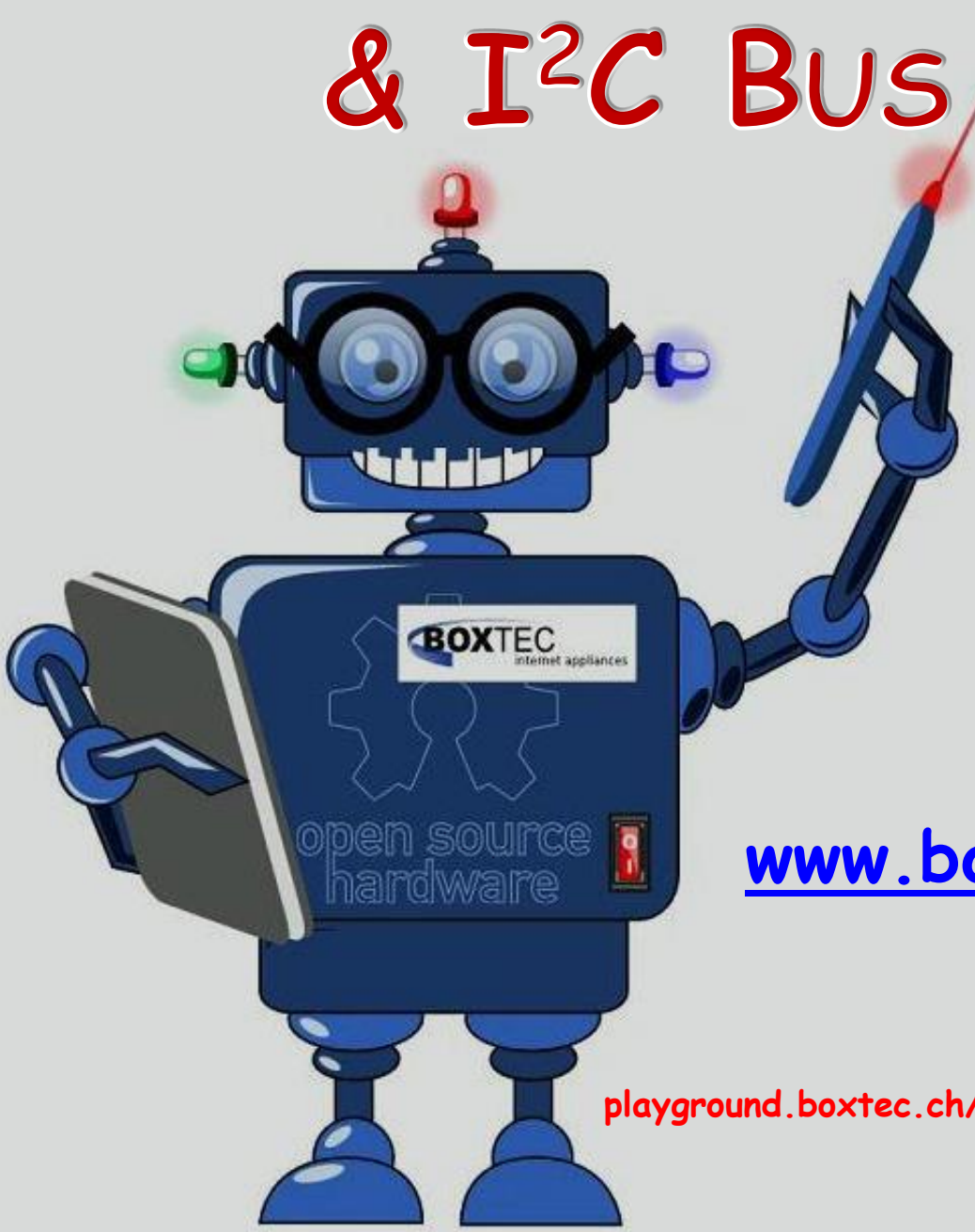
Mikrokontroller & I²C Bus



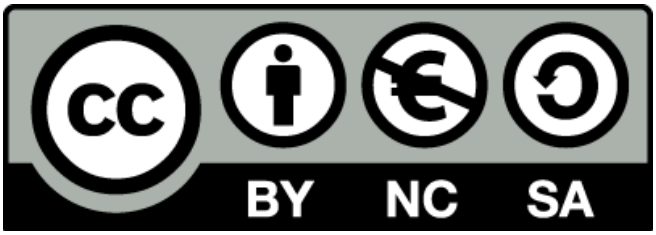
www.boxtec.ch

playground.boxtec.ch/doku.php/tutorial

I²C Bus und analoge Eingabe
= Teil 2 – Software (in) =

Analog 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 Nichtbe-achtung 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:

