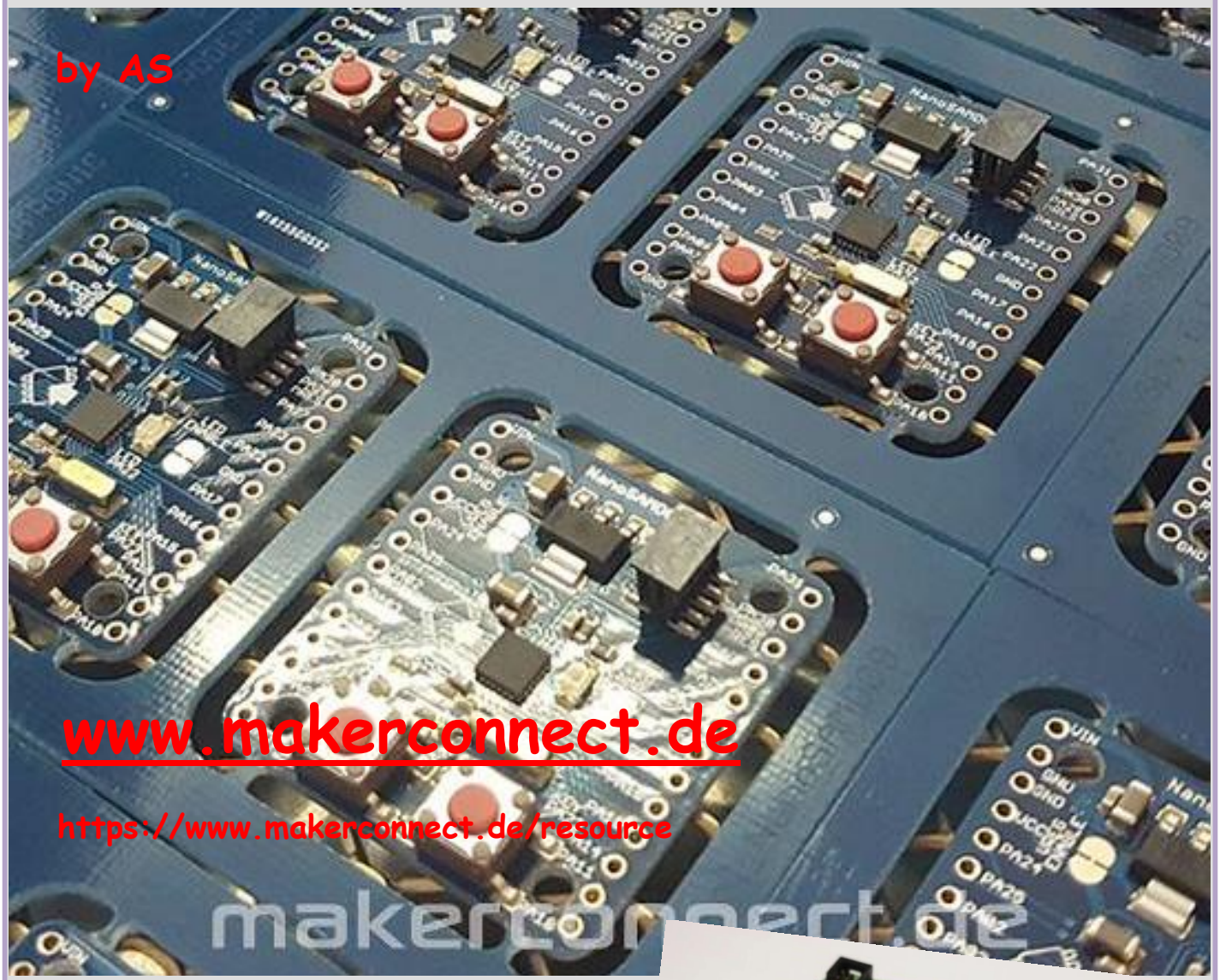
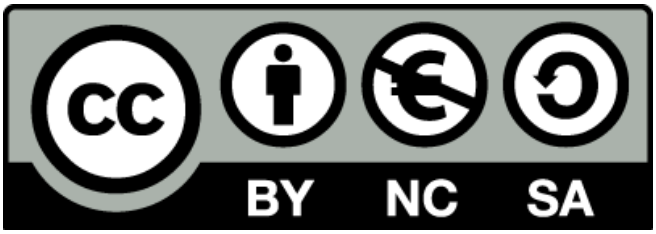
Mikrokontroller & I²C Bus



