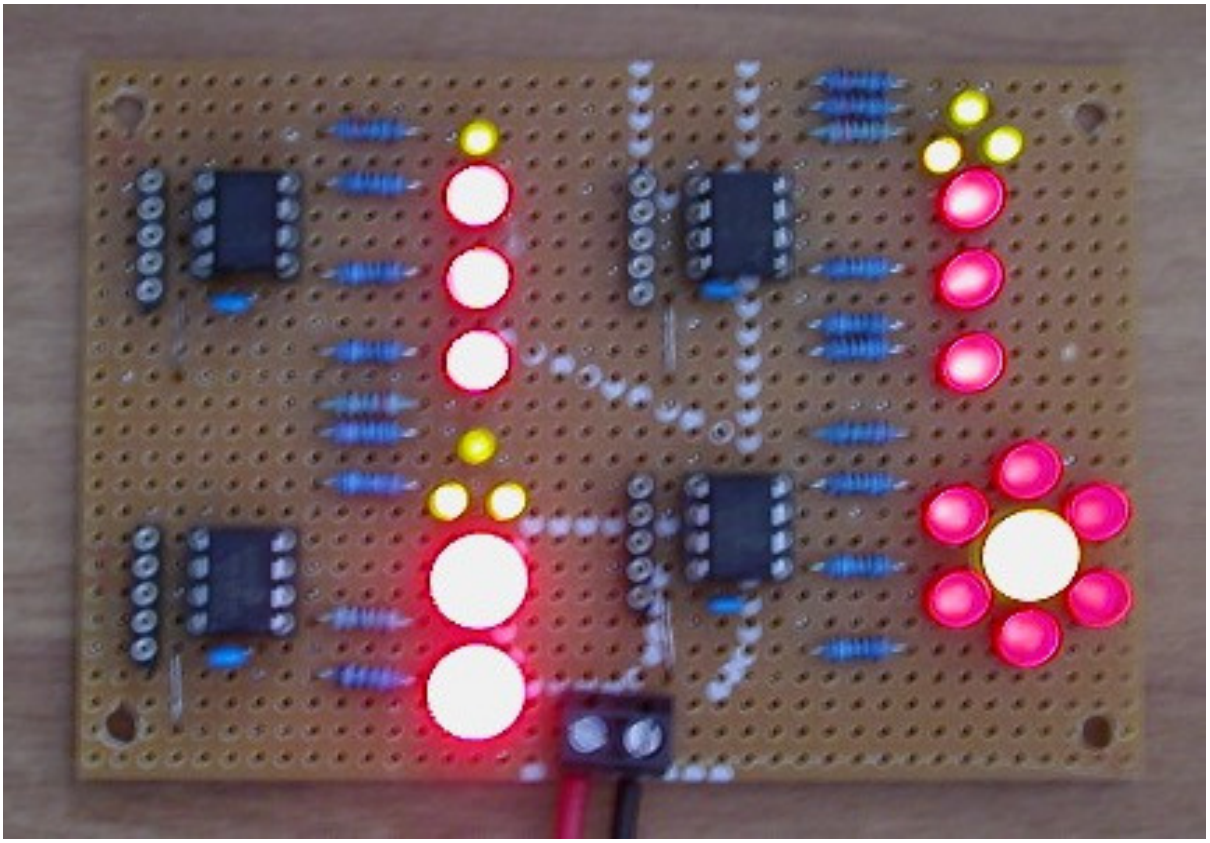
Elektronische Kerze
Flackerlicht mit PIC-Mikrocontroller



Stefan Buchgeher

7. Juli 2009

www.stefan-buchgeher.info
✉ kontakt@stefan-buchgeher.info
✉ Großambergstraße 106/3, A-4040 Linz (Österreich)
☎ ++43/699/11 29 63 38

Inhaltsverzeichnis

---

1 Einleitung

Das besondere an einer elektronischen Kerze ist das Nachbilden des „zufälligen“ Flackerns, so dass für den Betrachter der Eindruck entsteht, dass es sich hier um eine zufällige Helligkeitsänderung handelt. Das Erzeugen von Zufall ist aber eine sehr schwierige Aufgabe, da dies im Normalfall eher nicht gewollt wird. Die Elektronik, oder ein Gerät ganz allgemein, soll normalerweise nicht etwas zufälliges machen, sondern nach ganz bestimmten, definierten Vorgaben reagieren. Das Erzeugen einer zufälligen Zahl ist aber genau das Gegenteil davon.

Diese Arbeit zeigt wie man mit einem Mikrocontroller (hier mit einem kleinen 8pinigen PIC-Mikrocontroller) eine Kerze nach bilden (simulieren) kann.

Der Mikrocontroller benötigt keine besonderen Hardwaremodule. Die Idee der hier beschriebene Methode kann daher mit jedem Mikrocontroller (auch von anderen Herstellern) realisiert werden.

2 Realisierungsidee

Für das Nachbilden einer Kerze wird hier eine Tabelle verwendet, wobei die Tabellenwerte ein Maß für die Helligkeit der Leuchtdioden sind. Die Leuchtdiode leuchtet mal hell mal etwas dunkler, wieder eine Spur heller usw., so wie es die Werte in der Tabelle vorgeben. Wenn diese Helligkeitsänderungen sehr schnell sind, nimmt das menschliche Auge dies als Flackern war, so wie das Flackern einer Kerze.
Da eine Tabelle aber nur eine begrenzte Anzahl an Werten beinhaltet, wiederholt sich dieses Flackern immer wieder. Je länger die Tabelle ist, desto unbedeutender wird dieser kleine Nachteil. Hier, bei dieser Arbeit, wiederholt sich das Flackern erst nach ca. 20 Sekunden, was genau genommen schon sehr lange ist.

Die Abbildung 1 zeigt die Tabellenwerte in grafischer Form für eine mögliche Flackersequenz, wobei null gleich bedeutend mit dunkel ist (die Leuchtdiode oder eine Glühlampe leuchtet nicht) und beim Maximalwert von 31 leuchten die Leuchtdiode (oder eine Glühlampe) mit voller Helligkeit. Bei den Werten zwischen 0 und 31 leuchtet diese dementsprechend weniger hell.

