TFT-Farbdisplays für Mikrocontroller

Ansteuerung des Minimoduls mit BASCOM-AVR

Dr. Claus Kühnel, Dr. Klaus Zahnert

1. Vorbemerkung

In Mikrocontrollerapplikationen mit eher beschränkten Ressourcen werden zur Ausgabe von Informationen und zur Präsentation von Ergebnissen sehr oft LC-Displays eingesetzt, die uns allen in sehr unterschiedlichen Bauformen und Konfigurationen bekannt sind.

Abbildung 1 zeigt ein sehr preiswertes Dot-Matrix-LCD zur Anzeige alphanumerischer Zeichen. Displays mit einer Zeile zu acht Zeichen bis hin zu vier Zeilen mit bis zu 40 Zeichen/Zeile sind von verschiedenen Anbietern erhältlich.

Abbildung 2 zeigt ein Grafik-LCD, welches neben der Anzeige alphanumerischer Zeichen auch grafische Elemente darstellen kann. Viele Displays besitzen bereits einen integrierten Controller, der das Setzen und Löschen einzelner Pixel ermöglicht. Der Bereich der Auflösung erstreckt sich für die Displays von 98 x 32 Pixel bis hin 320 x 240 Pixel (1/4 VGA).

Abbildung 3 zeigt ein Aktiv-Matrix-LCD (TFT) wie es heute in Mobiltelefonen, PDA und Digitalkameras zu finden ist. Auf Grund der Massenproduktion ist ein drastischer Preisverfall festzustellen und diese Displays sind daher auch für Einzelanwendungen und Kleinserien interessant geworden.

Abbildung 4 zeigt schließlich noch ein auf der Basis organischen LEDs (OLED) aufgebautes Grafik-Display, welches ohne Hintergrundbeleuchtung eine brillante Abbildung mit hohem Kontrast bei nahezu unbegrenztem Ablesewinkel ermöglicht. Die Technologie der OLED-Displays hat noch erhebliches Wachstumspotenzial.

Beim Aufbau von Geräten der Mess-, Steuerungs- und Automatisierungstechnik können grafische Displays neue Akzente hinsichtlich Bedienung und Visualisierung setzen. Die oft recht unansehnlichen grünen oder blauen 2x16 oder 4x20 LCDs können nun durch eine farbenfrohe Alternative ergänzt werden. Kostengünstige und einfach ansteuerbare Displays sind dafür eine wesentliche Voraussetzung.

Im Beitrag wird ein Farb-TFT der Fa. Speed IT up – Peter Küsters vorgestellt, mit welchem schnell und kostengünstig jede Mikrocontroller-Lösung mit einem farbigen Grafik-LCD mit hoher Auflösung erweitert werden kann.

Abbildung 1 Dot-Matrix LCD

Abbildung 2 LCD-Grafikdisplay

Abbildung 3 TFT-Grafikdisplay

Abbildung 4 OLED-Grafikdisplay

2. 2.1” Farbdisplay-Modul mit Eingabeeinheit

Die 2.1“ Farbdisplay-Module bieten eine Auflösung von 176 x 132 Pixel mit 65536 Farben.

Mit dem integrierten AVR-Mikrocontroller steht somit ein vollkommen autonomes Modul zur Verfügung, welches über eine RS232-, I^2C- und/oder SPI-Schnittstelle mit der Außenwelt kommunizieren kann. Zum Betreiben des Farbdisplay-Moduls ist keine zusätzliche Verdrahtung mehr erforderlich. Auf dem Farbdisplay-Modul vorhandene Taster lassen sich als Eingabeeinheit verwenden.

Als Betriebsspannung kann eine Gleichspannung zwischen 4,5 und 18 V dienen. Alle vom Farbdisplay-Modul benötigten Spannungen werden intern erzeugt.

Das TFT-Display wird auf die Mikrocontrollerplatine aufgesteckt und könnte auch unabhängig von der hier vorgestellten modularen Lösung eingesetzt werden.

Tabelle 1 zeigt die verfügbaren 2.1“ Farbdisplay-Module und deren Ausstattung. Die folgenden Ausführungen beziehen sich auf das Modul D072, was aber grundsätzlich keine Einschränkung gegenüber den anderen Modulen bedeutet.

Abbildung 5 zeigt das Farbdisplay-Modul D072 in Komplettausstattung. Die seitlichen Montagerahmen und das Tastenfeld können entfernt werden, so dass ein sehr kompaktes Modul erzeugt werden kann.

2.1” Farbdisplay-Module (Komplettlösungen mit AVR-Mikrocontroller)			
Artikel Nr.	**D071**	**D073**	**D072**
Mikrocontroller	ATmega128, ATmega2561, AT90CAN128		
Gesamtgröße in mm (inkl. montiertem Display)	103 x 58	variabel: max: 68 x 63 min: 60 x 41	variabel: max: 73 x 54 min: 59 x 41
Displaygröße	2,1” (53mm), 1310 mm²		
Betriebsspannung im Auslieferungszustand	4,5V-18V	4,5V-18V	4,5V-18V
Taster auf Modul	6 +Reset	6 +Reset	5(6) +Reset
Interface on Board	ISP, 2xRS232, JTAG, RS485	ISP, RS232	ISP, RS232
freie Ports an Steckern	51	50	38

Tabelle 1 Verfügbare 2.1“ Farbdisplay-Module

Abbildung 5 Farbdisplay-Modul D072

Die Programmierung des Farbdisplay-Moduls erfolgt über einen ISP-Programmieradapter. Es stehen ISP-Programmieradapter für Parallel- und COM-Port sowie den USB vom gleichen Anbieter zur Verfügung (Tabelle 2).

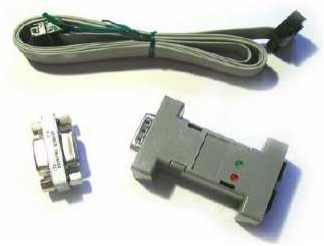
ISP-Programmieradapaters für		
Parallelport	***COM Port***	***USB***
		

Tabelle 2 ISP-Programmieradapter

3. Initialisierung und Ausgabe

Bei der Ansteuerung des Farbdisplay-Moduls unterscheiden wir zwischen der Initialisierung und der eigentlichen Ausgabe.

Durch die Initialisierung des Farbdisplay-Moduls wird die wunschgemäße Betriebsart eingestellt bevor die eigentlichen Ausgaben erfolgen können.

Für die Programmierung des Farbdisplay-Moduls wird hier BASCOM-AVR eingesetzt, wodurch die Notation der folgenden Anweisungen festgelegt ist. BASCOM-AVR wurde in diesem Handbuch schon mehrfach eingesetzt, weshalb an dieser Stelle nicht weiter auf Details dieser Entwicklungsumgebung eingegangen wird. Für die Programmierung in C sind Beispiele Im Lieferumfang des Farbdisplay-Moduls enthalten

3.1. Initialisierung des Farbdisplay-Moduls

Durch die Initialisierung des Farbdisplay-Moduls werden die Orientierung des Bildinhaltes und der Darstellungsmode für Bitmaps festgelegt. Tabelle 3 zeigt die Initialisierungskommandos und deren Parameter.

Initialisierung 2.1“ Farbdisplay		
`Orientation =`	**`Portrait | Portrait180 | Landscape | Landscape180`**	Orientierung des Bildinhaltes
`Graphics_mode =`	**`65k_uncompressed | 65k_compressed | 256low_uncompressed | 256low_compressed | 256high_uncompressed | 256high_compressed`**	Legt den Darstellungsmodus eines Bitmaps fest

Tabelle 3 Initialisierungskommandos

Orientierung des Displayinhalts und Fensterdefinition werden an Hand von Abbildung 6 erläutert. Beim D072 Farbdisplay-Modul besitzt die rechte, untere Ecke im Portraitmode die Koordinate (0,0). Um von dem in der linken Abbildung gezeigten Koordinatensystem ausgehen zu können, muss bei einem D072 *Orientation = Portrait180* initialisiert werden.

