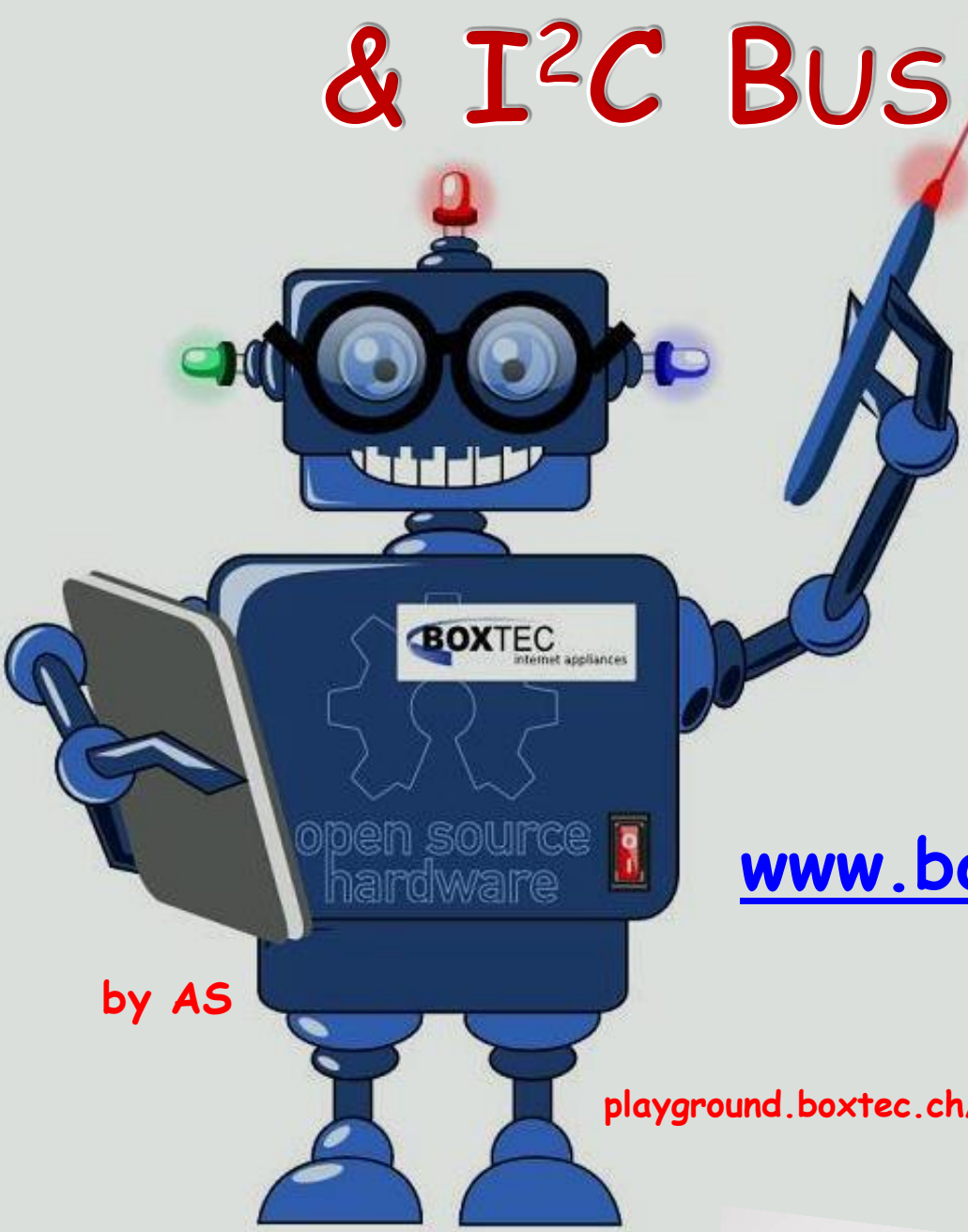
Mikrokontroller & I²C Bus

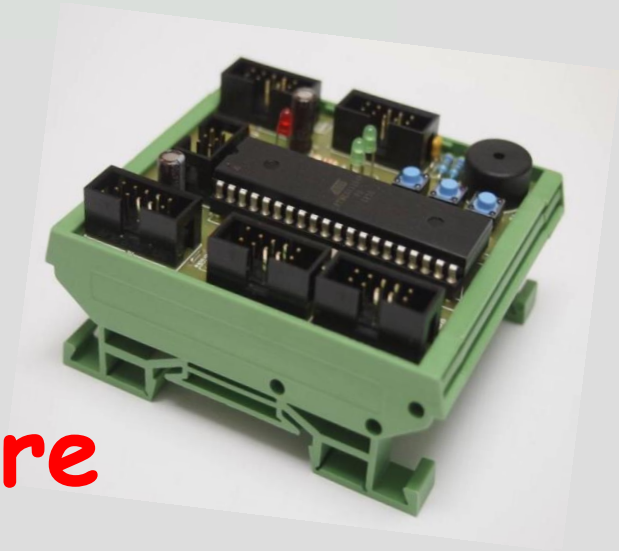


www.boxtec.ch

by AS

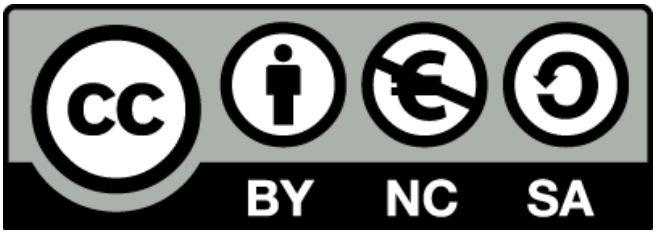
playground.boxtec.ch/doku.php/tutorial

I²C Bus zu USI
Eine Verbindung mit mehreren Prozessoren (Master – Slave)



USI - Software

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 Fehler 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

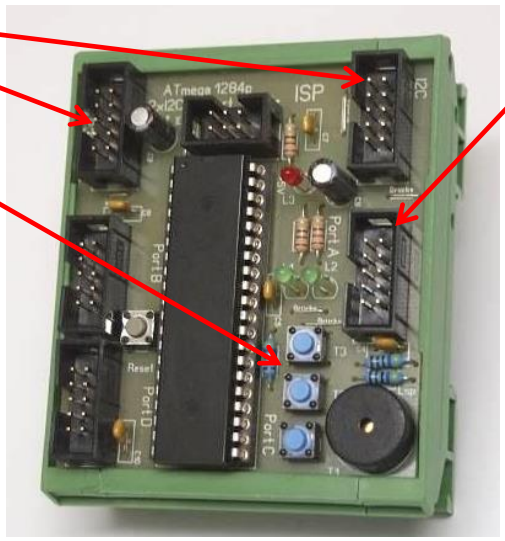
- Auf keinen Fall darf 230 V~ Netzspannung angeschlossen werden. Es besteht dann Lebensgefahr!
- Dieser Bausatz ist nur zum Einsatz unter Lern- und Laborbedingungen konzipiert worden. Er ist nicht geeignet, reale Steuerungsaufgaben jeglicher Art zu übernehmen. Ein anderer Einsatz als angegeben ist nicht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

I²C - Bus zu USI - Software

Eine Verbindung von Controller zu Controller

Ein Nachteil ist, das Master und Slave jeweils andere Programme brauchen. In diesem Teil möchte ich euch die Software und die Bedienung dazu vorstellen.

