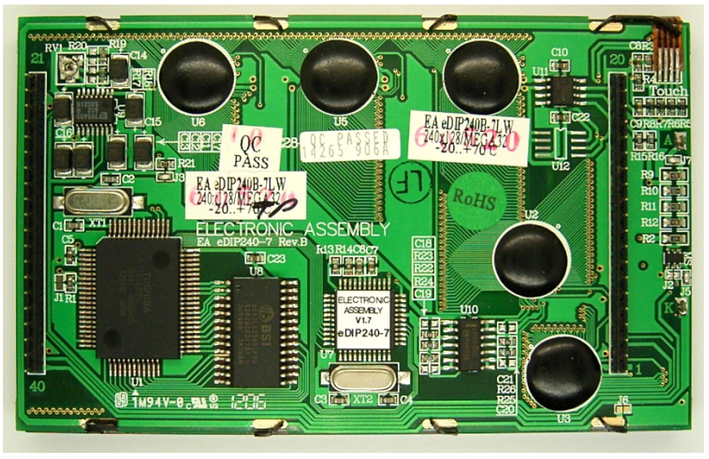
Steuerplatine BE-Display für das Touch-Display eDIP 240-7

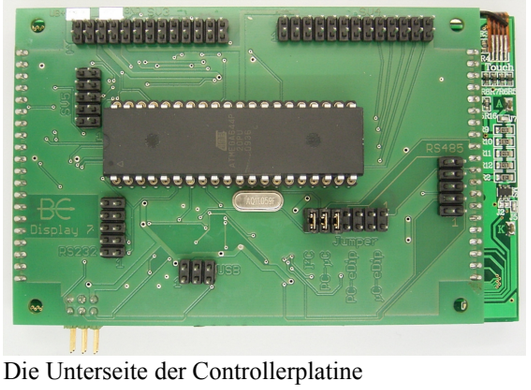
Professionelle Geräte der Mess- und Regeltechnik werden oft mit grafikfähigen Displays ausgestattet. Mit dem eDIP 240-7 hat man ein Display mit 240 mal 128 Punkten bei einer Größe 113 mm * 70 mm, das darüber hinaus auch noch über ein Touch-Screen-Bedienfeld verfügt.

Das eDIP 240-7

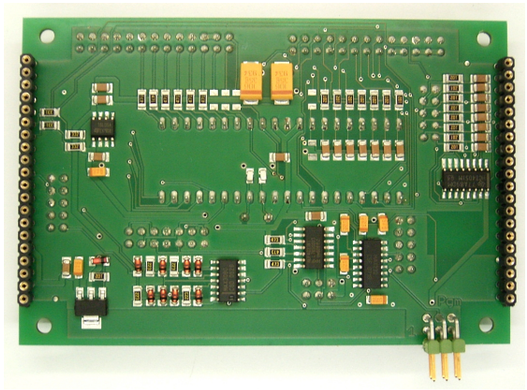


Unterseite des eDIP 240-7

Die hier vorgestellte Steuerplatine von Bürger-Elektronik wird über AK Modul-Bus vertrieben und erleichtert den Umgang mit dem Display erheblich. Auf dem Board befindet sich ein ATmega644P mit zahlreichen Schnittstellenanschlüssen. Das eDIP240-7 enthält zwar bereits einen Mikrocontroller, der alle Display-Funktionen zur Verfügung stellt. Die eigentliche Anwendung erfordert jedoch einen Hauptcontroller. Das Display kann wahlweise über eine RS232 oder über den I2C-Bus angesteuert werden. Beide Schnittstellen sind auf dem Display-Board vorbereitet und können über Jumper gewählt werden. In den meisten Fällen ist die I2C-Steuerung günstiger, weil damit die serielle Schnittstelle des Mega644P frei bleibt. In diesem Fall muss der I2C-Jumper gesetzt sein.