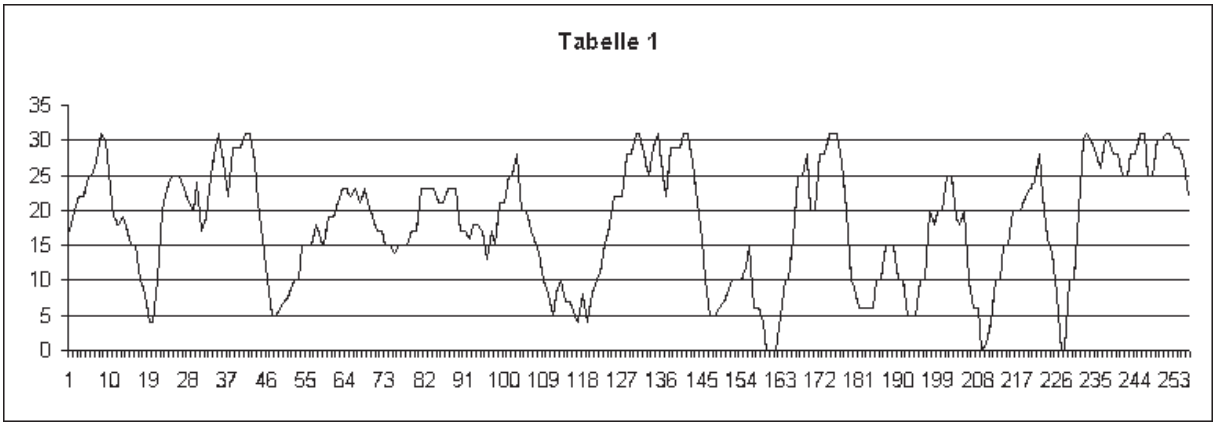


Abbildung 1: Flackerzyklus (Tabelle)

Die zur Abbildung 1 dazugehörigen Tabellenwerte sind in Listing 1 so wie sie auch im Quellcode für den Mikrocontroller verwendet werden angegeben. Bei diesem Listing handelt es sich um einen Auszug aus dem gesamten Quellcode, der im Abschnitt 3.2 (ab Seite 6) vollständig abgedruckt ist. Die Zeilennummern in Listing 1 entsprechen denen vom Gesamtquelltext (Listing) von Abschnitt 3.2.

```
68 // Tabelle fuer das Flackerlicht
69 const char TabFlackerlicht[256] =
70 {
71     17,20,22,22,25,25,27,31,30,23,19,18,19,17,15,
72     15,10,8,4,4,10,20,23,25,25,25,23,21,20,24,17,
73     19,25,29,31,27,22,29,29,29,31,31,27,21,16,10,5,
74     5,6,7,8,10,10,15,15,15,18,16,15,19,19,21,23,
75     23,22,23,21,23,21,19,17,17,15,15,14,15,15,15,17,
76     17,23,23,23,23,21,21,23,23,23,17,17,16,18,18,17,
77     13,17,15,21,21,25,25,28,20,20,17,16,14,10,8,5,
78     8,10,7,7,5,4,8,4,8,10,11,15,18,22,22,22,
79     28,28,31,31,28,25,29,31,27,22,29,29,29,31,31,27,
80     21,16,10,5,5,6,7,8,10,10,10,12,15,6,6,4,0,
81     0,0,5,10,10,16,25,25,28,20,20,28,28,31,31,31,
82     27,20,10,9,6,6,6,6,10,10,15,15,15,10,10,5,
83     5,5,10,10,20,18,20,20,25,25,19,18,20,10,6,6,
84     0,1,4,10,10,15,15,20,20,20,22,23,24,28,22,16,
85     14,8,0,0,10,10,20,30,31,30,28,26,30,30,28,28,
86     25,25,28,28,31,31,25,25,30,30,31,31,29,29,27,22
87 };
```

Listing 1: Tabelle `TabFlackerlicht` (Auszug aus dem Quellcode)

Für die Ausgabe des analogen Tabellenwertes an einem digitalen Ausgang wird das Prinzip der Puls-Weiten-Modulation (kurz: PWM) verwendet. Die Abbildung 2 zeigt das Prinzip der Puls-Weiten-Modulation in Form von Zeitdiagrammen.

Bei der Puls-Weiten-Modulation wird eine rechteckförmige Ausgangsspannung mit fixer

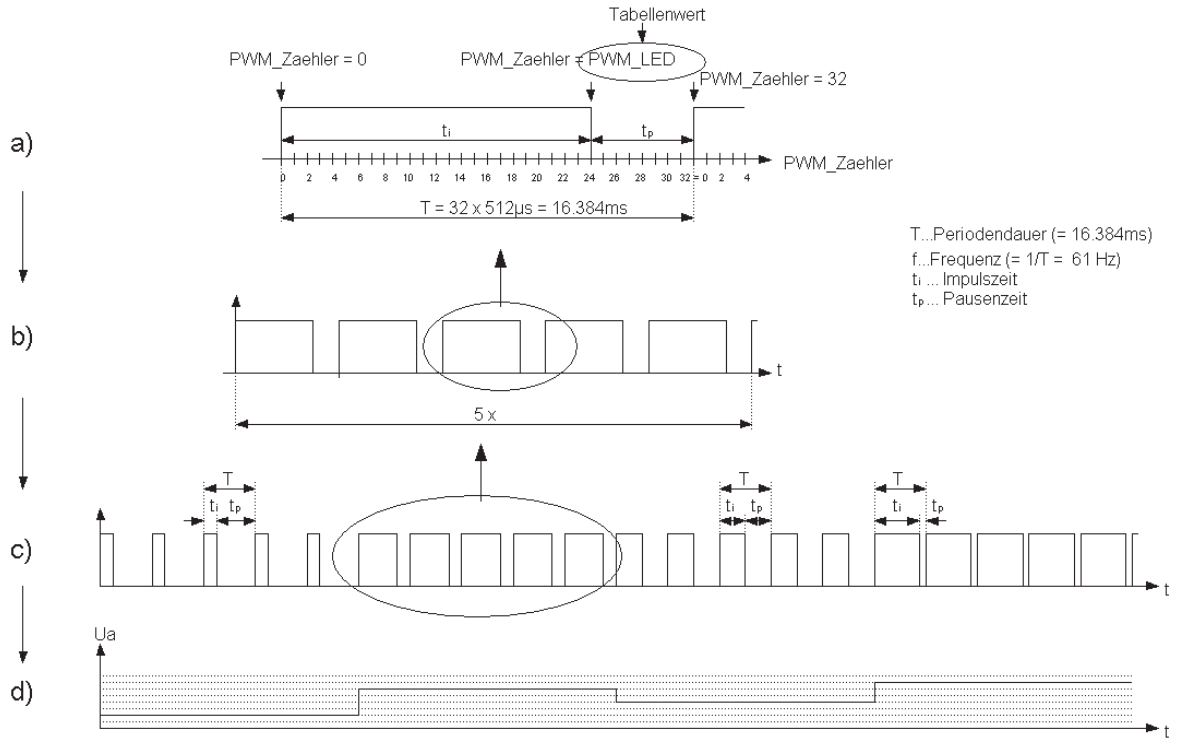


Abbildung 2: Flackerzyklus (Zeitdiagramme)