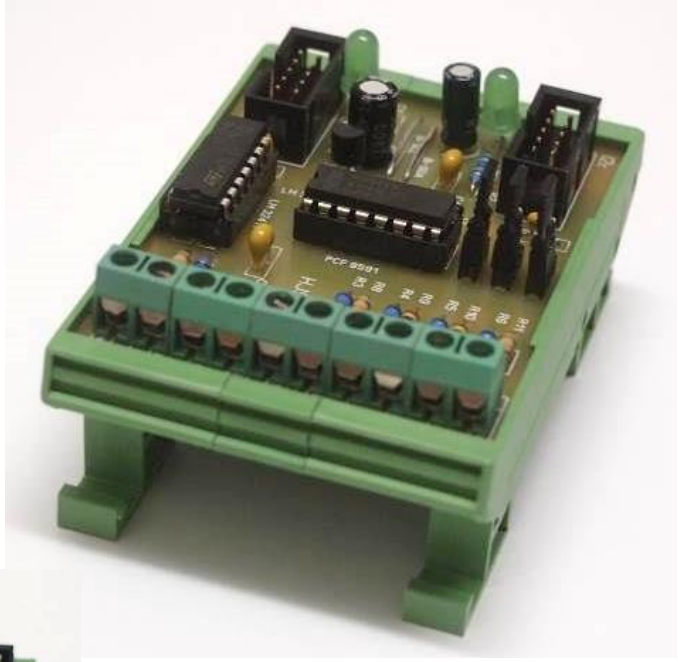
- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige Person anschließen.
- In Schulen, Ausbildungseinrichtungen, Hobby- und Selbsthilfewerkstätten ist das Betreiben die-ser 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 Versor-gungsspannung 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 Spannungsver-sorgung 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 un-verzüglich 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 Baugrup-pe 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 Ge-fahren, 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.

I^2C Bus und analoge Eingabe
= Teil 2 – Software (in)=

Im zweiten Teil von Analog 2 wollen wir uns die Messung von 0 bis 10 V ansehen. Als Grundlage verwenden wir wieder diese Platine.

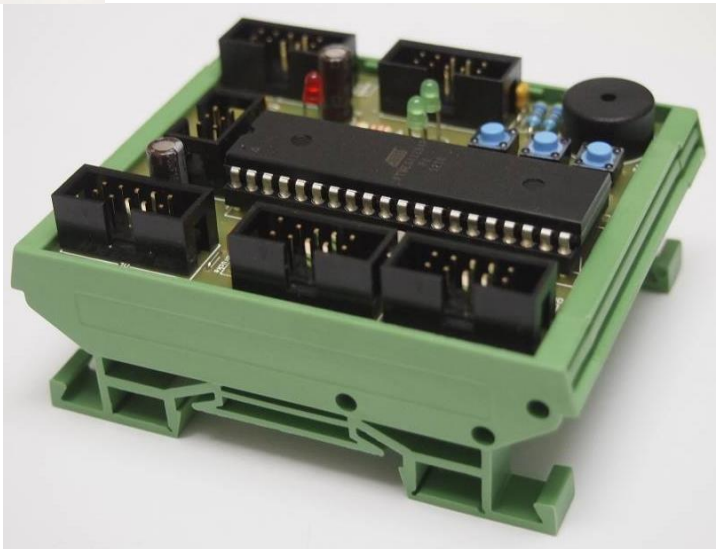


Platine mit dem PCF 8591 und Zusätzlichen OPV, Spannungsteiler und Schraubklemmen

Als Anzeige nutze ich ein Graphikdisplay, das über den ATmega 1284p angesteuert wird.
Zum Betrieb sind zusätzliche Dateien notwendig.

Graphikmodul 1 (P 35) zum Anschluss an den ATmega1284p

Spannungsnormal 10,00V

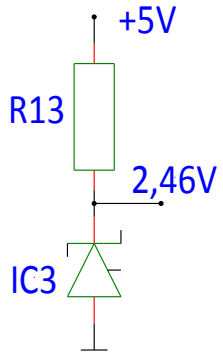


Prozessorplatine Board 1 mit dem ATmega 1284p

Zur Spannungsversorgung nutze ich wieder mein NT2. Die Verbindung der Module erfolgt über den I^2C Bus bzw. über die Verbindung Board1 und Graphikmodul.
Bei diesem Aufbau verwende ich zusätzlich die Platine P25. Dabei handelt es sich um eine sehr genaue +10V Spannungsquelle. Durch den Anschluss mehrerer Potis kann ich die Eingangsspannung zwischen 0 und +10V verändern. Die Schaltung und Funktion habe ich bereits im ersten Teil erläutert.

