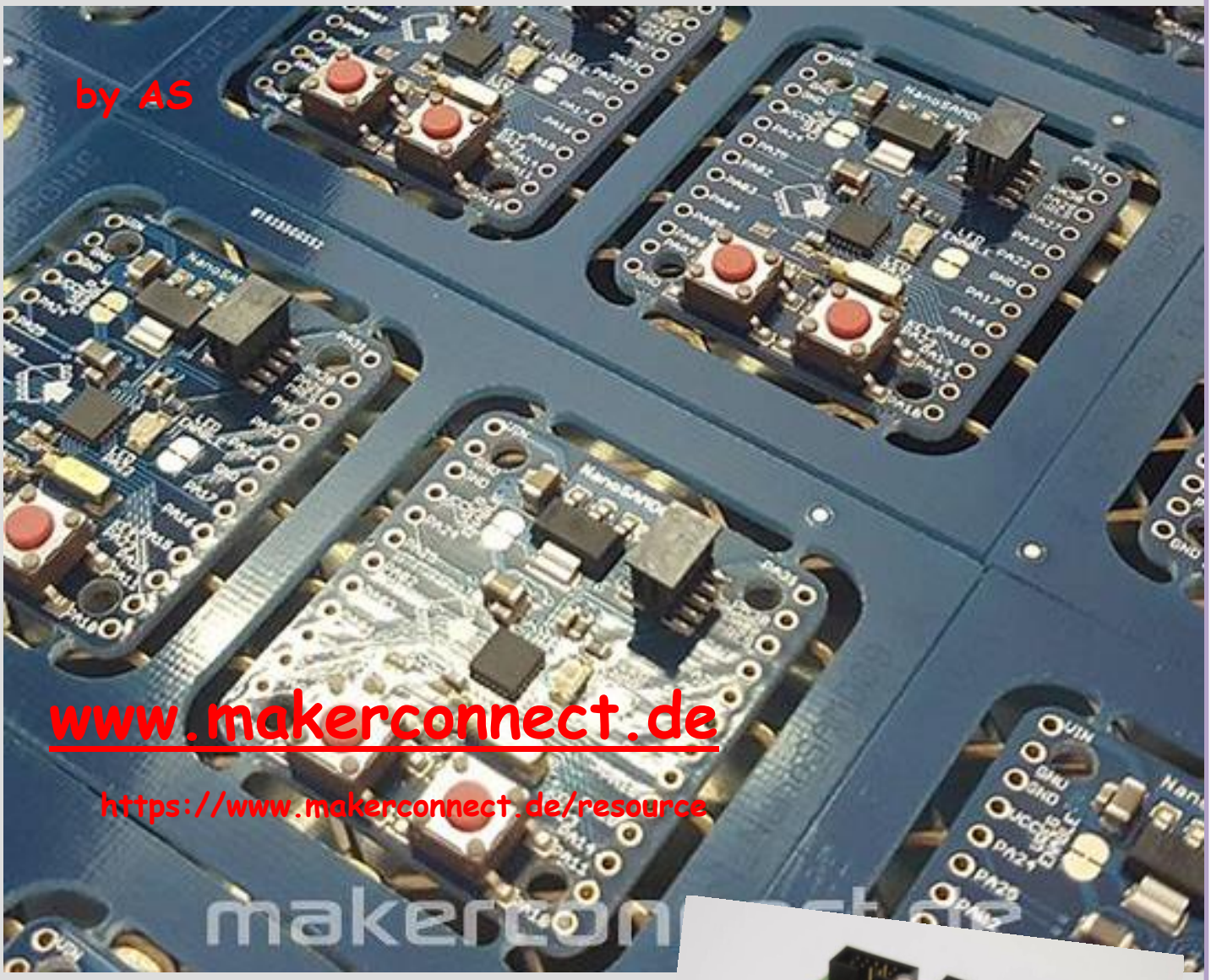
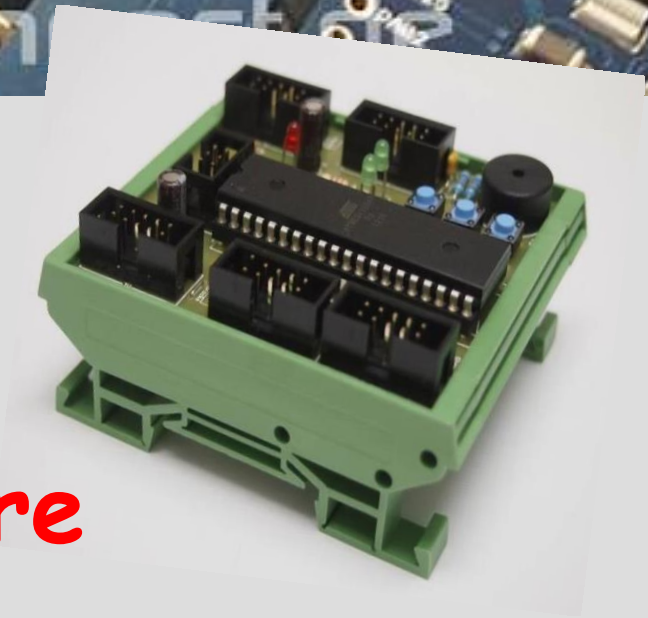
MIKROKONTROLLER & I²C BUS



