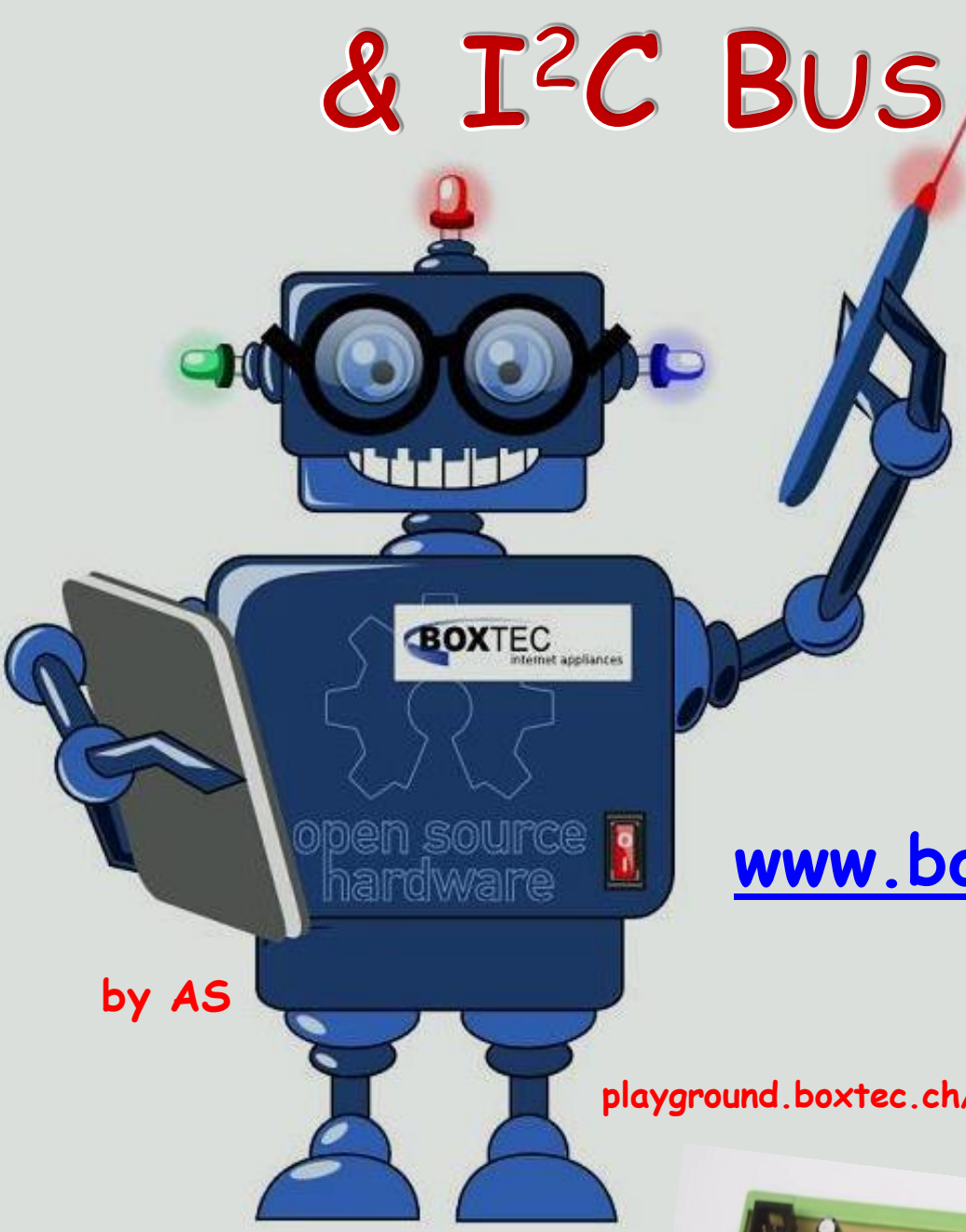
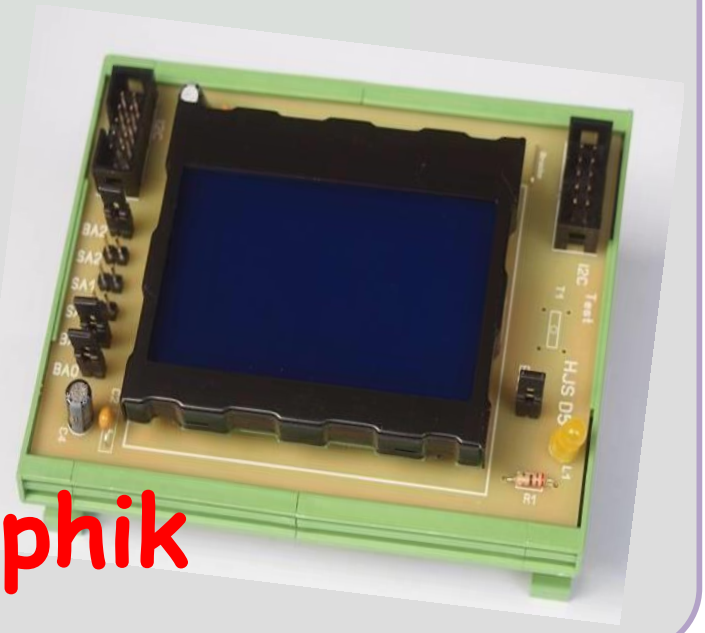
Mikrokontroller & I²C Bus