Anzeige 1 mit dem HT16K33, 2 x LED Matrix Anzeigen (8x8), 2 x I²C – Bus
Teil 2 – Software

Anzeige 1 - 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

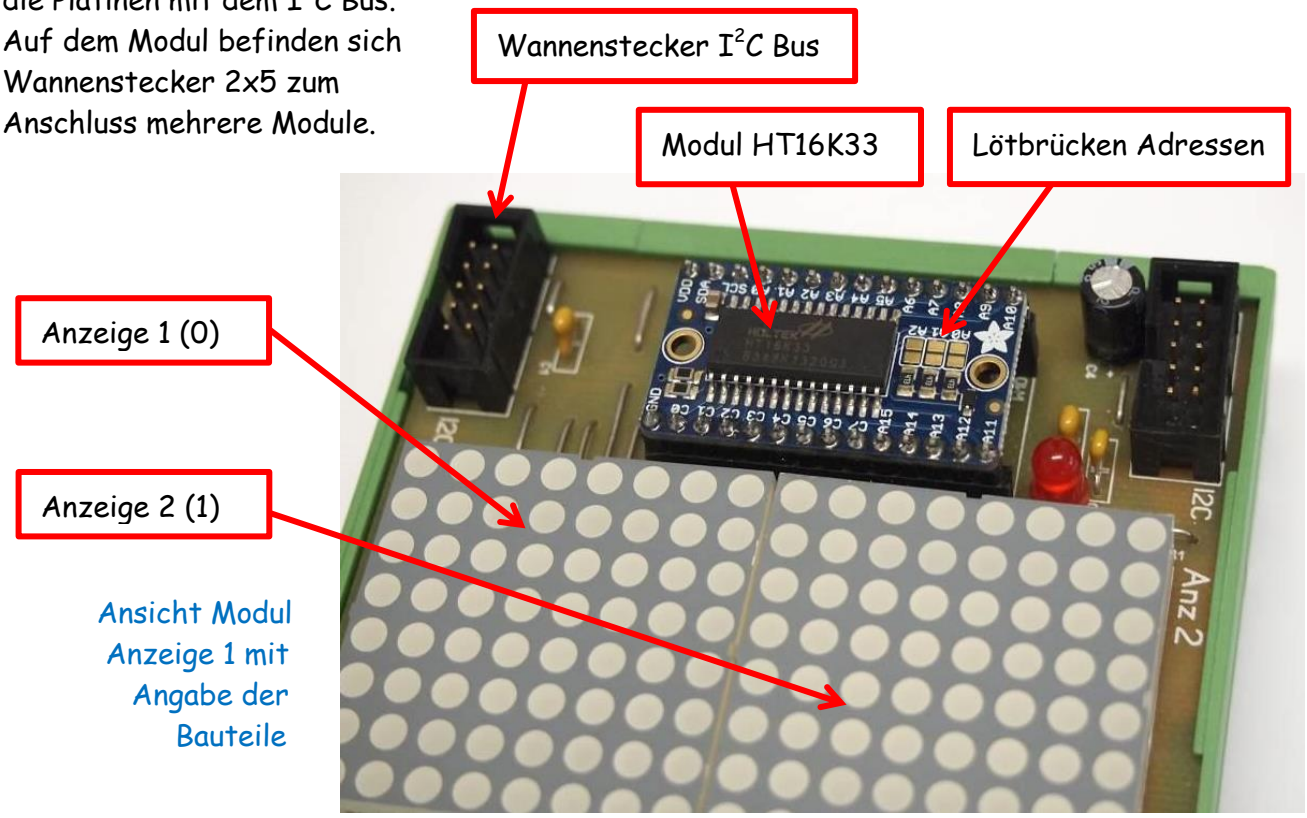
Bestimmungsgemäße Verwendung

- 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 1 – Teil 2 (Software)