Periodendauer T, aber variabler Impulsdauer t_i erzeugt. Ist die Impulsdauer t_i sehr kurz im Vergleich zur Pausendauer t_p, dann ist die „Fläche“ sehr klein. Für den Verbraucher (hier für die Leuchtdiode) bedeutet dies eine kleine Spannung. Die Leuchtdiode leuchtet daher schwach, oder gar nicht. Ist die Impulsdauer t_i dagegen groß, dann ist die Pausendauer t_p gering, und die „Fläche“ ist daher groß, und die Leuchtdiode leuchtet sehr hell. Das Verhältnis zwischen Pulsdauer und Pausendauer bestimmt daher die Helligkeit der Leuchtdiode. Wichtig ist nur, dass die Periodendauer T (also die Summe von Impulsdauer t_i und Pausendauer t_p) immer gleich bleibt ($T = t_i + t_p$).
In Abbildung 2 ist dies im unteren Teil noch einmal grafisch für einige Fälle dargestellt: links ist $t_i = \frac{1}{4}T$ (c), daher ist auch die Ausgangsspannung U_a nur ein viertel der maximal möglichen Spannung (d). Beim Fall etwa in der Mitte ist $t_i = t_p = \frac{1}{2}T$, daher ist auch die Ausgangsspannung U_a nur die Hälfte der maximal möglichen Spannung. Im letzen Fall ist $t_i = \frac{7}{8}T$, und die Ausgangsspannung U_a ist daher auch nur $\frac{7}{8}$ der maximal möglichen Spannung

Damit das Fackern auch vom Auge als Flackern wahrgenommen wird, darf die Periodendauer T nicht zu groß werden, andererseits darf das Flackern auch nicht zu schnell erfolgen, das Flackern einer „echten“ Kerze ist relativ langsam. Ein realistisches Flackern lässt sich etwa mit einer Periodendauer T von ca. 16ms (entspricht einer Frequenz von ca. 61Hz) erzeugen, und wenn jede „Periode“ fünf mal wiederholt wird, dann ist das Flackern schon sehr realitätsgetreu.

Listing 2 zeigt die für die softwaretechnische Realisierung des hier beschriebenen Flacker-

lichts das wichtigste Unterprogramm. Dieses Unterprogramm muss regelmäßig (hier alle $512\mu s$) aufgerufen werden.

```
360 void FLACKERLICHT(void)
361 {
362     if (PWM_Zaehler == 0)
363     {
364         LED1 = 1;              // LED1 einschalten
365         LED2 = 1;              // LED2 einschalten
366         LED3 = 1;              // LED3 einschalten
367 
368         Gleichheitszaehler--;
369         if (Gleichheitszaehler == 0)
370         {
371             Gleichheitszaehler = KONST_GLEICHHEIT;
372 
373             TabZaehler1++;
374             PWM_LED1 = TabFlackerlicht[TabZaehler1];
375 
376             TabZaehler2++;
377             PWM_LED2 = TabFlackerlicht[TabZaehler2];
378 
379             TabZaehler3++;
380             PWM_LED3 = TabFlackerlicht[TabZaehler3];
381         }
382     }
383     else
384     {
385         if (PWM_Zaehler == PWM_LED1) LED1 = 0;
386         if (PWM_Zaehler == PWM_LED2) LED2 = 0;
387         if (PWM_Zaehler == PWM_LED3) LED3 = 0;
388     }
389 
390     PWM_Zaehler++;
391     if (PWM_Zaehler == KONST_PWM_OBERGRENZE)
392     {
393         PWM_Zaehler = 0;
394     }
395 }
```

Listing 2: Unterprogramm `FLACKERLICHT` (Auszug aus dem Quellcode)

Dieses Unterprogramm ist sehr einfach, und erzeugt genau das Zeitdiagramm gemäss Abbildung 2a) und 2b). Anzumerken ist, dass dies gleichzeitig für drei Leuchtdioden (unabhängig voneinander) erfolgt. Daher sind auch jeweils drei Register notwendig. Das Register `Gleichheitsszaehler` ist für das fünfmalige Wiederholen notwendig. Die Konstante `KONST_GLEICHHEIT` beinhaltet demnach den Wert 5.
Bei `LED1` bis `LED3` handelt es sich um die Portpins, an denen die Leuchtdioden angeschlossen sind.
Die Register `TabZaehler1` bis `TabZaehler3` geben an, welcher Wert aus der Tabelle gelesen werden soll, und die Register `PWM_LED1` bis `PWM_LED3` beinhalten den Wert aus der Tabelle und somit die „Impulsdauer“.
Die Konstante `KONST_PWM_OBERGRENZE` beinhaltet den Wert 32, und somit die Periodendauer.

Damit alle benötigten Register klar deinierte Startwert besitzen müssen diese vor dem ersten Aufruf des Unterprogramms `FLACKERLICHT` initialisiert werden. Diese Aufgabe übernimmt unter anderem ein Unterprogramm namens `INIT`. Siehe Listing im Abschnitt 3.2 (Zeilen 287 bis 297).

Wichtig:
Das Unterprogramm `FLACKERLICHT` muss regelmässig vom Hauptprogramm aufgerufen werden. (siehe Listing im Abschnitt 3.2, Zeile 414).

3 Demonstrationsbeispiel 1

Das folgende Beispiel dient nur zur Demonstration. Es zeigt eine mögliche Einbindung des zuvor beschriebenen Unterprogramms.

3.1 Schaltung (Hardware)

Die Abbildung 3 zeigt die sehr einfache Schaltung für das erste Demonstrationsbeispiel.

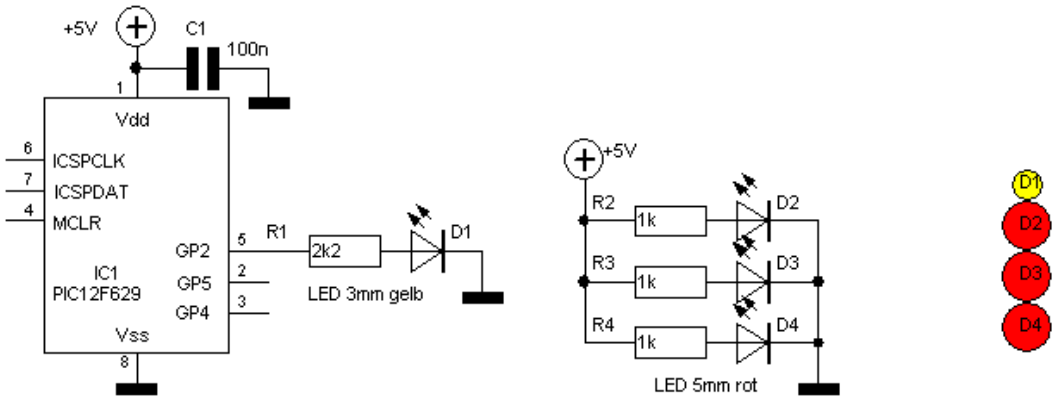


Abbildung 3: Demonstrationsbeispiel 1 (Schaltung)

Bei diesem Demonstrationsbeispiel besteht die Kerze aus drei roten Leuchtdioden (D2 bis D4) und einer gelben Leuchtdiode (D1). Die drei roten Leuchtdioden repräsentieren das Wachs der Kerze und die gelbe Leuchtdiode die Flamme.