Für die Positionierung von grafischen Objekten gelten bei dieser Initialisierung dann die folgenden Beziehungen:

$$0 \le x \le 131$$

$$0 \le y \le 175$$

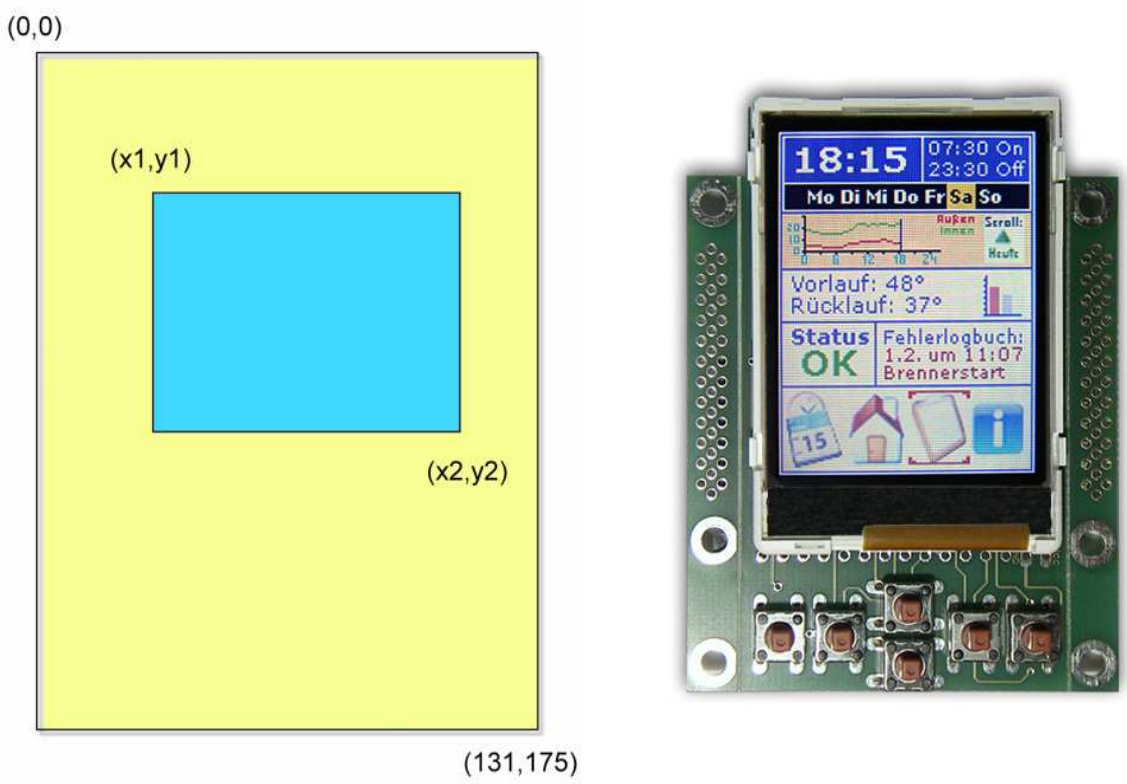


Abbildung 6 Koordinatensystem (Portrait180 bei D072)

3.2. Ausgabebefehle für grafische Objekte

Für die Ausgabe der grafischen Objekte stehen eine Reihe von Befehlen zur Verfügung, die in Tabelle 4 in einer Übersicht zusammengestellt sind.

Neben den Befehlen *LCD_Init* und *LCD_CLS* sind die Grundbefehle zur Ausgabe eines Textes (*LCD_Print()*), Setzen eines Pixels (*LCD_Plot()*), Zeichnen einer Linie (*LCD_Draw()*), Zeichnen eines Rechtecks (*LCD_Rect()*) und eines ausgefüllten Rechtecks (*LCD_Box()*) sowie der Ausgabe eines Bitmaps (*LCD_Bitmap()*) vorhanden.

Wie mit den einzelnen Befehlen gearbeitet werden kann, zeigen die im Abschnitt 5 vorgestellten Programmbeispiele.

Befehlsübersicht 2.1“ Farbdisplay	
LCD_Init	Initialisierung des LCDs
LCD_CLS	Löscht den Bildschirminhalt
LCD_Print(String, x, y, Font, ScaleX, ScaleY, FColor, BColor)	Druckt einen Textstring an eine beliebige Position auf den Bildschirm. Erlaubt eine Skalierung der Größe (Breite und Höhe) sowie die Festlegung der Schriftfarbe und der Hintergrundfarbe
LCD_Plot(x, y, Pixel, Color)	Setzt Pixel (1, 2x2) an einer gewünschte Position in der gewünschten Farbe
LCD_Draw (x1, y1, x2, y2, Pixel, Color)	Zeichnet eine Linie mit 1 oder 2 Pixel Breite von der Koordinate x1, y1 zur Koordinate x2, y2. Die Richtung ist egal. Dieser Algorithmus arbeitet nur mit Integerzahlen und ist sehr schnell.
LCD_Rect (x1, y1, x2, y2, Pixel, Color)	Zeichnet ein Rechteck (nicht gefüllt) mit 1 oder 2 Pixel Breite von der Koordinate x1, y1 zur Koordinate x2, y2.
LCD_Box(x1, y1, x2, y2, Color)	Zeichnet eine farbig gefüllte Box (x1, y1 ist links oben; x2, y2 ist rechts unten)
LCD_Bitmap(x1, y1, x2, y2)	Stellt eine Bitmapgrafik (z.B. Icons, Logos etc.) mit beliebiger Größe an eine gegebene Position dar. Für das 2.1” Display erlauben erweiterte Funktionen die Dekomprimierung komprimierter Bitmapdaten sowie die Nutzung von Daten mit indizierten Farbtabellen.

Tabelle 4 ***Befehlsübersicht***

4. Aufbereitung von Bitmap-Grafiken

Aus Tabelle 4 kennen wird nun den Befehl *LCD_Bitmap()* zur Ausgabe einer Bitmap-Grafik.

Dieser Befehl öffnet ein durch die Koordinaten (x1, y1) und (x2, y2) definiertes Ausgabefenster und füllt dieses mit der betreffenden Anzahl von Bitmap-Daten.

Für BASCOM-AVR müssen die Bitmap-Daten in Form von *DATA*-Anweisungen vorliegen. Diese *DATA*-Anweisungen können mit dem im Lieferumfang enthaltenen *Image Converter* erzeugt und als Binärdatei abgespeichert werden. Über das Kommando *$inc*… wird die Binärdatei (und damit die DATA-Anweisung(en)) in den Quelltext des Programms eingefügt. Mit dem Befehl *Restore* ist ein Pointer vorgängig auf den Anfang des betreffenden Datenfeldes zu setzen.

Die folgenden Programmzeilen zeigen beispielhaft die wenigen erforderlichen Schritte.

```
Restore T2a                                  ' adjust pointer to pixel data for Image2
Call Lcd_bitmap(0 , 0 , 131 , 175)           ' output Image2
…
$inc T1a , Nosize , "C:\...\Image1.bin"      ' pixel data for Image1
$inc T2a , Nosize , "C:\...\Image2.bin"      ' pixel data for Image2
```

Das Umsetzen eines Bildes in die Binärdatei wird nun vom bereits erwähnten *Image Converter* vorgenommen. Abbildung 7 zeigt die Oberfläche des Image Converters mit den Einstellmöglichkeiten, die mit den Angaben im Quelltext des zu erstellenden Programms abgeglichen sein müssen.

