Mikrokontroller & I²C Bus

www.boxtec.ch

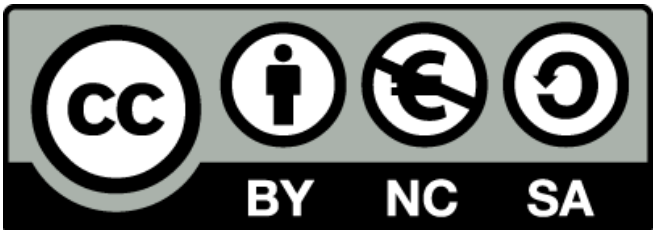
by AS

playground.boxtec.ch/doku.php/tutorial

I²C Bus und E-Paper
Teil 2 - Software

I²C Bus und E-Paper

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:

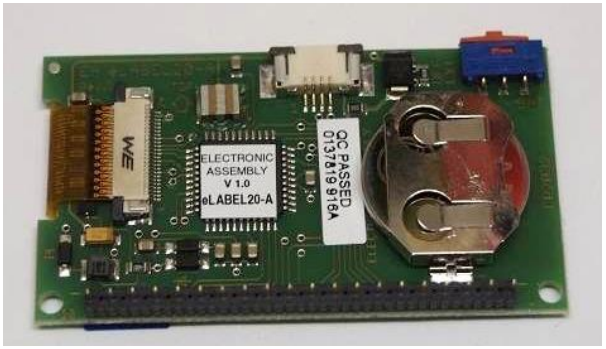
- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige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

I²C – Bus und E – Paper - Software

Auf der Rückseite unseres E-Papers befindet sich ein Prozessor mit den notwendigen Bauteilen zur Ansteuerung. Dadurch ist die Ausgabe von Texten und Graphik relativ einfach möglich.



Buchsenstecker zum Anschluss I²C Bus

Stecker für Vcc (+5V) I²C Bus

Stecker (Jumper) für Adressen (Basis und Slave)

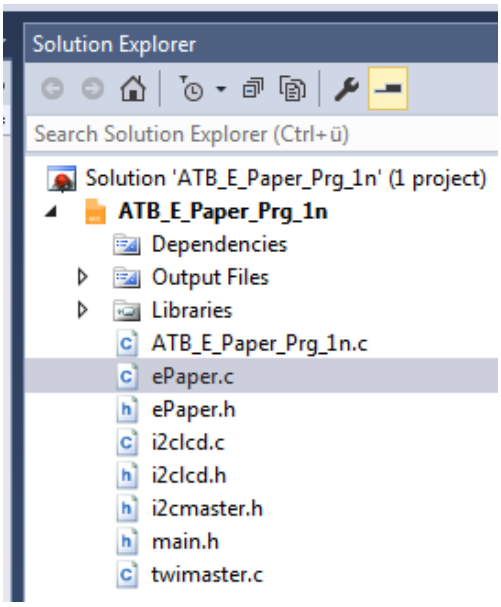
Anzeige Vcc +5V

Inbetriebnahme

- Anschluss der Platine über ein 20 poligen Steckverbinder an das Grundboard. Die Stromversorgung erfolgt über das Anschlusskabel
- Mit dem Busstecker wird der I²C Bus auf +5V gelegt. Wiederstände befinden auf der Platine
- Durch die LED wird Vcc (+5V) angezeigt
- Mit den Steckern (Jumper), BA0 – BA1 und SA0 – SA1, erfolgt eine Auswahl der Busadresse. Ohne korrekte Busadresse erfolgt keine Verbindung zum Display
- Es wird der letzte Inhalt des Displays dargestellt. Die Anzeige erfolgt auch ohne Betriebsspannung

Damit haben wir unser Display in Betrieb genommen. Sollte es nicht funktioniere, hilft nur Fehlersuche. Dabei unbedingt den korrekten Sitz der Steckverbinder am Display 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.
Achtet auf die Unterseite des Displays. Die Anschlüsse und Batterie liegen frei. Die Einbauhöhe beträgt nur ca. 4mm!