www.boxtec.ch

by AS

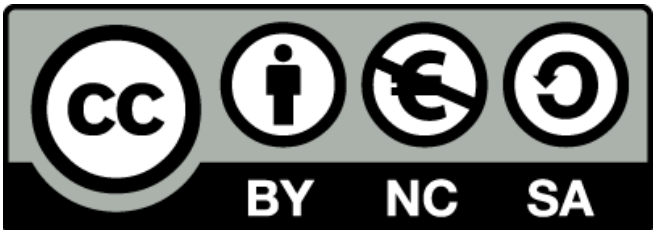
playground.boxtec.ch/doku.php/tutorial



Ein Graphik Display eDIP128 am I²C Bus Teil 2 - Software

I2C Bus + Graphik

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:

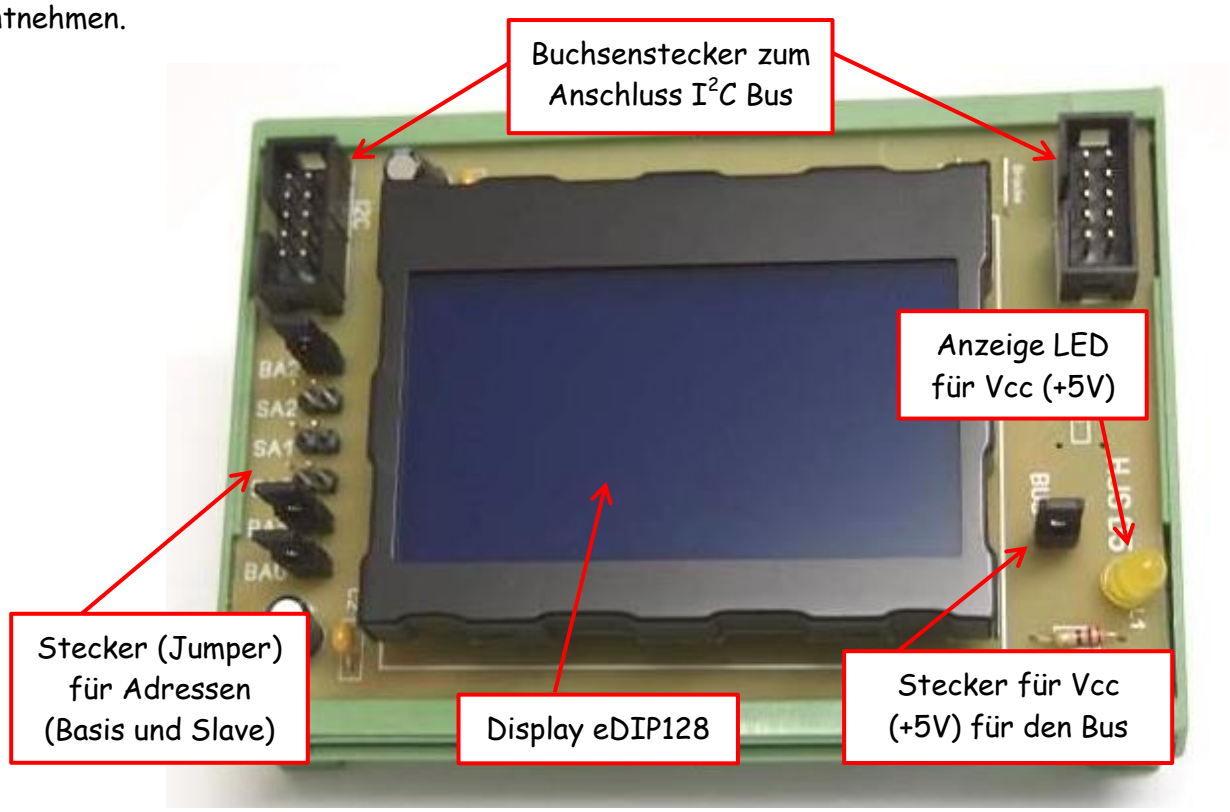
- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige 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 und Graphik - Software

Im zweiten Teil wollen wir uns die „Bedienung" und die Programmierung genauer ansehen. Bedienung ist allerdings übertrieben. Eigentlich sind auf der Platine nur die zwei Buchsenstecker zum Anschluss des I²C Bus, die Adressstecker, der Bus-Stecker und die Anzeige der Vcc (+5V). Den Aufbau und genaue Lage der Bauteile bzw. Platinen bitte dem Teil Hardware entnehmen.