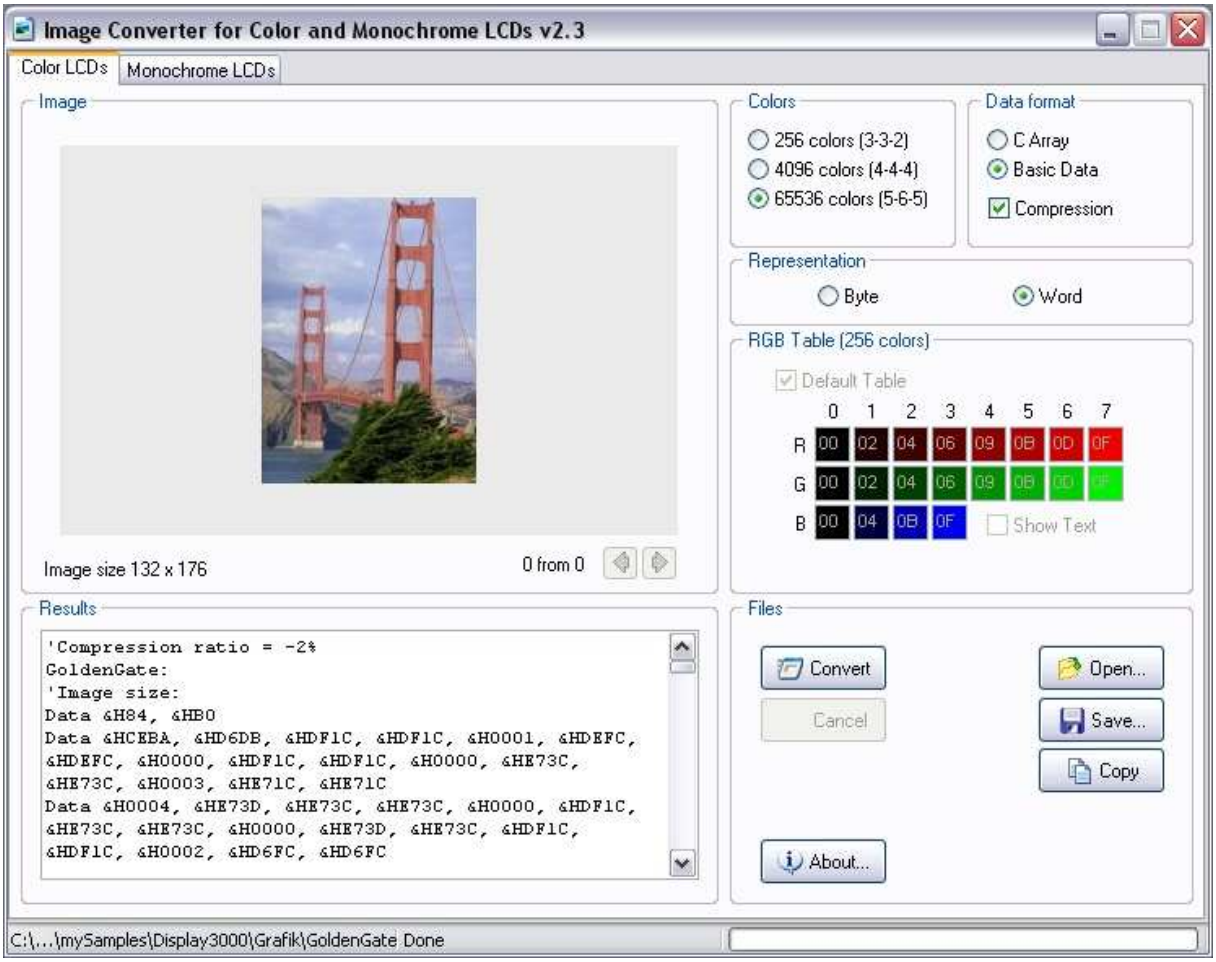


Abbildung 7 Aufbereitung einer Bitmap-Grafik im Image Converter

Das hier verwendete Bild der „Golden Gate Bridge“ deckt das gesamte Display (132 x 176 Pixel) ab. Arbeitet man mit größeren Vorlagen sind diese in ihrer Größe entsprechend zu reduzieren. Ein beliebiges Bildverarbeitungsprogramm kann dazu verwendet werden.

Gemäß den gesetzten Optionen wird hier mit größter Farbtiefe (65536 Farben) und Kompression gearbeitet. Nach Anklicken des Buttons *Convert* und kurzer Rechenzeit präsentiert sich das Ergebnis im Feld *Result*.

Die *DATA*-Anweisungen kann man gelassen zur Kenntnis nehmen und schließlich als Binärdatei abspeichern. Wie mit diesem dann im Quelltext des BASCOM-AVR Programms zu verfahren ist, war bereits erläutert worden.

5. *D072* Programmbeispiele

Der folgende Abschnitt zeigt einige Programmbeispiele, die die Anwendung der Grafikbefehle und die erzielten Ergebnisse aufzeigen.

5.1. Lizenzbestimmungen

Die Ansteuerung des Grafikdisplays wurde von Peter Küsters durch Reverse Engineering ermittelt. Das Display wurde hierzu in seiner ursprünglichen Anwendung betrieben und der Datenverkehr zwischen ansteuerndem Mikrocontroller und dem Display protokolliert. Nach Auswertung der umfangreichen Protokolldateien (mehrere Hundert MByte) war die Ansteuerung offen gelegt.

Mit dem Kauf eines solchen Grafikdisplay erwirbt der Käufer eine Source-Code-Lizenz, die den folgenden Bedingungen unterliegt.

> *Display-Software-Grundlagen wurden von Peter Küsters, www.display3000.com ermittelt.*
>
> *Dieser Display-Code ist urheberrechtlich geschützt. Sie erhalten eine Source-Code-Lizenz, d.h. Sie dürfen den Code in eigenen Programmen verwenden, diese aber nur in kompilierter Form weitergeben. Die Weitergabe dieses Codes in lesbarer Form oder die Publizierung im Internet etc. ist nicht gestattet und stellen einen Verstoß gegen das Urheberrecht dar.*

Um die Lizenzbedingungen zu erfüllen und dennoch die Möglichkeiten des Grafikdisplays an Hand von Programmbeispielen vorzustellen, werden die der Lizenz unterliegenden Programmteile in das im folgenden Abschnitt vorgestellte Template über Includes eingefügt.

Auf der beigefügten CD sind die auf dem Template aufbauenden Quelltexte sowie die kompletten Hexdateien enthalten. Die Includedateien erhält man beim Kauf eines Grafikdisplays automatisch.

5.2. Template für Programmbeispiele

Listing 1 zeigt den Quelltext der Datei *Template.bas*, die Grundlage für alle weiteren Programmbeispiele ist.

Fett markiert und nummeriert wurden alle Stellen, die einer anwendungsspezifischen Anpassung bedürfen. Tabelle 5 beschreibt die anzupassenden Stellen im Quelltext.

```
'####################################################################################################
' Program sample template       (1)
' Display D072
' Claus Kühnel 2007-03-29
'####################################################################################################

$hwstack = 64
$swstack = 128
$framesize = 16                 'you might need to raise these numbers if your code grows (2)

$regfile = "m128def.dat"
$crystal = 14745600             'enter the used clock of your actual microcontroller (3)

'####################################################################################################
'Application variables
'… your application data        (4)
```

```
'#####################################################################################################
'Definition of used ports and pull up resistors (5)

'At our boards we are using Port B for the SPI-communication to the LCD.
'Now we need to select Port B as to an output port (data output to the display)
'DDR = Data direction register; Port B1, B2, B4, B5, B6 switched to output (1) as needed by the display
Ddrb = &B01110110
Portb = &B10001001      '... the other ports of Port B are inputs with switched on pull up resistors

Ddra = &B00000000       'switch all 8 Ports of Port A to input (0), Pin (PA.0 - PA.7)
Porta = &B11111111      'All port pins have individually selectable pull-up resistors.
                        'Here we enable these pull-up-resistors, so these Pins are always at logical 1
                        'You need to pull these Pins against ground (GND)
Ddrc = &B00000000       'switch all Ports of Port C to input
Portc = &B11111111      'all pull-up-Resistors turned on

Ddrd = &B00000000       'switch all Ports of Port D to input
Portd = &B11111111      'all pull-up-Resistors turned on

Ddre = &B00000000       'switch all Ports of Port E to input
Porte = &B11111111      'all pull-up-Resistors turned on

Ddrf = &B00000000       'switch all Ports of Port F to input
Portf = &B11111111      'all pull-up-Resistors turned on

Ddrg = &B00000000       'switch all Ports of Port G to input
Portg = &B11111111      'all pull-up-Resistors turned on

$include C:\Programme\BASCOM-AVR\mySamples\DisplaySamples\Includes\Init21_display3000.bas ' (6)

'#####################################################################################################
'Initialisation of the display (needs to be done only for changes from default) (7)
Orientation = Portrait180               'select neeeded orientation, here: Portrait180 mode
Graphics_mode = 65k_compressed          'select the needed color mode, here 65.536 colors
Gosub Lcd_init     'Initialisation routine of the display, Needs to be done only once at the beginning
Gosub Lcd_cls

'#####################################################################################################
'Main program
Do
'… your application code        (8)
Loop

End
'#####################################################################################################

$include C:\Programme\BASCOM-AVR\mySamples\DisplaySamples\Includes\Glcd21_display3000.bas ' (9)

'-----------------------------------------------------------------------------------------
' Includes a graphics file shown at start up - you need to change the directory.
'-----------------------------------------------------------------------------------------
'$inc T2a , Nosize , "C:\Programme\BASCOM-AVR\mySamples\Display3000\Grafik\Goldengate.bin" ' (10)

'#####################################################################################################
$include C:\Programme\BASCOM-AVR\mySamples\DisplaySamples\Includes\Glcd21_fonts.bas ' (11)
'Dummy Data um Fehlermeldungen bei der Kompilierung der Standardroutinen zu vermeiden
'Die Tabelle wird dann bei Nutzung eines indizierten Grafikdatei mit "echten" Daten ausgetauscht
Colortable:
Data 0
```

