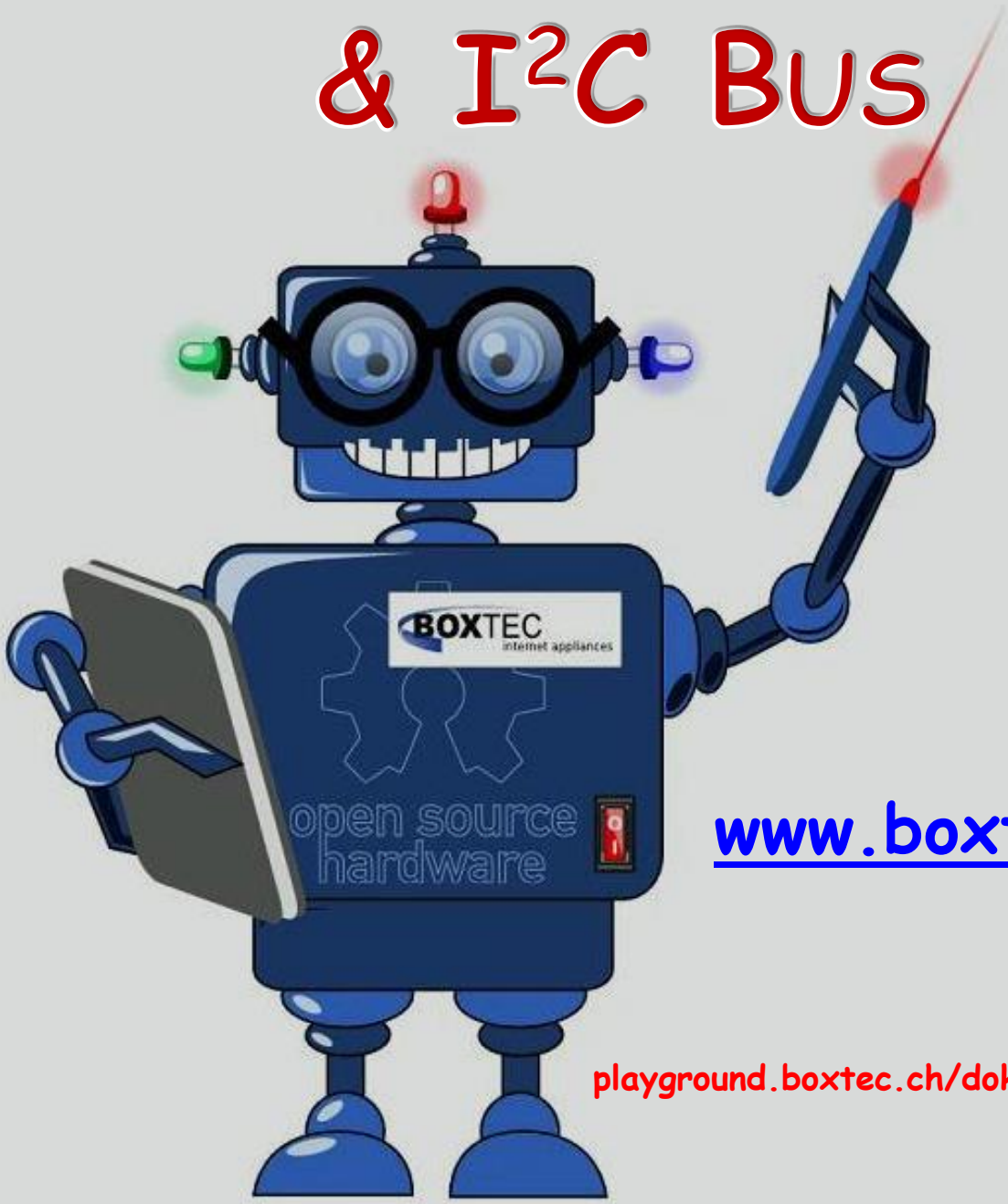
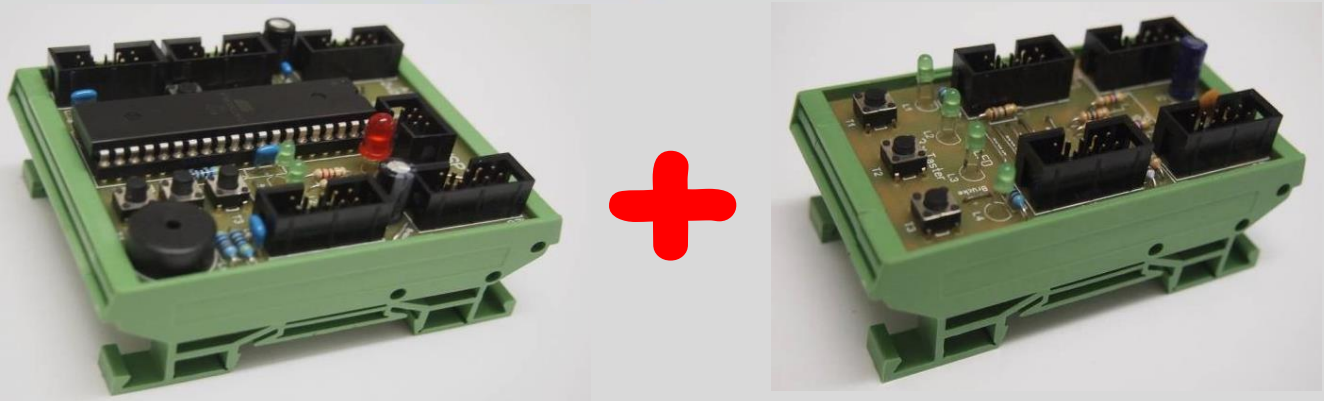
Mikrokontroller & I²C Bus