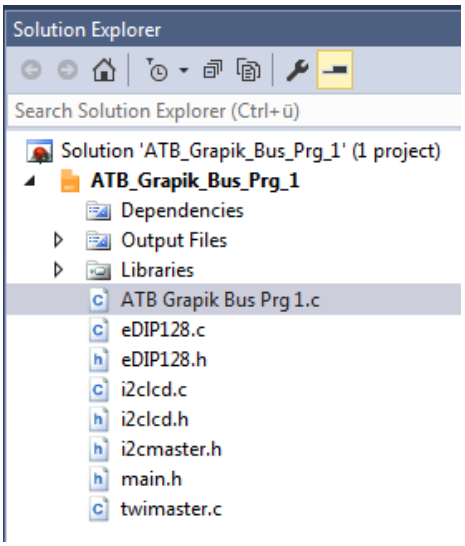


Inbetriebnahme

- Anschluss der Platine über die 10 poligen Steckverbinder an das Grundboard. Die Stromversorgung erfolgt über das Anschlusskabel
- Mit dem Busstecker wird der I²C Bus auf +5V gelegt. Wiederstände befinden sich unterhalb des Displays
- Durch die LED wird Vcc (+5V) angezeigt
- Mit den Steckern (Jumper), BA0 – BA2 und SA0 – SA1, erfolgt eine Auswahl der Busadresse. Ohne korrekte Busadresse erfolgt keine Verbindung zum Display
- Die Hintergrundbeleuchtung des Displays leuchtet und der Cursor blinkt in der linken oberen Ecke

Damit haben wir unser Display in Betrieb genommen. Sollte es nicht funktioniere, hilft nur Fehlersuche. Dabei unbedingt die Richtung des Displays (oben und unten) und den korrekten Sitz der Steckverbinder kontrollieren.
Ein verdrehen oder verschieben der Steckverbinder kann zu eine sofortigen Zerstörung des Displays führen.
Stecker, Kabelverbinder oder das Display dürfen nur im ausgeschalteten Zustand entfernt oder umgesteckt werden.

Die Programmierung erfolgt in C mit dem Atmel Studio 7.0. Die folgenden Dateien sind notwendig und in das Programm einzubinden:



- ATB Graphik Bus Prg 1.c
- eDIP128.c
- eDIP128.h
- i2cmaster.h
- main.h
- twimaster.c
- i2clcd.c
- i2clcd.h

Die beiden Dateien i2clcd sind für die Funktion nicht notwendig bzw. noch aus anderen Programmen vorhanden. Bei einer Kontrolle der Prüfsumme können sie für ein anderes Display verwendet werden. Dazu kommen wir aber im Teil Fehlersuche.

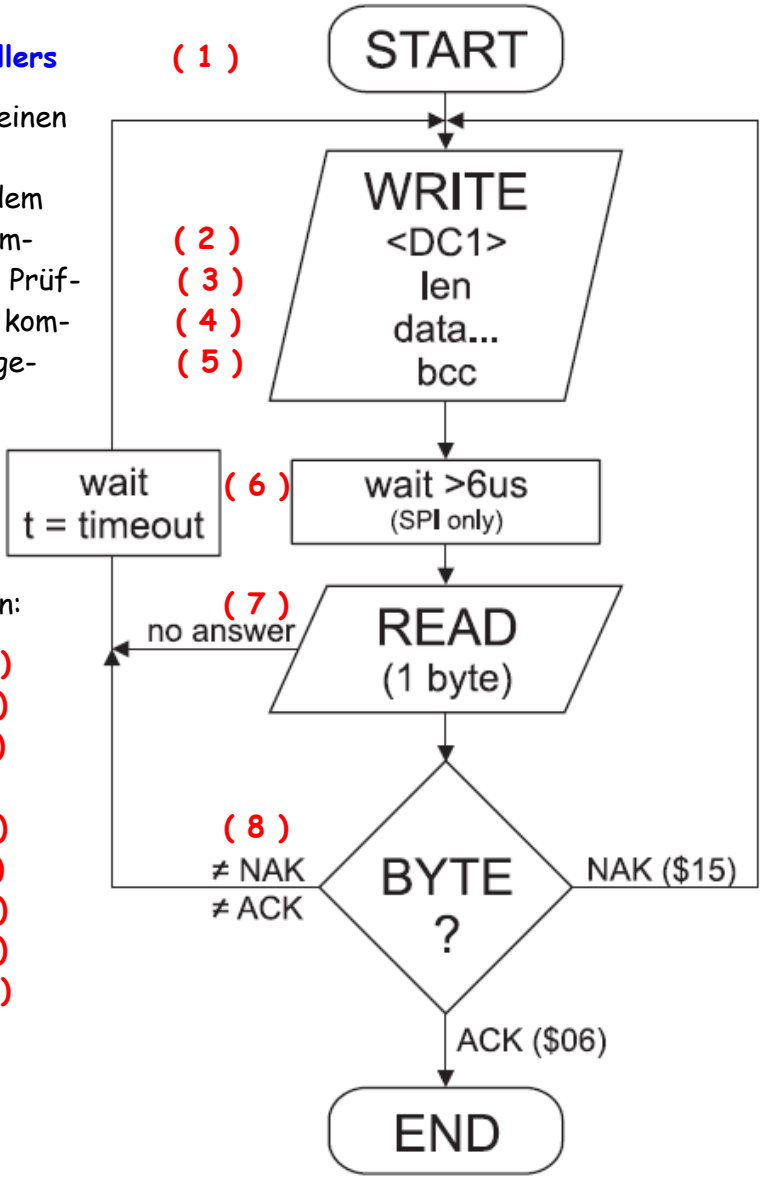
Auszug aus dem Datenblatt des Herstellers