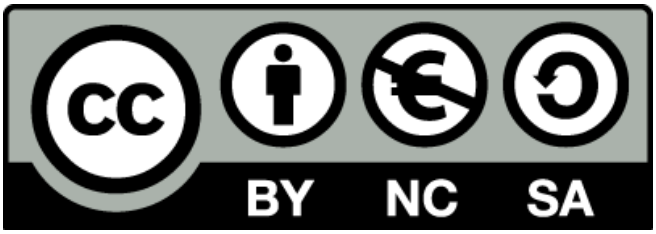
I²C Bus zu USI
Eine Verbindung mit mehreren Prozessoren (Master – Slave)



USI - Software

Copyright

Sofern nicht anders angegeben, stehen die Inhalte dieser Dokumentation unter einer „Creative Commons - Namensnennung-NichtKommerziell-Weitergabe unter gleichen Bedingungen 3.0 DE Lizenz“



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

Im ersten Teil über USI habe ich die Verbindung zwischen 2 Controller mit dem I²C-Bus und 4 Verbindungen (SCL, SDA, Vcc, GND) vorgestellt. Ein Nachteil oder Vorteil ist, das Master und Slave jeweils andere Programme brauchen. Dadurch kann der Slave eigenständige Programme ausführen und den Master entlasten.
In diesem Teil möchte ich euch die Software und die Bedienung dazu vorstellen.