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



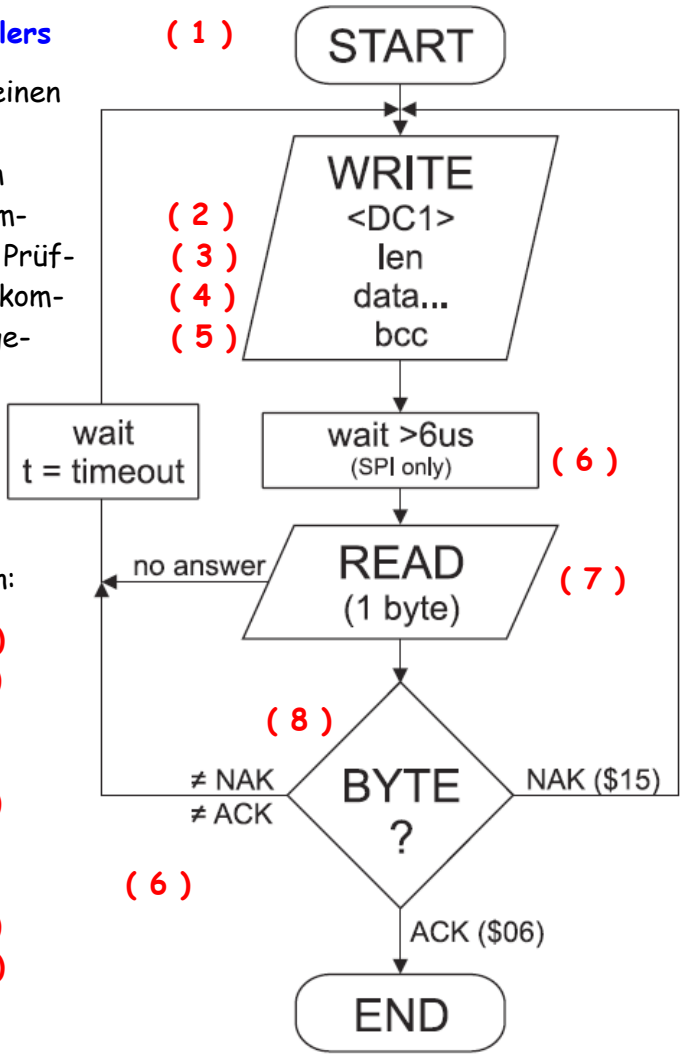
- ATB_E_Paper_Prg_1n.c
- ePaper.c
- ePaper.h
- i2cmaster.h
- main.h
- twimaster.c
- i2clcd.c
- i2clcd.h

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 **eLabel** 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 ausgelesen 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)
- Mindestens **> 6µs** warten (6)
- Lesen der Daten (7)
- Auswertung der Daten (8)

Vom Hersteller werden **> 6µs** angegeben. Das kann zu Problemen führen, z.B. das Programm bricht willkürlich ab oder hängt sich total auf. Ohne jede Vorwarnung oder Zeitweise.
Habe diesen Wert auf **30µs** gesetzt.

Beispiel:

- Display löschen und eine Linie von 0, 0 nach 171, 71 zeichnen

Clear display and draw a line from 0,0 to 171,71

<DC1>	len	ESC D L	ESC G D 0 0 171 71	bcc	>
$11	$0A	$1B $44 $4C	$1B $47 $44 $00 $00 $AB $47	$5E	
					< <ACK>
					$06

Beispiel für ein komplettes Datenpaket

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

Displayfunktionen (Wirkung auf das gesamte Display)								nach Reset
Befehl	**Codes**						**Anmerkung**	
Display Orientierung	ESC	D	O	n1			n1=0: 0° = 172x 72 pixel; Terminal: 21 Spalten, 9 Zeilen n1=1: 90° = 72x172 pixel; Terminal: 9 Spalten, 21 Zeilen	0°
Display Update			U				Der aktuelle Inhalt des Display-RAM wird ins ePAPER kopiert	
Display Autoupdate Zeiten			Z	n1	n2		n1: Verzögerungszeit nach dem letzten Befehl bevor geupdatet wird n2: Zwangsupdatezeit nach dem ersten Befehl, wenn ohne Unterbrechung gesendet wird n1,n2=1..255 in 1/10sec; (n1,n2=0: Autoupdate Aus; ESC DU für Update benutzen)	3, 50
Display löschen	ESC	D	L				Displayinhalt löschen (alle Pixel aus)	
Display invertieren			I				Displayinhalt invertieren (alle Pixel umkehren)	
Display füllen			F	n1			Displayinhalt mit Farbe n1 füllen 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß	

