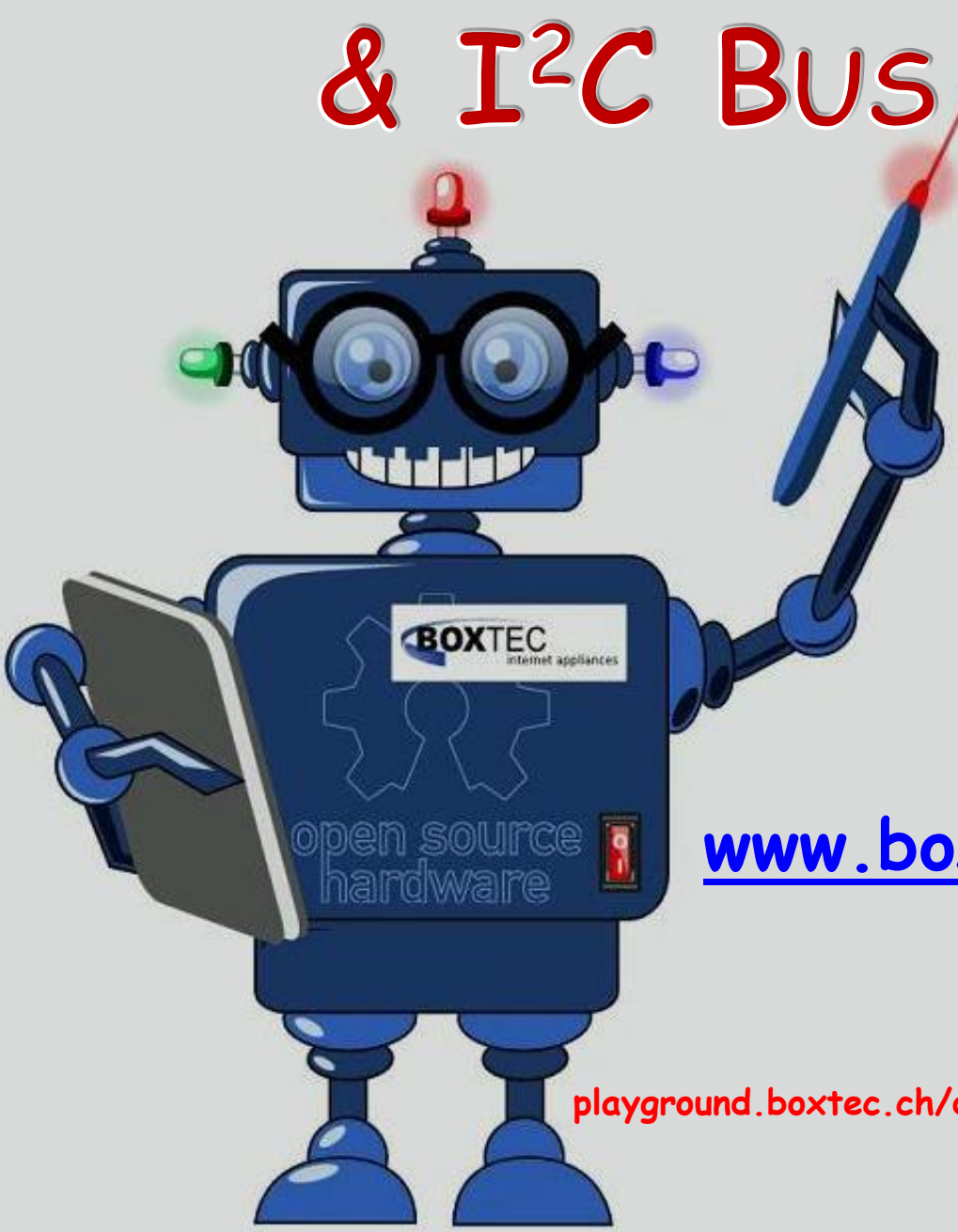
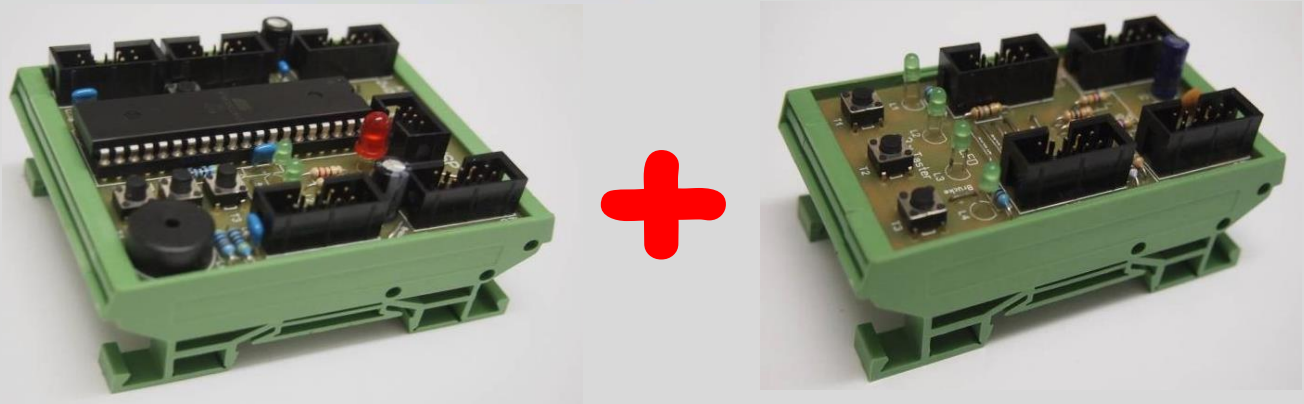
Mikrokontroller & I²C Bus