Als Mikrocontroller (IC1) dient hier ein PIC12F629. Auffällig beim Mikrocontroller ist die fehlende Oszillatorschaltung sowie die fehlende Resetschaltung. Die Oszillatorschaltung ist hier auch nicht notwendig, da diese Anwendung nicht zeitkritisch ist und daher der interne 4-MHz-Oszillator verwendet werden kann.

Wichtig:
Bei der Konfiguration des Mikrocontrollers müssen die entsprechenden Bits für die Verwendung des internen Oszillators und des internen Reset gesetzt werden! (siehe Listing im folgenden Abschnitt, Zeilen 99 bis 131).

Der Kondensator C1 dient zur Entkoppelung der Betriebsspannung für den Mikrocontroller IC1. Für diesen Koppelkondensator sollte ein Keramiktyp verwendet werden. Dieser muss möglichst nahe beim Mikrocontroller angebracht werden.

3.2 Software

```
/*****************************************************************************/
/* Flackerlicht (elektronische Kerze) mit Tabelle in C (CC5X)                */
/*                                                                           */
/* Fuer drei Kerzen, oder für eine Flamme mit drei Leuchtdioden, wobei nur eine */
/* Tabelle verwendet wird. Damit die drei Kerzen trotz nur einer Tabelle "unabhaengig" */
/* voneinander "flackern" beginnt jeder Tabellenzaehler an einer anderen Position in */
/* der Tabelle (siehe Konstanten KONST_TABSTART_LEDx).                       */
/* Die Zykluswiederholzeit des "Flackerns" betraegt bei einer Tabelle mit 256 Werten */
/* 21 Sekunden (siehe Unterprogramm FLACKERLICHT). Durch Kaskadierung mehrerer */
/* Tabellen kann diese Zykluswiederholzeit verdoppelt, verdreifacht usw. werden. */
/*                                                                           */
/* Takt: interner Takt                                                       */
/*                                                                           */
/* Entwickler:                    Stefan Buchgeher                           */
/* Entwicklungsbeginn der Software: 14. Oktober 2006                         */
/* Funktionsfaehig seit:          14. Oktober 2006                           */
/* Letzte Bearbeitung:            12. Januar 2009                            */
/* Compiler:                      CC5X                                       */
/*****************************************************************************/

/****************** Include-Dateien ***************************************/
#include <int16CXX.H>              // Ist fuer die Interrupts notwendig
#include <INLINE.H>                // Ist fuer Assembleranweisungen notwendig


/****************** Pragma-Anweisungen ************************************/
#pragma library 1                  //.. library functions that are deleted if unused


/****************** Globale Register  *************************************/
uns8    FlagISRHP;                 // beinhaltet Botschaftsflags ISR -> HP

uns8    PWM_Zaehler;               // Zaehlregister fuer Puls-Weiten-Modulation (PWM)
uns8    Gleichheitszaehler;        // Zaehlregister zum 5maligen wiederholen

uns8    TabZaehler1;               // Zeigt auf den aktuellen Index der Tabelle (fuer
uns8    TabZaehler2;               // jede LED/Gluehbirne
uns8    TabZaehler3;

uns8    PWM_LED1;                  // PWM-Wert aus der Tabelle fuer LED/Gluehbirne 1
uns8    PWM_LED2;                  // PWM-Wert aus der Tabelle fuer LED/Gluehbirne 2
uns8    PWM_LED3;                  // PWM-Wert aus der Tabelle fuer LED/Gluehbirne 3


/****************** Bits in den globalen Registern ************************/
/* Register FLAGSISRHP */
bit FlagZeitbasis              @ FlagISRHP.0;


/****************** Portbelegung ******************************************/
/* Port GPIO */
#pragma bit LED1               @ GPIO.2
#pragma bit LED2               @ GPIO.4
#pragma bit LED3               @ GPIO.5


/****************** Konstanten ********************************************/
#define KONST_PWM_OBERGRENZE    32
#define KONST_GLEICHHEIT        5

// Startwert in der Tabelle fuer die einzelnen LEDs
#define KONST_TABSTART_LED1     0
#define KONST_TABSTART_LED2     50
#define KONST_TABSTART_LED3     100


/****************** Tabellen **********************************************/
// Tabelle fuer das Flackerlicht
const char TabFlackerlicht[256] =
```

```
{
    17,20,22,22,25,25,27,31,30,23,19,18,19,17,15,
    15,10,8,4,4,10,20,23,25,25,25,23,21,20,24,17,
    19,25,29,31,27,22,29,29,29,31,31,27,21,16,10,5,
    5,6,7,8,10,10,15,15,15,18,16,15,19,19,21,23,
    23,22,23,21,23,21,19,17,17,15,15,14,15,15,15,17,
    17,23,23,23,23,21,21,23,23,23,17,17,16,18,18,17,
    13,17,15,21,21,25,25,28,20,20,17,16,14,10,8,5,
    8,10,7,7,5,4,8,4,8,10,11,15,18,22,22,22,
    28,28,31,31,28,25,29,31,27,22,29,29,29,31,31,27,
    21,16,10,5,5,6,7,8,10,10,10,12,15,6,6,4,0,
    0,0,5,10,10,16,25,25,28,20,20,28,28,31,31,31,
    27,20,10,9,6,6,6,6,10,10,15,15,15,10,10,5,
    5,5,10,10,20,18,20,20,25,25,19,18,20,10,6,6,
    0,1,4,10,10,15,15,20,20,20,22,23,24,28,22,16,
    14,8,0,0,10,10,20,30,31,30,28,26,30,30,28,28,
    25,25,28,28,31,31,25,25,30,30,31,31,29,29,27,22
};


/***************** Funktionsprototypen ********************************************/
/* Initialisierung des Mikrocontroller */
void INIT(void);

/* Unterprogramm zur Erzeugung des Flackerlichts */
void FLACKERLICHT(void);


/***************** Konfigurations-Bits ********************************************/
#pragma config |= 0b.00000110000100
/*                    ++--------------- Bit 13-12 (BG1:BG0): Bandgap-Calibration
                      ||||||||||||    -> 0  0 :  Lowest bandgap voltage
                      ||||||||||||       1  1 :  Highest bandgap voltage
                     +++------------- Bit 11-9: Reserve
                        +------------ Bit 8 (CPD): Data Code Protection (Data Memory)
                        ||||||||         0 : CPD on    (= enabled)
                        ||||||||     -> 1 : CPD off   (= disabled)
                        +------------ Bit 7 (CP): Code Protection (Programm Memory)
                        |||||||          0 : CP on     (= enabled)
                        |||||||      -> 1 : CP off    (= disabled)
                        +------------ Bit 6 (BODEN): Brown-Out Detection
                        ||||||       -> 0 : BODEN on  (= disabled)
                        ||||||          1 : BODEN off (= enabled)
                        +------------ Bit 5 (MCLRE)
                        |||||        -> 0 : GP3 = I/O
                        |||||           1 : GP3 = MCLR
                        +------------ Bit 4 (PWRTE): Power Up Timer
                        ||||        -> 0 : PWRT on   (= enabled)
                        ||||           1 : PWRT off  (= disabled)
                        +------------ Bit 3 (WDTE): Watchdog Timer
                        |||         -> 0 : WDT off   (= disabled)
                        |||            1 : WDT on    (= enabled))
                        +++---------- Bit 2-0 (FOSC2:FOSC0): Oszillator Selectio
                                         000 : LP Oszillator     (GP4=CLKOUT, GP5=CLKIN)
                                         001 : XT Oszillator     (GP4=CLKOUT, GP5=CLKIN)
                                         010 : HS Oszillator     (GP4=CLKOUT, GP5=CLKIN)
                                         011 : EC Oszillator     (GP4=I/O, GP5=CLKIN)
                                      -> 100 : INTOSC Oszillator (GP4=I/O, GP5=I/O)
                                         101 : INTOSC Oszillator (GP4=CLKOUT, GP5=I/O)
                                         110 : RC Oszillator     (GP4=I/O, GP5=RC)
                                         111 : RC Oszillator     (GP4=CLKOUT, GP5=RC)
*/


/***************** ISR - Timer0 ***************************************************/

/*****************************************************************************/
/* Interrupt Service Routine:                                                */
/*                                                                           */
/* Aufruf:                                                                   */
/*      alle 512 us                                                          */
```