Sehen wir uns einige Berechnungen an:

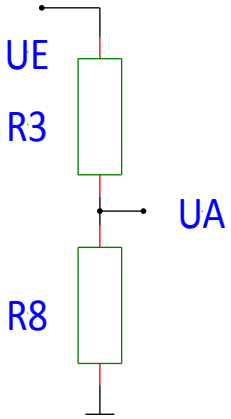
Referenzspannung



R13 – Widerstand 2,49 kOhm
IC3 – LM 336 – Z 2,5

Die Referenzspannung von 2,46V für den PCF 8591 wird aus +5V gewonnen. Dazu verwenden ich den IC LM 336-Z2,5 und einem Vorwiderstand von 2,49 kOhm.
Die Daten/Schaltung habe ich dem Datenblatt des Herstellers entnommen.

Eingangspannungsteiler



R3 – Widerstand 31,6 kOhm
R8 – Widerstand 10 kOhm
U_E = max. 10V

Die maximale Eingangsspannung kann +10V betragen. Da unsere Referenzspannung 2,46V beträgt, kann kein höherer Wert gelesen werden. Durch den Spannungsteiler wird die Eingangsspannung von max. +10V auf max. 2,40V geteilt.

$$U_A = U_E \times \frac{R_8}{R_8 + R_3} = 10\,V \times \frac{10\,k\Omega}{10\,k\Omega + 31{,}6\,k\Omega} = 2{,}404\,V \approx 2{,}40\,V$$

Die Eingangsspannung wird durch den Spannungsteiler mit 4,16 geteilt

$$T = \frac{U_E}{U_A} = \frac{10\,V}{2{,}404\,V} = 4{,}16$$

Da die Software an vielen Stellen mit dem Programm aus Analog 1 - Software identisch ist, möchte ich nur das wichtigste und erfolgten Änderungen erläutern.

Sehen wir uns als nächste das Programm in kleinen Schritten an:

```
#include "ks0108.h"                 // Bibliothek für den verwendeten Graphik IC
#include "arial_bold_14.h"          // Zeichensätze für die Schriften
#include "corsiva_12.h"
#include "arial8.h"
#include "small_font.h"
```

Zum Betrieb des Graphikmoduls müssen diese Dateien/Bibliotheken zuzüglich den bereits vorhandenen geladen werden.

```
#define PCF8591w   0x90                 // Adressen PCF 8591
#define PCF8591r   0x91
```

Angabe der Bus Adressen für den PCF 8591

```
void startanzeige()
 {
   ks0108ClearScreen();                                      // Display löschen
   ks0108DrawRoundRect(1, 2, 125, 61, 8, BLACK);             // Ausgabe Rand
   ks0108DrawLine(110, 32, 67, 60, BLACK);                   //Ausgabe Linie
   ks0108DrawRect(12, 47, 10, 10, BLACK);                    //Ausgabe Viereck leer
   ks0108FillRect(40, 50, 10, 10, BLACK);                    //Ausgabe Viereck gefüllt
   ks0108Circle(110, 50, 10, BLACK);                         // Ausgabe Kreis
   ks0108SelectFont(Arial_Bold_14, ks0108ReadFontData, BLACK); // Auswahl Schrift
   ks0108GotoXY(14,5);                                       // Angabe Position
   ks0108Puts_P(PSTR("Display - Modul\nmit dem KS0108"));    // Ausgabe Text
   _delay_ms(1000);                                          // Pause
   ks0108SelectFont(small_font, ks0108ReadFontData, BLACK);
   ks0108GotoXY(14,35);                                      //Angabe Position
   ks0108Puts_P(PSTR("Spannungsmessung"));                   // Ausgabe Text
 }
```