Die Datenübertragung ist eingebettet in einen festen Rahmen mit Prüfsumme „**bcc**".
Das **eDIP128** quittiert dieses Paket mit dem Zeichen <**ACK**> (**$06**) bei erfolgreichem Empfang oder <**NAK**> (**$15**) bei fehlerhafter Prüfsumme. Bei Rückgabe von <**NAK**> wird das komplette Paket verworfen und muss erneut gesendet werden. Jeder Befehl beginnt mit **ESCAPE** gefolgt von einem oder zwei Befehlsbuchstaben und einigen Parametern.

<ACK> muss eingelesen werden.

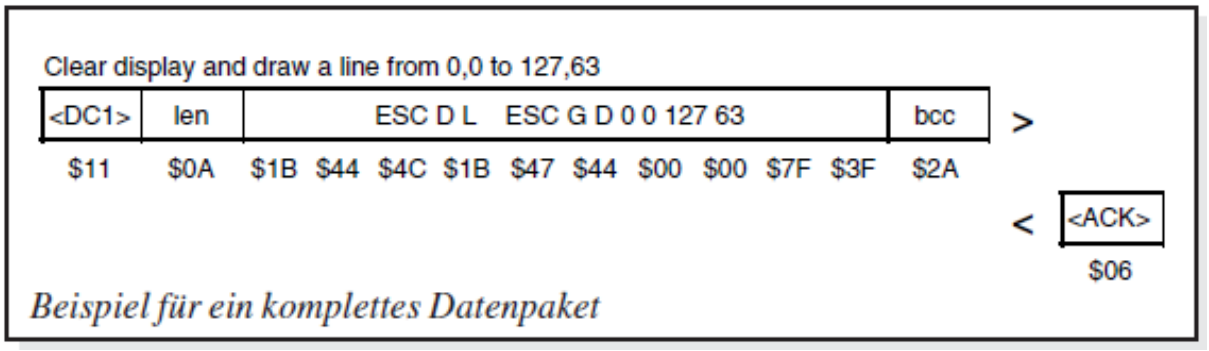
Sehen wir uns die Übertragung genauer an:

- Start der Übertragung (1)
- Senden von **DC1** ($11) (2)
- Senden von **len** (3)
- Senden von ESC, Data und Parameter (4)
- Senden der Prüfsumme **bcc** (5)
- **> 6µs** warten (6)
- Lesen der Daten (7)
- Auswertung der Daten (8)



Beispiel:

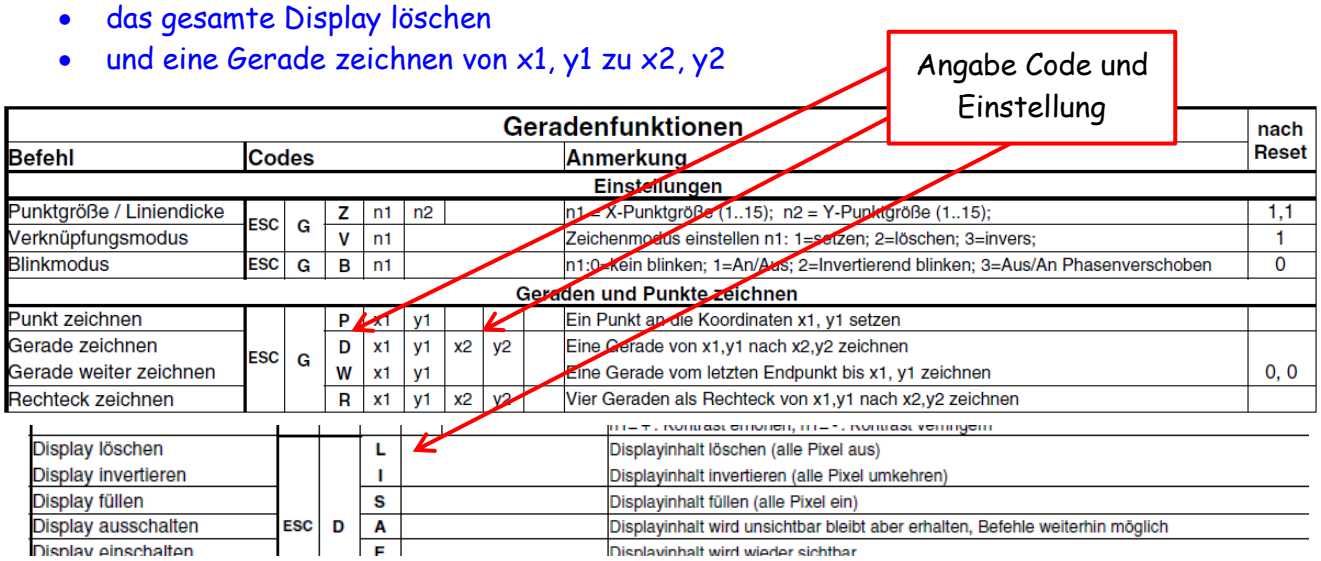
- Display löschen und eine Linie von 0,0 nach 127,63 zeichnen