Anzeige 1 mit dem HT16K33 (I^2C), 2x LED Matrix Anzeigen (8 x 8 mit 38mm) und 2 x I^2C – Bus

Im ersten Teil dieser Folge habe ich den Aufbau (Hardware) beschrieben. Nur allein mit dem Modul können wir nichts anfangen. Der HT16K33 macht 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.
Auf dem Modul befinden sich Wannenstecker 2x5 zum Anschluss mehrere Module.



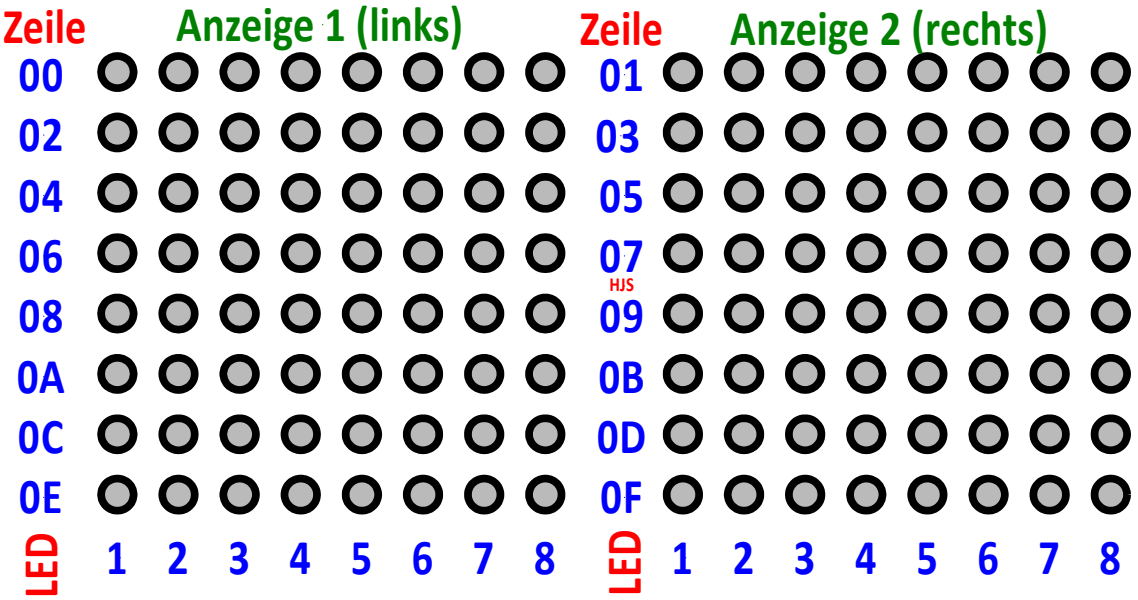
Ansicht Modul Anzeige 1 mit Angabe der Bauteile

Im ersten Teil habe ich die Auswahl der Adressen, die Schaltung und den Anschluss der LED an den HT16K33 bereits beschrieben.

Zum Betrieb sind die folgenden Dateien notwendig:

- **ATB_Anz1_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 Tut beschrieben.