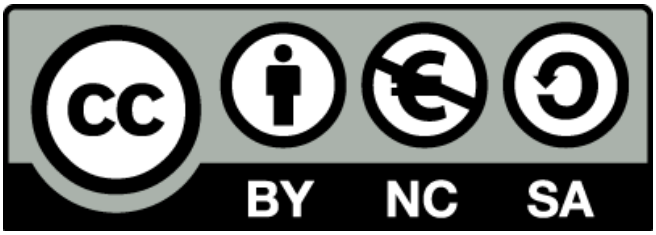
www.boxtec.ch

playground.boxtec.ch/doku.php/tutorial



Multitasking 3

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

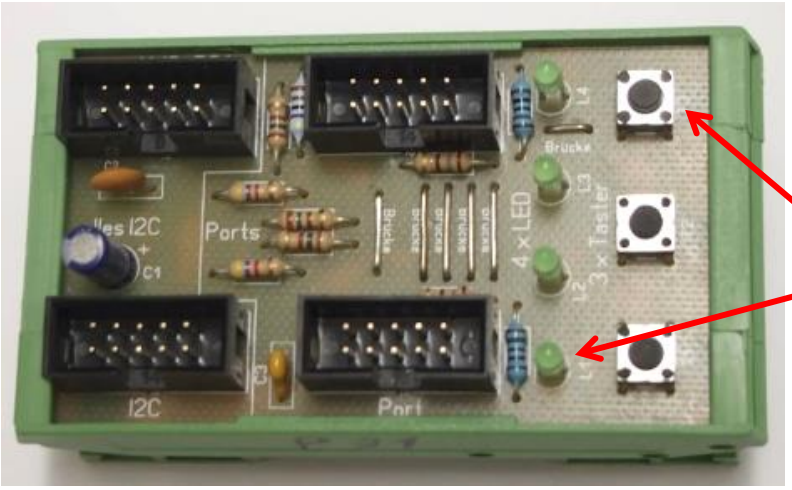
- Auf keinen Fall darf 230 V~ Netzspannung angeschlossen werden. Es besteht dann Lebensgefahr!
- Dieser Bausatz ist nur zum Einsatz unter Lern- und Laborbedingungen konzipiert worden. Er ist nicht geeignet, reale Steuerungsaufgaben jeglicher Art zu übernehmen. Ein anderer Einsatz als angegeben ist nicht 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 3 (einfacher Ansatz)