Die Unterseite der Controllerplatine

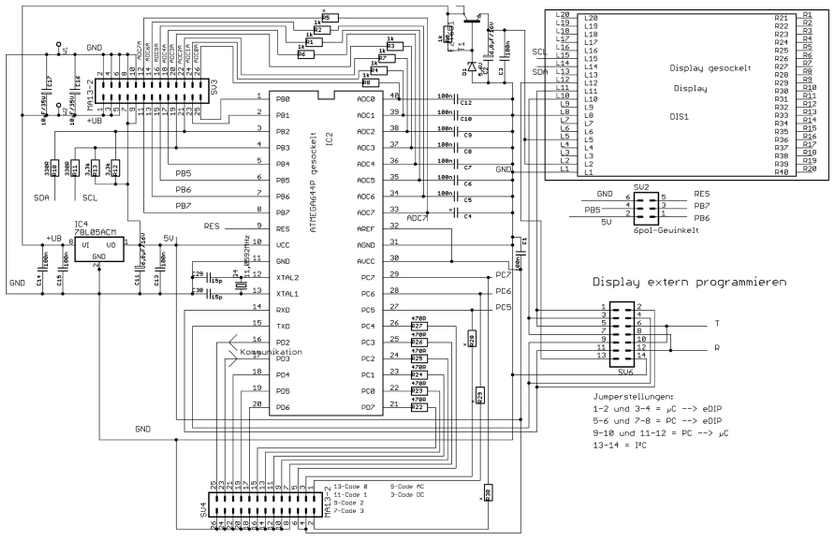


Die Oberseite mit Anschlussleisten für das Display

Das Board verfügt über einen 5-V-Spannungsregler und wird extern mit 9 … 12 V versorgt. Ein ISP-Stecker zur Programmierung des Controllers z.B. über das STK500 ist seitlich herausgeführt, sodass Software problemlos getestet werden kann. Auf der Platine befinden sich ein MAX232 und Anschlüsse für eine serielle Schnittstelle, eine RS485 und eine externe USB-Schnittstelle. Über zwei Jumper auf dem Pfostenstecker BU1 kann die Verwendung der seriellen Schnittstelle eingestellt werden. Dabei gibt es drei Möglichkeiten:

1. Der Mega32 steuert das Display über die RS232
2. Der PC erhält eine direkte Verbindung zum Display, um es zu programmieren
3. Der Mega32 kommuniziert mit dem PC

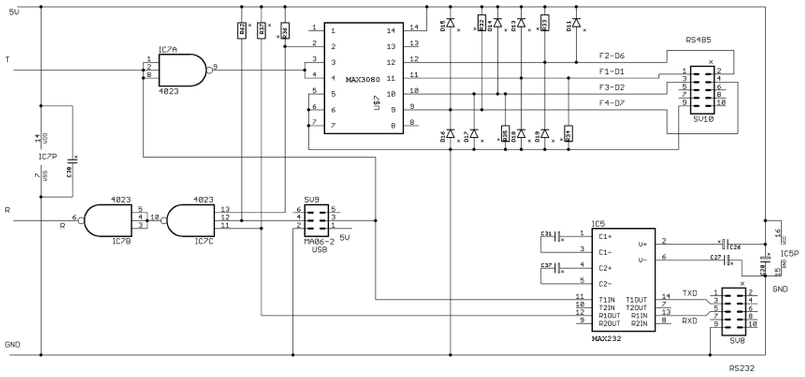
Zusätzlich gibt es auf der Platine einen Jumper (I2C), der die Schnittstelle des Displays festlegt: RS232 (offen) oder I2C (geschlossen).



Der Schaltplan der Steuerplatine

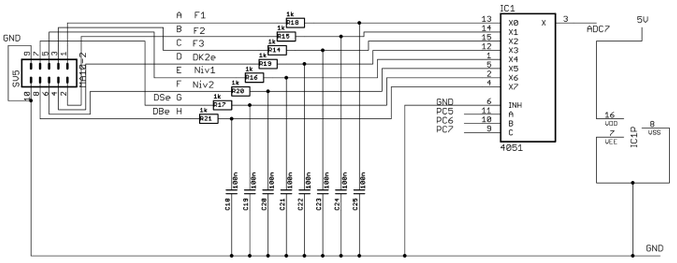
Die Serielle Schnittstelle verfügt über drei Anschlüsse nach außen:
- RS232-Schnittstelle zum direkten Anschluss an eine DB9-Buchse
- RS485-Schnittstelle mit einem MAX3088 für Bus-Anwendungen
- USB-Schnittstelle über einen extern anzuschließenden FT232R

Eine logische Verknüpfung der seriellen Signale erlaubt einen gleichzeitigen Anschluss mehrerer Schnittstellen. So könnte man z.B. ein Gerät mit USB- und RS232 betreiben, wobei der Anwender ohne spezielle Umschaltung zwischen den Schnittstellen wählen kann.