Listing 1 Template.bas

Position	Anpassung
(1)	Programmkopf: Programmtitel, Beschreibung, Autor und Version
(2)	Stackangaben für BASCOM-AVR, müssen je nach Anwendungsprogramm ggf. erweitert werden
(3)	Taktfrequenz des verwendeten AVR-Mikrocontrollers
(4)	Deklaration der Konstanten und Variablen des Anwendungsprogramms
(5)	Definition der verwendeten Ports und PullUp-Widerstände
(6)	Include für Datei *Init21_display3000.bas* (Pfad anpassen)
(7)	Initialisierung des Grafikdisplay
(8)	Anwendungsprogramm
(9)	*Include für Datei Glcd21_display3000.bas* (Pfad anpassen)
(10)	Include(s) für Bitmap-Grafik(en) incl. Pfadangabe
(11)	Include für Datei *Glcd21_fonts.bas* (Pfad anpassen)

Tabelle 5 Anwendungsspezifische Anpassungen am Template

5.3. GoldenGate

Im Programmbeispiel „GoldenGate“ sollen die Verwendung einer Bitmap-Grafik und die Textausgabe vorgestellt werden.

Listing 2 zeigt den gesamten Quelltext zum Vergleich mit dem Template. Wieder sind die angepassten Stellen im Quelltext fett markiert. Es ist deutlich ersichtlich, dass die erforderlichen Anpassungen wenig dramatisch sind. Hat man im Template die Pfadangaben (6), (9) und (11) auf sein jeweiliges Entwicklungsumfeld eingestellt, dann fallen an diesen Stellen die Anpassungen auch noch weg.

```
'#####################################################################################################
' Program sample for demonstration of opening screen and text message
' Goldengate.bas
' Display D072
' Claus Kühnel 2007-04-06
'#####################################################################################################

$hwstack = 64
$swstack = 128
$framesize = 16
'you might need to raise these numbers if your code grows

$regfile = "m128def.dat"
$crystal = 14745600             'enter the used clock of your actual microcontroller

'#####################################################################################################
'Application variables
Dim Txt$ As String * 20

'#####################################################################################################
'Definition of used ports and pull up resistors
'At our boards we are using Port B for the SPI-communication to the LCD.
'Now we need to select Port B as to an output port (data output to the display)
Ddrb = &B01110110
'DDR = Data direction register; Port B1, B2, B4, B5, B6 switched to output (1) as needed by the display
Portb = &B10001001      '... the other ports of Port B are inputs with switched on pull up resistors
```

```
Ddra = &B00000000       'switch all 8 Ports of Port A to input (0), Pin (PA.0 - PA.7)
Porta = &B11111111      'All port pins have individually selectable pull-up resistors.
                        'Here we enable these pull-up-resistors, so these Pins are always at logical 1
                       'You need to pull these Pins against ground (GND)
Ddrc = &B00000000       'switch all Ports of Port C to input
Portc = &B11111111      'all pull-up-Resistors turned on

Ddrd = &B00000000       'switch all Ports of Port D to input
Portd = &B11111111      'all pull-up-Resistors turned on

Ddre = &B00000000       'switch all Ports of Port E to input
Porte = &B11111111      'all pull-up-Resistors turned on

Ddrf = &B00000000       'switch all Ports of Port F to input
Portf = &B11111111      'all pull-up-Resistors turned on

Ddrg = &B00000000       'switch all Ports of Port G to input
Portg = &B11111111      'all pull-up-Resistors turned on

$include C:\Programme\BASCOM-AVR\mySamples\DisplaySamples\Includes\Init21_display3000.bas

'#####################################################################################################
'Initialisation of the display (needs to be done only for changes from default)
Orientation = Portrait180       'select neeeded orientation, here: Portrait180 mode
Graphics_mode = 65k_compressed  'select the needed color mode, here 65.536 colors
Gosub Lcd_init    'Initialisation routine of the display, Needs to be done only once at the beginning
Gosub Lcd_cls

'#####################################################################################################
'Main program
Do
   Restore T2a
   Call Lcd_bitmap(0 , 0 , 131 , 175)
   Wait 2

   Call Lcd_box(0 , 0 , 131 , 175 , Bright_yellow)
   Txt$ = "Visit"
   Call Lcd_print(txt$ , 10 , 70 , 2 , 1 , 2 , Blue , Bright_yellow)
   Txt$ = "Golden Gate"
   Call Lcd_print(txt$ , 10 , 100 , 2 , 1 , 2 , Blue , Bright_yellow)
   Wait 1
   Txt$ = "Visit"
   Call Lcd_print(txt$ , 10 , 70 , 2 , 1 , 2 , Bright_yellow , Bright_yellow)
   Txt$ = "Golden Gate"
   Call Lcd_print(txt$ , 10 , 100 , 2 , 1 , 2 , Bright_yellow , Bright_yellow)
Loop

End
'#####################################################################################################

$include C:\Programme\BASCOM-AVR\mySamples\DisplaySamples\Includes\Glcd21_display3000.bas

'-----------------------------------------------------------------------------------------------------
' Includes agraphics file shown at start up - you need to change the directory.
'-----------------------------------------------------------------------------------------------------
$inc T2a , Nosize , "C:\Programme\BASCOM-AVR\mySamples\Display3000\Grafik\Goldengate.bin"

'#####################################################################################################
$include C:\Programme\BASCOM-AVR\mySamples\DisplaySamples\Includes\Glcd21_fonts.bas
'Dummy Data um Fehlermeldungen bei der Kompilierung der Standardroutinen zu vermeiden
'Die Tabelle wird dann bei Nutzung eines indizierten Grafikdatei mit "echten" Daten ausgetauscht
Colortable:
Data 0
```

Listing 2 Quelltext Programmbeispiel „GoldenGate.bas“

In einer Endlosschleife wird das gemäß Abschnitt 4 vorbereitete Bild der „Golden Gate Bridge“ für zwei Sekunden angezeigt. Nach dem Löschen des Bildschirminhalts durch Ausgabe einer gelben Box mit den Massen des ganzen Bildschirms erfolgen zwei Textausgaben in blauer Schrift vor gelbem Hintergrund. Nach einer Sekunde werden die beiden Texte durch das Schreiben der gleichen Texte mit der Hintergrundfarbe an identischer Position gelöscht. Abbildung 8 zeigt die beiden Bildschirmausgaben des Programmbeispiels.

Das Fotografieren der Inhalte des Grafikdisplay ist nicht ganz trivial, wenn man Ergebnisse präsentieren will, die der Anzeige entsprechen. Durch die hier vorgenommene Darstellung in Grauwerten wird der Aufwand deshalb in Grenzen gehalten und die jeweilige Darstellung nicht unbedingt optimal sein. Wichtig ist an dieser Stelle, den Zusammenhang zum jeweiligen Programmbeispiel deutlich machen zu können.

Abbildung 8 Bildschirmausgaben des Programmbeispiels „GoldenGate.bas“

5.4. Boxes

Im Programmbeispiel „Boxes“ sollen die Darstellung von gefüllten Rechtecken und Textausgaben bei zufälliger Positionierung auf dem Bildschirm vorgestellt werden. Über die Tasten des Farbdisplay-Moduls wird die Anzeigedauer des jeweiligen Bildinhalts beeinflusst.