Eingerahmt von <**DC1**>, der Anzahl der Daten "**len**" und der Prüfsumme "**bcc**" werden die jeweiligen Nutzdaten übertragen. Als Antwort sendet das Display <**ACK**> zurück.

An dem Beispiel möchte ich die Datenübertragung erläutern. Wir wollen als erstes

- das gesamte Display löschen
- und eine Gerade zeichnen von x1, y1 zu x2, y2



dazu müssen wir als erstes dem Datenblatt den Code entnehmen. Für

- Display löschen, **D** und **L**
- Gerade zeichnen, **G** und **D** und die Positionen **x1, y1, x2, y2**

Sehen wir uns den Ablauf genauer an.

In der Datei **eDIP128.h** geben wir dem Programm bekannt, mit welchen Teilen wir arbeiten:

```
// alles löschen
void Graphik_loeschen(void);

// Gerade von x1, y1 zu x2, y2
void Graphik_Gerade(int8_t x1, int8_t y1, int8_t x2, int8_t y2);
```

Im Programm **ATB_Graphik_Bus_Prg 1** werden diese Befehle aufgerufen

```
Graphik_loeschen();

Graphik_Gerade(5,5,85,5);          // Gerade von 5, 5 zu 85, 5
```

In der **eDIP128.c** werden die entsprechenden Befehle ausgeführt:

```
void Graphik_loeschen(void)
 {
   int bcc;
   bcc = 0x11 + 0x03 + 0x1b + 'D' + 'L' ;          // Berechnung len
   i2c_start(slave_adresse_1);                     ( 1 )
   i2c_write(DC1);   // DC1                        ( 2 )
   i2c_write(0x03);  // len 0x03                   ( 3 )
   i2c_write(ESC);   // ESC                        ( 4 )
   i2c_write('D');   // D                          ( 4 )
   i2c_write('L');   // L                          ( 4 )
   i2c_write(bcc);                                 ( 5 )
   i2c_start(slave_adresse_2);
   e = i2c_readAck();                              ( 7 )
   i2c_stop();
   _delay_us(10);                                  ( 6 )
 }
```

- Es wir das Unterprogramm **Graphik_loeschen** aufgerufen
- Es wird die Variable **bcc** (Prüfsumme) bekannt gegeben
- Die wird die Prüfsumme **bcc** aus **DC1, len, ESC** und den Codes **D** und **L** gebildet
- Es wird der Bus gestartet und die Daten an die Adresse **slave_adresse_1** gesendet
- Es werden die Daten **DC1, len, ESC, D** und **L** übertragen
- Es wird die Prüfsumme **bcc** übertragen
- Es werden Daten von **slave_adresse_2** ausgelesen und in **e** gespeichert
- Bus wird gestoppt
- Pause von **10 µs** gewartet

Das Auslesen der Daten und die Pause sind zwingend notwendig. Sonst können Befehle oder Teile überlesen werden und es kann zu recht eigenartigen Fehler kommen. Mit **e** wird der aktuelle Fehler angegeben. Wird kein Fehler erkannt, wird „**6**" zurückgegeben. Wird „**21**" oder eine andere Zahl zurückgegeben, ist ein Fehler innerhalb der Befehlsfolge. Es wird ACK ausgelesen, aber in diesem Programm nicht weiter verarbeitet.

```
void Graphik_Gerade(int8_t x1, int8_t y1, int8_t x2, int8_t y2)
 {
   uint8_t bcc;
   bcc = 0x11 + 0x07 + 0x1b + 'G' + 'D' + x1 + y1 + x2 + y2;
   i2c_start(slave_adresse_1);
   i2c_write(DC1);          // DC1
   i2c_write(0x07);         // len 07
   i2c_write(ESC);          // ESC
   i2c_write('G');          // G
   i2c_write('D');          // D
   i2c_write(x1);
   i2c_write(y1);
   i2c_write(x2);
   i2c_write(y2);
```

```
  i2c_write(bcc);
  i2c_start(slave_adresse_2);
  e = i2c_readAck();
  i2c_stop();
  _delay_us(10);
 }
```

