Ralf Pongratz

7 Segment Grundversion

Reaktivlicht mit 7 Segment Anzeige

Grundversion
Reaktivlicht mit 7 Segment Anzeige

Ralf Pongratz

13. September 2013

Inhaltsverzeichnis

Teil I.
Die Schaltung

1. Funktionsbeschreibung

Diese Schaltung ist die Grundversion der Reaktivlichter. Sie ist einfach mit nur wenigen Bauteilen zu bauen und ohne Parametrierung einsatzbereit. Statt einer einfachen Leuchtdiode (LED) wird bei dieser Variante eine 7 Segment Anzeige genutzt. Somit lässt sich eine Folge von Zahlen und Punkten ausgeben.

Die Helligkeitsmessung erfolgt über einen Fotowiderstand (LDR). Tagsüber fällt die Schaltung in einen Ruhezustand, indem sie inaktiv ist. Es wird lediglich periodisch die Helligkeit abgefragt um festzustellen, wann die Nacht anbricht und die Schaltung sich

wieder scharf schaltet. Wird der LDR in diesem Zustand angeleuchtet, gibt die Schaltung eine fest einprogrammierte Zahlenfolge mit der 7 Segment Anzeige zurück und wartet dann auf eine erneute Aktivierung.

Aufgrund des äußerst geringen Stromverbrauchs ist die Schaltung mit einem Satz Batterien über Jahre hinweg einsatzbereit.

2. Schaltplan

Abbildung 1 zeigt den Schaltplan des Reaktivlichtes.

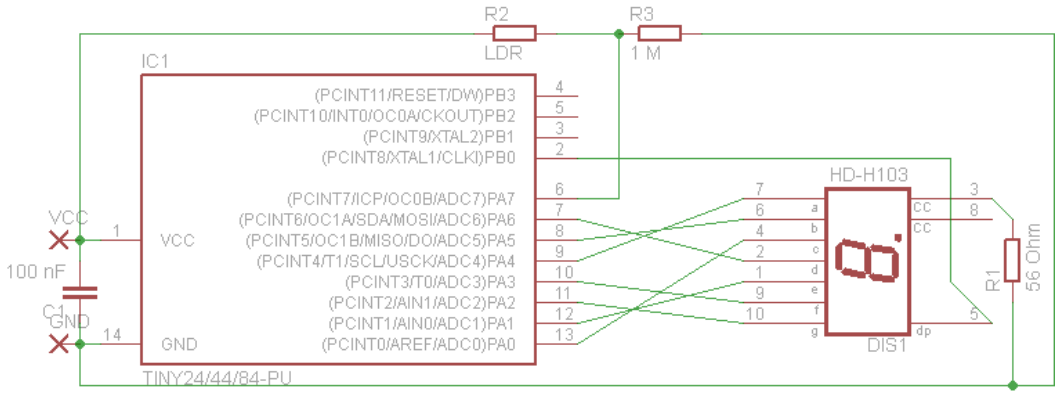

Abbildung 1: Schaltplan.

In der linken Hälfte befindet sich der Microcontroller, links davon die Spannungsversorgung. Die Schaltung benötigt eine Spannung von 3 V. Zwei Standardbatterien in Reihe geschaltet ergeben diese Spannung. Der Pluspol muss an den Pin `VCC`, der Minuspol an den Pin `GND` angeschlossen werden. Oberhalb des ICs befindet sich die Helligkeitsmessung, bestehend aus R3 und dem Fotowiderstand R2. Ganz rechts findet sich die 7-Segment-Anzeige (gemeinsame Kathode) mit dem Vorwiderstand. Aufgrund der Programmstruktur ist für alle acht LEDs der Anzeige nur ein Widerstand nötig.

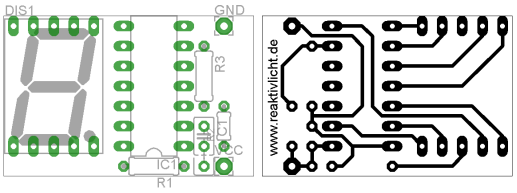

Abbildung 2: Platinenlayout.