Im vorangehenden Programmbeispiel war das gesamte Listing des Quelltextes dargestellt. Aus Platzgründen wird im Folgenden darauf verzichtet. Es werden nur die anwendungsspezifischen Programmteile gezeigt (Listing 3). Der gesamte Quelltext ist aber auf der beiliegenden CD enthalten.

```
'##########################################################################################
' Program sample for box and text display
' Display D072
' Claus Kühnel 2007-04-04
'##########################################################################################

…

'##########################################################################################
'Application variables
Dim Xx1 As Byte , Xx2 As Byte , Yy1 As Byte , Yy2 As Byte
Dim Txt$ As String * 10
Dim Duration As Integer
```

```
Key_right Alias Pind.6
Key_left Alias Pind.5

…

'#########################################################################################################
'Initialisation of the display (needs to be done only for changes from default)
Orientation = Landscape                 'landscape mode
Graphics_mode = 65k_uncompressed        'select the needed color mode, here 65.536 colors
Gosub Lcd_init                          'Initialisation routine of the display
Gosub Lcd_cls

'#########################################################################################################
'Main program
Txt$ = "Test"
Duration = 100

Do
   If Key_right = 0 Then Duration = 100
   If Key_left = 0 Then Duration = 500

   Xx1 = Rnd(125) : Xx2 = Xx1 + 50              ' random values limited to byte by arguments
   Yy1 = Rnd(81) : Yy2 = Yy1 + 50
   Call Lcd_box(xx1 , Yy1 , Xx2 , Yy2 , Red)
   Waitms Duration
   Call Lcd_box(xx1 , Yy1 , Xx2 , Yy2 , White)

   Xx1 = Rnd(125) : Yy1 = Rnd(81)
   Call Lcd_print(txt$ , Xx1 , Yy1 , 1 , 3 , 2 , Yellow , Blue)
   Waitms Duration
   Call Lcd_print(txt$ , Xx1 , Yy1 , 1 , 3 , 2 , White , White )
Loop


End
…
```

Listing 3 Quelltextauszug Programmbeispiel „Boxes.bas“

Für den bereits skizzierten Programmablauf werden im Deklarationsteil die Variablen (xx1, yy1) und (xx2, yy2) als Eckpunkte des darzustellenden Rechtecks im Byteformat vereinbart. Die Stringvariable *Txt$* nimmt den auszugebenden Text auf und die Variable *Duration* bestimmt die Zeit des Bildwechsels in Millisekunden.

Über die Tasten D5 und D6 soll der Bildwechsel beeinflusst werden. Mit Hilfe des Alias geschieht die Zuordnung der Tasten in lesbarer Form. Abbildung 9 zeigt die Zuordnung der Tasten auf dem Farbdisplay-Modul D072.

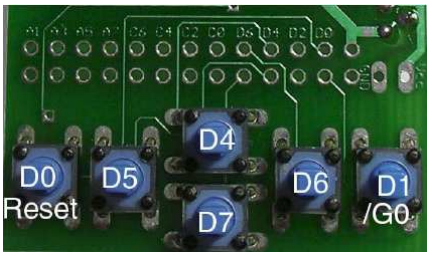


Abbildung 9 D072 Tastenfeld

Nach erfolgter Initialisierung des Farbdisplay-Moduls und der Variablen tritt das Programm in eine Endlosschleife.

Zuerst werden die beiden Tasten abgefragt, um die Variable *Duration* entsprechend zu setzen. Solange die Taste D5 nicht betätigt wurde, erfolgt der Wechsel des angezeigten Bildinhalts alle 100 ms.

Die Durchlaufzeit der Hauptschleife wird im Wesentlichen durch die vorgegebenen Wartezeiten (2 x Duration) bestimmt. Da die Tasten des Farbdisplay-Moduls nur einmal pro Schleifendurchlauf abgefragt werden, erfordert das gegebenenfalls eine etwas längere Betätigung der betreffenden Taste. Eine Alternative hierzu wird im letzten Programmbeispiel vorgestellt.

Abbildung 10 zeigt die beiden Bildschirminhalte (rote Box und blaues Testfeld mit gelber Schrift). Die Ausgabeorientierung war mit Landscape vorgewählt, wodurch die Orientierung des Textes und die Werte der x,y-Koordinaten bestimmt sind.

Die linke, obere Ecke der roten Box sowie die Koordinaten des Textfeldes werden mit der Random-Funktion von BASCOM-AVR bestimmt. Diese Random-Funktion liefert einen Rückgabewert im Integerformat (16 Bit). Da der Randomwert (durch die verwendeten Argumente) aber auf Byteformat beschränkt wird (das Hi-Byte bleibt 0), kann er problemlos einer Bytevariablen zugewiesen werden. Bei der Portierung eines solchen Programms muss dem aber ggf. Rechnung getragen werden.

Durch den wiederholten Aufruf der Random-Funktion werden die rote Box und die Textausgabe wild auf dem Farbdisplay „umherspringen“.

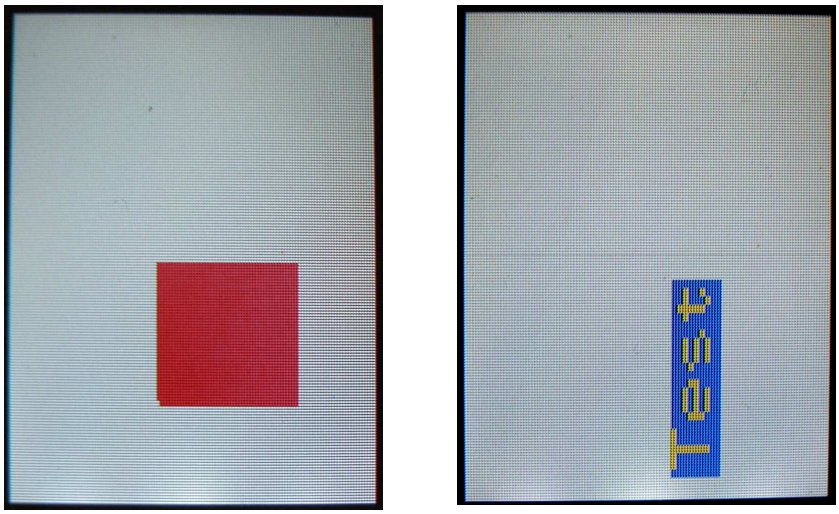


Abbildung 10 Bildschirmausgaben des Programmbeispiels „Boxes.bas

5.5. Anzeige von Werten in einem Liniendiagramm

Im Programmbeispiel „Plot“ wird die grafische Darstellung von Zahlenwerten in einem Liniendiagramm vorgestellt. Hierzu werden mit einem Pseudo-Zufallsgenerator die darzustellenden Werte erzeugt und als Originaldaten und gefilterte Daten zur Anzeige gebracht.

```
'#################################################################################################
' Program sample demonstrating plotting measuring data (simulated by pseudo-random numbers)
' Display D072
' Claus Kühnel 2007-04-03
'#################################################################################################

…

'#################################################################################################
'Application variables
Dim Idx As Byte
Dim Tmpx As Byte , Tmpy As Byte , Tmpxold As Byte , Tmpyold As Byte
```

```
Dim Value(128) As Byte
Dim Filtered_value(128) As Byte
Dim Mean As Word
Dim Txt$ As String * 20
Const Filter = 5

…

'####################################################################################################
'Initialisation of the display (needs to be done only for changes from default)
Orientation = Portrait180               'portrait180 mode
Graphics_mode = 65k_compressed          'select the needed color mode, here 65.536 colors
Gosub Lcd_init
Gosub Lcd_cls                           'clear screen (White)

'####################################################################################################
'Main program
' Print text
Txt$ = "Load array..."
Call Lcd_print(txt$ , 2 , 2 , 1 , 1 , 1 , White , Dark_blue)
Wait 2
Call Lcd_print(txt$ , 2 , 2 , 1 , 1 , 1 , White , White)

For Idx = 1 To 128
   Value(idx) = Rnd(128)                ' random value is limited to byte by argument 128
   Filtered_value(idx) = 0
Next

For Idx = 1 To 123
   Mean = Value(idx)
   Mean = Mean + Value(idx + 1)
   Mean = Mean + Value(idx + 2)
   Mean = Mean + Value(idx + 3)
   Mean = Mean + Value(idx + 4)
   Mean = Mean \ 5
   Filtered_value(idx) = Low(mean)
Next

Do
   Gosub Lcd_cls                                        'clear screen (White)

   ' Draw frame
   Call Lcd_rect(2 , 12 , 129 , 139 , 0 , Black)

   ' Print text
   Txt$ = "Show original data"
   Call Lcd_print(txt$ , 2 , 2 , 1 , 1 , 1 , White , Dark_blue)

   ' Draw line in frame
   Tmpxold = 2 : Tmpyold = 129
   For Idx = 1 To 128
      Tmpx = Idx + 1
      Tmpy = 139 - Value(idx)
      Call Lcd_draw(tmpxold , Tmpyold , Tmpx , Tmpy , 0 , Red)
      Tmpxold = Tmpx
      Tmpyold = Tmpy
'      Waitms 100
   Next
   Wait 2
   ' Clear text
   Call Lcd_print(txt$ , 2 , 2 , 1 , 1 , 1 , White , White)

   ' Print text
   Txt$ = "Add filtered data"
   Call Lcd_print(txt$ , 2 , 2 , 1 , 1 , 1 , White , Dark_blue)

   ' Draw line in frame
   Tmpxold = 2 : Tmpyold = 129
   For Idx = 1 To 128
      Tmpx = Idx + 1
```

```
      Tmpy = 139 - Filtered_value(idx)
      Call Lcd_draw(tmpxold , Tmpyold , Tmpx , Tmpy , 0 , Blue)
      Tmpxold = Tmpx
      Tmpyold = Tmpy
'      Waitms 100
   Next
   Wait 2

   ' Clear text
   Call Lcd_print(txt$ , 2 , 2 , 1 , 1 , 1 , White , White)
Loop

End
…
```

