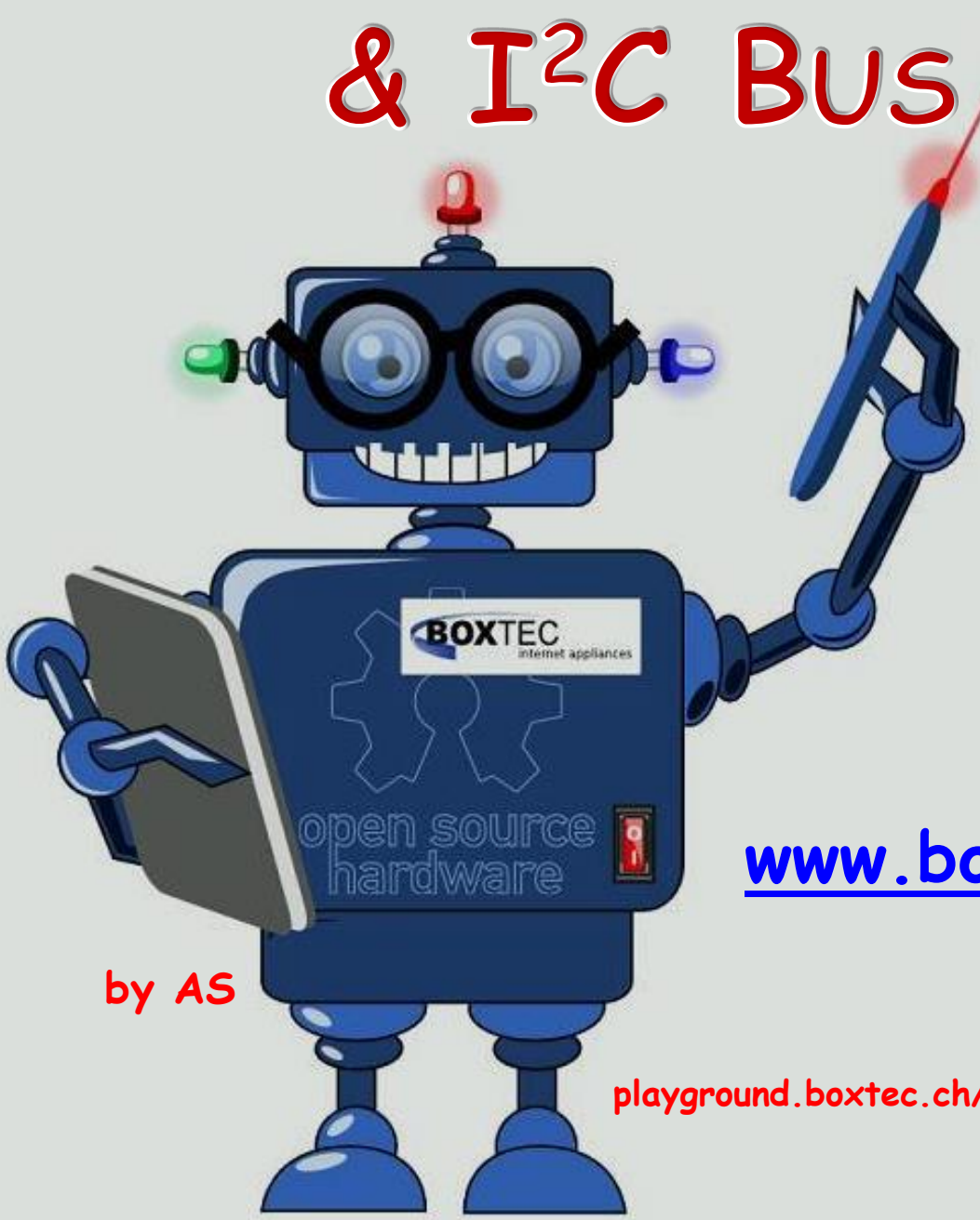
Mikrokontroller & I²C Bus



www.boxtec.ch

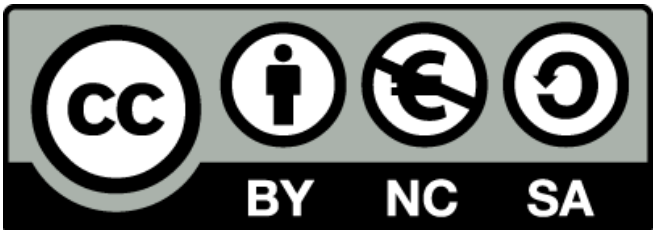
by AS

playground.boxtec.ch/doku.php/tutorial

Anzeige 2 mit dem
HT16K33, 3 x LED Matrix
Anzeigen (8x8), 2 x I²C – Bus
= Teil 2 – Software =

Anzeige 2 - Teil 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 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

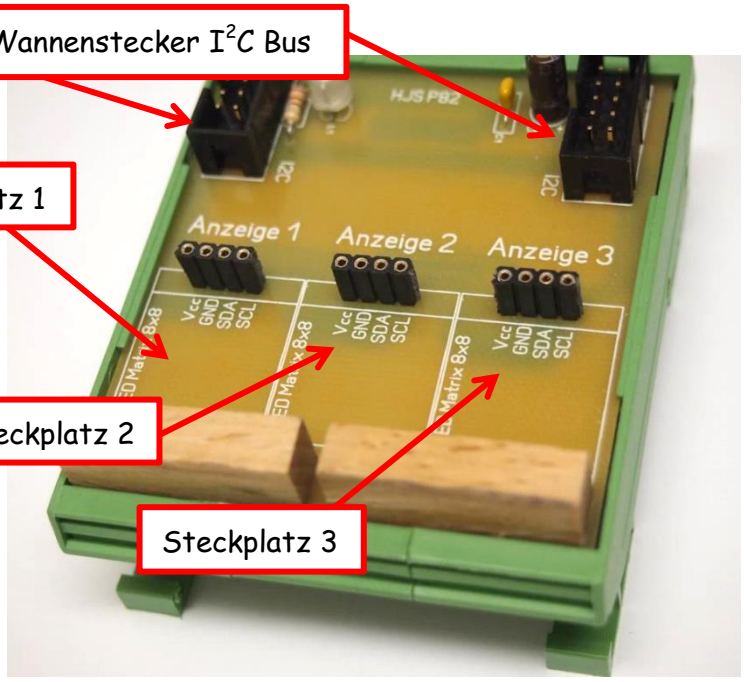
Bestimmungsgemäße Verwendung

- 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.

Anzeige 2 – Teil 2 (Software)

Anzeige 2 mit dem HT16K33 (I^2C), 3 x LED Matrix Anzeigen (8 x 8 mit 19mm) und 2 x I^2C – Bus

Im diesem Teil wollen wir uns ansehen, wie diese Matrix Module angesteuert bzw. programmiert werden. Eigentlich sind sie genauso wie die anderen Boards aufgebaut. Dabei wird auch ein HT16K33 und eine LED Matrix 8x8 verwendet. Der Unterschied besteht einmal im HT16K33, dieser IC hat „nur" 24 Pins und einer LED Matrix mit ca. 19x19mm. Daraus ergeben sich auch Änderungen in der Programmierung.



Ansicht Modul Anzeige 2
ohne HT16K33 Module

Die Holzstücken dienen nur zur Auflage der Matrix Module

Auf unserem Modul befinden sich 3 Steckplätze für Matrix Module. An diesen 3 Steckplätzen liegt der I^2C-Bus parallel an.
Die Zuordnung der Busadressen erfolgt auf der Rückseite der Matrix-Module. Das habe ich bereits im ersten Teil beschrieben.