Im oberen Bild habe ich noch mal die Anordnung der Zeile und LED dargestellt.
Dabei ist zu beachten:

- Die ganze Anzeige besteht aus 2 x Matrix mit 8 x 8 LED, bis zu 16 Matrix Anzeigen möglich
- Die Anzeige wird aber nur über einen IC HT16K33 gesteuert (mit 2 x Matrix Anzeige)
- Die linke Anzeige bezeichne ich als Anz 1, hat aber im Programm die 0
- Die rechte Anzeige bezeichne ich als Anz 2, hat aber im Programm die 1
- Die linke Anzeige beginnt bei der Zeilenzahl 0 und verwendet jede zweite Zeile, also die Zeilen 00, 02, 04, 06, 08, 0A, 0C und 0E
- Die rechte Anzeige beginnt bei der Zeilenzahl 1 und verwendet jede zweite Zeile, also die Zeile 01, 03, 05, 07, 09, 0B, 0D und 0F
- Die LED werden Zeilenweise durch 0x00 (keine) bzw. 0xFF (alle) und alle Werte dazwischen angegeben
- Die Angabe ist dabei vertauscht (oberes bzw. unteres Bit)
- Die 4 linken LED werden durch die rechte Zahl angegeben (siehe Tabelle)
- Die 4 rechten LED werden durch die linke Zahl angegeben (siehe Tabelle)
- 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 1024 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

Anordnung der LED:

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

Beispiel:

0 x 00								
0 x 99	■			■	■			■
0 x AA		■		■		■		■
0 x 55	■		■		■		■	
0 x C3	■	■					■	■
0 x 3C			■	■	■	■		
0 x 66		■	■			■	■	
0 x FF	■	■	■	■	■	■	■	■

Bitte beachten:

Die linke Zahl gibt die Anzahl der 4 rechten LED an
Die rechte Zahl gibt die Anzahl der 4 linken LD an

Als nächste sehen wir uns das Hauptprogramm an:

```
/* ATB_Anz1_Prg_1.c  Created: 29.04.2015 19:48:15 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"
#include "util/delay.h"
int main (void)
 {
   i2c_init();
   HT16K33_modul1_addr = 0x70;              // Address 0x70 ..0x77)
   HT16K33_Init();
   while(1)
    {             // 0 - linke Matrix, 1 - rechte Matrix
      HT16K33_WriteBild1(0);                // schreibt Bild1 auf Matrix 0
      _delay_ms(2000);                      // Pause
      HT16K33_ClearDisplay(0);              // löscht Matrix 0
      _delay_ms(2000);                      // Pause
      HT16K33_WriteBild1Zeit(1);            // Schreibt Bild1 mit Zeit auf Matrix 1
      _delay_ms(2000);                      // Pause
      HT16K33_ClearDisplay(1);              // löscht Matrix 1
      _delay_ms(2000);                      // Pause
      HT16K33_WriteBild2(0);                // schreibt Bild2 auf Matrix 0
      _delay_ms(2000);                      // Pause
      HT16K33_WriteBild2(1);                // schreibt Bild1 auf Matrix 1
      _delay_ms(2000);                      // Pause
      HT16K33_ClearDisplay(1);              // löscht Matrix 1
      _delay_ms(2000);                      // Pause
      HT16K33_FillDisplay(0);               // füllt Matrix 0 einheitlich
      _delay_ms(2000);                      // Pause
      HT16K33_FillDisplay(1);               // füllt Matrix 1 einheitlich
      _delay_ms(2000);                      // Pause
      HT16K33_WriteDisplay(0);              // Angabe schreiben mit adresse
      _delay_ms(2000);                      // Pause
      HT16K33_ClearDisplay(0);              // löscht Matrix 0-links, 1-rechts
      _delay_ms(2000);                      // Pause
      HT16K33_WriteDisplay(1);              // Angabe schreiben mit adresse
      _delay_ms(2000);                      // Pause
      HT16K33_FillDisplay(0);               // füllt Matrix 0 einheitlich
      _delay_ms(2000);                      // Pause
      HT16K33_FillDisplay(1);               // füllt Matrix 1 einheitlich
      _delay_ms(2000);                      // Pause
      HT16K33_ClearDisplay(0);              // löscht Matrix 0
      HT16K33_ClearDisplay(1);              // löscht Matrix 1
      _delay_ms(2000);                      // Pause
    }
   return 0;
 }
```

Sehen wir uns das Programm in kleinen Stücken 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"
#include "util/delay.h"
```

Damit gebe ich an, welche Dateien geladen werden sollen.