- Es wir das Unterprogramm **Graphik_Gerade** aufgerufen und Werte übergeben
- Es wird die Variable **bcc** (Prüfsumme) bekannt gegeben
- Die wird die Prüfsumme **bcc** aus **DC1, len, ESC** und den Codes **G** und **D** und **x1, y1, x2, y2** gebildet
- Es wird der Bus gestartet und die Daten an die Adresse **slave_adresse_1** gesendet
- Es werden die Daten **DC1, len, ESC, D, L x1, y1, x2, y2** übertragen
- Es wird die Prüfsumme **bcc** übertragen
- Es werden Daten von **slave_adresse_2** ausgelesen und in **e** gespeichert
- Bus wird gestoppt
- Pause von **10 µs** gewartet

Das Auslesen der Daten und die Pause sind zwingend notwendig. Sonst können Befehle oder Teile überlesen werden und es kann zu recht eigenartigen Fehler kommen. Mit **e** wird der aktuelle Fehler angegeben. Wird kein Fehler erkannt, wird „**6**" zurückgegeben. Wird „**21**" oder eine andere Zahl zurückgegeben, ist ein Fehler innerhalb der Befehlsfolge. Es wird ACK ausgelesen, aber in diesem Programm nicht weiter verarbeitet.

Nach Angabe des Herstellers soll eine Pause von **> 6 µs** vor dem lesen der Daten erfolgen. Ich habe diese Pause am Ende eingefügt. Es kommt bei meinem Hardware Aufbau sonst zu zeitweisen Störungen oder es werden keine korrekten Umschalten zwischen den Befehlen durchgeführt. Der Grund dazu ist mir unklar.

Sehen wir uns als nächste die Ausgabe von Texten an:

In der Datei **eDIP128.h** geben wir dem Programm wieder bekannt, mit welchen Teilen wir arbeiten:

```
// Ausgabe Text am gewählten Ort links
void Graphik_Text_ausgabe_L(int8_t x1, int8_t y1, char * Text);
```

Im Programm **ATB_Graphik_Bus_Prg 1** werden diese Befehle wieder aufgerufen. Zusätzlich rufen wir vorher noch **Graphik_Auswahl_Font** auf

```
Graphik_Auswahl_Font(5);                    // Auswahl eines Fonts (Schriftart)
Graphik_Text_ausgabe_L(19,10,"Demo Board");
```

In der **eDIP128.c** werden die entsprechenden Befehle ausgeführt:

```
void Graphik_Text_ausgabe_L(int8_t x1, int8_t y1, char * Text)
 {
  int bcc,i;
  bcc = 0x11 + 0x06 + 0x1b + 'Z' + 'L' + x1 + y1 + strlen(Text);
  for (i=0; i<=strlen(Text); i++)
  bcc += Text[i];
```

```
  i2c_start(slave_adresse_1);
  i2c_write(DC1);                  // DC1
  i2c_write(0x06 + strlen(Text));// len 0x06+Text
  i2c_write(ESC);                  // ESC
  i2c_write('Z');                  // Z
  i2c_write('L');                  // L
  i2c_write(x1);
  i2c_write(y1);
  for (i=0; i<=strlen(Text); i++)
  i2c_write(Text[i]);
  i2c_write(bcc);
  i2c_start(slave_adresse_2);
  e = i2c_readAck();
  i2c_stop();
  _delay_us(10);
}
```

- Es wir das Unterprogramm **Graphik_Text_ausgabe_L** aufgerufen und Werte für die Position und der Text selber übergeben
- Es wird die Variable **bcc** (Prüfsumme) und **i** bekannt gegeben
- Die wird die Prüfsumme **bcc** aus **DC1, len, ESC** und den Codes **Z** und **L** und **x1, y1,** und **Text** gebildet. Dabei wird mit **strlen** die Länge des Textes ermittelt
- Es wird der Bus gestartet und die Daten an die Adresse **slave_adresse_1** gesendet
- Es werden die Daten **DC1, len, ESC, Z, L x1, y1** übertragen
- Der Text wird durch **i** einzelne gelesen und einzeln übertragen. Da wir vorher die Länge ermittelt haben, wird diese Länge zu **len** addiert
- Es wird die Prüfsumme **bcc** übertragen
- Es werden Daten von **slave_adresse_2** ausgelesen und in **e** gespeichert
- Bus wird gestoppt
- Pause von **10 µs** gewartet

Durch diesen Teil des Programmes wird an der gewählten Position „Demo Board" ausgegeben. Es muss aber vorher immer ein entsprechender Font (Schriftart) ausgewählt werden.