Listing 4 Quelltextauszug Programmbeispiel „Plot.bas“

Das Programmbeispiel „Plot.bas“ besteht neben dem Initialisierungsteil im Wesentlichen aus drei separaten Programmteilen:

1. Load array
2. Display original data
3. Add filtered data

Abbildung 11 zeigt diese drei Programmteile an Hand der zugehörigen Bildschirmausgaben.

Abbildung 11 Bildschirmausgaben des Programmbeispiels „Plot.bas“

Im ersten Teil (Load array) werden die beiden Bytearrays *Value(128)* und *Filtered_value(128*) mit Pseudozufallszahlen bzw. Null initialisiert. Anschließend werden die im Array *Value()* abgelegte Werte einer Mittelwertbildung unterzogen. Die so gefilterten Werte werden im Array *Filtered_value()* abgespeichert. Dieser Vorgang erfolgt einmalig nach dem Programmstart.

Die beiden nächsten Programmteile befinden sich in der Endlosschleife und werden deshalb ständig wiederholt.

Auf dem gelöschten Bildschirm wird ein Rechteck ausgegeben, welches den Ausgabebereich für die darzustellenden Messwerte markiert. Bevor die 128 darzustellenden Messwerte in einer Liniengrafik in Rot ausgegeben werden, folgt noch die Textausgabe „Display original data“ oberhalb der Grafik.

Nach einer Wartezeit von zwei Sekunden wird der ausgegebene Text gelöscht und das Programm tritt in die dritte Phase.

Die dritte Phase beginnt mit der Textausgabe „Add filtered data“. Es folgt die Ausgabe der gefilterten Daten als Liniengrafik in Blau. Nach zwei Sekunden Anzeigezeit wird der Text gelöscht und der Zyklus beginnt von neuem.

5.6. Anzeige Messwerte in einem Balkendiagramm

Im folgenden Programmbeispiel sind mehrere Funktionen umgesetzt. Im Einzelnen handelt es sich um:

- Analogwerteingabe an PinF.0 mit numerischer Anzeige des Zahlenwertes und grafischer Darstellung des Analogwertes in einer Balkenanzeige
- Anzeige von zwei weiteren Werten im Balkendiagramm
- Digitale Ansteuerung einer LED an PinD.5 zur Kennzeichnung des Schleifendurchlaufes
- Digitaler Ausgangsimpuls an PinD.6 zur Signalisation des Interrupts INT0
- Digitale Eingabe über 2 Tasten zur Auslösung eines Interrupts über PinD.0 (Int0) zur Auslösung einer Anzeigefunktion und Polling der an PinD.4 angeschlossenen Taste zum Löschen der durch Int0 initiierten Einblendung
- Serielle Ausgabe von Daten über die RS-232 (Pin TX)

Abbildung 12 zeigt die Beschaltung des Farbdisplay-Moduls D072 für die beschriebene Aufgabe.

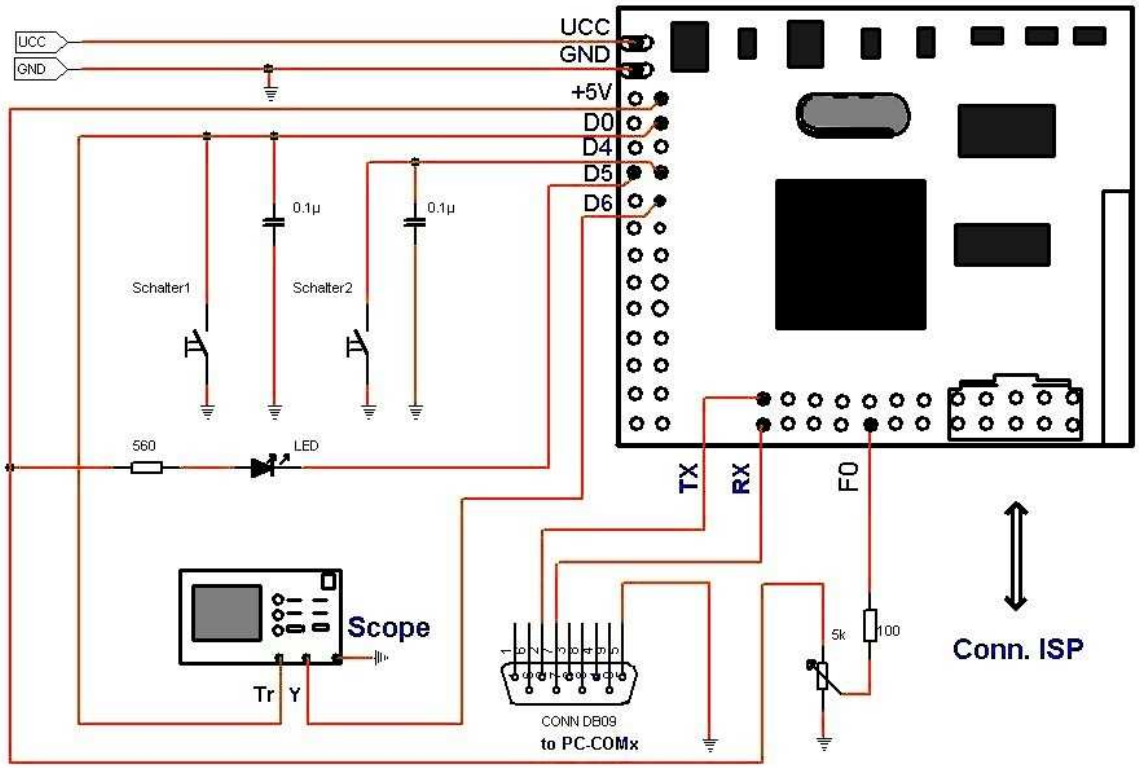


Abbildung 12
Beschaltung des Farbdisplay-Moduls zum Programmbeispiel „ADU-0_Beam4D.bas“

Zur Simulation der Analogwerteingabe wird ein Potentiometer als Spannungsteiler eingesetzt. Der Rest der Beschaltung weist keine Besonderheiten auf und kann dem Schaltbild entnommen werden.