8. Hardware
9. Software Blinker
10. Software Blinker und Taster
11. Erhöhte Belastung

8. Hardware

Wir nutzen auch für die nächsten Programme weiter unsere Hardware. Die besteht immer noch aus dem NT2, dem Board 1 und unserem BPM In / Out 1.



Bei unseren nächsten Programmen nutzen wir aber zusätzlich zu den LEDs auch die Taster.

Sehen wir uns nochmal die Belegung unseres Ports A an:

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

Bisher haben wir PA4 bis PA7 für die LED verwendet. Jetzt nutzen wir auch die Pins PA1, PA2 und PA3 für unsere Taster.

In dem nachfolgenden Programm habe ich nur einen Blinker für die LED 3 und 4 gelassen. In dieser Art werde ich ihn jetzt immer verwenden, um eine Kontrolle des Programmes zu haben. So lange es blinkt, läuft das Programm
Es hat noch einen anderen Vorteil. Auf Grund der kurzen Blinkzeit kann ich fast jede Änderung der Durchlaufzeiten erkennen. Wenn z.B. ein Taster den Durchlauf stoppt, dann stoppt auch das Blinklicht.

9. Software Blinker

```
/* ATB_Multi_8.c Created: 20.08.2014 17:50:20 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;
void led_blinken1()
 {
   led1++;
   if(led1==300)
    {
      PORTA &= ~(1<<PA6);       // Schaltet Pin
      PORTA |= (1<<PA7);        // Schaltet Pin
    }
   else
    {
      if(led1==600)
        {
          PORTA |= (1<<PA6);    // Schaltet Pin
          PORTA &= ~(1<<PA7);   // Schaltet Pin
          led1=0;
        }
    }
 }
int main(void)
 {
   DDRA=0b11000000;             // Port A auf Ausgang schalten
   while(1)                     // Programmschleife
    {
      led_blinken1();           // Aufruf Unterprogramm 1
      _delay_ms(1);
    }
 }
```

Dieses Programm lässt die beiden LED ständig mit einer Zeit von 300 ms blinken. Dabei erfolgt bei jedem Durchlauf der Aufruf des Unterprogrammes led_blinken1. Innerhalb des Unterpro-grammes erfolgt dabei das wechselseitige schalten der beiden Led und das setzen und auslesen der Zähler.

Im nächsten Programm wollen wir den Zustand eines Tasters auslesen. Dabei darf die Durch-laufzeit auf keinen Fall verzögert oder unterbrochen werden.
Die einzelnen Teile unseres Programmes werden in Unterprogrammen ausgeführt. Das hat den Vorteil, dass ich kleinere „Häppchen" machen kann und jeden Teil extra programmieren und beschriften kann. Das wirkt sich besonders bei einer Fehlersuch aus, was ich persönlich nur bestätigen kann.
Zusätzlich gebe ich Daten aus einem Unterprogramm zurück und nutze diese im nächsten.