Master



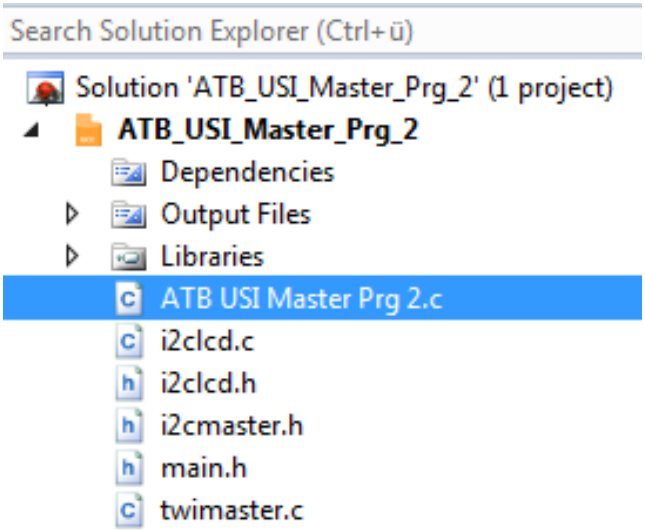

```
/* ATB_USI_Master_Prg_1.c Created: 29.04.2016 21:31:23 Author : AS */
/* I2C-Master-Routinen von Peter Fleury verwenden siehe:
http://homepage.hispeed.ch/peterfleury/avr-software.html#libs */
/*  Abgestimmt auf meine Hardware P30, D2, P52, NT2, 2xP20   Bedienung:
Slave - Port B Anzeige welche LED, T2 Umschaltung der Sende byte, L2 und L4 Anzeige der Sende byte
Master - T1 und T3 sende byte zum Slave, T2 hole byte vom Slave, Anzeige am Port A    */

#include <stdbool.h>
#include <avr/pgmspace.h>
#include "main.h"
#include <util/delay.h>
#include "i2clcd.h"
#include "i2cmaster.h"
#include "avr/io.h"
#include "util/delay.h"
#include "avr/interrupt.h"

#define slave_adresse 0x52                  // Slave Adresse schreiben
#define slave_adresse_r 0x53                // Slave Adresse lesen

uint8_t ret;                                // Kontrollvariable ob Slave vorhanden
uint8_t byte1;                              // Daten zum Senden vom Master zu Slave
uint8_t byte2;
uint8_t byte10;                             // Daten zum holen vom Slave zu Master
```

```
void startanzeige()                    // Titelbild
 {
   lcd_command(LCD_CLEAR);             // Leere Display
   _delay_ms(2);                       // Warte 2ms
   lcd_printlc(1,6,"USI Bus");         // Zeile 1
   lcd_printlc(2,2,"Verbindung von");  // Zeile 2
   lcd_printlc(3,2,"Prz ueber I2C ");  // Zeile 3
   lcd_printlc(4,4,"(achim S.)");      // Zeile 4
   _delay_ms(5000);                    // Warte 5000ms
 }

void slavetest()                       // Abfrage Slave, Fehlermeldung und Anzeige
 {                                     // Abfarge ob Slave vorhanden ist
   ret = i2c_start(slave_adresse);     // Start i2C mit Adresse Slave
   i2c_write(0x00);                    // Sende Daten
   i2c_stop();                         // I2C Stop
   if (ret == 0)                       // Bus ok - 0, kein Bus 1
    {                                  // Anzeige Slave ok
      lcd_command(LCD_CLEAR);          // Leere Display
      _delay_ms(2);                    // Warte 2ms
      lcd_printlc(2,4,"Slave ist ");   // Ausgabe Text
      lcd_printlc(3,5,"OK !!!!!");     // Slave OK
      _delay_ms(2000);                 // Warte 2s
    }
   else                                // Fehlermeldung
    {                                  // Anzeige Slave nicht ok
      lcd_command(LCD_CLEAR);          // Leere Display
      _delay_ms(2);                    // Warte 2ms
      lcd_printlc(2,6,"Slave ist");    // Ausgabe Schrift
      lcd_printlc(3,5,"Nicht OK");     // Slave Nicht O
      _delay_ms(2000);                 // Warte 2s
    }
 }

void s_write1 (void)                   // schreibe Daten 1 von Master zu Slave
 {
   i2c_start(slave_adresse);           // Slave ist bereit zum Schreiben
   i2c_write(0x00);                    // Buffer Startadresse setzen
   i2c_write(42);                      // Drei Bytes schreiben...
   i2c_stop();                         // Zugriff beenden
 }

void s_write2 (void)                   // schreibe Daten 2 von Master zu Slave