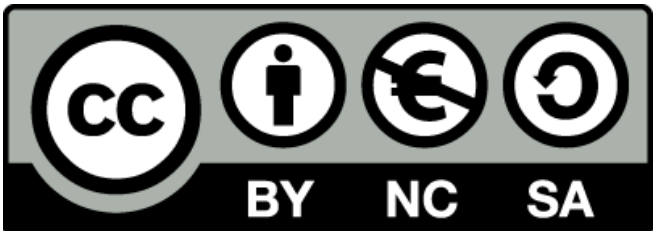
www.boxtec.ch

playground.boxtec.ch/doku.php/tutorial



Multitasking 2

Copyright



Sicherheitshinweise

Lesen Sie diese Gebrauchsanleitung, bevor Sie diesen Bausatz in Betrieb nehmen und bewahren Sie diese an einem für alle Benutzer jederzeit zugänglichen Platz auf. Bei Schäden, die durch Nichtbeachtung dieser Bedienungsanleitung verursacht werden, erlischt die Gewährleistung/Garantie. Für Folgeschäden übernehmen wir keine Haftung! Bei allen Geräten, die zu ihrem Betrieb eine elektrische Spannung benötigen, müssen die gültigen VDE-Vorschriften beachtet werden. Besonders relevant sind für diesen Bausatz die VDE-Richtlinien VDE 0100, VDE 0550/0551, VDE 0700, VDE 0711 und VDE 0860. Bitte beachten Sie auch nachfolgende Sicherheitshinweise:

- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige Person anschließen.
- In Schulen, Ausbildungseinrichtungen, Hobby- und Selbsthilfewerkstätten ist das Betreiben dieser Baugruppe durch geschultes Personal verantwortlich zu überwachen.
- In einer Umgebung in der brennbare Gase, Dämpfe oder Stäube vorhanden sind oder vorhanden sein können, darf diese Baugruppe nicht betrieben werden.
- Im Falle eine Reparatur dieser Baugruppe, dürfen nur Original-Ersatzteile verwendet werden! Die Verwendung abweichender Ersatzteile kann zu ernsthaften Sach- und Personenschäden führen. Eine Reparatur des Gerätes darf nur von fachkundigen Personen durchgeführt werden.
- Spannungsführende Teile an dieser Baugruppe dürfen nur dann berührt werden (gilt auch für Werkzeuge, Messinstrumente o.ä.), wenn sichergestellt ist, dass die Baugruppe von der Versorgungsspannung getrennt wurde und elektrische Ladungen, die in den in der Baugruppe befindlichen Bauteilen gespeichert sind, vorher entladen wurden.
- Sind Messungen bei geöffnetem Gehäuse unumgänglich, muss ein Trenntrafo zur Spannungsversorgung verwendet werden
- Spannungsführende Kabel oder Leitungen, mit denen die Baugruppe verbunden ist, müssen immer auf Isolationsfehler oder Bruchstellen kontrolliert werden. Bei einem Fehlers muss das Gerät unverzüglich ausser Betrieb genommen werden, bis die defekte Leitung ausgewechselt worden ist.
- Es ist auf die genaue Einhaltung der genannten Kenndaten der Baugruppe und der in der Baugruppe verwendeten Bauteile zu achten. Gehen diese aus der beiliegenden Beschreibung nicht hervor, so ist eine fachkundige Person hinzuzuziehen