Startbildschirm mit kurzer Angabe der Anwendung

```
void anzeige1()
 {
   ks0108ClearScreen();                                      // Display löschen
   // Text 1 überschrift
   ks0108SelectFont(Arial_Bold_14, ks0108ReadFontData, BLACK);
   ks0108GotoXY(4,2);                                        // Angabe Position
   ks0108Puts_P(PSTR("Anzeige Spannung"));                   // Ausgabe Text
   // Text 2 normale Anzeige
   ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);
   ks0108GotoXY(8,20);                                       // Angabe Position
   ks0108Puts_P(PSTR("P1 ="));                               // Ausgabe P
   ks0108GotoXY(60,20);                                      // Angabe Position
   ks0108Puts_P(PSTR("U1 ="));                               // Ausgabe U
   ks0108GotoXY(117,20);                                     // Angabe Position
   ks0108Puts_P(PSTR("V"));                                  // Ausgabe V
   ks0108GotoXY(92,20);                                      // Angabe Position
   ks0108Puts_P(PSTR("."));                                  // Ausgabe Punkt
   // Text 3 normale Anzeige
   ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);
```

```
  ks0108GotoXY(8,30);                                   // Angabe Position
  ks0108Puts_P(PSTR("P2 ="));                           // Ausgabe P
  ks0108GotoXY(60,30);                                  // Angabe Position
  ks0108Puts_P(PSTR("U2 ="));                           // Ausgabe U
  ks0108GotoXY(117,30);                                 // Angabe Position
  ks0108Puts_P(PSTR("V"));                              // Ausgabe V
  ks0108GotoXY(92,30);                                  // Angabe Position
  ks0108Puts_P(PSTR("."));                              // Ausgabe Punkt
  // Text 4 normale Anzeige
  ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);
  ks0108GotoXY(8,40);                                   // Angabe Position
  ks0108Puts_P(PSTR("P3 ="));                           // Ausgabe P
  ks0108GotoXY(60,40);                                  // Angabe Position
  ks0108Puts_P(PSTR("U3 ="));                           // Ausgabe U
  ks0108GotoXY(117,40);                                 // Angabe Position
  ks0108Puts_P(PSTR("V"));                              // Ausgabe V
  ks0108GotoXY(92,40);                                  // Angabe Position
  ks0108Puts_P(PSTR("."));                              // Ausgabe Punkt
  // Text 5 normale Anzeige
  ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);
  ks0108GotoXY(8,50);                                   // Angabe Position
  ks0108Puts_P(PSTR("P4 ="));                           // Ausgabe P
  ks0108GotoXY(60,50);                                  // Angabe Position
  ks0108Puts_P(PSTR("U4 ="));                           // Ausgabe U
  ks0108GotoXY(117,50);                                 // Angabe Position
  ks0108Puts_P(PSTR("V"));                              // Ausgabe V
  ks0108GotoXY(92,50);                                  // Angabe Position
  ks0108Puts_P(PSTR("."));                              // Ausgabe Punkt
 }
```

Mit diesem Code erzeuge ich eine „Maske" die einmalig auf das Graphikdisplay während des Startvorganges gelegt wird

```
int main(void)
 {
  DDRD=0b00001000;                                      // Angabe Pin am AT1284p
  //PORTD &=~(1<<PD3);                                  // LED Licht aus
  PORTD |=(1<<PD3);                                     // LED Licht an
  ks0108Init(0);                                        // Initialize GLCD
  _delay_ms(5);                                         // Pause
  i2c_init();                                           // Starte I2C Bus
  startanzeige();
  _delay_ms(2000);                                      // Pause
  ks0108ClearScreen();                                  // Display löschen
  anzeige1();
  char Buffer[20];
```

Start des Programmes, Buffer definieren, Bus und LCD initiieren, LCD-Licht an, LCD Befehle senden, Aufruf Startbildschirm und anzeige1, Display löschen und 5 bzw. 2000 ms Pause

```
// Zeile 2 Wert P1
itoa( results[0], Buffer, 10 );                    // Berechne Wert
ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);  //Auswahl Schrift
ks0108FillRect(29, 20, 20, 8, WHITE);              // Angabe Position überschreibe Wert
ks0108GotoXY(31,20);                               // Angabe Position
ks0108Puts(Buffer);                                // Ausgabe Wert an Position
```

results[0] wird durch itoa gewandelt (itoa(results[0], Buffer, 10);), eine Schriftart ausgewählt und eine Position zur Ausgabe angegeben. Vor Ausgabe des neuen Wertes wird der alte Wert durch überschreiben gelöscht.
Das wiederholen wir für die nächsten 3 Zeilen. Dadurch wird die „Maske" nicht verändert.

```
// Zeile 2 rechts
      int16_t adc1l, adc1a, adc1r1, adc1r2, adc1b;          // notwendige Variablen
      adc1=((results[0]*398)/100);                          // 1. Schritt
      if (adc1>1025)
       {
         adc1=1025;                                         // Begrenzung
       }
      adc1a=adc1;                                           // 2. Schritt
      adc1r2=adc1a % 10;                                    // 3. Schritt rechts 2
      adc1b=adc1a/10;                                       // 4. Schritt
      adc1r1=adc1b % 10;                                    // 5. Schritt rechts 1
      adc1l=adc1b/10;                                       // 6. Schritt  links
      ks0108FillRect(79, 20, 11, 8, WHITE);          // Angabe Position überschreibe Wert
      ks0108FillRect(94, 20, 14, 8, WHITE);          // Angabe Position überschreibe Wert
      ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);  //Auswahl Schrift
      itoa( adc1l, Buffer, 10 );                            // Zeile 2 Zahl 1
      ks0108GotoXY(85,20);                                  // Angabe Position
      ks0108Puts(Buffer);                                   // Ausgabe Wert an Position
      itoa( adc1r1, Buffer, 10 );                           // Zeile 2 Zahl 2
      ks0108GotoXY(95,20);                                  // Angabe Position
      ks0108Puts(Buffer);                                   // Ausgabe Wert an Position
      itoa( adc1r2, Buffer, 10 );                           // Zeile 2 Zahl 3
      ks0108GotoXY(100,20);                                 // Angabe Position
      ks0108Puts(Buffer);                                   // Ausgabe Wert an Position
```

adc1=((results[0]*398)/100); // 1. Schritt

Mit dieser Zeile berechnen wir den adc1 Wert. Dazu wird results[0] mit dem Korrekturfaktor von 398 multipliziert.