Nach dieser Vorlage können alle anderen Codes, Positionen und Auswahlen übertragen werden. Habe es bereits für die folgenden Anwendungen gemacht.

Im Einzelnen sind es:

```
Graphik_Cursor(1);                    // Cursor aus=0, ein=1
Graphik_Cursor_Position(1,1);         // Cursor Position max x=16 y=8
Graphik_Blinken(0);                   // Blinken Display aus=0, ein=1, 2=inv 3=Phase
Graphik_Punkt(96,39);                 // Ausgabe eines Punktes an x, y
Graphik_Punkt_Groesse(1,1);           // allgemein für alle x, y
Graphik_Gerade(5,15,72,42);           // Gerade von x1, y1 zu x2, y2
Graphik_Gerade_weiter(90,35);               // Gerade weiter zu x,y
Graphik_Text_ausgabe_Z(41,41,"(Text)");     // Angabe Position, Textausgabe zentriert
Graphik_Text_ausgabe_R(41,41,"(Text)");     // Angabe Position, Textausgabe rechts
Graphik_Text_ausgabe_L(41,41,"(Text)");     // Angabe Position, Textausgabe links
```

```
Graphik_Auswahl_Font(f);                    // Auswahl Schrift
Graphik_Font_Winkel(w);                     // Winkel Schrift
Graphik_Font_Zoom(z1, z2);                  // Vergrösserung Schrift in x und y
Graphik_Bereich_invertieren(18,15,24,27);   // Bereich invertieren blinke
Graphik_Bereich_fuellen(12,12,34,34);       // Bereich fuellen nicht blinken
Graphik_Bereich_loeschen(10,10,94,54);      // Bereich loeschen

Graphik_Bereich_Muster(35,12,95,28,12);     // Bereich mit Muster füllen
Graphik_Rechteck(0,0,127,63);               // min x1=0, y1=0 max x2=127, y2=63

Graphik_Beleuchtung(1);                     // Beleuchtung 1=ein 0=aus
Graphik_Helligkeit(100);                    // Helligkeit von 0 bis 100%
Graphik_Info(void);                         // Anzeige Software Version und mehr
Graphik_Version(void);                      // Anzeige Version
Graphik_loeschen(void);                     // alles löschen
Graphik_invertieren(void);                  // alles invertieren
Graphik_Box_zeichnen(35,12,95,28,9);        // Box mit Muster füllen
Graphik_Rahmen_zeichnen(35,12,95,28,8);     // Rahmen zeichnen ohne Muster
Graphik_Rahmenbox_zeichnen(35,12,95,28,8);  // ok Rahmen zeichnen Muster füllen
Graphik_Bargraph_R(1,10,40,75,55,25,100,1,1); // Nr,x1,y1,x2,y2,Anfang,Ende, Strichbreite,
                                               Muster Füllung nach rechts
Graphik_Bargraph_L(1,10,40,75,55,25,100,1,1); // Nr,x1,y1,x2,y2,Anfang,Ende,Strichbreite,
                                               Muster Füllung nach links
Graphik_Bargraph_O(1,10,40,75,55,25,100,1,1); // Nr,x1,y1,x2,y2,Anfang,Ende,Strichbreite,
                                               Muster Füllung nach oben
Graphik_Bargraph_U(1,10,40,75,55,25,100,1,1); // Nr,x1,y1,x2,y2,Anfang,Ende,Strichbreite,
                                               Muster Füllung nach unten
Graphik_Bargraph_aktuell(1,70);             // Nr Bargraph, Angabe in %
Graphik_Menue_zeichnen(x1, y1, a1, char * Text);      // Ausgabe Menue Inhalt und Rahmen
Graphik_Menue_weiter(void);                 // Menue Balken nach unten - weiter
Graphik_Menue_zurueck(void);                // Menue Balken nach oben - zurück
Graphik_Menue_Font(f);                      // Auswahl Schrift Menue
```

In diesen Dateien habe ich die Befehle bereits eingetragen

- **eDIP128.c**
- **eDIP128.h**

Aus dem Hauptprogramm **ATB Graphik Bus Prg 1.c** können diese Befehle einfach aufgerufen werden. In der Datei **main.h** habe ich die verwendeten Bus Adressen und die Quarzfrequenz eingetragen.

So einfach wie man etwas aufs Display schreiben oder zeichnen kann, so einfach und schnell kommt man aber auch zu Fehlern. Leider ist die Fehlersuch nicht so einfach.
Habe eine Möglichkeit gefunden, wie man schnell und sicher durch auslesen des ACK den Fehler eingrenzen bzw. finden kann.