Bestimmungsgemäße Verwendung

- Auf keinen Fall darf 230 V~ Netzspannung angeschlossen werden. Es besteht dann Lebensgefahr!
- Dieser Bausatz ist nur zum Einsatz unter Lern- und Laborbedingungen konzipiert worden. Er ist nicht geeignet, reale Steuerungsaufgaben jeglicher Art zu übernehmen. Ein anderer Einsatz als angegeben ist nicht zulässig!
- Der Bausatz ist nur für den Gebrauch in trockenen und sauberen Räumen bestimmt.
- Wird dieser Bausatz nicht bestimmungsgemäß eingesetzt kann er beschädigt werden, was mit Gefahren, wie z.B. Kurzschluss, Brand, elektrischer Schlag etc. verbunden ist. Der Bausatz darf nicht geändert bzw. umgebaut werden!
- Für alle Personen- und Sachschäden, die aus nicht bestimmungsgemäßer Verwendung entstehen, ist nicht der Hersteller, sondern der Betreiber verantwortlich. Bitte beachten Sie, dass Bedien- und /oder Anschlussfehler außerhalb unseres Einflussbereiches liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen.
- Der Autor dieses Tutorials übernimmt keine Haftung für Schäden. Die Nutzung der Hard- und Software erfolgt auf eigenes Risiko.

Multitasking 2 (einfacher Ansatz)

Diese Tutorial entstand unter Verwendung von Beiträgen von Falk. Dabei werden Textstellen wörtlich zitiert. http://www.mikrocontroller.net/articles/Multitasking

5. Zusammenfassung Teil 1

Kurze Zusammenfassung des 1. Teils:

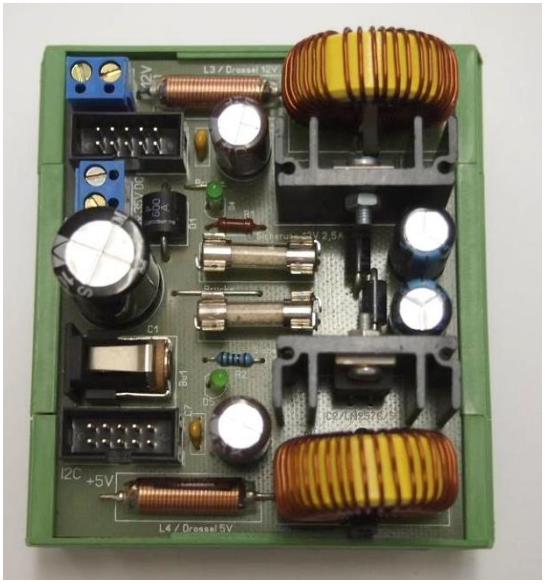
- Kurze Erklärung zum EVA Prinzip
- Erster Versuch mit einem einfachen Ansatz – keine richtige Funktion
- Zweiter Versuch mit einem einfachen Ansatz – unser Programm läuft korrekt

Den ersten Teil haben wir geschafft. Das Programm läuft auf unserem Board 1 ohne Fehler. Leider ist das Ergebnis mit den 2 LED sehr klein ausgefallen. Ich möchte diese Anwendung jetzt Stück für Stück erweitern. Dazu möchte ich auch andere Platinen und Erweiterungen nutzen.