Ein Entwurf für eine Platine ist in Abbildung 2 zu sehen. Die linke Abbildung zeigt die Bestückungsseite, die rechte die Leiterbahnen von der Unterseite aus gesehen. Bei

der Bestückung ist auf die Polung des ICs (die Einkerbung muss zum Kondensator hin zeigen) zu achten.

3. Programmierung

```
$regfile = "ATtiny84.DAT"
$crystal = 16000
$hwstack = 3

Config Adc = Single , Prescaler = Auto
Ddra = &B01111111
Porta = &B00000000
Ddrb = &B00000001
Portb = &B00000000

Stop Ac

Wdtcr = &B11010000
Enable Interrupts

Const Threshold = 15
Const Daylight = 950
Const Maxdigits = 512

Dim Ldr As Integer
Dim Oldldr As Integer
Dim Deltaldr As Integer

Dim Daylightcounter As Integer

Dim E As Byte
Dim J As Integer
Dim X As Integer

Do

  Start Adc
  Ldr = Getadc(7)
  Stop Adc
  Deltaldr = Ldr - Oldldr
  Oldldr = Ldr

  If Deltaldr > Threshold Then
```

```
    Gosub Show
    Oldldr = 1024
  End If

  If Ldr > Daylight Then
    If Daylightcounter < 255 Then
      Daylightcounter = Daylightcounter + 1
    End If
  Else
    Daylightcounter = 0
  End If
  If Daylightcounter > 200 Then
    Gosub Pause_8
  End If

  Gosub Pause_0125

Loop

Show:
   X = 0
   E = 0
   While E < 11 And X < Maxdigits
     Readeeprom E , X
     If E < 11 Then
       For J = 0 To 100
         Gosub Show_dig
       Next J
       Gosub Pause_0125
     End If
     X = X + 1
   Wend
Return

Pause_0125:
  Wdtcr = &B11010011
  Reset Watchdog
  Powerdown
Return

Pause_8:
  Wdtcr = &B11110001
  Reset Watchdog
  Powerdown
```

```
Return

Show_dig:
  If E = 0 Then
    Gosub Dig_0
  Elseif E = 1 Then
    Gosub Dig_1
  Elseif E = 2 Then
    Gosub Dig_2
  Elseif E = 3 Then
    Gosub Dig_3
  Elseif E = 4 Then
    Gosub Dig_4
  Elseif E = 5 Then
    Gosub Dig_5
  Elseif E = 6 Then
    Gosub Dig_6
  Elseif E = 7 Then
    Gosub Dig_7
  Elseif E = 8 Then
    Gosub Dig_8
  Elseif E = 9 Then
    Gosub Dig_9
  Elseif E = 10 Then
    Gosub Dig_dot
  End If
Return

Dig_dot:
  Portb.0 = 1
  Waitms 5
  Portb.0 = 0
  Waitms 5
  Waitms 5