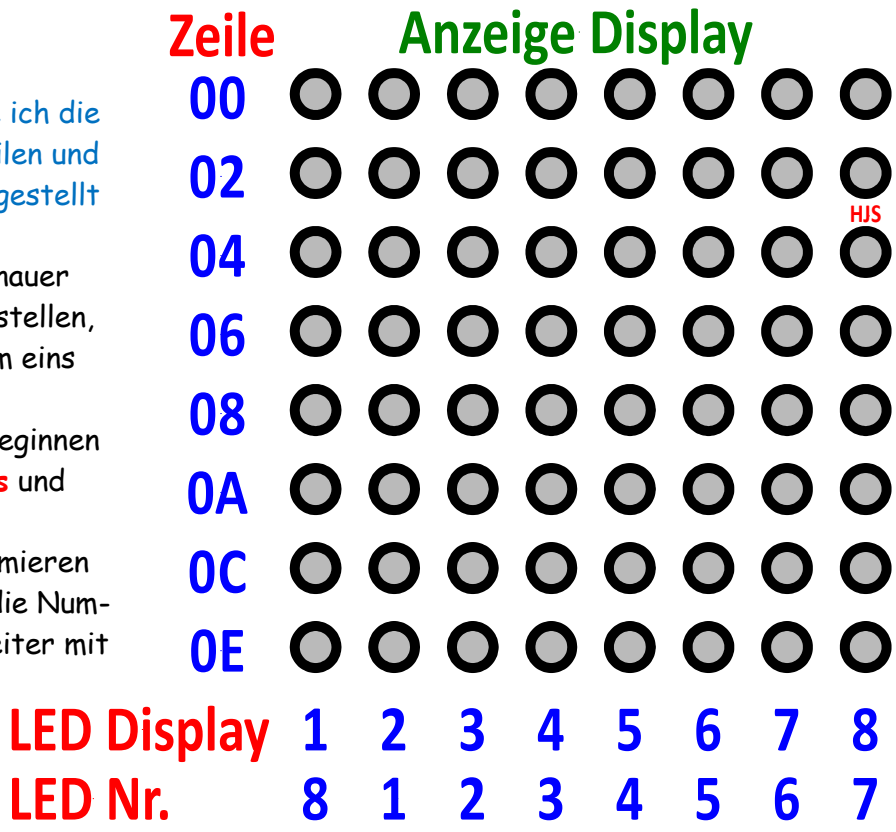
Auch in diesem Fall macht der HT16K33 nur das, was wir ihm sagen. Das geht z.B. über einen Prozessor. Ich verwende mein Board 1 dazu mit dem AT1284p und verbinde die Platinen mit dem I^2C Bus.
Über den I^2C Bus kann jedes Matrix-Modul und jede LED einzeln angesteuert werden.

Zum Betrieb sind die folgenden Dateien notwendig:

- **ATB_Anz2_Prg_1.c** (das eigentlich Programm)
- **HT16K33.c** (alle Unterprogramme)
- **HT16K33.h** (erster Aufruf Unterprogramme)
- **i2cmaster.h** (Programm für den Bus)
- **twimaster.c** (Programm für den Bus)
- **main.h** (allgemeine Angaben)

Die notwendigen Programme habe ich als Datei mit angehängt. Ich arbeite mit dem AVR Studio 6 (6.2) und programmiere in C. Die verwendeten Programme müssen dem Programm mitgeteilt werden. Wie man das macht habe ich bereits in einem anderem Tut beschrieben. Bitte die Dateien nicht mit anderen verwechseln. Die Programme sind jeweils den Anzeigen angepasst. Bitte nicht verwechseln.

In diesem Bild habe ich die Anordnung der Zeilen und der LED dargestellt



Wenn man sich das Bild genauer betrachtet, kann man feststellen, das die LED nach rechts um eins verschoben sind.
Die LEDs auf der Matrix beginnen an der linken Seite mit **eins** und gehen bis zur **achten** LED.
Wenn ich die LED programmieren will, so hat die erste LED die Nummer **acht** und geht dann weiter mit **eins** bis zur **sieben.**
Das muss bei der Programmierung unbedingt beachtet werden

Wichtiges:

- Die ganze Anzeige besteht aus 3 x Matrix mit je 8 x 8 LED, es sind bis zu 4 Matrix Anzeigen möglich – Begrenzt durch die Adressen
- Die Anzeigen werden über den I^2C Bus gesteuert
- Jede Anzeige hat einen HT16K33 und nur eine LED Matrix mit einer Adresse
- Jede Anzeige beginnt bei der Zeilenzahl 0 und verwendet jede zweite Zeile, also die Zeilen 00, 02, 04, 06, 08, 0A, 0C und 0E
- Die LED werden Zeilenweise durch 0x00 (keine) bzw. 0xFF (alle) und alle Werte dazwischen angegeben
- Die LED sind um eins nach rechts verschoben (siehe Bild oben)
- Es müssen zu Anfang bestimmte Parameter z.B. interner Osz. eingestellt werden
- Jeder HT16K33 muss seine eigene Adresse haben
- Beachte den Adressrahmen bei 20 Pins, 24 Pins und 28 Pins
- Es sind keine Vorwiderstände notwendig
- Betriebsspannung + 5V
- Es können bis zu 256 LED, jede einzeln angesteuert werden
- In Abhängigkeit der Anzahl der LED die mögliche Belastung des Netzteiles und der Leiterzüge beachten
- Werte sind gespeichert bis sie wieder überschrieben werden
- Zum Beginn des Programmes deshalb alle LED ausschalten

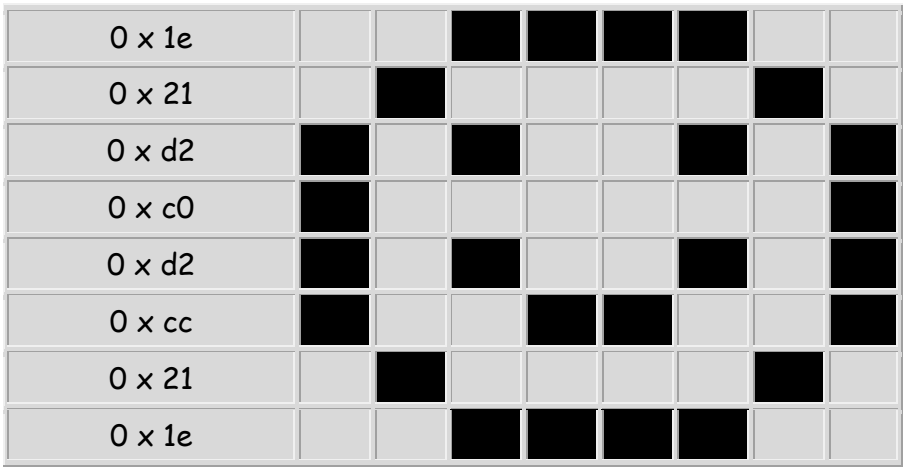
Die Belegung der LED und Zuordnung der Nummern

Bitte unbedingt beachten:

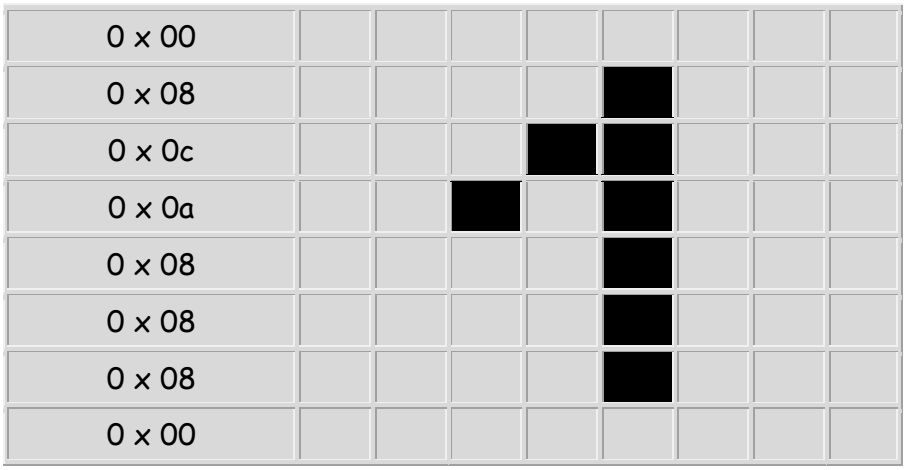
Die LED sind um eins nach rechts verschoben. Damit ergeben sich andere Werte bei der Zuordnung der LED