6. Hardware Erweiterung

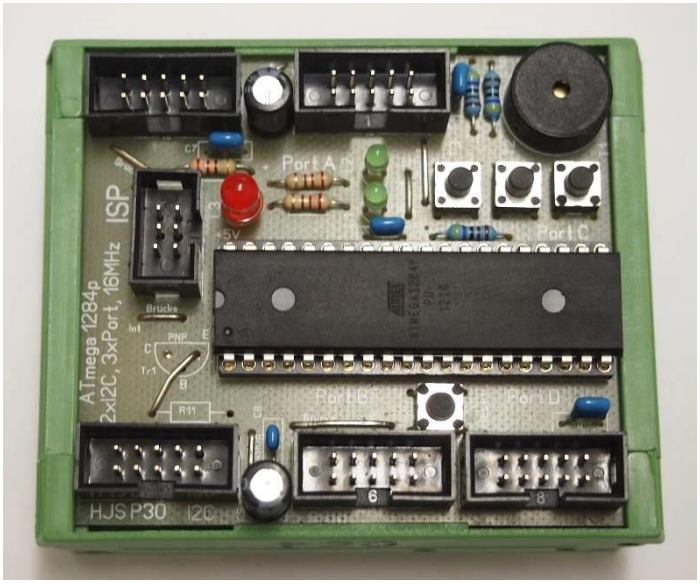
Im ersten Teil haben wir das Netzteil NT 2 und unser Board 1 verwendet. Jetzt möchte ich eine weitere Hardware verwenden.

Bisher verwendet:



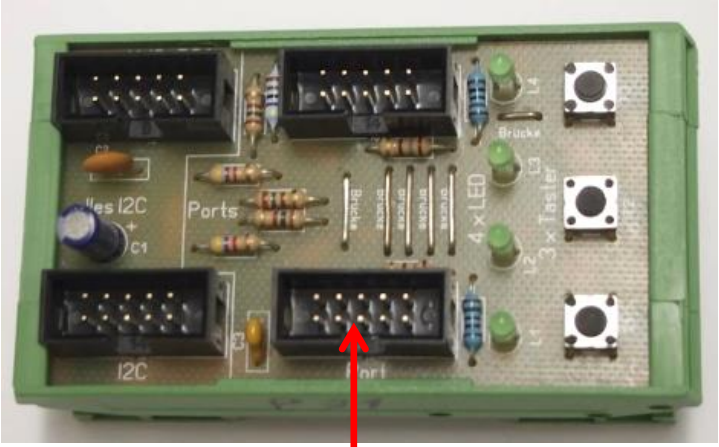
Netzteil NT 2

(+5V, +12V mit je 3A)



Board 1

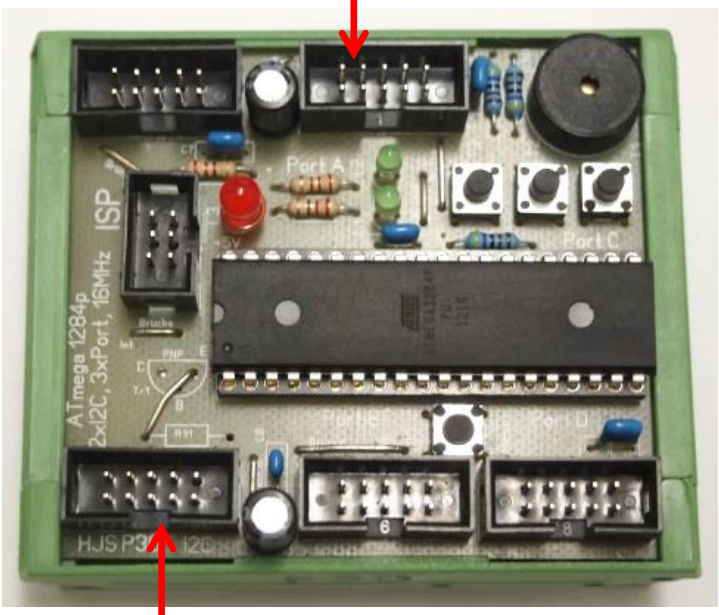
(Atmega 1284)