 {
   i2c_start(slave_adresse);           // Slave ist bereit zum Schreiben
   i2c_write(0x00);                    // Buffer Startadresse setzen
   i2c_write(43);                      // Drei Bytes schreiben...
   i2c_stop();                         // Zugriff beenden
 }

void s_read()                          // lese Slave
 {
   i2c_start(slave_adresse);           // Start Bus schreiben
   i2c_write(0x00);
```

```
  i2c_start(slave_adresse_r);          // starte Slave lesen
  //byte2= i2c_readAck();              // 2. Byte lesen und in "gelesen" ablegen
  byte10=i2c_readNak();                // letztes Byte lesen, darum kein ACK
  i2c_stop();                          // Zugriff beenden
 }

int main(void)
 {
  cli();                               // Interrupts deaktiviert
  i2c_init();                          // Starte I2C Bus
  lcd_init();                          // Starte I2CLCD
  // Display Befehle
  lcd_command(LCD_DISPLAYON | LCD_CURSOROFF | LCD_BLINKINGOFF);
  lcd_light(0);                        // Licht an
  startanzeige();
  lcd_command(LCD_CLEAR);              // Leere Display
  _delay_ms(2);                        // Warte 2ms
  slavetest();
  _delay_ms(2);

  DDRC=0b00000000;                     // Taster
  DDRA=0b11111111;                     // LED
  PORTC|=0b01111111;                   // Taster
  PORTA|=0b11111111;                   // LED
  while(1)
   {
    // ========>> AUF GELESENE DATEN REAGIEREN
    if (byte10 == 30)
     {                                 // Wenn Wert 30 gelesen wurde...
      PORTA &= ~( (1<<PINA6));         // schalte Port A6
     }
    else
     {
      PORTA |=(1<<PINA6);              // wenn nicht lösche Port A6
     }
    if (byte10 == 40)
     {                                 // Wenn Wert 40 gelesen wurde...
      PORTA &=~(1<<PINA7);             // schalte Port A7 ein
     }
    else
     {
      PORTA |=(1<<PINA7);              // wenn nicht lösche
     }

    // ========>> TASTENEINGABEN
    if (!(PINC & (1<<PINC2)) )         // Taster T1
     {                                 // Wenn T1 gedrückt...
      PORTA &=~(1<<PINA1);             // LED 2 A1 ein
      s_write1();                      // Schreib 1 Funktion aufrufen
     }
    else
     {
      PORTA |=(1<<PINA1);              // sonst LED 2 A1 aus
```

```
      }
    // ========>> TASTENEINGABEN
    if (!(PINC & (1<<PINC4)) )          // Taster T 3
      {                                 // Wenn T1 gedrückt...
        PORTA &=~(1<<PINA2);            // LED 3 A2 ein
        s_write2();                     // Schreib 2 Funktion aufrufen
      }
    else
      {
        PORTA |=(1<<PINA2);             // sonst LED 3 aus
      }
    // vom Slave lesen
    if (!(PINC & (1<<PINC3)) )          // Taster T2 mitte
      {                                 // Wenn T 2 gedrückt...
        PORTA &=~(1<<PINA5);            // LED  6 A5 ein
        s_read();                       // ...Lese-Funktion aufrufen
      }
    else
      {
        PORTA |=(1<<PINA5);             // LED 6 A5 aus
      }
  }
}
```