Byte hexadezimal	Anzeige auf der Matrix							
LED Display	1	2	3	4	5	6	7	8
LED Nr	8	1	2	3	4	5	6	7
0 x 00								
0 x 01		■						
0 x 02			■					
0 x 03		■	■					
0 x 04				■				
0 x 05		■		■				
0 x 06			■	■				
0 x 07		■	■	■				
0 x 08					■			
0 x 09		■			■			
0 x 0A			■		■			
0 x 0B		■	■		■			
0 x 0C				■	■			
0 x 0D		■		■	■			
0 x 0E			■	■	■			
0 x 0F		■	■	■	■			
0 x 10						■		
0 x 20							■	
0 x 30						■	■	
0 x 40								■
0 x 50						■		■
0 x 60							■	■
0 x 70						■	■	■
0 x 80	■							
0 x 90	■					■		
0 x A0	■						■	
0 x B0	■					■	■	
0 x C0	■							■
0 x D0	■					■		■
0 x E0	■						■	■
0 x F0	■					■	■	■

Beispiel Smiley



Beispiel Zahl 1



Ein paar Beispiele für Smiley und verschiedene Zahlen und Buchstaben:

```
int bild1 [8] = {0x1e, 0x21, 0xc0, 0xc0, 0xc0, 0xc0, 0x21, 0x1e};      // smiley 1
int bild2 [8] = {0x1e, 0x21, 0xd2, 0xc0, 0xc0, 0xc0, 0x21, 0x1e};      // smiley 2
int bild3 [8] = {0x1e, 0x21, 0xd2, 0xc0, 0xd2, 0xcc, 0x21, 0x1e};      // smiley 3

int zahl1 [8] = {0x00, 0x08, 0x0c, 0x0a, 0x08, 0x08, 0x08, 0x00};
int zahl2 [8] = {0x00, 0x0c, 0x12, 0x10, 0x08, 0x04, 0x1e, 0x00};
int zahl3 [8] = {0x00, 0x0c, 0x10, 0x08, 0x10, 0x12, 0x0c, 0x00};
int zahl4 [8] = {0x00, 0x08, 0x0a, 0x0a, 0x1e, 0x08, 0x08, 0x00};
int zahl5 [8] = {0x00, 0x1e, 0x02, 0x0e, 0x10, 0x12, 0x0c, 0x00};
int zahl6 [8] = {0x00, 0x1c, 0x02, 0x02, 0x0e, 0x12, 0x0c, 0x00};
int zahl7 [8] = {0x00, 0x1e, 0x10, 0x10, 0x10, 0x08, 0x04, 0x00};
int zahl8 [8] = {0x00, 0x0c, 0x12, 0x0c, 0x12, 0x12, 0x0c, 0x00};
int zahl9 [8] = {0x00, 0x0c, 0x12, 0x1c, 0x10, 0x10, 0x0c, 0x00};
int zahl0 [8] = {0x00, 0x0c, 0x12, 0x12, 0x12, 0x12, 0x0c, 0x00};
int zahlA [8] = {0x00, 0x0c, 0x12, 0x12, 0x1e, 0x12, 0x12, 0x00};
int zahlB [8] = {0x00, 0x1e, 0x10, 0x08, 0x04, 0x02, 0x1e, 0x00};
```

Am besten selber was erstellen und testen

Als nächstes sehen wir uns ein Programm an. Ich habe dem Tut mehrere Dateien mit Programmbeispielen angehängt.