BPM In / Out 1

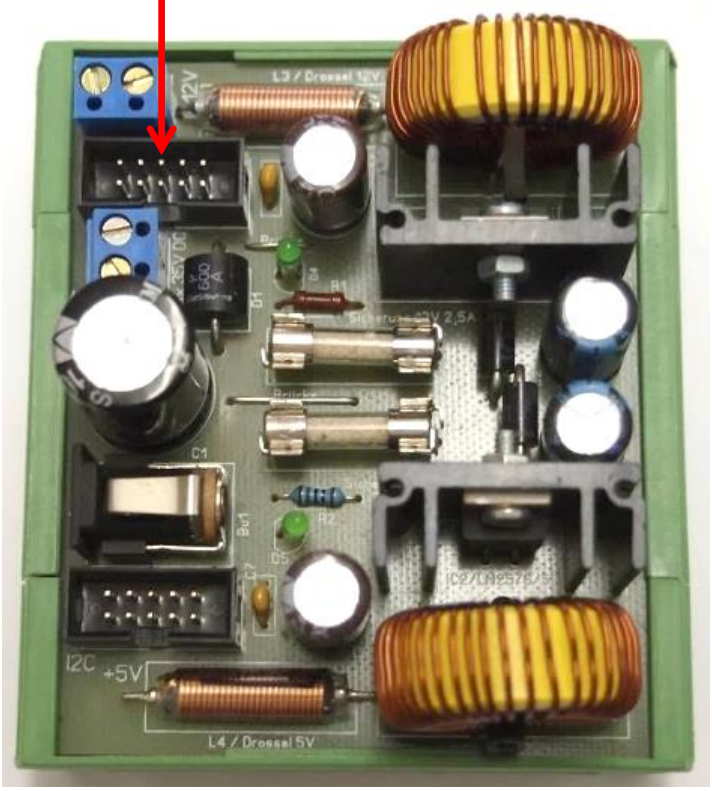
(Neue Hardware)

Verbindung
Board 1 – BPM In / Out 1



Board 1

Verbindung
NT2 – Board 1



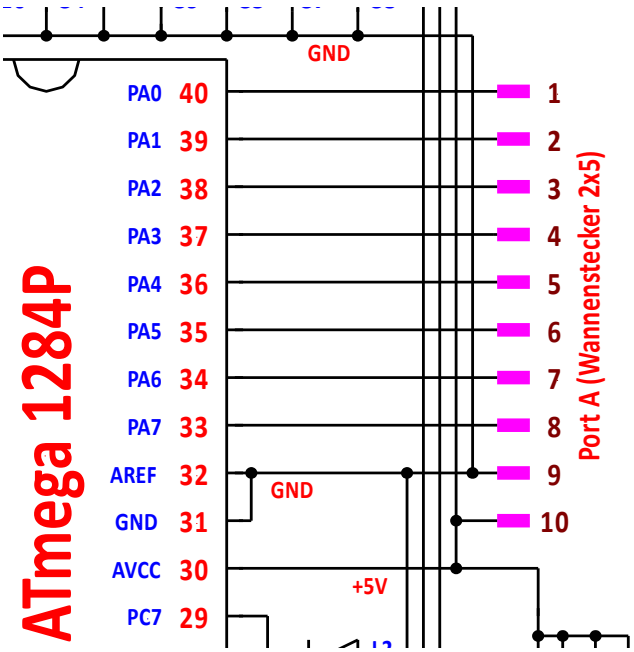
Netzteil NT 2

So könnte die neue Anordnung der Hardware aussehen.
Ich habe die notwendigen Verbindungen zwischen den einzelnen Modulen eingezeichnet. Es werden die 10 polg. Flachkabel mit Buchsenstecker genutzt.

Als nächste brauchen wir die Bezeichnungen der einzelnen Pins, die wir verwenden wollen. Aus dem Tut BPM In / Out 1 habe ich die Bezeichnungen übernommen.

P? 0	→	frei – wird nicht verwendet
P? 1	→	Taster T1
P? 2	→	Taster T2
P? 3	→	Taster T3
P? 4	→	LED L1
P? 5	→	LED L2
P? 6	→	LED L3
P? 7	→	LED L4

Ausschnitt aus der Schaltung des Board 1



Wir nutzen also den Port A des ATmega 1284 mit den Pins

- PA 4 → LED L1
- PA 5 → LED L2
- PA 6 → LED L3
- PA 7 → LED L4

Damit haben wir alle Angaben die wir brauchen.

7. Software Erweiterung

Als nächstes möchte ich das vorhandene Programm an die neue Hardware anpassen. Dazu werde ich einfach die entsprechenden Adressen/Pins ändern.

```
/* ATB_Multi_4.c Created: 20.08.2014 11:10:28 Author: AS */

#define F_CPU 16000000UL        // Angabe der Quarzfrequenz, wichtig für die Zeit
#include <util/delay.h>         // Einbindung Datei Pause
#include <avr/io.h>             // Einbindung Datei Ausgänge

int16_t led1=0;
int16_t led2=0;

void led_blinken1()
 {
   led1++;
   if(led1==1999)
   PORTA &= ~(1<<PA5);          // Schaltet Pin
   else
    {
      if(led1==3999)
        {
          PORTA |= (1<<PA5);    // Schaltet Pin
          led1=0;
        }
    }
 }
```

```
void led_blinken2()
 {
   led2++;
   if(led2==599)
   PORTA &= ~(1<<PA6);              // Schaltet Pin
   else
    {
      if(led2==1199)
        {
          PORTA |= (1<<PA6);        // Schaltet Pin
          led2=0;
        }
    }
 }
int main(void)
 {
   DDRA=0b01100000;                 // Port A auf Ausgang schalten
   while(1)                         // Programmschleife
    {
      led_blinken1();               // Aufruf Unterprogramm 1
      led_blinken2();               // Aufruf Unterprogramm 2
      _delay_ms(1);
    }
 }
```