Diese Dateien sind zum Betrieb notwendig:



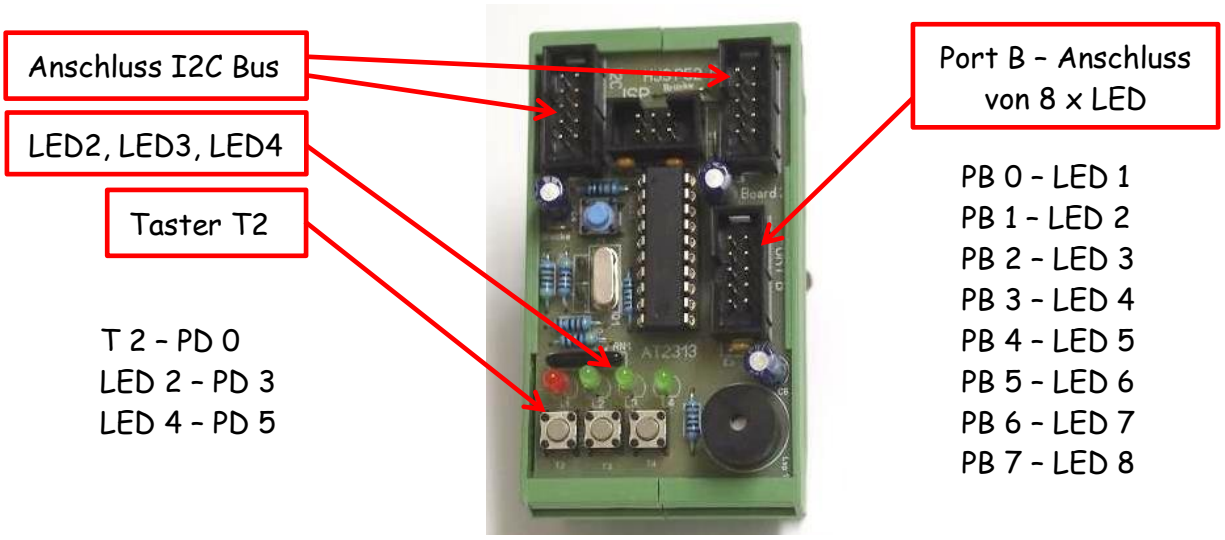
Funktion:

T1 sendet 42 an Slave
T3 sendet 43 an Slave
T2 liest 30 oder 40 vom Slave

Kontrolle Slave wird über das Display angezeigt.

Die notwendigen Dateien müssen in das System eingebunden werden. Es handelt sich dabei um die bekannten Dateien von Peter. Diese Dateien sind auch zum Betrieb mit anderen ICs im I²C Bus notwendig.
Die verwendeten Adressen, für den Master und Slave, muss innerhalb des Programmes eingetragen werden. Innerhalb des Programmes des Masters erfolgt eine Kontrolle, ob der angegebene Slave vorhanden ist. Ist der Slave oder die Adresse nicht vorhanden, stoppt das Programm.
Zur Kontrolle der Taster Bedienung und Anzeige der Daten werden verschiedene LED angesteuert.
Es können weitere Slave angeschlossen werden. Dabei immer auf die korrekten Adressen achten. Jeder Slave bekommt seine eigene Adresse (Lesen/Schreiben).
Bitte die Frequenz (SCL_CLOCK) im twimaster.c von 400 kHz auf 100 kHz umstellen.