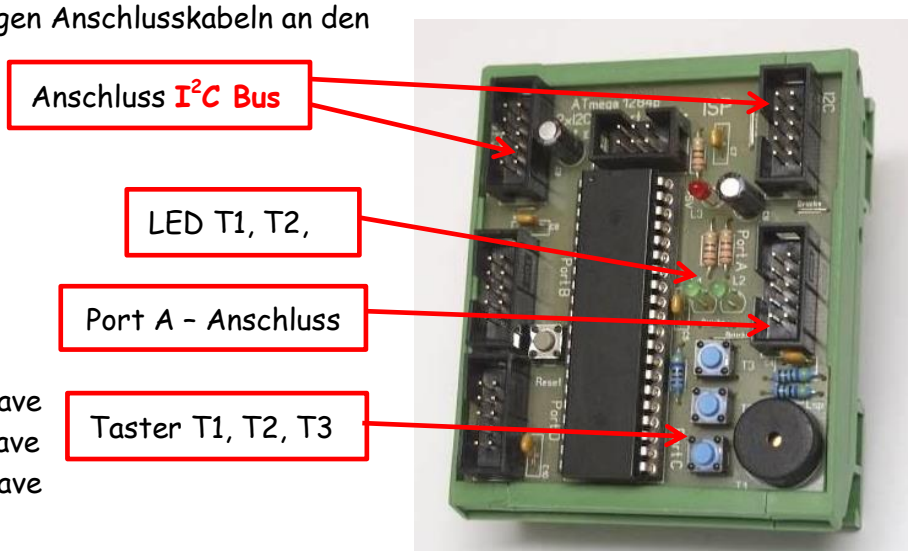
Master

Sehen wir uns als ersten den Master an. Die Verbindung zum Slave und anderen Modulen erfolgt mit den 10 poligen Anschlusskabeln an den I²C Buchsen.

Taster T1 – PC 2
Taster T2 – PC 3
Taster T3 – PC 4

LED 1 – PC 5
LED 2 – PC 6

Bedienung:
T1 – sende Wert an Slave
T2 – lese Wert vom Slave
T3 – sende Wert an Slave



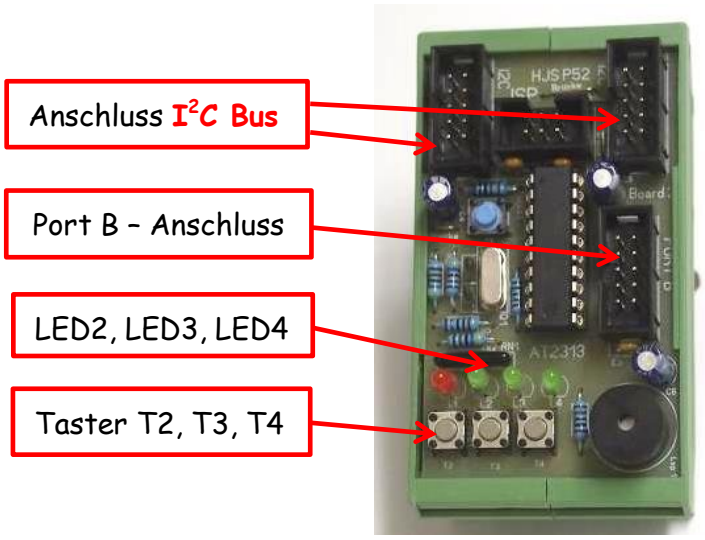
Es kann sowohl ein Slave mit einem Prozessor z.B. Attiny 2313 (mit USI) oder andere Module mit I²C Bus IC's angeschlossen werden. Den Anschluss mehrerer Slave mit Prozessor habe ich nicht getestet. Am Port A können weitere LEDs angeschlossen werden.

Slave

Als nächstes sehen wir uns den Slave an. Am Port B können weitere LEDs angeschlossen werden.

LED 2 – PD 3
LED 3 – PD 4
LED 4 – PD 5

Taster T 2 – PD 0
Taster T 3 – PD 1
Taster T 4 – PD 2



Bedienung: Taster T2 schaltet LED 2 ein oder aus

Kommen wir nun zum eigentlichen Programm. Als erstes kommt der Code für den Master. Habe in fast jede Zeile Kommentare zur Erklärung eingefügt. Damit dürfte es keine Probleme bei der Ausführung oder Fehlersuche geben.

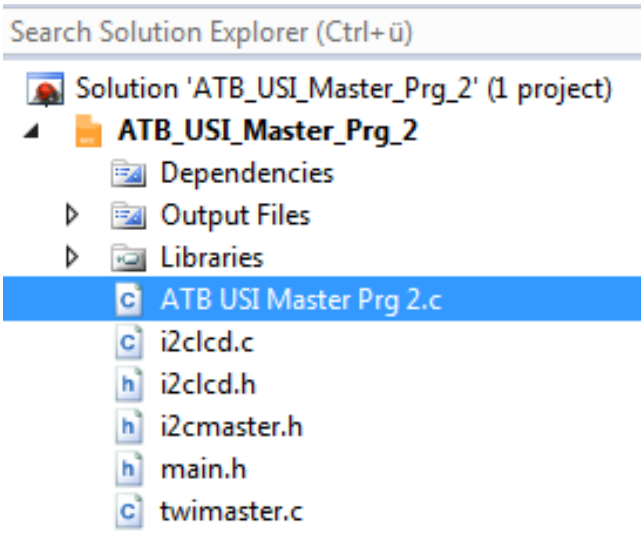
Master:

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

#define slave_adresse 0x52              // Angabe Slave Adresse

uint8_t ret;                            // Kontrollvariable ob Slave vorhanden
uint8_t byte1, byte2;                   // Daten zum senden vom Master zu Slave
uint8_t byte3, byte4;                   // Daten zum holen vom Slave zu Master
char Buffer[20];                        // Angabe Grösse Buffer

void startanzeige()                     // Titelbild
 {
   lcd_command(LCD_CLEAR);              // Leere Display
   _delay_ms(2);                        // Warte 2ms
   lcd_printlc(1,2,"USI Bus M-Prg1");   // Zeile 1
   lcd_printlc(2,2,"Verbindung von");   // Zeile 2
   lcd_printlc(3,2,"Prz ueber I2C ");   // Zeile 3
   lcd_printlc(4,2,"(by achim S.)");    // Zeile 4
   _delay_ms(3000);                     // Warte 3000ms
 }

void slavetest()                        // Abfrage Slave, Fehlermeldung und Anzeige
 {                                      // Abfrage 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(2,5,"Slave ist");     // Ausgabe Schrift
      lcd_printlc(3,5,"Nicht OK");      // Slave Nicht ok
      _delay_ms(2000);                  // Warte 2s
    }
  lcd_command(LCD_CLEAR);               // Leere Display
  _delay_ms(2);                         // Warte 2ms
  lcd_printlc(1,1,"Taste aus ");        // Ausgabe Text
 }

void s_write (uint8_t wert)             // schreibe Daten von Master zu Slave
 {
  i2c_start(slave_adresse);             // Slave ist bereit zum schreiben
  i2c_write(0x00);                      // Buffer Startadresse setzen
  i2c_write(wert);                      // Bytes schreiben...
  i2c_stop();                           // Zugriff beenden
 }

void s_read()                           // lesen Slave
 {
  i2c_start(slave_adresse);             // Start Bus schreiben
  i2c_write(0x00);
  i2c_rep_start(slave_adresse+1);       // starte Slave lesen
  byte3=i2c_readAck();                  // Byte lesen und in "gelesen" ablegen
  byte4=i2c_readNak();                  // letztes Byte lesen
  i2c_stop();                           // Zugriff beenden

  lcd_printlc(3,3,"byte 3:");           // Anzeige byte 3
  itoa(byte3, Buffer, 10 );             // umrechnung
  lcd_printlc(3,12,Buffer);             // Anzeige
  lcd_printlc(4,3,"byte 4:");           // Anzeige byte 4
  itoa(byte4, Buffer, 10 );             // umrechnung
  lcd_printlc(4,12,Buffer);             // Anzeige
 }

int main(void)
 {
  cli();                                // Interrupts deaktiviert
  i2c_init();                           // Starte I2C Bus
  lcd_init();                           // Starte LCD
  // 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=0b01100000;                    // setze Port C
   PORTC|=0b01001111;                  // setze Port C