Als nächste wollen wir alle LEDs zu unterschiedlichen Zeiten schalten. Das geht leider nur mit 6 Unterprogrammen.

```
/* ATB_Multi_5.c Created: 20.08.2014 14:30:43 Author: AS */

#define F_CPU 16000000UL        // Angabe der Quarzfrequenz, wichtig für die Zeit
#include <util/delay.h>         // Einbindung Datei Pause
#include <avr/io.h>             // Einbindung Datei Ausgänge

int16_t led1=0;
int16_t led2=0;
int16_t led3=0;
int16_t led4=0;
int16_t led5=0;
int16_t led6=0;

void led_blinken1()
 {
   led1++;
   if(led1==999)
   PORTC &= ~(1<<PC5);              // Schaltet Pin
   else
    {
      if(led1==1999)
```

```
        {
          PORTC |= (1<<PC5);          // Schaltet Pin
          led1=0;
        }
    }
 }
void led_blinken2()
 {
   led2++;
   if(led2==599)
   PORTC &= ~(1<<PC6);               // Schaltet Pin
   else
    {
      if(led2==1199)
        {
          PORTC |= (1<<PC6);         // Schaltet Pin
          led2=0;
        }
    }
 }
void led_blinken3()
 {
   led3++;
   if(led3==799)
   PORTA &= ~(1<<PA7);               // Schaltet Pin
   else
    {
      if(led3==1599)
       {
         PORTA |= (1<<PA7);          // Schaltet Pin
         led3=0;
       }
    }
 }
void led_blinken4()
 {
   led4++;
   if(led4==399)
   PORTA &= ~(1<<PA6);               // Schaltet Pin
   else
    {
      if(led4==799)
        {
          PORTA |= (1<<PA6);         // Schaltet Pin
          led4=0;
```

```
        }
    }
}
void led_blinken5()
{
  led5++;
  if(led5==150)
  PORTA &= ~(1<<PA5);              // Schaltet Pin
  else
   {
    if(led5==300)
      {
        PORTA |= (1<<PA5);         // Schaltet Pin
        led5=0;
      }
    }
}
void led_blinken6()
{
  led6++;
  if(led6==450)
  PORTA &= ~(1<<PA4);              // Schaltet Pin
  else
   {
    if(led6==900)
      {
        PORTA |= (1<<PA4);         // Schaltet Pin
        led6=0;
      }
    }
}
int main(void)
{
  DDRA=0b11110000;                 // Port A auf Ausgang schalten
  DDRC=0b01100000;                 // Port A auf Ausgang schalten
  while(1)                         // Programmschleife
   {
    led_blinken1();                // Aufruf Unterprogramm 1
    led_blinken2();                // Aufruf Unterprogramm 2
    led_blinken3();                // Aufruf Unterprogramm 3
    led_blinken4();                // Aufruf Unterprogramm 4
    led_blinken5();                // Aufruf Unterprogramm 5
    led_blinken6();                // Aufruf Unterprogramm 6
    _delay_ms(1);
    }
}
```