  Waitms 5
  Waitms 5
  Waitms 5
  Waitms 5
Return

Dig_0:
  Porta.5 = 1
  Waitms 5
  Porta.5 = 0
```

```
127   Porta.0 = 1
128   Waitms 5
129   Porta.0 = 0
130   Porta.6 = 1
131   Waitms 5
132   Porta.6 = 0
133   Porta.1 = 1
134   Waitms 5
135   Porta.1 = 0
136   Porta.3 = 1
137   Waitms 5
138   Porta.3 = 0
139   Porta.4 = 1
140   Waitms 5
141   Porta.4 = 0
142   Waitms 5
143 Return
144
145 Dig_1:
146   Waitms 5
147   Porta.5 = 1
148   Waitms 5
149   Porta.5 = 0
150   Porta.0 = 1
151   Waitms 5
152   Porta.0 = 0
153   Waitms 5
154   Waitms 5
155   Waitms 5
156   Waitms 5
157 Return
158
159 Dig_2:
160   Porta.4 = 1
161   Waitms 5
162   Porta.4 = 0
163   Porta.5 = 1
164   Waitms 5
165   Porta.5 = 0
166   Waitms 5
167   Porta.6 = 1
168   Waitms 5
169   Porta.6 = 0
170   Porta.1 = 1
```

```
171   Waitms 5
172   Porta.1 = 0
173   Waitms 5
174   Porta.2 = 1
175   Waitms 5
176   Porta.2 = 0
177 Return
178 
179 Dig_3:
180   Porta.4 = 1
181   Waitms 5
182   Porta.4 = 0
183   Porta.5 = 1
184   Waitms 5
185   Porta.5 = 0
186   Porta.0 = 1
187   Waitms 5
188   Porta.0 = 0
189   Porta.6 = 1
190   Waitms 5
191   Porta.6 = 0
192   Waitms 5
193   Waitms 5
194   Porta.2 = 1
195   Waitms 5
196   Porta.2 = 0
197 Return
198 
199 Dig_4:
200   Waitms 5
201   Porta.5 = 1
202   Waitms 5
203   Porta.5 = 0
204   Porta.0 = 1
205   Waitms 5
206   Porta.0 = 0
207   Waitms 5
208   Waitms 5
209   Porta.3 = 1
210   Waitms 5
211   Porta.3 = 0
212   Porta.2 = 1
213   Waitms 5
214   Porta.2 = 0
```

```
Return

Dig_5:
  Porta.4 = 1
  Waitms 5
  Porta.4 = 0
  Waitms 5
  Porta.0 = 1
  Waitms 5
  Porta.0 = 0
  Porta.6 = 1
  Waitms 5
  Porta.6 = 0
  Waitms 5
  Porta.3 = 1
  Waitms 5
  Porta.3 = 0
  Porta.2 = 1
  Waitms 5
  Porta.2 = 0
Return

Dig_6:
  Porta.4 = 1
  Waitms 5
  Porta.4 = 0
  Porta.3 = 1
  Waitms 5
  Porta.3 = 0
  Porta.0 = 1
  Waitms 5
  Porta.0 = 0
  Porta.6 = 1
  Waitms 5
  Porta.6 = 0
  Porta.1 = 1
  Waitms 5
  Porta.1 = 0
  Waitms 5
  Porta.2 = 1
  Waitms 5
  Porta.2 = 0
Return