Die Programmfunktionen selbst werden an Hand des Quelltextes zum Programmbeispiel „ADU-0_Beam4D.bas“ betrachtet (Listing 5).

```
'#########################################################################################
' Test ADU, Kanal 0 mit serieller Ausgabe an COM-Port und ext.Interrupt INT0 für
' zusätzliche Bildschirmanzeige
' Darstellung  des Messergebnisses auf Screen als Zahl und in zwei Balken
' Dritter Balken zur Darstellung eines mit jedem Schleifenumlauf inkrementierten Wertes
' Alle Balken mit Farbumschlag bei Überschreitung eines Anzeige - Grenzwertes
'                                                 16% Flash used
' klaus.zahnert@freenet.de                        26.Febr.07
'#########################################################################################

…

'#########################################################################################
'Application variables
Dim Incbeam As Byte                     'Counter value for incremented-in-loop lowest beam
Dim W As Word                           'ADC numeric result as word
Dim Xbeam As Byte                       'right x-position of beam (makes actual length of beam)
Dim S As String * 6                     'alphanumeric string displayed left in line
Dim Str_w As String * 3                 'W changed from word- to string-formatted
Dim Formstr_w As String * 3
Dim Walker_mark As Bit                  'Marker for INT0

…

'#########################################################################################
'Initialisation of the display (needs to be done only for changes from default) (7)
Orientation = Landscape                 'landscape mode
```

```
Graphics_mode = 65k_uncompressed        'select the needed color mode, here 65.536 colors
Gosub Lcd_init                          'Initialisation routine of the display,
Gosub Lcd_cls

'#######################################################################################################
'Main program

Config Pind.5 = Output , Pind.6 = Output        'PinD.4 for used polling key   set as input
                                                'PinD.5 for LED                set as output
                                                'PinD.6 for scope              set as output
Config Adc = Single , Prescaler = Auto
Start Adc

$baud1 = 9600                                   'select baud rate
Open "COM2:" For Binary As #1                   'open serial communication (RS232) to PC

Incbeam = 0                                     'preset value for lowest beam
Walker_mark = 0

''''''''''''''''''''''''''''''' INT0 settings '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
On Int0 Key_int0                                'INT0 calls his alligned routine
Enable Int0                                     'Enable Int0 from PortD.0
Enable Interrupts                               'Enable global interrupts
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Do                                              'LOOP
   If Pind.4 = 0 Then
      Call Lcd_box(1 , 53 , 176 , 132 , White)  'Clear Message from INT0
   End If

   W = Getadc(0)                                'Get ADC value, resolution 10 bit
   Shift W , Right , 3                          'scale W  (makes div 8)      1024 shiftr3==>128
   Print #1 , "IncBeam = " ; Incbeam ; "    ADU-Wert/8 = " ; W          'Output values to RS232

   Call Lcd_draw(1 , 0 , 175 , 0 , 0 , Black)   ' border line (uppest line)

'******   upper beam (W)  ***************************************************************************
   Call Lcd_draw(1 , 1 , 175 , 0 , 0 , Black)   'upper border line
   Xbeam = W + 48                               'calculate actual right x-pos. of beam
   Call Lcd_box(48 , 2 , 175 , 16 , White)      'clear actual text and beam, high = 15, length = max

   If W > 60 Then                               'decide for beam color green-red from XBeam-Value
      Call Lcd_box(48 , 2 , Xbeam , 16 , Red)
   Else                                         'write actual beam high = 15, length = ADU Wert/8
      Call Lcd_box(48 , 2 , Xbeam , 16 , Green)
   End If

   Str_w = Str(w)
   Formstr_w = Format(str_w , "000")
   S = "A=" + Formstr_w                         'String for alphanumeric display

   Call Lcd_print(s , 1 , 2 , 2 , 1 , 1 , Blue , Yellow)    'Display value at left position in line

'*****  middle beam  (128 - W)   ******************************************************************
   Call Lcd_draw(1 , 18 , 175 , 18 , 0 , Black) 'upper border line
   W = 128 - W                                  'used W now as residual from max-beam
   Xbeam = W + 48                               'calculate actual right x-pos. of beam
   Call Lcd_box(48 , 19 , 175 , 33 , White)     'clear actual text and beam, high = 15, length = max

   If W > 60 Then                               'decide for beam color green-red from XBeam-Value
      Call Lcd_box(48 , 19 , Xbeam , 33 , Red)
   Else                                         'write actual beam high = 15, length = ADU Wert/8
      Call Lcd_box(48 , 19 , Xbeam , 33 , Green)
   End If

   Str_w = Str(w)
   Formstr_w = Format(str_w , "000")
   S = "B=" + Formstr_w
   Call Lcd_print(s , 1 , 19 , 2 , 1 , 1 , Blue , Yellow)   'Display value at left position in line
```

```
'*****  lower  beam  (incremented from loop)   ***************************************************
   Call Lcd_draw(1 , 35 , 175 , 35 , 0 , Black) 'upper border line
   Incr Incbeam : If Incbeam = 128 Then Incbeam = 1
   W = Incbeam                                  'let's give W 1...127 as incremented value

   Xbeam = W + 48                               'calculate actual right x-pos. of beam
   Call Lcd_box(48 , 36 , 175 , 50 , White)     'clear actual text and beam, high = 15, length = max

   If W > 60 Then                               'decide for beam color green-red from XBeam-Value
      Call Lcd_box(48 , 36 , Xbeam , 50 , Red)
   Else                                         'write actual beam high = 15, length = ADU Wert/8
      Call Lcd_box(48 , 36 , Xbeam , 50 , Green)
   End If

   Str_w = Str(w)
   Formstr_w = Format(str_w , "000")
   S = "C=" + Formstr_w
   Call Lcd_print(s , 1 , 36 , 2 , 1 , 1 , Blue , Yellow)   'Display value at left position of line

   Call Lcd_draw(1 , 52 , 175 , 52 , 0 , Black) 'bottom border line

   Toggle Portd.5                               'Toggle LED

   If Walker_mark = 1 Then Gosub Walker         'If INT-0 then call subroutine to display image
Loop

Close #1                                        'Closing the RS232 channel
End

'#############################################################################################

'*****************   own subroutines alligned to main ***************************************
Key_int0:
   Walker_mark = 1                              'Interrupt from Key-Input
   Return

Walker:
   Walker_mark = 0                              'Reset Walker-Marker

   Pulseout Portd , 6 , 5                       'Single Pulse 10µsec for test by scope
   Restore T1a                                  'Restore picture memory
   Call Lcd_bitmap(1 , 66 , 68 , 127)           'Display image

   Call Lcd_print( "Keyed INT0: " , 70 , 95 , 2 , 1 , 1 , Blue , Yellow)        'Text output
   Call Lcd_print( "Stay cool ! " , 70 , 110 , 2 , 1 , 1 , Blue , Yellow)
   Call Lcd_draw(70 , 128 , 175 , 128 , 1 , Red)           'Red double line

   Return

'#############################################################################################
$include C:\Programme\BASCOM-AVR\mySamples\DisplaySamples\Includes\Glcd21_display3000.bas
'-------------------------------------------------------------------------------------------
' Includes a graphics file
'-------------------------------------------------------------------------------------------
$inc T1a , Nosize , "C:\Programme\BASCOM-AVR\mySamples\Display3000\Grafik\BACKPACK62x65A.bin"

…
```

Listing 5 Quelltextauszug Programmbeispiel „ADU-0_Beam4D.bas“

Für die Ausgabe ist im Grafik-Initialisierungsteil der Landscape Mode vereinbart. Abbildung 13 zeigt den Bildschirminhalt in der betreffenden Orientierung.

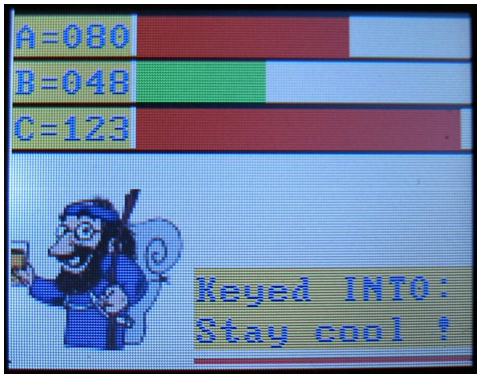


Abbildung 13 Bildschirmausgaben des Programmbeispiels „ADU-0_Beam4D.bas“

Bevor das Programm in die Endlosschleife eintritt, werden noch die eingesetzten I/O-Pins konfiguriert. PinD.5 und PinD.6 dienen als Ausgang, während PinD.0 und PinD.4 als Eingang konfiguriert werden.

Im Auslieferungszustand des D072 Farbdisplay-Moduls ist die Taste D0 mit Reset verbunden. Durch Trennen der Brücke *Reset* und Verbinden der Brücke *D0* mit etwas Zinn ist die Taste mit PinD.0 verbunden. Abbildung 14 zeigt den erforderlichen Eingriff am D072 Farbdisplay-Modul.

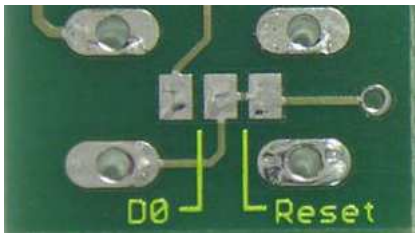


Abbildung 14 Umschaltung PinD.0

Der über PinF.0 zugängliche Kanal0 des internen AD-Umsetzers wird konfiguriert und gestartet bevor die zur Verfügung stehende serielle Schnittstelle mit einer Baudrate von 9600 Baud initialisiert und geöffnet wird.