Den Korrekturfaktor errechnen wir mit (Berechnet 399, verwendet 398)

$$K = \frac{Referensspannung}{2^8} \times T = \frac{2,46\,V}{256} \times 4,16 = 0,0399746 \times 10000 \approx 399$$

Damit haben wir auch die Auflösung der Spannungsmessung berechnet. Diese beträgt bei 2^8 0,0399746 V oder rund 40 mV.

Für die Berechnung wird K mit 10000 multipliziert. Dadurch führen wir nur Festkommaarithmetik durch. Das spart sehr viel Rechenzeit.
Mit dem Korrekturfaktor K kann ein Abgleich der Eingangsspannung vorgenommen werden.

Berechnung der möglichen maximalen Anzeige:

Maximale Ausgabe des PCF 8591 = **255**
Korrekturfaktor (K) = **399**

$$Ausgabe\,\max = (Ausgabe\ PCF8591 \times K) \div 100 = (255 \times 399) \div 100 = 1018$$

```
if (adc1>1025)
 {
   adc1=1025;        // Begrenzung
 }
```

Sehen wir uns dazu ein Beispiel an: **adc1 = (results[0] x K) / 100**
adc1 = (188 x 399) / 100
adc1 = 750,12

Begrenzung der möglichen maximalen Anzeige

Da **adc1** ein int Wert ist, werden nur ganze Zahlen gespeichert **adc1 = 750**

```
adc1a=adc1;                    // 2. Schritt
```

adc1a = 750

im zweiten Schritt speichern wir **adc1** als **adc1a**

```
adc1r2=adc1a % 10;
```

im dritten Schritt wird **adc1a** durch **Modulo 10** geteilt **adc1r2 = 0**
(Modulo (%) bedeutet, Rest einer Division)

```
adc1b=adc1a/10;                // 4. Schritt
```

im vierten Schritt wird **adc1a** durch 10 dividiert **adc1b = 75**
und als **adc1b** gespeichert

```
adc1r1=adc1b % 10;             // 5. Schritt rechts 1
```

im fünften Schritt wird **adc1b** durch **Modulo 10** geteilt **adc1r1 = 5**
und als **adc1r1** gespeichert

```
adc1l=adc1b/10;                // 6. Schritt  links
```

im sechsten Schritt wird **adc1b** durch 10 dividiert **adc1l = 7**
und als **adc1l** gespeichert

```
itoa( adc1l, Buffer, 10 );          // Zeile 2 Zahl 1
ks0108GotoXY(85,20);                // Angabe Position
ks0108Puts(Buffer);                 // Ausgabe Wert an Position   Anzeige 7
itoa( adc1r1, Buffer, 10 );         // Zeile 2 Zahl 2
ks0108GotoXY(95,20);                // Angabe Position
ks0108Puts(Buffer);                 // Ausgabe Wert an Position   Anzeige 5
itoa( adc1r2, Buffer, 10 );         // Zeile 2 Zahl 3
ks0108GotoXY(100,20);               // Angabe Position
ks0108Puts(Buffer);                 // Ausgabe Wert an Position   Anzeige 0
```

in diesem Teil des Programmes wird durch **itoa** die gespeicherten Werte **adc1l, adc1r1** und **adc1r2** gewandelt und an den gewünschten Stellen unserer Anzeige angezeigt.

Die Zahl **750** wird angezeigt durch:

1. Zahl - **adc1l** - **7**
2. Zahl - **adc1r1** - **5**
3. Zahl - **adc1r2** - **0**

Die Berechnung und Darstellung habe ich recht ausführlich dargestellt. Es können einige Schritte gekürzt oder zusammengefasst werden.
Die Berechnung der anderen Zeilen erfolgt genauso. Achtet bitte genau auf die Bezeichnungen der Variablen. Auf Grund der Ähnlichkeit kommt man recht schnell durcheinander.