```

```
Dig_7:
  Porta.4 = 1
  Waitms 5
  Porta.4 = 0
  Porta.5 = 1
  Waitms 5
  Porta.5 = 0
  Porta.0 = 1
  Waitms 5
  Porta.0 = 0
  Waitms 5
  Waitms 5
  Waitms 5
  Waitms 5
Return

Dig_8:
  Porta.5 = 1
  Waitms 5
  Porta.5 = 0
  Porta.0 = 1
  Waitms 5
  Porta.0 = 0
  Porta.6 = 1
  Waitms 5
  Porta.6 = 0
  Porta.1 = 1
  Waitms 5
  Porta.1 = 0
  Porta.3 = 1
  Waitms 5
  Porta.3 = 0
  Porta.4 = 1
  Waitms 5
  Porta.4 = 0
  Porta.2 = 1
  Waitms 5
  Porta.2 = 0
Return

Dig_9:
  Porta.4 = 1
  Waitms 5
  Porta.4 = 0
```

```
303   Porta.5 = 1
304   Waitms 5
305   Porta.5 = 0
306   Porta.0 = 1
307   Waitms 5
308   Porta.0 = 0
309   Porta.6 = 1
310   Waitms 5
311   Porta.6 = 0
312   Waitms 5
313   Porta.3 = 1
314   Waitms 5
315   Porta.3 = 0
316   Porta.2 = 1
317   Waitms 5
318   Porta.2 = 0
319 Return
320
321 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 drei 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 werden die Pins 0 - 6 des Port A und der Pin 0 des Port B als Ausgang konfiguriert und auf low gesetzt. Sie dienen der Ansteuerung der LEDs. Die restlichen Port sind Eingänge. In Zeile 11 wird der Analog-Komparator abgestellt, um Strom zu sparen. Er wird in diesem Programm nicht benötigt. Zeile 13 und 14 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. `Threshold` gibt die Mindestgröße der Helligkeitsänderung zwischen zwei Durchläufen an, damit das Reaktivlicht ausgelöst wird. Hiermit kann die Empfindlichkeit der Schaltung eingestellt werden. `Daylight` gibt den Wert vor, oberhalb dessen die Schaltung in den Tagmodus wechselt. `Maxdigits` gibt die Anzahl der Bytes im EEPROM an. Je nach verwendetem Controller muss der Wert angepasst werden. Der ATtiny24A hat 128 Byte, der ATtiny44A 256 Byte und der ATtiny84A 512 Byte. Der aktuelle Helligkeitswert wird in der Variablen `Ldr`, derjenige des vorangegangenen Durchlauf in `Oldldr` gespeichert. `Deltaldr` ist eine Variable, in der die Differenz zwischen diesen beiden Werten gespeichert wird. In `Daylightcounter` werden die Durchläufe gezählt, in denen eine Helligkeit größer als `Daylight` gemessen wurde. `E` enthält den Code des

Zeichens, das angezeigt werden soll. `J` ist eine Zählvariable. `X` enthält die Adresse der EEPROM-Speicherzelle, die gelesen werden soll.

Die Zeilen 30 bis 56 enthalten das Hauptprogramm. In Zeile 32 bis 41 wird 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 `Oldldr` gespeichert. Ist der Helligkeitsunterschied größer als die eingestellte Schwelle, wird die Unterroutine `Show` aufgerufen. Anschließend wird `Oldldr` auf dem Maximalwert gesetzt, um eine Doppelauslösung zu verhindern. Dazu könnte es kommen, wenn während des Blinkvorganges die Helligkeit weiter zunimmt. Ab Zeile 43 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_8` aufgerufen. Zuletzt wird der Controller für 0,125 s in den Schlafmodus versetzt, bevor mit dem nächsten Zyklus begonnen wird.

Die Unterprozedur `Show` befindet sich in den Zeilen 58 bis 71. Hier werden solange nacheinander die Bytes des EEPROM ausgelesen, bis eine Wert größer als 10 gelesen wird oder das Ende des EEPROMs erreicht wurde. Für jeden Wert wird 100 Mal die Prozedur `Show_dig` aufgerufen. Über die Anzahl der Aufrufe kann die Anzeigedauer der Werte eingestellt werden.

Die Zeilen 73 bis 83 enthalten Prozeduren, um den Controller 125 ms bzw. 8 s warten zu lassen. Zu Beginn wird die Zeit des Watchdog-Timers auf die gewünschte Wartezeit gesetzt. Anschließend wird der Controller in den Schlafmodus versetzt. Wenn er daraus wieder erwacht ist, wird mit dem Programmablauf fortgefahren.

Die Prozedur `Show_dig` in den Zeilen 85 - 109 ruft in Abhängigkeit vom Inhalt der Variablen `E` andere Prozeduren auf, die das entsprechende Zeichen auf der 7-Segment-Anzeige anzeigen. Diese sind in den Zeilen 111 - 319 zu finden. Jede Prozedur besteht aus einer identischen Anzahl Warteschritten, sodass jedes Zeichen gleich lang und gleich hell erscheint. Nacheinander werden die benötigten Segmente jeweils einen Warteschritt lang eingeschaltet. Dadurch leuchtet immer nur ein Segment gleichzeitig und es kann ein gemeinsamer Vorwiderstand verwendet werden.

Die wiederzugebende Zahlenfolge muss in das EEPROM einprogrammiert werden. Dabei repräsentieren die Werte 0x00 - 0x09 die Ziffern 0 - 9. Der Punkt wird durch 0x0A codiert. Werte von 0x0B bis 0xFF zeigen das Ende der Folge an.

Teil II.
Häufig gestellte Fragen und bekannte Probleme

4. Zwei Segmente der Anzeige leuchten nicht.

Wahrscheinlich ist während des Lötens ein Kurzschluss zwischen den Anschlüssen dieser beiden Segmente entstanden. Dieser muss beseitigt werden.

5. Die Schaltung funktioniert nicht und der Microcontroller wird sehr heiß.

Entweder ist der Microcontroller verkehrt herum in den Sockel gesteckt oder die Versorgungsspannung ist verpolt.