Durch die Betätigung der Taste D0 (PinD.0) soll der Interrupt INT0 ausgelöst werden. Diesem Interrupt wird die Interrupt-Serviceroutine ISR Key_int0 zugewiesen. Bevor der Interrupt selbst und auch der globale Interrupt freigegeben werden.

Zu Beginn der Endlosschleife erfolgt die Abfrage von PinD.4 (Polling). Bei Betätigung der Taste D4 wird der Bereich des Bildschirms gelöscht, der im Falle eines INT0 Interrupts beschrieben wurde.

Die nächste Programmaktivität besteht im Auslesen des letzten Resultats der AD-Umsetzung und Start einer neuen durch die Instruktion *W = Getadc(0)*. Genau an diesen Kanal0 (PinF.0) hatten wir das Potentiometer angeschlossen.

Der mit 10 Bit- Auflösung vorliegende Wert wird durch Acht geteilt (0…1023 -> 0… 127) und in der Variablen *W* abgespeichert. Die Ausgabe der Werte *Incbeam* und *W* über RS232 erfolgt zu Kontrollzwecken. Abbildung 15 zeigt einen Ausschnitt einer Aufzeichnung der Datenausgabe mit dem Programm Hyperterminal auf einem über RS232 angeschlossenen PCs.

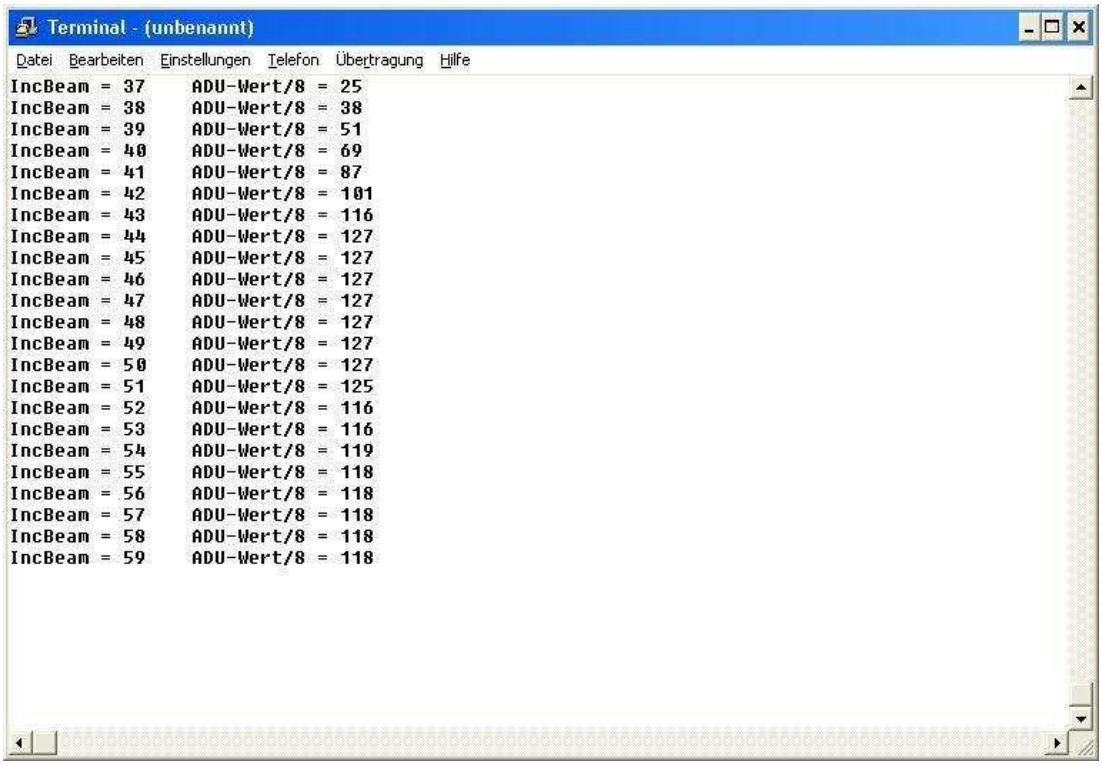


Abbildung 15 Datenausgaben am Terminal

Im weiteren Programmablauf folgen drei praktisch identische Abschnitte zur Darstellung der drei Balken.

Im oberen Balken wird der vom AD-Umsetzer über Kanal0 ermittelte Messwert skaliert auf einen Bereich auf kleiner 128 als Balken ausgegeben. Für Werte über 60 erfolgt die Darstellung des Balkens in Rot, sonst in Grün. An der linken Seite des Balkens erfolgt eine numerische Ausgabe des Messwerts gekennzeichnet mit „A=xxx“.

Im mittleren Balken erfolgt die Darstellung des gegenläufigen Wertes (128 – ADC-Wert) nach den gleichen Kriterien. An der linken Seite des Balkens erfolgt eine numerische Ausgabe des Messwerts gekennzeichnet mit „B=yyy“.

Im unteren Balken wird ein bei jedem Schleifendurchlauf inkrementierter Wert, nach ansonsten gleichen Kriterien, dargestellt. An der linken Seite des Balkens erfolgt eine numerische Ausgabe dieses Inkrements gekennzeichnet mit „C=zzz“.

Kurz vor Ende der Hauptschleife des Programms wird noch PinD.5 getoggelt und dadurch die angeschlossene LED umgeschaltet. Dieses Blinken signalisiert die wiederholten Schleifendurchläufe.

Drückt man nun zu einem beliebigen Zeitpunkt auf Taste D0, dann erfolgt ein Interrupt INT0. Die zugehörige ISR *Key_int0* setzt die Bitvariable *Walker_mark*.

Am Ende der Hauptschleife des Programms wird genau diese Variable noch abgefragt und so ein erfolgter Interrupt INT0 detektiert.

War die Taste D0 auch nur kurzzeitig betätigt worden, dann sorgt der Aufruf der Subroutine *Walker* für das Rücksetzen der Variablen *Walker_mark* und einen kurzen Impuls von 10 µs an PinD.6, der z.B. mit einem Oszilloskop erfasst werden kann. Anschließend wird das Bild des Wandersmanns in einem Bereich außerhalb des Balkendiagramms gefolgt von einigen Textausgaben dargestellt.

Die Triggerung des Oszilloskops erfolgt mit Schalter S1 zum Zeitpunkt des Interrupts. Der Y-Puls von PinD.6 ist ein Impuls kurz vor dem Bildaufbau (siehe Subroutine Walker) zur Dar-

stellung der Reaktionszeit, die, je nach relativer zeitlicher Lage der Interrupts zum Schleifenablauf, auf die Zeit eines Schleifendurchlaufs statistisch verteilt ist.

Die durch einen Interrupt behandelte Tasteneingabe bedarf nicht einer längeren Betätigung der Taste sondern setzt die Bitvariable umgehend. Die resultierende Verzögerung der Anzeige des Bildes kommt nur durch die verzögerte Abfrage der Bitvariablen zustande. Will man diese Verzögerung reduzieren, dann muss die Abfrage (*If Walker_mark = 1 Then Gosub Walker*) nur an mehreren Orten innerhalb der Hauptschleife des Programms eingefügt werden.

6. Schlussbemerkung

Im vorliegenden Beitrag wurde einige Bespiele für den Einsatz eines Farddisplay-Moduls gezeigt. Kenntnisse der AVR-Mikrocontroller und der eingesetzten Programmierumgebung BASCOM-AVR vorausgesetzt, ist der Einstieg recht einfach.

Der C-Programmierer findet in Lieferumfang des Farbdisplay-Moduls vergleichbare Beispielprogramme für die GNU-Umgebung WinAVR.

7. Literatur

[1] Kühnel, C.:

Programmieren der AVR RISC Mikrocontroller mit BASCOM-AVR.
Eine Einführung anhand von Programmbeispielen.

2. Auflage, 2004

ISBN 3-907857-04-6

8. Links

Speed IT up Homepage	http://www.display3000.com
BASCOM-AVR	http://www.mcselec.com
Author's Homepage	http://www.ckuehnel.ch