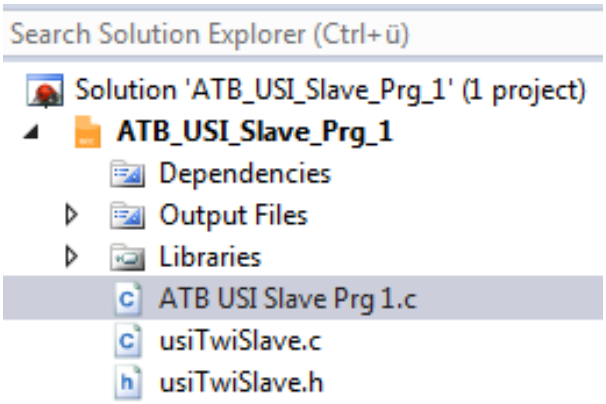
Slave




```
/* ATB_USI_Slave_Prg_1.c Created: 15.05.2016 10:39:23 Author : AS */
/* I2C-Master-Routinen von Peter Fleury verwenden siehe:
http://homepage.hispeed.ch/peterfleury/avr-software.html#libs  Abgestimmt auf meine Hardware
P30, D2, P52, NT2, 2xP20   Bedienung: Slave - Port B Anzeige welche LED, T2 Umschaltung der Sende
byte, L2 und L4 Anzeige der Sende byte
Master - T1 und T3 sende byte zum Slave, T2 hole byte vom Slave, Anzeige am Port A    */

#define F_CPU 16000000UL
#include <stdlib.h>                    // für den Ati 2313A im usiTwiSlave.c A zufügen
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include "usiTwiSlave.h"
#define slave_adresse 0x52             // Adresse Slave
uint8_t  byte1, byte2,byte10;
uint16_t buffer;
int main(void)
 {
   cli();                              // Interrupts aus
   usiTwiSlaveInit(slave_adresse);     // Init TWI slave
   sei();                              // Interrupts ein
   DDRD=0b00111000;                    // setze Taster + LED DDR D
   DDRB=0b10100111;                    // setze LED DDR B
   PORTD|=0b00111000;                  // setze Taster + LED Port D
   PORTB|=0b00000111;                  // setze LED Port B
   while(1)
    {
      byte1 = rxbuffer[0];             // schreibe Daten in den Eingangspuffer
      byte2 = rxbuffer[1];
      if (byte1==43)                   // Abfrage byte1 auf 43
         {                             // wenn dann ...
           PORTB &=~(1<<PINB1);        // LED 2 B1 ein
           PORTB |=(1<<PINB2);         // LED 3 B2 aus
```

```
    }
  if (byte1==42)                  // Abfrage byte1 auf 42
    {                             // wenn dann ...
      PORTB &=~(1<<PINB2);        // LED 3 B2 ein
      PORTB |=(1<<PINB1);         // LED 2 B1 aus
    }
  if (PIND & (1<<PIND0))          // Taster T2 D0
    {                             // Wenn T1 gedrückt...
      PORTD &=~(1<<PIND3);        // LED 2 D2 ein
      byte10=40;                  // setze byte1 auf 40
      PORTD |=(1<<PIND5);         // LED 4 D5 aus
    }
  else
    {
      PORTD &=~(1<<PIND5);        // LED 4 D5 ein
      PORTD |=(1<<PIND3);         // LED 2 D3 aus
      byte10=30;                  // setze byte1 auf 30
    }
  txbuffer[0]=byte10;             // byte1 in den Sendepuffer auf [0]
 }                                // end.while
}                                 // end.main
```

Diese Dateien sind zum Betrieb des Slave notwendig:



Funktion:

T2 – Umschaltung auf 30 oder 40
LED2 und 4 – Anzeige der Umschaltung

rxbuffer - Eingangspuffer
txbuffer - Sendepuffer

Der Zustand der übermittelten Werte wird jeweils durch LED angezeigt. Innerhalb des Slave erfolgt keine Kontrolle der Adresse.
Die korrekte Adresse im Master und Slave unbedingt überprüfen.

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

Quellenangabe:

http://jump.to/fleury
http://timogruss.de/
http://www.jtronics.de/avr-projekte/atmega-grbl-cnc-controller.html