   while(1)
    {
// ========>> TASTENEINGABEN T1
      if (!(PINC & (1<<PINC2)) )       // Taster T1
       {                               // Wenn T1 gedrückt...
         s_write(42);                  // Schreib Funktion aufrufen
         lcd_printlc(1,1,"Taste 1   "); // Ausgabe Text
       }
// ========>> TASTENEINGABEN T3
      if (!(PINC & (1<<PINC4)) )       // Taster T 3
       {                               // Wenn T3 gedrückt...
         s_write(43);                  // Schreib Funktion aufrufen
         lcd_printlc(1,1,"Taste 2   "); // Ausgabe Text
       }
//  =======>> Tastereingabe 2 - vom Slave lesen
      if (!(PINC & (1<<PINC3)) )       // Taster T2
       {                               // Wenn T 2 gedrückt...
         PORTC &=~(1<<PINC5);          // LED  5 ein
         s_read();                     // Lese-Funktion aufrufen
       }
      else
       {
         PORTC |=(1<<PINC5);           // LED 5 aus
       }
    }
 }
```

Diese Dateien sind zum Betrieb notwendig:



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, wird das auf dem Display angezeigt.
Es können weitere Slave angeschlossen werden. Dabei immer auf die korrekten Adressen

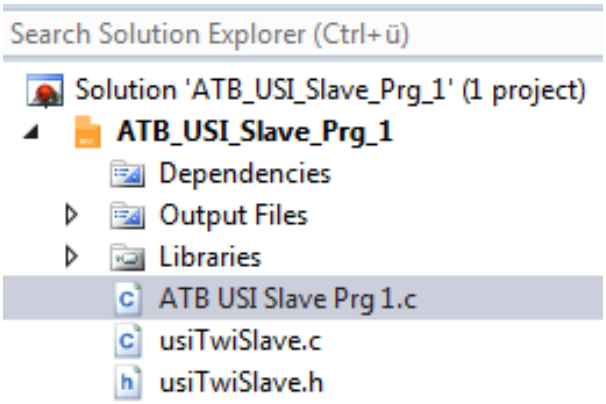
achten. Jeder Slave bekommt seine eigene Adresse (Lesen/Schreiben).
Bitte die Frequenz (SCL_CLOCK) im twimaster.c von 400 kHz auf 100 kHz umstellen.
In der main habe ich die Bus Adresse für das Display und die CPU Frequenz angegeben.

Slave:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include "usiTwiSlave.h"

#define slave_adresse 0x52                    // Adresse Slave

uint8_t byte1, byte2;
uint8_t byte3, byte4;

int main(void)
 {
   cli();                                     // Interrupt aus
   usiTwiSlaveInit(slave_adresse);            // Init TWI slave
   sei();                                     // Interrupts ein
   DDRD=0b10111000;                           // Taster + LED DDR D
   PORTD|=0b00111000;                         // Taster + LED Port D
   while(1)
    {
      byte1 = rxbuffer[0];                    // Daten in den Eingangspuffer
      byte2 = rxbuffer[1];
      if (byte1==43)                          // Abfrage byte1 auf 43
       {                                      // wenn dann ...
         PORTD &=~(1<<PIND5);                 // LED 2 ein
         PORTD |=(1<<PIND4);                  // LED 3 aus
       }
      if (byte1==42)                          // Abfrage byte1 auf 42
       {                                      // wenn dann ...
         PORTD &=~(1<<PIND4);                 // LED 3 ein
         PORTD |=(1<<PIND5);                  // LED 2 aus
       }
      if (PIND & (1<<PIND0))                  // Taster T2
       {                                      // Wenn T2 gedrückt...
         byte3=11;                            // setze byte
         byte4=22;
         PORTD |=(1<<PIND3);                  // LED 2 ein
       }
      else
       {                                      // wenn nicht
         PORTD &=~(1<<PIND3);                 // LED 2 aus
         byte3=33;                            // setze byte
         byte4=44;
       }
```

```
    txbuffer[0]=byte3;                    // Daten in Sendepuffer
    txbuffer[1]=byte4;
  }                                       // end.while
}                                         // end.main
```

Diese Dateien sind zum Betrieb des Slave notwendig:



Funktion:

T2 – Umschaltung byte, Anzeige mit **LED 3**
LED 3 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://timogruss.de/
http://www.jtronics.de/category/avr-tutorial/
http://homepage.hispeed.ch/peterfleury/avr-software.html