```
141 /*                                                                              */
142 /* Aufgaben:                                                                    */
143 /*      + w-Register (=Arbeitsregister) und Status-Register zwischenspeichern   */
144 /*        (int_save_registers)                                                  */
145 /*      + Zeitbasis fuer 512 Mikrosekunde erzeugen                              */
146 /*      + Das Timer-Interrupt-Flag T0IF wieder loeschen                         */
147 /*      + w-Register (=Arbeitsregister) und Statusregister wiederherstellen     */
148 /*        (int_restore_registers)                                               */
149 /********************************************************************************/
150 #pragma origin 4
151
152 interrupt InterruptRoutine(void)        // Interruptroutine
153 {
154     int_save_registers                  // W, STATUS (und PCLATH) retten
155
156     // Beginn der eigentlichen ISR-Routine
157     FlagZeitbasis = 1;
158
159     T0IF = 0;                           // Timer-Interrupt-Flag T0IF wieder loeschen
160     // Ende der eigentlichen ISR-Routine
161
162     int_restore_registers               // W, STATUS (und PCLATH) Wiederherstellen
163 }
164
165
166 /***************** Unterprogramme und Funktionen ***********************************/
167
168 /********************************************************************************/
169 /* INIT:                                                                        */
170 /*                                                                              */
171 /* Aufgabe:                                                                     */
172 /*   Initialisierung des Prozessor:                                             */
173 /*      + Timer 0 (TMR0) loeschen                                               */
174 /*      + Timer 0-ISR soll ca. alle 512us aufgerufen werden, daher den Vorteiler mit 2 */
175 /*        laden (bei einem internen 4-MHz-Takt)                                 */
176 /*      + Comparator/Analog-Eingaenge deaktivieren                              */
177 /*      + Port als Ausgang (fuer LEDs) definieren                               */
178 /*      + externen Variablen initialisieren                                     */
179 /*      + Interrupt freigeben (Timer0 freigeben durch Setzen von GIE und T0IE im */
180 /*        INTCON-Register                                                       */
181 /*                                                                              */
182 /* Uebergabeparameter:                                                          */
183 /*   keiner                                                                     */
184 /*                                                                              */
185 /* Rueckgabeparameter:                                                          */
186 /*   keiner                                                                     */
187 /********************************************************************************/
188 void INIT(void)
189 {
190     // Timer-0-Interrupt
191     TMR0 = 0;                      // Timer 0 auf 0 voreinstellen
192     OPTION = 0b.1000.0000;         // Option-Register (Bank 1)
193 /*              +------------------ Bit 7 (nGPPU): Pull-Up-Widerstaende am GPIO
194                  |||  ||||                 0 : Pull-Up aktiviert
195                  |||  ||||             ->  1 : Pull-Up deaktiviert
196                  +-------------------- Bit 6 (INTEDG): Interrupt Edge Select Bit
197                   ||  ||||             ->  0 : Interrupt on falling edge of GP2/INT pin
198                   ||  ||||                 1 : Interrupt on rising edge of GP2/INT pin
199                   +-------------------- Bit 5 (T0CS): Timer 0 Clock source Select Bit
200                    |  ||||             ->  0 : Transition on GP2/T0CKI pin
201                    |  ||||                 1 : Internal instruction cycle clock (CLKOUT)
202                    +-------------------- Bit 4 (T0SE): TMR0 Source Edge Select bit
203                       ||||             ->  0 : Incr. on low-to-high tranition (GP2/T0CKI)
204                       ||||                 1 : Incr. on high-to-low tranition (GP2/T0CKI)
205                       +---------------- Bit 3 (PSA): Prescaler Assignment bit
206                        |||             ->  0 : Prescaler is assigned to the TIMER0 module
207                        |||                 1 : Prescaler is assigned to the WDT
208                        +++-------------- Bit 2-0 (PS2:PS0): Prescaler Rate Select bits
209                                            Bit Value      TMR0 Rate       WDT Rate
210                                        ->  000 :           1:2             1:1
211                                            001 :           1:4             1:2
```

```
                                        010 :           1:8             1:4
                                        011 :           1:16            1:8
                                        100 :           1:32            1:16
                                        101 :           1:64            1:32
                                        110 :           1:128           1:64
                                        111 :           1:256           1:128
*/

    // Comparator/Analog-Eingaenge deaktivieren
    GPIO = 0b.0000.0000;            // PORT (Bank 0)
/*              ++----------------- Bit 7-6 : Reserve
                +------------------ Bit 5 (GPIO5): hier: Portpin is low
                 +----------------- Bit 4 (GPIO4): hier: Portpin is low
                  +---------------- Bit 3 (GPIO3): hier: Portpin is low
                   +--------------- Bit 2 (GPIO2): hier: Portpin is low
                    +-------------- Bit 1 (GPIO1): hier: Portpin is low
                     +------------- Bit 0 (GPIO0): hier: Portpin is low
*/

    CMCON = 0b.0000.0111;           // Comparator Control-Register (Bank 0)
/*              +-+---------------- Bit 7,5 : Reserve
                +------------------ Bit 6 (COUT): Comparator Output bit
                  | ||||                When CINV = 0:
                  | ||||            ->    0 : Vin+ < Vin-
                  | ||||                  1 : Vin+ > Vin-
                  | ||||                When CINV = 1:
                  | ||||                  0 : Vin+ > Vin-
                  | ||||                  1 : Vin+ < Vin-
                  +------------------ Bit 4 (CINV): ComparatorOutput Inversion Bit
                    ||||            ->  0 : Output not inverted
                    ||||                1 : Output inverted
                   +----------------- Bit 3 (CIS): Comparator Input Switch Bit
                    |||                 When CM2:CM0 = 110 or 101:
                    |||             ->  0 : Vin- connects to CIN-
                    |||                 1 : Vin- connects to CIN+
                   +++--------------- Bit 2-0 (CM2:CM0): Comp. Mode bits (Datenbl. S.37)
                                        000 : Comparator Reset
                                        001 : Comparator with Output
                                        010 : Comparator without Output
                                        011 : Comp. with Output and Internal Ref.
                                        100 : Comp. w/o Output and Internal Ref.
                                        101 : Multiplexed Input with Int. Ref and Outp.
                                        110 : Multiplexed Input with Int. Ref
                                    ->  111 : Off
*/

    ANSEL = 0b.0000.0000;           // Analog Select-Register (Bank 1)
/*              +------------------ Bit 7 : Reserve
                +++---------------- Bit 6-4 (ADCS2:ADCS0): A/D Conv. Clock Select Bits
                   ||||             ->  000 : Fosc/2
                   ||||                 001 : Fosc/8
                   ||||                 010 : Fosc/32
                   ||||                 x11 : Frc
                   ||||                 100 : Fosc/4