Das erste Programm:

```
/* ATB_Anz2_Prg_1.c Created: 09.05.2015 14:53:52 Author: AS */

#define F_CPU 16000000                  // CPU clock in Hz
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include "HT16K33.h"
#include "i2cmaster.h"                  // Aufruf twimaster
#include "util/delay.h"

int main (void)
 {
   i2c_init();
   // Address 0x70 ..0x73)
   HT16K33_modul1_addr = 0x70;
   HT16K33_modul2_addr = 0x71;
   HT16K33_modul3_addr = 0x72;
   HT16K33_Init1();
   HT16K33_Init2();
   HT16K33_Init3();
   while(1)
    {
      HT16K33_ClearDisplay1(0);         // löscht Matrix 1
      _delay_ms(500);                   // Pause
      HT16K33_ClearDisplay2(0);         // löscht Matrix 2
      _delay_ms(500);                   // Pause
      HT16K33_ClearDisplay3(0);         // löscht Matrix 3
      _delay_ms(500);                   // Pause

      HT16K33_WriteDisplay1(0);         // füllt Matrix
      _delay_ms(2000);                  // Pause
      HT16K33_WriteDisplay2(0);         // füllt Matrix
      _delay_ms(2000);                  // Pause
      HT16K33_WriteDisplay3(0);         // füllt Matrix
      _delay_ms(2000);                  // Pause

      HT16K33_FillDisplay1(0);          // füllt Matrix
      _delay_ms(2000);                  // Pause
      HT16K33_FillDisplay2(0);          // füllt Matrix
      _delay_ms(2000);                  // Pause
      HT16K33_FillDisplay3(0);          // füllt Matrix
      _delay_ms(2000);                  // Pause
    }
   return 0;                            // by AS
 }
```

Sehen wir uns das Programm im Einzelnen an:

```
#define F_CPU 16000000                  // CPU clock in Hz
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include "HT16K33.h"
#include "i2cmaster.h"                  // Aufruf twimaster
#include "util/delay.h"
```

Damit werden die notwendigen Dateien und Programme geladen

```
int main (void)
 {
   i2c_init();
```

Start des Hauptprogrammes und Initialisierung I^2C Bus

```
   HT16K33_modul1_addr = 0x70;
   HT16K33_modul2_addr = 0x71;
   HT16K33_modul3_addr = 0x72;
```

Es werden die notwendigen Busadressen definiert

```
   HT16K33_Init1();
   HT16K33_Init2();
   HT16K33_Init3();
```

Es müssen 3 x HT16K33 initiiert werden

```
      HT16K33_ClearDisplay1(0);         // löscht Matrix 1
      _delay_ms(500);                   // Pause
      HT16K33_ClearDisplay2(0);         // löscht Matrix 2
      _delay_ms(500);                   // Pause
      HT16K33_ClearDisplay3(0);         // löscht Matrix 3
      _delay_ms(500);                   // Pause
```

Es werden alle drei HT16K33 gelöscht (mit 0x00 überschrieben), danach jeweils 500ms Pause

```
      HT16K33_WriteDisplay1(0);         // füllt Matrix
      _delay_ms(2000);                  // Pause
      HT16K33_WriteDisplay2(0);         // füllt Matrix
      _delay_ms(2000);                  // Pause
      HT16K33_WriteDisplay3(0);         // füllt Matrix
      _delay_ms(2000);                  // Pause
```

Jedes HT16K33 Modul wird mit einem anderen Wert beschrieben, danach jeweils 2s Pause

Mit der Angabe **0** innerhalb des Aufrufes HT16K33_WriteDisplay3(**0**); wird die Berechnung der Zeilen mit **0** begonnen. Dadurch wird als erstes die Zeile **0** (links) beschrieben.

```
HT16K33_FillDisplay1(0);          // füllt Matrix
_delay_ms(2000);                  // Pause
HT16K33_FillDisplay2(0);          // füllt Matrix
_delay_ms(2000);                  // Pause
HT16K33_FillDisplay3(0);          // füllt Matrix
_delay_ms(2000);                  // Pause
```

Jedes HT16K33 Modul wird mit dem gleichen Wert beschrieben, danach jeweils 2s Pause