```
int main (void)
 {
   i2c_init();
   HT16K33_modul1_addr = 0x70;          // Address 0x70 ..0x77)
   HT16K33_Init();
   while(1)
```

Beginn des Hauptprogrammes, initiiere den I^2C Bus, weise dem HTK33 die Adresse 0x70 zu und initiiere den HT16K33

```
      // 0 - linke Matrix, 1 - rechte Matrix
      HT16K33_WriteBild1(0);            // schreibt Bild1 auf Matrix 0
      HT16K33_WriteBild1Zeit(1);        // Schreibt Bild1 mit Zeit auf Matrix 1
      HT16K33_WriteBild2(0);            // schreibt Bild2 auf Matrix 0
      HT16K33_WriteBild1(1);            // schreibt Bild1 auf Matrix 1
      HT16K33_FillDisplay(0);           // füllt Matrix 0 einheitlich
      HT16K33_FillDisplay(1);           // füllt Matrix 1 einheitlich
      HT16K33_WriteDisplay(0);          // Angabe schreiben
      HT16K33_WriteDisplay(1);          // Angabe schreiben
      HT16K33_ClearDisplay(0);          // löscht Matrix 0
      HT16K33_ClearDisplay(1);          // löscht Matrix 1
```

Vom Hauptprogramm erfolgt der Aufruf der gewünschten Unterprogramme. Dabei erfolgt eine Übergabe der Zeilenzahl, mit der begonnen werden soll. Wird eine 0 übergeben, wird auf der linken Matrix die LED geschaltet. Wird eine 1 übergeben, werden auf der rechten Matrix LED geschaltet.

Sehen wir uns dazu einige Teile der Unterprogramme an:

```
void HT16K33_Init(void)                 // HT16K33 initieiren
 {
   HT16K33_SetOscillator(true);         // schaltet Oszillator ein
   HT16K33_Blinken(HT16K33_BLINK_OFF);  // SETZ 0-AUS ODER 1-EIN
   HT16K33_SetBrightness(15);           // setzt Helligkeit max 15 (0..15)
 }
```

Beim initiieren des HT16K33 werden entsprechende Parameter des HT16K33 gesetzt. Auch hier erfolgt ein Aufruf von Unterprogrammen. In diesem Programm werden nur die Daten für einen HT16K33 übergeben. Es wird der interne Osz. eingeschaltet, Blinken ausgeschaltet und maximale Helligkeit eingeschaltet.

```
void HT16K33_ClearDisplay(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 Adresse modul 1
    i2c_write(i);                            // angabe Zeile
    i2c_write(0x00);                         // angabe LED - alle aus
   }
 }
```

Mit diesem Unterprogramm wird der Inhalt der Matrix gelöscht, in dem alle LED ausgeschaltet werden. Dabei wird der HT16K33 mit der Adresse 0x70 aufgerufen und jede zweite Zeile geschaltet. Die Auswahl der Matrix erfolgt durch die Übergabe von **start**.

```
void HT16K33_FillDisplay(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
   }
 }
```

Wir können auch alle LED mit dem gleichen Inhalt schalten. Dazu verwenden wir fast das gleiche Programm wie vorher. Es wird nur ein anderer Inhalt übergeben.