In unserem letzten Programm schalten alle 6 LEDs mit unterschiedlichen Zeiten. Dabei sind die Zeiten des Ein- und Ausschalten gleich lang. Wir wollen jetzt mal unterschiedliche Zeiten für Ein- und Ausschalten programmieren.

```
/* ATB_Multi_6.c Created: 20.08.2014 15:02:56 Author: AS */

#define F_CPU 16000000UL        // Angabe der Quarzfrequenz, wichtig für die Zeit
#include <util/delay.h>         // Einbindung Datei Pause
#include <avr/io.h>             // Einbindung Datei Ausgänge

int16_t led1=0;
int16_t led2=0;
int16_t led3=0;
int16_t led4=0;

void led_blinken1()
 {
   led1++;
   if(led1==800)
   PORTA &= ~(1<<PA4);          // Schaltet Pin
   else
    {
      if(led1==1000)
        {
          PORTA |= (1<<PA4);    // Schaltet Pin
          led1=0;
        }
    }
 }

void led_blinken2()
 {
   led2++;
   if(led2==500)
   PORTA &= ~(1<<PA5);          // Schaltet Pin
   else
    {
      if(led2==1000)
        {
          PORTA |= (1<<PA5);    // Schaltet Pin
          led2=0;
        }
    }
 }

void led_blinken3()
 {
   led3++;
   if(led3==1000)
   PORTA &= ~(1<<PA6);          // Schaltet Pin
```

```
   else
    {
      if(led3==2000)
        {
          PORTA |= (1<<PA6);          // Schaltet Pin
          led3=0;
        }
    }
 }
void led_blinken4()
 {
   led4++;
   if(led4==200)
   PORTA &= ~(1<<PA7);               // Schaltet Pin
   else
    {
      if(led4==1000)
        {
          PORTA |= (1<<PA7);          // Schaltet Pin
          led4=0;
        }
    }
 }
int main(void)
 {
   DDRA=0b11110000;                  // Port A auf Ausgang schalten
   while(1)                          // Programmschleife
    {
      led_blinken1();                // Aufruf Unterprogramm 1
      led_blinken2();                // Aufruf Unterprogramm 2
      led_blinken3();                // Aufruf Unterprogramm 3
      led_blinken4();                // Aufruf Unterprogramm 4
      _delay_ms(1);
    }
 }
```

In diesem Programm habe ich die 4 LEDs alle auf 1 Sekunde bezogen. Dabei habe ich die Zeiten unterschiedlich angeordnet. Eine länger aus, die andere länger an, eine mit gleichmässige Zeiten und die letzte im doppelten Takt. Es kann sich jeder anschauen und raten welche was macht (oder ins Programm schauen).

Im nächsten Programm wollen wir uns mal das Blinken genauer ansehen. Es gibt im Grund zwei verschiedene Arten:

- Das gleiche blinken mit mind. 2 LED
- Das Wechsel blinken mit mind. 2 LED (kein Lauflicht)

… und natürlich ein paar Sonderformen.

```
/* ATB_Multi_7.c Created: 20.08.2014 15:38:22 Author: AS */

#define F_CPU 16000000UL        // Angabe der Quarzfrequenz, wichtig für die Zeit
#include <util/delay.h>         // Einbindung Datei Pause
#include <avr/io.h>             // Einbindung Datei Ausgänge

int16_t led1=0;
int16_t led2=0;

void led_blinken1()
 {
   led1++;
   if(led1==500)
    {
      PORTA &= ~(1<<PA4);       // Schaltet Pin
      PORTA |= (1<<PA5);        // Schaltet Pin
    }
   else
    {
      if(led1==1000)
        {
          PORTA |= (1<<PA4);    // Schaltet Pin
          PORTA &= ~(1<<PA5);           // Schaltet Pin
          led1=0;
        }
    }
 }

void led_blinken2()
 {
   led2++;
   if(led2==700)
    {
      PORTA &= ~(1<<PA6);       // Schaltet Pin
      PORTA &= ~(1<<PA7);       // Schaltet Pin
    }
   else
    {
      if(led2==1400)
        {
          PORTA |= (1<<PA6);    // Schaltet Pin
          PORTA |= (1<<PA7);    // Schaltet Pin
          led2=0;
        }
    }
 }

int main(void)
 {
   DDRA=0b11110000;             // Port A auf Ausgang schalten
```

```
  while(1)                    // Programmschleife
   {
     led_blinken1();          // Aufruf Unterprogramm 1
     led_blinken2();          // Aufruf Unterprogramm 2
     _delay_ms(1);
   }
}
```

Das wichtigste am letzten Programm, das ich frei auswählen kann, welche Art von Blinken ich möchte und kann es z.B. durch einen Taster frei geben.
Damit sind wir aber schon beim nächsten Thema. Der Taster …

Einige Teile des Textes wurden zur besseren Übersicht farblich gestaltet.
Die Nutzung erfolgt auf eigenes Risiko.
Ich wünsche viel Spaß beim Bauen und programmieren
Achim

myroboter@web.de