Was sind die beliebtesten Fehler? (**Fehlersuche**)

- Eine falsche Angabe der Prüfsumme (bcc)
- Eine falsche Angabe von der Codes
- Eine falsche Angabe der Position
- Vergessene Codes
- Pause vergessen
- Bus ist zu schnell

Zur Fehlersuche habe ich ein zweites Display 4x16 und einem PCF8574 am selben Bus betrieben. Dieses Display habe ich wie gewohnt mit einer anderen Adresse betrieben und mir darauf verschiedene Zustände und Abläufe des Programmes anzeigen lassen. Die wichtigste Ausgabe auf diesem Display ist aber der ausgelesen Wert **ACK** vom **eDIP128**.
Wird der Wert „**6**" ausgeben, liegt kein Fehler in der Datenübertragung vor. Wird aber der Wert „**21**" oder ein andere Wert ausgegeben, stimmt was mit der Datenübertragung nicht. Dann hilft nur Vergleich des Datenblattes mit meinem Programm oder wo liegt der Schreibfehler.

Dazu verwende ich den folgenden Code:

```
void Graphik_Auswahl_Font(int8_t f)            // Ausgabe Font (Schriftart)
 {
   int bcc;                                                              (1)
   bcc = 0x11 + 0x04 + 0x1b + 'Z' + 'F' + f;                             (2)
   i2c_start(slave_adresse_1);                                           (3)
   i2c_write(DC1);            // DC1 oder 0x11                           (4)
   i2c_write(0x04);           // len 0x04                                (5)
   i2c_write(ESC);            // ESC oder 0x1b                           (6)
   i2c_write('Z');                                                       (7)
   i2c_write('F');                                                       (8)
   i2c_write(f);                                                         (9)
   i2c_write(bcc);                                                       (10)
   i2c_start(slave_adresse_2);                                           (11)
   e = i2c_readAck();                                                    (12)
   i2c_stop();                                                           (13)
   itoa( e, Buffer, 10 );                                                (14)
   lcd_printlc(4,8,Buffer);                                              (15)
   _delay_us(10);                                                        (16)
 }
```

An diesem Beispiel werde ich die Möglichen Fehler beschreiben. Sicher gibt es noch andere Fehler.

- In der Zeile **(1)** wird **bcc** deklariert. Diese Bezeichnung muss mit der Zeile **(10)** übereinstimmen
- In der Zeile **(2)** wird die Prüfsumme **bcc** berechnet. Die Bezeichnungen müssen mit den Zeilen **(4)**, **(5)**, **(6)**, **(7)**, **(8)** und **(9)** übereinstimmen. Ein korrekter Code und Gross- und Kleinschreibung beachten

- Keine oder falsche slave_adresse_1 in Zeile **(3)** angegeben
- In der Zeile **(5)** muss ich die korrekte Prüfsumme **(len)** per Hand eingeben. Grösse kann man durch testen ermitteln
- In den Zeilen **(4)** und **(6)** korrekte Einstellungen für **DC1** und **ESC** eingeben
- In den Zeilen **(7)** und **(8)** den korrekten Code eingeben (Kontrolle). Muss mit der Zeile **(2)** übereinstimmen
- In der Zeile **(9)** wird der gewählte Font übermittelt. Bitte Bereich beachten
- In der Zeile **(10)** wird die ermittelte Prüfsumme **(bcc)** übermittelt
- In der Zeile **(11)** wird die eingestellte Leseadresse angegeben
- In der Zeile **(12)** wird **ACK** ausgelesen und in **e** gespeichert
- Bitte **i2c_stop** in Zeile **(13)** nicht vergessen
- In der Zeile **(14)** wird der Wert in **e** durch **itoa** „umgerechnet"
- In der Zeile **(15)** wird dieser Wert auf dem zweiten Display in der Zeile **4** Position **8** ausgegeben. Der ausgegebene Wert gibt den „Fehler" an
- Bitte die Pause von min. **> 6 µs** nicht vergessen
- In der Datei twimaster.c muss die Busfrequenz geändert werden

```
/* I2C clock in Hz */
// #define SCL_CLOCK  400000L
# define SCL_CLOCK  100000L
```

Bei Angabe der Grösse und Position ist unbedingt die Grösse des Display und der möglichen Pixel zu beachten. Bei Ausgabe von Texten und den Fontgrössen unbedingt die mögliche Fläche beachten.

Mit den angegebenen Dateien und der Anleitung zur Hard- und Software dürfte die Nutzung des Displays eDIP128 ohne Probleme möglich sein. Einen langsamen Aufbau oder eine verzögerte Darstellung konnte ich nicht feststellen. Sicher gibt es noch genügend Möglichkeiten das Programm zu verbessern oder zusätzliche Funktionen einzubinden.
Anzeige von Bildern und Makros sind nicht Bestandteil dieser Tuts.
Über Verbesserungen und Änderungen würde ich mich sehr freuen.

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://www.lcd-module.de/
http://www.lcd-module.de/produkte/edip.html
http://www.lcd-module.de/datenblaetter.html

Es wurden Teile aus dem Datenblatt zum eDIP128-6 des Herstellers EA verwendet.