Das ganze Programm:

```
/* ATB_Ana_Prg_7.c Created: 13.12.2015 17:08:23 Author: AS */
// Hardware
// P30 mit ATmega 1284p
// P35 Grapfigdisplay
// NT 2 Netzteil 5V + 12V
// Eingabe der analogwerte mit P33 - Anschluss Wandler 0 bis 10V über Poti
// Spannungsnormal 10V  P25
////////////////////////////////////////////////////////////////////////////////////////
// Achtung:  Basis 10 V
////////////////////////////////////////////////////////////////////////////////////////

#define F_CPU 16000000UL
#include <inttypes.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <util/delay.h>
#include <stdlib.h>
#include "i2cmaster.h"

#include "ks0108.h"
#include "arial_bold_14.h"
#include "corsiva_12.h"
#include "arial8.h"
#include "small_font.h"

#define PCF8591w 0x90        // Adresse analog Modul
#define PCF8591r 0x91

int8_t i;
float results[5],adc1 ,adc2 ,adc3 ,adc4;

void startanzeige()
 {
   ks0108ClearScreen();                                          // Display löschen
   ks0108DrawRoundRect(1, 2, 125, 61, 8, BLACK);                 // Ausgabe Rand
   ks0108DrawLine(110, 32, 67, 60, BLACK);                       //Ausgabe Linie
   ks0108DrawRect(12, 47, 10, 10, BLACK);                        //Ausgabe Viereck leer
   ks0108FillRect(40, 50, 10, 10, BLACK);                        //Ausgabe Viereck gefüllt
   ks0108Circle(110, 50, 10, BLACK);                             // Ausgabe Kreis
   ks0108SelectFont(Arial_Bold_14, ks0108ReadFontData, BLACK); // Auswahl Schrift
   ks0108GotoXY(14,5);                                           // Angabe Position
   ks0108Puts_P(PSTR("Display - Modul\nmit dem KS0108"));        // Ausgabe Text
   _delay_ms(1000);                                              // Pause
   ks0108SelectFont(small_font, ks0108ReadFontData, BLACK);
   ks0108GotoXY(14,35);                                          //Angabe Position
   ks0108Puts_P(PSTR("Spannungsmessung"));                       // Ausgabe Text
 }
```

```
void anzeige1()
 {
   ks0108ClearScreen();                                  // Display löschen
   // Text 1 überschrift
   ks0108SelectFont(Arial_Bold_14, ks0108ReadFontData, BLACK);
   ks0108GotoXY(4,2);                                    // Angabe Position
   ks0108Puts_P(PSTR("Anzeige Spannung"));               // Ausgabe Text
   // Text 2 normale Anzeige
   ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);
   ks0108GotoXY(8,20);                                   // Angabe Position
   ks0108Puts_P(PSTR("P1 ="));                           // Ausgabe P
   ks0108GotoXY(60,20);                                  // Angabe Position
   ks0108Puts_P(PSTR("U1 ="));                           // Ausgabe U
   ks0108GotoXY(117,20);                                 // Angabe Position
   ks0108Puts_P(PSTR("V"));                              // Ausgabe V
   ks0108GotoXY(92,20);                                  // Angabe Position
   ks0108Puts_P(PSTR("."));                              // Ausgabe Punkt
   // Text 3 normale Anzeige
   ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);
   ks0108GotoXY(8,30);                                   // Angabe Position
   ks0108Puts_P(PSTR("P2 ="));                           // Ausgabe P
   ks0108GotoXY(60,30);                                  // Angabe Position
   ks0108Puts_P(PSTR("U2 ="));                           // Ausgabe U
   ks0108GotoXY(117,30);                                 // Angabe Position
   ks0108Puts_P(PSTR("V"));                              // Ausgabe V
   ks0108GotoXY(92,30);                                  // Angabe Position
   ks0108Puts_P(PSTR("."));                              // Ausgabe Punkt
   // Text 4 normale Anzeige
   ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);
   ks0108GotoXY(8,40);                                   // Angabe Position
   ks0108Puts_P(PSTR("P3 ="));                           // Ausgabe P
   ks0108GotoXY(60,40);                                  // Angabe Position
   ks0108Puts_P(PSTR("U3 ="));                           // Ausgabe U
   ks0108GotoXY(117,40);                                 // Angabe Position
   ks0108Puts_P(PSTR("V"));                              // Ausgabe V
   ks0108GotoXY(92,40);                                  // Angabe Position
   ks0108Puts_P(PSTR("."));                              // Ausgabe Punkt
   // Text 5 normale Anzeige
   ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);
   ks0108GotoXY(8,50);                                   // Angabe Position
   ks0108Puts_P(PSTR("P4 ="));                           // Ausgabe P
   ks0108GotoXY(60,50);                                  // Angabe Position
   ks0108Puts_P(PSTR("U4 ="));                           // Ausgabe U
   ks0108GotoXY(117,50);                                 // Angabe Position
   ks0108Puts_P(PSTR("V"));                              // Ausgabe V
   ks0108GotoXY(92,50);                                  // Angabe Position