                   ||||                 101 : Fosc/16
                   ||||                 110 : Fosc/64
                  ++++-------------- Bit 3-0 (ANS3-ANS0): Analog Select Bit
                                     (Between analog or digital function on pins
                                     AN3:AN0, respectively)
                                     ->  0 : Digital I/O
                                         1 : Analog input
*/

    // Port
    TRISIO = 0b.0000.0000;          // Richtungsregister PORT (Bank 1)
/*              ++----------------- Bit 7-6 : Reserve
                +------------------ Bit 5 (TRISIO5): hier Ausgang
                 +----------------- Bit 4 (TRISIO4): hier Ausgang
                  +---------------- Bit 3 (TRISIO3): hier Ausgang
                   +--------------- Bit 2 (TRISIO2): hier Ausgang
                    +-------------- Bit 1 (TRISIO1): hier Ausgang
```

```
283                         +------------ Bit 0 (TRISIO0): hier Ausgang
284 */
285
286     // alle externen Variablen initialisieren
287     PWM_Zaehler = 0;
288
289     Gleichheitszaehler = KONST_GLEICHHEIT;
290
291     LED1 = 1;
292     LED2 = 1;
293     LED3 = 1;
294
295     TabZaehler1 = KONST_TABSTART_LED1;
296     TabZaehler2 = KONST_TABSTART_LED2;
297     TabZaehler3 = KONST_TABSTART_LED3;
298
299     // Interrupt freigeben
300     INTCON = 0b.1010.0000;          // Interrupt-Control-Register (Bank 1)
301 /*              +-------------------- Bit 7 (GIE): Global Interrupt Enable bit
302                  ||| ||||                    0 : Disables all interrupts
303                  ||| ||||                ->  1 : Enables all unmasked interrupts
304                  +-------------------- Bit 6 (PEIE): Peripheral Interrupt Enable bit
305                   || ||||                ->  0 : Disables all peripheral interrupts
306                   || ||||                    1 : Enables all unmasked peripheral interrupts
307                   +-------------------- Bit 5 (T0IE): Timer 0 Overflow Interrupt Enable Bit
308                    | ||||                    0 : Disabled
309                    | ||||                ->  1 : Enabled
310                    +-------------------- Bit 4 (INTE): GP2/INT External Interrupt Enable Bit
311                      ||||                ->  0 : Disabled
312                      ||||                    1 : Enabled
313                      +-------------------- Bit 3 (GPIE): Port Change Interrupt Enable Bit
314                       |||                ->  0 : Disabled
315                       |||                    1 : Enabled
316                       +-------------------- Bit 2 (T0IF): Timer 0 Overflow Interrupt Flag bit
317                        ||                ->  0 : TMR0 register did not overflow
318                        ||                    1 : TMR0 register has overflowed
319                        +-------------------- Bit 1 (INTF): GP2/INT External Interrupt Flag bit
320                         |                ->  0 : GP2/INT external interrupt did not occur
321                         |                    1 : GP2/INT external interrupt occurred
322                         +-------------------- Bit 0 (GPIF): Port Change Interrupt Flag bit
323                                          ->  0 : None of GP5:GP0 pins have changed state
324                                              1 : One of the GP5:GP0 pins changed state
325 */
326 }
327
328
329 /*****************************************************************************/
330 /* FLACKERLICHT:                                                             */
331 /*                                                                           */
332 /* Aufgabe:                                                                  */
333 /*   PWM fuer LEDs erzeugen:                                                 */
334 /*      + PWM_Zaehler = 0: Ausgang LEDx setzen und den naechsten Wert aus der Tabelle  */
335 /*        TabFlackerlicht lesen.                                             */
336 /*      + PWM_Zaehler = PWM_LEDx: Ausgang LEDx ruecksetzen.                  */
337 /*      + TabZaehlerx gibt an, welcher PWM-Wert aus der Tabelle (TabFlackerlicht) aus- */
338 /*        gegeben werden soll.                                               */
339 /*                                                                           */
340 /* Uebergabeparameter:                                                       */
341 /*   keiner                                                                  */
342 /*                                                                           */
343 /* Rueckgabeparameter:                                                       */
344 /*   keiner                                                                  */
345 /*                                                                           */
346 /* Anmerkung:                                                                */
347 /*  + Zykluswiederholzeit des "Flackerns" bei einer Tabelle mit 256 Werten: 21 Sek.  */
348 /*                                                                           */
349 /*            4                                                              */
350 /*    T = ------- * 256 * 2 * 32 * 5 * 256 = ca. 21 Sekunden                 */
351 /*        4 [MHz]                                                            */
352 /*         |         |    |    |   |    |                                   */
353 /*         |         |    |    |   |    +-- Anz. der Tabellenwerte (hier: 256)  */
```

```
354 /*          |          |     |     |   +-------- Anz. der gleichen PWM (hier: 5)          */
355 /*          |          |     |     +------------ max. PWM-Zyklen -> 61 Hz                  */
356 /*          |          |     +------------------ Vorteiler fuer Timer 0 (hier: 2)          */
357 /*          |          +------------------------ Timer-0-Ueberlauf                         */
358 /*          +---------------------------------- PIC-interner Takt                         */
359 /*****************************************************************************************/
360 void FLACKERLICHT(void)
361 {
362     if (PWM_Zaehler == 0)
363     {
364         LED1 = 1;             // LED1 einschalten
365         LED2 = 1;             // LED2 einschalten
366         LED3 = 1;             // LED3 einschalten
367 
368         Gleichheitszaehler--;
369         if (Gleichheitszaehler == 0)
370         {
371             Gleichheitszaehler = KONST_GLEICHHEIT;
372 
373             TabZaehler1++;
374             PWM_LED1 = TabFlackerlicht[TabZaehler1];
375 
376             TabZaehler2++;
377             PWM_LED2 = TabFlackerlicht[TabZaehler2];
378 
379             TabZaehler3++;
380             PWM_LED3 = TabFlackerlicht[TabZaehler3];
381         }
382     }
383     else
384     {
385         if (PWM_Zaehler == PWM_LED1) LED1 = 0;
386         if (PWM_Zaehler == PWM_LED2) LED2 = 0;
387         if (PWM_Zaehler == PWM_LED3) LED3 = 0;
388     }
389 
390     PWM_Zaehler++;
391     if (PWM_Zaehler == KONST_PWM_OBERGRENZE)
392     {
393         PWM_Zaehler = 0;
394     }
395 }
396 
397 
398 /***************** Hauptprogramm *********************************************************/
399 
400 /*****************************************************************************************/
401 /* Aufgaben des Hauptprogramms:                                                          */
402 /*   + Controller initialisieren (Unterprogramm INIT)                                    */
403 /*   + Taetigkeiten/Unterprogramme, die alle zyklisch durchgefuehrt werden muessen:      */
404 /*       + Wert fuer Flackerlichtausgabe ermitteln und an den Port-Pins ausgeben.        */
405 /*****************************************************************************************/
406 void main(void)
407 {
408     INIT();                           // Controller initialisieren
409 
410     while(1)                          // Endlosschleife
411     {
412         if (FlagZeitbasis)
413         {
414             FLACKERLICHT();
415             FlagZeitbasis = 0;        // Botschaftsflag zuruecksetzen
416         }
417     }
418 }
```