Innerhalb unseres Hauptprogrammes werden verschiedene Unterprogramme aufgerufen. Das eigentliche beschreiben unserer HT16K33 Module erfolgt in diesen Unterprogrammen. Sehen wir uns mal einige Teile dazu an:

```
//////////////////////  Init 1
void HT16K33_Init1(void)                    // HT16K33 initieiren
 {
  HT16K33_SetOscillator1(true);             // schaltet Oszillator ein
  HT16K33_Blinken1(HT16K33_BLINK_OFF);      // SETZ 0-AUS ODER 1-EIN
  HT16K33_SetBrightness1(3);                // setzt Helligkeit max 15 (0..15)
 }
void HT16K33_SetOscillator1(bool osc_on)    // schaltet Oszi ein
 {
  i2c_start(HT16K33_modul1_addr<<1);
  if (osc_on)
  i2c_write(HT16K33_CMD_OSCILLATOR | 1);    // oscillator ein
  else
  i2c_write(HT16K33_CMD_OSCILLATOR);        // oscillator aus
  i2c_stop();                               // Bus stop
 }
void HT16K33_Blinken1(uint8_t b)            // schaltet Blinken
 {
  i2c_start(HT16K33_modul1_addr<<1);
  if (b > 3) b = 0;                         // Sicherheitsabfrage
  i2c_write(HT16K33_CMD_BLINK | HT16K33_BLINK_DISPLAYON | (b << 1));
  i2c_stop();                               // Bus stop
 }
void HT16K33_SetBrightness1(uint8_t b)      // schaltet Helligkeit
 {
  if (b > 15) b = 15;                       // Sicherheitsabfrage
  i2c_start(HT16K33_modul1_addr<<1);        // Bus start, Angabe Adresse
  i2c_write(HT16K33_CMD_BRIGHTNESS | b);    // schreibt Wert
  i2c_stop();                               // Bus stop
 }
```

Es wird void HT16K33_Init1(void) vom Hauptprogramm aufgerufen. Innerhalb dieses Programmes werden

```
HT16K33_SetOscillator1(true);             // schaltet Oszillator ein
HT16K33_Blinken1(HT16K33_BLINK_OFF);      // SETZ 0-AUS ODER 1-EIN
HT16K33_SetBrightness1(3);                // setzt Helligkeit max 15 (0..15)
```

aufgerufen und entsprechende Einstellungen vorgenommen.
Für das Modul 2 und 3 werden gleiche Einstellungen vorgenommen.

```
void HT16K33_ClearDisplay1(uint8_t start)      // löscht komplette Matrix
 {
  for (uint8_t i=start; i<16; i+=2)            // Auswahl der Zeile jede 2.
   {
    i2c_start(HT16K33_modul1_addr<<1);         // schreibt modul 1
    i2c_write(i);                              // angabe Zeile
    i2c_write(0x00);                           // angabe LED - alle aus
    i2c_stop();                                // Bus stop
   }
 }
```

Mit diesem Unterprogramm wird der Inhalt der kompletten Matrix gelöscht bzw. mit 0 überschrieben. Die Funktion jeder Zeile habe ich kommentiert. Für das Modul 2 und 3 werden gleiche Einstellungen vorgenommen.

```
void HT16K33_FillDisplay1(uint8_t start)       // füllt komplette Matrix
 {
  for (uint8_t i=start; i<16; i+=2)            // Auswahl der Zeile jede 2.
   {
    i2c_start(HT16K33_modul1_addr<<1);         // schreibt modul 1
    i2c_write(i);                              // angabe Zeile
    i2c_write(0xff);                           // angabe LED - alle ein
    i2c_stop();                                // Bus stop
   }
 }
```

Im Grunde mach ich mit diesem Unterprogramm wieder das gleiche. Der Inhalt der kompletten Matrix wird mit dem gleichen Wert überschrieben. Die Funktion jeder Zeile habe ich kommentiert. Für das Modul 2 und 3 werden gleiche Einstellungen vorgenommen.

Mit dem Programm Anz2_Prg_1 müssen die drei HT16K33 Module nacheinander ein LED Muster, alle LED uns alles aus darstellen. Dabei werden die Bus-Adressen 0x70, 0x71 und 0x72 verwendet.
Zum besseren Verständnis des HT16K33 bitte auch die anderen Tut`s zu diesem Thema lesen.

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