   ks0108Puts_P(PSTR("."));                              // Ausgabe Punkt
 }
```

```
int main(void)
 {
   DDRD=0b00001000;                                    // Angabe Pin am AT1284p
   //PORTD &=~(1<<PD3);                                // LED Licht aus
   PORTD |=(1<<PD3);                                   // LED Licht an
   ks0108Init(0);                                      // Initialize GLCD
   _delay_ms(5);                                       // Pause
   i2c_init();                                         // Starte I2C Bus
   startanzeige();
   _delay_ms(2000);                                    // Pause
   ks0108ClearScreen();                                // Display löschen
   anzeige1();
   char Buffer[20];
   while(1)
    {
      i2c_start(PCF8591w);
      i2c_write(0x44);                        // Analog out enable, auto increment, channel 0
      i2c_stop();
      i2c_start(PCF8591r);
      i2c_readAck();                                   // first data is old
      for (i=0; i<4; i++)
       {
         results[i]=i2c_readAck();
       }
      i2c_readNak();
      i2c_stop();
      _delay_ms(2);
      // Zeile 2 Wert P1
      itoa( results[0], Buffer, 10 );                  // Berechne Wert
      ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);   //Auswahl Schrift
      ks0108FillRect(29, 20, 20, 8, WHITE);          // Angabe Position überschreibe Wert
      ks0108GotoXY(31,20);                             // Angabe Position
      ks0108Puts(Buffer);                              // Ausgabe Wert an Position
      // Zeile 3 Wert P2
      itoa( results[1], Buffer, 10 );                  // Berechne Wert
      ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);   //Auswahl Schrift
      ks0108FillRect(29, 30, 20, 8, WHITE);          // Angabe Position überschreibe Wert
      ks0108GotoXY(31,30);                             // Angabe Position
      ks0108Puts(Buffer);                              // Ausgabe Wert an Position
      // Zeile 4 Wert P3
      itoa( results[2], Buffer, 10 );                  // Berechne Wert
      ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);   //Auswahl Schrift
      ks0108FillRect(29, 40, 20, 8, WHITE);          // Angabe Position überschreibe Wert
      ks0108GotoXY(31,40);                             // Angabe Position
      ks0108Puts(Buffer);                              // Ausgabe Wert an Position
      // Zeile 5 Wert P4
      itoa( results[3], Buffer, 10 );                  // Berechne Wert
      ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);   //Auswahl Schrift
      ks0108FillRect(29, 50, 20, 8, WHITE);          // Angabe Position überschreibe Wert
      ks0108GotoXY(31,50);                             // Angabe Position
      ks0108Puts(Buffer);                              // Ausgabe Wert an Position
```

```
// Zeile 2 rechts
int16_t adc1l, adc1a, adc1r1, adc1r2, adc1b;
adc1=((results[0]*398)/100);                                  // 1. Schritt
if (adc1>1025)
 {
   adc1=1025;                                                 // Begrenzung
 }
adc1a=adc1;                                                   // 2. Schritt
adc1r2=adc1a % 10;                                            // 3. Schritt rechts 2
adc1b=adc1a/10;                                               // 4. Schritt
adc1r1=adc1b % 10;                                            // 5. Schritt rechts 1
adc1l=adc1b/10;                                               // 6. Schritt  links
ks0108FillRect(79, 20, 11, 8, WHITE);          // Angabe Position überschreibe Wert
ks0108FillRect(94, 20, 14, 8, WHITE);          // Angabe Position überschreibe Wert
ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);      //Auswahl Schrift
itoa( adc1l, Buffer, 10 );                                    // Zeile 2 Zahl 1
ks0108GotoXY(85,20);                                          // Angabe Position
ks0108Puts(Buffer);                                           // Ausgabe Wert an Position
itoa( adc1r1, Buffer, 10 );                                   // Zeile 2 Zahl 2
ks0108GotoXY(95,20);                                          // Angabe Position
ks0108Puts(Buffer);                                           // Ausgabe Wert an Position
itoa( adc1r2, Buffer, 10 );                                   // Zeile 2 Zahl 3
ks0108GotoXY(100,20);                                         // Angabe Position
ks0108Puts(Buffer);                                           // Ausgabe Wert an Position
// Zeile 3 rechts
int16_t adc2l, adc2a, adc2r1, adc2r2, adc2b;
adc2=((results[1]*398)/100);                                  // 1. Schritt
if (adc2>1025)
 {
   adc2=1025;                                                 // Begrenzung
 }
adc2a=adc2;                                                   // 2. Schritt
adc2r2=adc2a % 10;                                            // 3. Schritt rechts 2