3.3 Anmerkungen zur Software

Der Quellcode für dieses Demonstrationsbeispiel wurde in C mit der freien Demoversion des CC5x-Compilers erstellt. (Anmerkung: Der CC5x-Compiler lässt sich sehr einfach in

die Entwicklungsumgebung MPLAB des PIC-Herstellers Microchip einbinden).

Die Software besteht im Wesentlichen aus einem sehr kurzen Hauptprogramm (Zeilen 406 bis 418, am Ende des Quellcodes), einer kurzen Interrupt-Service-Routine (kurz ISR, Zeilen 150 bis 163), das im Abschnitt 2 besprochen Unterprogramm zur Erzeugung des Flackerlicht (Zeilen 360 bis 395) und einem Unterprogramm zur Initialisierung des Mikrocontrollers (Unterprogramm `INIT`, Zeilen 188 bis 326).

Die ISR wird alle 512 μs aufgerufen und besitzt hier nur die Aufgabe eine 512-μs-Zeitbasis zu erzeugen, indem sie ein Botschaftsflag für das Hauptprogramm erzeugt. Das Hauptprogramm reagiert auf dieses Botschaftsflag, indem es das Unterprogramm zur Erzeugung des Flackerlichts aufruft. Durch diesen Mechanismus wird dieses Unterprogramm zyklisch, alle 512 μs aufgerufen.
Die Zeit von 512 μs ergibt sich, weil als Takt der interne 4-MHz-Takt verwendet wird, und weil der Vorteiler (VT) mit dem Wert 1:2 (TMR0 Rate, Zeile 210) geladen wird. Für diese Zeit gilt:

$$ISRAUFRUF[\mu s] = \frac{4 \cdot 256 \cdot VT}{f_{Quarz}\,[MHz]} = \frac{4 \cdot 256 \cdot 2}{4} = 512\mu s$$

Das Hauptprogramm (`main`, Zeilen 406 bis 418) befindet sich nach der Initialisierung (Unterprogramm `INIT`) in einer Endlosschleife. Diese Schleife besitzt nur die Aufgabe ständig das Botschaftsflag (hier das Bit `FLAGZEITBASIS`) zu prüfen. Ist dieses Botschaftsflag gesetzt, so muss vom Hauptprogramm eine bestimmte Aufgabe ausgeführt werden. Diese Aufgaben sind in Form von Unterprogrammen vorhanden. Hier ist die Aufgabe das Unterprogramm zur Erzeugung des Flackerlichts aufzurufen (Zeile 414). Wichtig ist, dass das Botschaftsflag wieder gelöscht werden muss (Zeile 415).

Das Unterprogramm (`INIT`, Zeilen 188 bis 326) dient zur Initialisierung des Mikrocontrollers. Hier werden unter anderem die Ports konfiguriert (Port dient als Eingang oder als Ausgang), der oder die Timer eingestellt usw. Dieses Unterprogramm ist vom Controllertyp abhängig und je nach Anwendung mehr oder weniger umfangreich.
In diesem Unterprogramm werden auch noch die für die Erzeugung der Flackerlichter notwendigen Register initialisiert (Zeilen 287 bis 297), und zum Schluss dieses Unterprogramms erfolgt die Freigabe des hier benötigten Timer-0-Interrupt.

Das für die Erzeugung des Flackerlichts wesentliche Unterprogramm (`FLACKERLICHT`, Zeilen 360 bis 395) wurde bereits im Abschnitt 2 behandelt und wird daher hier nicht nocheinmal beschrieben.