Angabe Code und Einstellung

Geradenfunktionen									nach Reset
Befehl	**Codes**							**Anmerkung**	
Einstellungen									
Geradenfarbe einstellen	ESC	F	G	n1				Linienfarbe n1 einstellen 1=Schwarz; 2=Dunkelgrau; 3=Grau; 4=Weiß oder 0=Invertieren	1
Punktgröße / Liniendicke		G	Z	n1	n2			n1 = X-Punktgröße (1..15); n2 = Y-Punktgröße (1..15);	1,1
Geraden und Punkte zeichnen									
Punkt zeichnen	ESC	G	P	x1	y1			Ein Punkt an die Koordinaten x1,y1 setzen	
Gerade zeichnen			D	x1	y1	x2	y2	Eine Gerade von x1,y1 nach x2,y2 zeichnen	
Gerade weiter zeichnen			W	x1	y1			Eine Gerade vom letzten Endpunkt bis x1, y1 zeichnen	0, 0
Rechteck zeichnen			R	x1	y1	x2	y2	Vier Geraden als Rechteck von x1,y1 nach x2,y2 zeichnen	

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 **ePaper.h** geben wir dem Programm bekannt, mit welchen Teilen wir arbeiten:

```
void eP_Display_loeschen();          // Display löschen

// Gerade zeichnen G + D, x= 0-171, y= 0-71
void eP_Gerade_zeichnen(int8_t x1, int8_t y1, int8_t x2, int8_t y2);
```

Im Programm **ATB_E_Paper_Prg_1.c** werden diese Befehle aufgerufen:

```
eP_Display_loeschen();                  // Display löschen D + L

eP_Gerade_zeichnen(0, 0, 171, 71);      // Gerade zeichnen von x1, y1 zu x2, y2; G + D
```

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

```
void eP_Display_loeschen()          // Display löschen
 {
   int8_t bcc;
   bcc = DC1 + 0x03 + ESC + 'D' + 'L' ;          // Berechnung bcc
   i2c_start(slave_adresse_1);                   ( 1 )
   i2c_write(DC1);          // DC1                ( 2 )
   i2c_write(0x03);         // 03                 ( 3 )
   i2c_write(ESC);          // ESC                ( 4 )
   i2c_write('D');                               ( 4 )
   i2c_write('L');                               ( 4 )
   i2c_write(bcc);                               ( 5 )
   i2c_stop();
   _delay_us (30);                               ( 6 )
   i2c_start(slave_adresse_2);                   ( 7 )
   e = i2c_readAck();                            ( 8 )
   i2c_stop();
   _delay_us (100);
 }
```

- Es wir das Unterprogramm **eP_Display_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
- Bus wird gestoppt
- Pause **30 µs**
- Der Bus wird gestartet
- Es werden Daten von **slave_adresse_2** ausgelesen und in **e** gespeichert
- Bus wird gestoppt
- Pause **100 µs**

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 eP_Gerade_zeichnen(int8_t x1, int8_t y1, int8_t x2, int8_t y2) // Gerade zeichnen G + D
 {
   int8_t bcc;
   bcc = DC1 + 0x07 + ESC + 'G' + 'D' + x1 + y1 + x2 + y2;      // Berechnung bcc
   i2c_start(slave_adresse_1);                   ( 1 )
   i2c_write(DC1);          // DC1                ( 2 )
   i2c_write(0x07);         // 07                 ( 3 )
   i2c_write(ESC);          // ESC                ( 4 )
   i2c_write('G');          // G                  ( 4 )
```