10. Software Blinker und Taster

```
/* ATB_Multi_9.c Created: 20.08.2014 20:50:21 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;
int taster;

int taste_lesen()               // Unterprogramm Taster lesen
 {
  if(PINA & 1<<PA3)             // Taster gedrückt?
  return 1;                     // wenn ja, Rückgabe 1
  else return 0;                // wenn nein, Rückgabe 0
 }

void led_taster(int taster)     // Unterprogramm led_taster
 {
  if(taster==1)                 // ist taster =1 (wahr)
   {                            // dann …
    PORTA &= ~(1<<PA4);         // Schaltet Pin
    PORTA |= (1<<PA5);          // Schaltet Pin
   }
  Else                          // wenn nicht …
   {                            // dann …
    PORTA |= (1<<PA4);          // Schaltet Pin
    PORTA &= ~(1<<PA5);         // Schaltet Pin
   }
 }

void led_blinken1()             // Unterprogramm led_blinken 1
 {
  led1++;
  if(led1==300)
   {
    PORTA &= ~(1<<PA6);         // Schaltet Pin
    PORTA |= (1<<PA7);          // Schaltet Pin
   }
  else
   {
    if(led1==600)
      {
       PORTA |= (1<<PA6);       // Schaltet Pin
       PORTA &= ~(1<<PA7);      // Schaltet Pin
       led1=0;
      }
   }
 }
```

```
int main(void)
 {
   DDRA=0b11110000;              // Port A Pins … auf Ausgang schalten
   while(1)                      // Programmschleife
    {
      taster = taste_lesen();    // Unterprogramm Taster lesen + Variable (1/0) in
                                 // taster speichern
      led_taster (taster);       // Unterprogramm led_taster + Übergabe Variable Taster
      led_blinken1();            // Aufruf Unterprogramm 1
      _delay_ms(1);              // Pause 1 ms
    }
 }
```

Sehen wir uns das Unterprogramm taste_lesen() einmal genauer an:

```
( 1 )    int taste_lesen()              // Unterprogramm Taster lesen
( 2 )     {                             // Klammer Anfang
( 3 )       if(PINA & 1<<PA3)           // Taster gedrückt ?
( 4 )       return 1;                   // wenn ja, Rückgabe 1
( 5 )       else return 0;              // wenn nein, Rückgabe 0
( 6 )     }                             // Klammer Ende
```

In der **Zeile 1** wird das Unterprogramm gestartet. In der **Zeile 3** wird der Zustand des Tasters (am Pin **PA3**) abgefragt. Ist der Taster **PA3 gedrückt**, erfolgt die **Rückgabe** von **1** in der **Zeile 4**.
Ist der Taster **PA3 nicht gedrückt**, erfolgt die **Rückgabe** von **0** in der **Zeile 5**.
Das ist die Taster abfrage, ohne Zeitverzögerung und Entprellung.

Als nächstes kommt der Aufruf der Unterprogramme und die Übergabe der Variablen:

```
( 1 )    while(1)                      // Beginn der Programmschleife
( 2 )     {                            // Klammer Anfang
( 3 )         taster = taste_lesen();  // Unterprogramm Taster lesen +
                                       // Variable (1/0) in taster speichern
( 4 )         led_taster (taster);     // Unterprogramm led_taster +
                                       // Übergabe Variable Taster
( 5 )         led_blinken1();          // Aufruf Unterprogramm 1
( 6 )         _delay_ms(1);            // Pause 1 ms
( 7 )     }                            // Klammer Ende
```

In der **Zeile 1** wird die Programmschleife gestartet. In der **Zeile 3** wird das Unterprogramm **taste_lesen** aufgerufen. In diesem Unterprogramm wird der Zustand des Tasters abgefragt und mit **0** (**offen/unwahr**) oder **1** (**geschlossen/wahr**) zurückgegeben und in der Variablen **taster** gespeichert. In der **Zeile 4** wird das Unterprogramm **led_taster** aufgerufen und die Variable **taster** übergeben. In der **Zeile 5** wird bei jedem Durchlauf **led_blinken1** aufgerufen. In der **Zeile 7** wird bei jedem Durchlauf **1 ms** verzögert