Am Beginn des Quellcodes befinden sich die folgenden Definitionen und Einstellungen für die Anwendung des Mikrocontrollers als „elektronische Kerze“:

- Einbinden externer Codeteile (Zeilen 22 und 23)
- Diverse Anweisungen für den CC5x-Compiler (Zeile 27)
- Definition der globalen Register für die ISR (Zeile 31) und für die Erzeugung des Flackerlichts für die drei LEDs und/oder Glühbirnen (Zeilen 33 bis 42)

- Konstanten für die Erzeugung des Flackerlichts (Zeilen 58 bis 64)
- Die im Abschnitt 2 angesprochene Tabelle mit den Werten, die das Flackern nachbilden (Zeilen 69 bis 87)
- Die Funktionsprototypen für die hier verwendeten Unterprogramme (Zeilen 92 und 95)
- Die Konfiguration für den verwendeten PIC-Mikrocontroller (Zeile 99). Ganz besonders wichtig ist die Einstellung des richtigen Oszillators (hier: INTOSC, Zeile 127), und das der interne Reset verwendet wird (Zeile 114)
- Ab Zeile 134 beginnt der Quellcode der hier ausreichend mit Kommentaren versehen ist

4 Demonstrationsbeispiel 2

Die Abbildung 4 zeigt die sehr einfache Schaltung für das zweite Demonstrationsbeispiel.

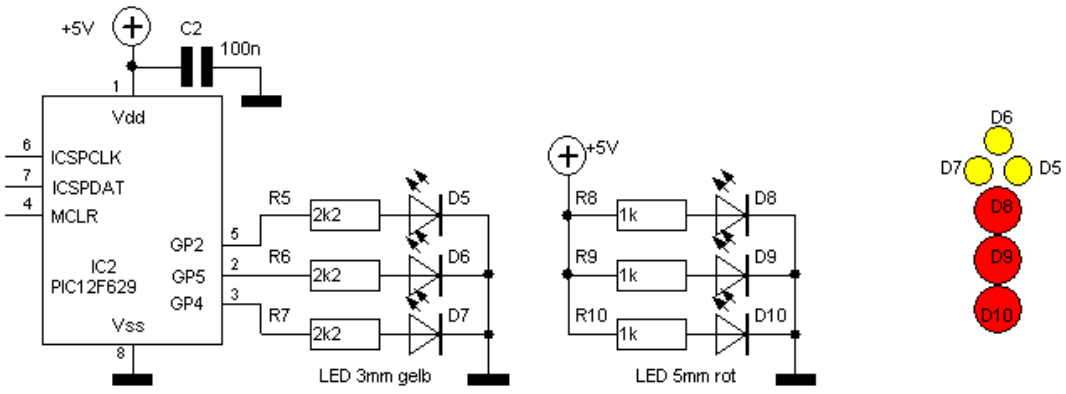


Abbildung 4: Demonstrationsbeispiel 2 (Schaltung)

Im Vergleich zum Demonstrationsbeispiel 1 hat sich nur die Form der Kerze verändert. Hier besteht die Kerze aus drei roten Leuchtdioden (D8 bis D10, Wachs) und drei gelben Leuchtdiode (D5 bis D7, Flamme).

Die Software ist die gleiche wie für das erste Demonstrationsbeispiel, und auch die weiteren Demonstrationsbeispiele verwenden die gleiche Software.

5 Demonstrationsbeispiel 3

Beim Demobeispiel gemäss Abbildung 5 besteht das Wachs der Kerze aus 2 größeren roten Leuchtdioden (D14 und D15) und die Flamme aus drei gelben Leuchtdiode (D11 bis D13)

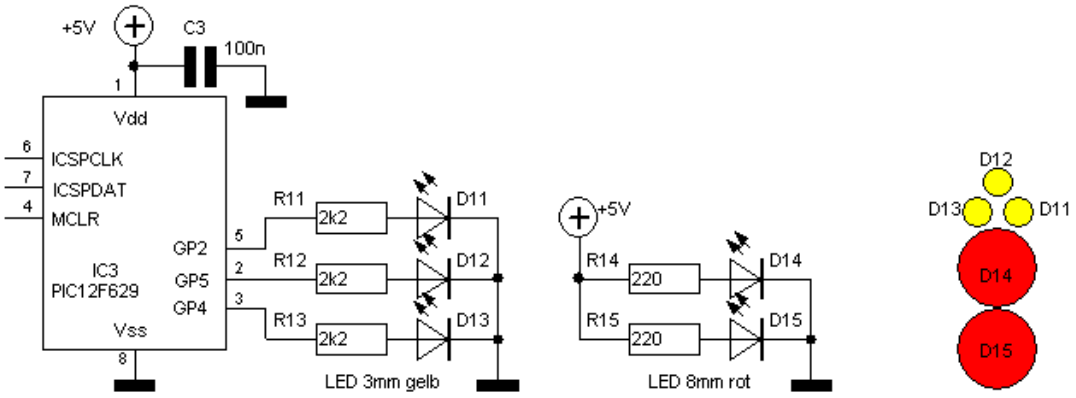


Abbildung 5: Demonstrationsbeispiel 3 (Schaltung)

6 Demonstrationsbeispiel 4

Die Kerze in Abbildung 6 hat eine ganz besondere Form. Hier bilden 6 rote Leuchtdioden (D17 bis D22) eine große Kerze und in der Mitte befindet sich eine große Flamme (D16).

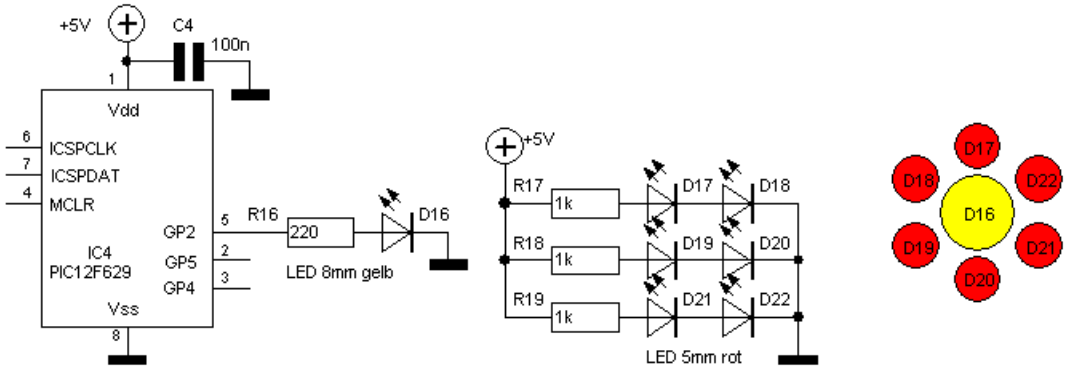


Abbildung 6: Demonstrationsbeispiel 4 (Schaltung)