```
  i2c_write('D');          // D              ( 4 )
  i2c_write(x1);           // x1             ( 4 )
  i2c_write(y1);           // y1             ( 4 )
  i2c_write(x2);           // x2             ( 4 )
  i2c_write(y2);           // y2             ( 4 )
  i2c_write(bcc);                           ( 5 )
  i2c_stop();
  _delay_us (30);                           ( 6 )
  i2c_start(slave_adresse_2);               ( 7 )
  e = i2c_readAck();                        ( 8 )
  i2c_stop();
  _delay_us (100);
}
```

- Es wir das Unterprogramm **eP_Gerade_zeichnen** 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, G, D, x1, y1, x2, y2** übertragen
- Es wird die Prüfsumme **bcc** übertragen
- Bus wird gestoppt
- Pause **30 µs**
- Der Bus wird gestartet
- Es werden Daten von **slave_adresse_2** ausgelesen und in **e** gespeichert
- Bus wird gestoppt
- Pause **100 µs**

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 auf **30 µs** vergrössert. Es kommt sonst bei meiner Hardware zu zeitweisen Störungen oder es werden keine korrekten Umschalten zwischen den Befehlen durchgeführt. Auch die Pause am Ende von **100 µs** muss deswegen sein. Der Grund dazu ist mir unklar.

Sehen wir uns als nächste die Ausgabe von Texten an:

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

```
// Ausgabe Text links Z + L
void eP_Text_L(int8_t x, int8_t y, char * Text);
```

Im Programm **ATB_E_Paper_Prg_1.c** werden diese Befehle wieder aufgerufen. Vorher müssen wir mit **eP_Font_Auswahl(3);** einen Font auswählen. Es können noch zusätzliche Befehle verwendet werden, z.B. eP_Font_Winkel(0);, eP_Font_Zoom(1,1); oder eP_Font_Farben(4,1);

```
void eP_Text_L(int8_t x, int8_t y, char * Text)        // Ausgabe Text links Z + L
 {
   int8_t bcc,i;
   bcc = DC1 + 0x06 + ESC + 'Z' + 'L' + x + y + strlen (Text);
   for (i=0; i<=strlen (Text); i++)
   bcc += Text[i];
   i2c_start(slave_adresse_1);
   i2c_write(DC1);
   i2c_write(0x06 + strlen (Text));
   i2c_write(ESC);
   i2c_write('Z');
   i2c_write('L');
   i2c_write(x);
   i2c_write(y);
   for (i=0; i<=strlen (Text); i++)
   i2c_write(Text[i]);
   i2c_write(bcc);
   i2c_stop();
   _delay_us (30);
   i2c_start(slave_adresse_2);
   e = i2c_readAck();
   i2c_stop();
   _delay_us (100);
 }
```

- Es wir das Unterprogramm **eP_Text_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 **x, y** und **Text** gebildet. Dabei wird mit **strlen** die Länge des Textes ermittelt und addiert
- Es wird der Bus gestartet und die Daten an die Adresse **slave_adresse_1** gesendet
- Es werden die Daten **DC1, len, ESC, Z, L x, y** 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
- Bus wird gestoppt
- Pause **30 µs**
- Es werden Daten von **slave_adresse_2** ausgelesen und in **e** gespeichert
- Bus wird gestoppt
- Pause **100 µs**

Durch diesen Teil des Programmes wird an der gewählten Position „Test 1" ausgegeben.
Es muss aber vorher immer ein entsprechender Font (Schriftart) ausgewählt werden.

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 oder zu kurz
- Bus ist zu schnell (max. 100kHz)

Zur Fehlersuche habe ich ein zweites Display 4x16 mit 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 **eLabel 20-A**.
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. Für eine genaue Fehlersuch und die Beschreibung im PDF I²C Graphik nachlesen.

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

Im Einzelnen sind es:

```
eP_Display_loeschen();                    // Display löschen D + L
eP_Version_Info();                        // Info abfragen T + V
eP_Geradenfarbe(1);                       // Geradenfarbe einstellen F + G
eP_Liniendicke(3, 3);                     // Punktgrösse / Liniendicke G + Z
eP_Gerade_zeichnen(5, 5, 100, 50);        // Gerade zeichnen G + D
eP_Punkt_zeichnen(15, 5);                 // Punkt zeichnen G + P
ep_Rechteck(5, 5, 120, 55);               // Rechteck zeichnen G + R
eP_Gerade_weiter(20, 70);                 // Gerade weiter zeichnen G + W
eP_Cursor_Position(5, 5);                 // Cursor Position T + P Ursprung 1,1
eP_Font_Winkel(0);                        // Auswahl Winkel Z + W 0=0°, 1=90°
eP_Display_invertieren();                 // Display invertieren D + I
eP_Display_fuellen(2);                    // Display füllen D + F
eP_Display_aus();                         // Display einschalten D + A
eP_Display_ein();                         // Display ausschalten D + E
ep_Bereich_loeschen(10, 32, 68, 49);      // Bereich löschen R + L
ep_Bereich_invertieren(10, 32, 68, 49);   // Bereich invertieren R + I
ep_Bereich_fuellen(10, 32, 128, 65, 3);   // Bereich füllen R + F
eP_Rahmen_Farbe(1, 4);                    // Rahmen Farbe einstellen F + R
ep_Rahmen_zeichnen( 10, 32, 128, 65, 8);  // Rahmen zeichnen R + R
ep_Bereich_Fuellmuster(10, 32, 128, 65, 15); // Bereich mit Füllmuster R + M
ep_Box_Fuellmuster( 10, 32, 128, 65, 12); // Box mit Füllmuster R + O
eP_Terminal_ein();                        // Terminal einschalten T + E
eP_Terminal_aus();                        // Terminal ausschalten T + A
eP_Display_Orientierung(0);               // Display Orientierung D + O
eP_Font_Farben(4,1);                      // Text Farben einstellen F + Z
```

```
eP_Font_Auswahl(3);                 // Auswahl Schriftart Z + F 0-8
eP_Font_Zoom(1,1);                  // Text Zoom einstellen Z + Z x=1x-4x, y=1x-4x
eP_Text_L(95,3,"Test L1");          // Ausgabe Text links Z + L
eP_Text_C( 55, 41, "Test L3");      // Ausgabe Text zentriert Z + C
eP_Text_R( 35, 15, "Test L2");      // Ausgabe Text rechts Z + R
eP_Projektname();                   // Projektname anzeigen T + J
eP_Info();                          // Informationen anzeigen T + I
eP_Summer(1);                       // Summer ein=1, aus=0, 2-255-> je 1/10 Y + S
```

Bei einigen Anweisungen habe ich bereits Angaben oder Parameter eingetragen. Die genauen Angaben bitte dem Datenblatt des Herstellers entnehmen.

In diesen Dateien habe ich die Befehle bereits eingetragen

- **ePaper.c**
- **ePaper.h**

Aus dem Hauptprogramm **ATB_E_Paper_Prg_1.c** können diese Befehle einfach aufgerufen werden. In der Datei **main.h** habe ich die verwendeten Bus Adressen und die Quarzfrequenz eingetragen.
Innerhalb der Aufrufe der einzelnen Funktion sind kurze Pausen einprogrammiert. Dies ist zwingend notwendig. Sonst kann es zu Fehlern oder einer Blockierung des I²C Bus kommen.

Durch den Aufruf einzelner Funktionen flackert das Display. Der Grund ist mir nicht bekannt.

Bitte unbedingt auf die Pegel von max. 3,3V achten. Betrieb nur mit Regelteil 3,3V und Pegelanpassung.

Beim Abschalten der Betriebsspannung bleibt der letzte Inhalt des Displays dauerhaft erhalten und kann zu einer Fehlinterpretation führen.

Für eine genaue Erklärung der Funktion und der Parameter bitte das Datenblatt des Herstellers benutzen.

http://www.lcd-module.de/fileadmin/pdf/grafik/elabel20-a.pdf

Alle notwendigen Dateien habe ich dem Tut angefügt. Einzelne Dateien und Teile habe ich den Programmen von Karl Heinz und PeDa entnommen.

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