```
void HT16K33_WriteBild1(uint8_t start)      // schreibt bild1
 {
  int8_t nr = 0;                             // array nr auf 0
  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(bild1[nr]);                    // anzeige array
    nr = nr +1;                              // array nr um 1 erhöht
   }
 }
```

Mit diesem Code können wir ein Bild auf der Matrix zeichnen. Ein „Bild" ist zwar übertrieben, aber alles was sich mit einer Matrix von 8x8 LED darstellen lässt.

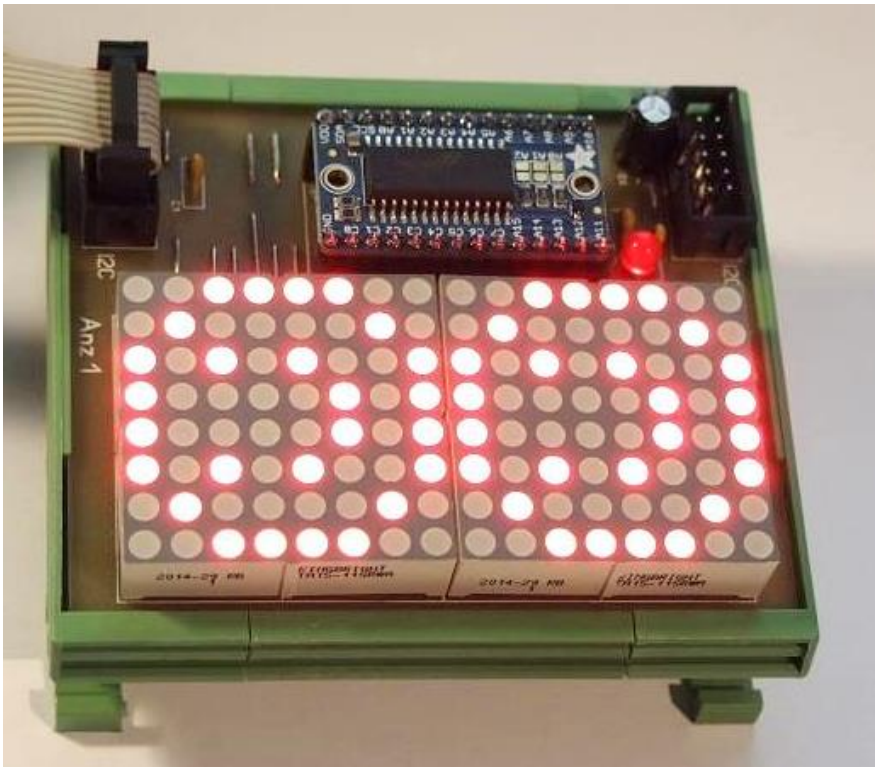
```
int bild1 [8] = {0x3c, 0x42, 0xa5, 0x81, 0xa5, 0x99, 0x42, 0x3c};    // Bild 1
```

Dazu habe ich ein Array mit Namen bild1 erstellt und mit 8 Werten gefüllt. Zum schreiben verwende ich wieder das gleiche Programm. Zusätzlich wird die Variable **nr** auf 0 gesetzt und bei jedem Durchlauf um 1 erhöht. Der Inhalt des Arrays **bild1** wird dadurch Wert für Wert eingelesen und dargestellt.
Bei der Erstellung des Arrays ist wieder die Anordnung der LED auf der Matrix zu berücksichtigen.

```
void HT16K33_WriteBild1Zeit(uint8_t start)// schreibt bild1
 {
   int8_t nr = 0;                          // array nr auf 0
   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(bild1[nr]);                // anzeige array
      nr = nr +1;                          // array nr um 1 erhöht
      _delay_ms(200);                      // Zeitverzögerung
    }
 }
```

Es ist eigentlich wieder das gleich wie vorher. Zusätzlich wird eine Verzögerung von 200ms bei jedem Durchlauf ausgeführt. Das führt zu einem verlangsamten Aufbau des „Bildes"

Wenn man alles richtig gemacht, könnte es z.B. so aussehen.



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