adc2b=adc2a/10;                                               // 4. Schritt
adc2r1=adc2b % 10;                                            // 5. Schritt rechts 1
adc2l=adc2b/10;                                               // 6. Schritt  links
ks0108FillRect(79, 30, 11, 8, WHITE);          // Angabe Position überschreibe Wert
ks0108FillRect(94, 30, 14, 8, WHITE);          // Angabe Position überschreibe Wert
ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);      //Auswahl Schrift
itoa( adc2l, Buffer, 10 );                                    // Zeile 3 Zahl 1
ks0108GotoXY(85,30);                                          // Angabe Position
ks0108Puts(Buffer);                                           // Ausgabe Wert an Position
itoa( adc2r1, Buffer, 10 );                                   // Zeile 3 Zahl 2
ks0108GotoXY(95,30);                                          // Angabe Position
ks0108Puts(Buffer);                                           // Ausgabe Wert an Position
itoa( adc2r2, Buffer, 10 );                                   // Zeile 3 Zahl 3
ks0108GotoXY(100,30);                                         // Angabe Position
ks0108Puts(Buffer);                                           // Ausgabe Wert an Position
// Zeile 4 rechts
int16_t adc3l, adc3a, adc3r1, adc3r2, adc3b;
adc3=((results[2]*398)/100);                                  // 1. Schritt
```

```
    if (adc3>1025)
      {
        adc3=1025;                                          // Begrenzung
      }
     adc3a=adc3;                                            // 2. Schritt
     adc3r2=adc3a % 10;                                     // 3. Schritt rechts 2
     adc3b=adc3a/10;                                        // 4. Schritt
     adc3r1=adc3b % 10;                                     // 5. Schritt rechts 1
     adc3l=adc3b/10;                                        // 6. Schritt  links
     ks0108FillRect(79, 40, 11, 8, WHITE);                  // Angabe Position überschreibe Wert
     ks0108FillRect(94, 40, 14, 8, WHITE);                  // Angabe Position überschreibe Wert
     ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);   //Auswahl Schrift
     itoa( adc3l, Buffer, 10 );                             // Zeile 2 Zahl 1
     ks0108GotoXY(85,40);                                   // Angabe Position
     ks0108Puts(Buffer);                                    // Ausgabe Wert an Position
     itoa( adc3r1, Buffer, 10 );                            // Zeile 2 Zahl 2
     ks0108GotoXY(95,40);                                   // Angabe Position
     ks0108Puts(Buffer);                                    // Ausgabe Wert an Position
     itoa( adc3r2, Buffer, 10 );                            // Zeile 2 Zahl 3
     ks0108GotoXY(100,40);                                  // Angabe Position
     ks0108Puts(Buffer);                                    // Ausgabe Wert an Position
     // Zeile 5 rechts
     int16_t adc4l, adc4a, adc4r1, adc4r2, adc4b;
     adc4=((results[3]*398)/100);                           // 1. Schritt
     if (adc4>1025)
      {
        adc4=1025;                                          // Begrenzung
      }
     adc4a=adc4;                                            // 2. Schritt
     adc4r2=adc4a % 10;                                     // 3. Schritt rechts 2
     adc4b=adc4a/10;                                        // 4. Schritt
     adc4r1=adc4b % 10;                                     // 5. Schritt rechts 1
     adc4l=adc4b/10;                                        // 6. Schritt  links
     ks0108FillRect(79, 50, 11, 8, WHITE);                  // Angabe Position überschreibe Wert
     ks0108FillRect(94, 50, 14, 8, WHITE);                  // Angabe Position überschreibe Wert
     ks0108SelectFont(Arial8, ks0108ReadFontData, BLACK);   //Auswahl Schrift
     itoa( adc4l, Buffer, 10 );                             // Zeile 2 Zahl 1
     ks0108GotoXY(85,50);                                   // Angabe Position
     ks0108Puts(Buffer);                                    // Ausgabe Wert an Position
     itoa( adc4r1, Buffer, 10 );                            // Zeile 2 Zahl 2
     ks0108GotoXY(95,50);                                   // Angabe Position
     ks0108Puts(Buffer);                                    // Ausgabe Wert an Position
     itoa( adc4r2, Buffer, 10 );                            // Zeile 2 Zahl 3
     ks0108GotoXY(100,50);                                  // Angabe Position
     ks0108Puts(Buffer);                                    // Ausgabe Wert an Position
     _delay_ms(20);
  }
}
```

So könnte das Ergebnis aussehen

Anzeige von links nach rechts

- Angabe des analogen Kanals (P1 bis P4)
- Werte das PCF 8591
- Angabe der U1 bis U4
- Umgerechnete Spannungen U1 bis U4

(Es sind nur 2 Potis angeschlossen)

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