** aufgerufen.

Die Unterprozedur **Blinken** befindet sich in den Zeilen 45 bis 54. Hier wird zehnmal in Folge die Port 3, an dem die LED angeschlossen ist, angeschaltet, mittels des Watchdog-Timers 0,125 s gewartet, der Ausgang wieder ausgeschaltet und erneut gewartet. Im Anschluss an diesen Vorgang wird der Wert der gemessenen Helligkeit auf den Maximalwert gesetzt, sodass eine Doppelauslösung verhindert wird.

Die Zeilen 57 bis 70 enthalten die Prozedur **Pause**. Damit wird der Microcontroller während des Tagmodus für 8 s ausgeschaltet. Zu Beginn wird die Zeit des Watchdog-Timers auf den Maximalwert gesetzt. Anschließend wird der Controller in den Schlafmodus versetzt. Wenn er daraus wieder erwacht ist, wird die Zeit des Watchdog-Timers wieder zurückgesetzt und der Schlafzähler inkrementiert. Erreicht der Schlafzähler den Wert, der in der Konstanten **Zwangsimpuls** angegeben ist, wird ein einzelner Lichtblitz auf der LED ausgegeben und der Schlafzähler auf 0 zurückgesetzt.

---

## Literaturverzeichnis

1. P. Dietz, W. Yeraunus, D. Leigh, „*Very Low-Cost Sensing and Communication Using Bidirectional LEDs*“.