Als letztes komm dann noch die Anzeige des Taster:

```
( 1 )    void led_taster(int taster)      // Unterprogramm led_taster
( 2 )     {                               // Klammer Anfang
( 3 )       if(taster==1)                 // ist taster = 1 (wahr), dann …
( 4 )        {                            // Klammer Anfang
( 5 )          PORTA &= ~(1<<PA4);        // Schaltet Pin
               PORTA |= (1<<PA5);         // Schaltet Pin
( 6 )        }                            // Klammer Ende
( 7 )       else                          // wenn nicht …
( 8 )        {                            // dann …
( 9 )          PORTA |= (1<<PA4);         // Schaltet Pin
               PORTA &= ~(1<<PA5);        // Schaltet Pin
( 10 )       }                            // Klammer Ende
( 11 )      }                             // Klammer Ende
```

In der **Zeile 1** wird das Unterprogramm gestartet. In der **Zeile 3** wird die Variable **taster** abgefragt. Ist Variable **taster** 1 (**wahr**), dann ist die Bedingung **if(taster==1) wahr** und die **Zeile 5** wird mit dem schalten der PINs **PA4** und **PA5** (**ein/aus**) ausgeführt.
Die **Zeile 7** wird ausgeführt, wenn die Bedingung **0** (**nicht wahr**) ist und die **Zeile 9** ausgeführt. Es werden wieder die PINs **PA4** und PA5 (**aus/ein**) geschaltet.

11. Erhöhte Belastung

Im nächsten Programm wollen wir die Belastung noch einmal erhöhen. Dazu habe ich einen zweiten Taster programmiert

```
/* ATB_Multi_10.c Created: 21.08.2014 18:11:24 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;
int taster1;
int taster2;

int taste_lesen1()
 {
  if(PINA & 1<<PA3)
  return 1;
  else return 0;
 }

int taste_lesen2()
 {
       if(PINA & 1<<PA1)
       return 1;
       else return 0;
 }
```

```
void led_taster1(int taster1)
 {
   if(taster1==1)
    {
      PORTA &= ~(1<<PA4);          // Schaltet Pin  A4
      PORTA |= (1<<PA5);           // Schaltet Pin  A5
    }
   else
    {
      PORTA |= (1<<PA4);           // Schaltet Pin  A4
      PORTA &= ~(1<<PA5);          // Schaltet Pin  A5
    }
 }
void led_taster2(int taster2)
 {
   if(taster2==1)
    {
      PORTC &= ~(1<<PC5);          // Schaltet Pin
      PORTC |= (1<<PC6);           // Schaltet Pin
    }
   else
    {
      PORTC |= (1<<PC5);           // Schaltet Pin
      PORTC &= ~(1<<PC6);          // Schaltet Pin
    }
 }
void led_blinken1()
 {
   led1++;
   if(led1==300)
    {
      PORTA &= ~(1<<PA6);          // Schaltet Pin
      PORTA |= (1<<PA7);           // Schaltet Pin
    }
   else
    {
      if(led1==600)
        {
          PORTA |= (1<<PA6);       // Schaltet Pin
          PORTA &= ~(1<<PA7);      // Schaltet Pin
          led1=0;
        }
    }
 }
```

```
int main(void)
 {
   DDRA=0b11110000;              // Port A auf Ausgang schalten
   DDRC=0b01100000;              // Port C auf Ausgang schalten
   while(1)                      // Programmschleife
    {
      taster1 = taste_lesen1();  // Aufruf Unterprogramm 1
      led_taster1 (taster1);     // Aufruf Unterprogramm 2
      taster2 = taste_lesen2();  // Aufruf Unterprogramm 3
      led_taster2 (taster2);     // Aufruf Unterprogramm 4
      led_blinken1();            // Aufruf Unterprogramm 5
      _delay_ms(1);              // Pause 1 ms
    }
 }
```

Zu diesem Programm brauche ich nicht viel zu erklären, da es sich im Grund nur um Erweiterung des Vorläufers handelt. Habe dazu nur die Variablen angepasst und ein paar zusätzliche Kommentare eingefügt.
Wer Lust hat kann ja mal versuchen den Prozessor zu überlasten oder aus dem Tritt zu bringen. Wünsche viel Spass dabei

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