Die Schnittstellen

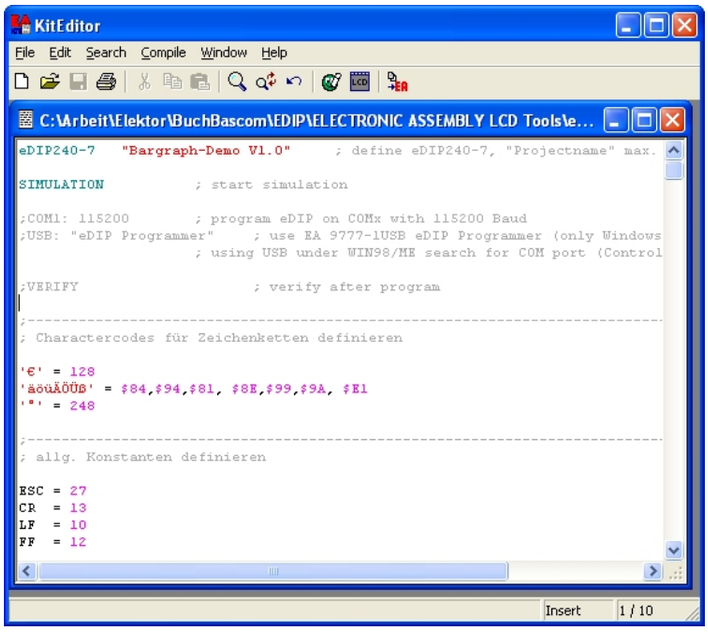
Die häufigste Anwendung der Platine liegt im Bereich Messdatenerfassung in Prüf- und Steuergeräten. Dazu wird häufig eine große Anzahl analoger Eingänge benötigt. Der Mega644P verfügt zwar bereits über acht analoge Eingänge. Mit einem Multiplexer 4051 wird jedoch der Kanal ADC7 auf acht zusätzliche Eingänge umgeschaltet. Insgesamt hat das Gerät damit 15 analoge Eingänge. Alle weiteren noch freien Ports des Mega664P sind an externe Anschlüsse gelegt und können für beliebige Aufgaben eingesetzt werden.



Multiplexer für acht weitere Kanäle

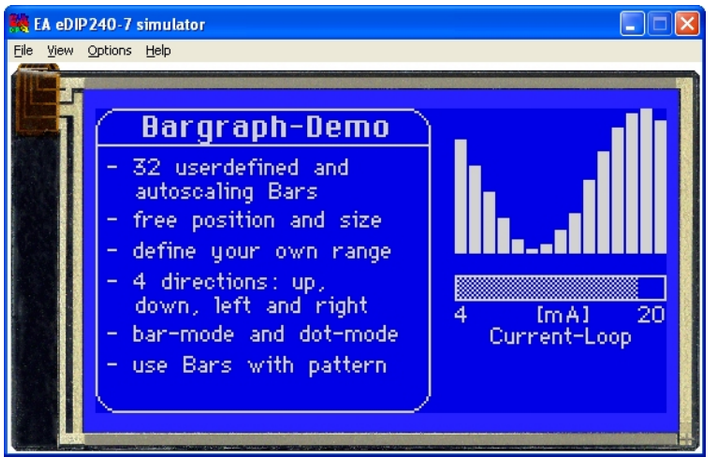
Einschaltbildschirm

Electronic Assembly stellt eine umfangreiche Software zur Programmierung des Displays bereit, die auch von der CD installiert werden kann. Es lohnt sich die verschiedenen Beispiele in den Kit-Editor zu laden und anzusehen, um die Möglichkeiten kennenzulernen. Laden Sie z.B. d ein Beispiel zur Darstellung einer Bargraph-Anzeige in den Editor. Die verschiedenen Elemente werden hier zwar direkt programmiert, sie können jedoch in gleicher Weise auch in einem Bascom-Programm festgelegt werden, das dann die entsprechenden Befehlssequenzen an das Display sendet.



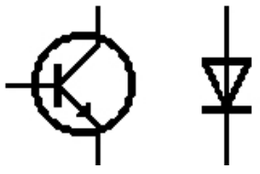
Ein Demo-Programm im Editor

Im Quelltext steht die Anweisung „SIMULATION“. Ein Klick auf die Compile-Schaltfläche startet deshalb den Simulator. Verwendet man stattdessen die auskommentierte Zeile „COM1: 115200“, wird das Programm seriell in die Hardware übertragen.



Das Demo-Programm im Simulator

In einer typischen Anwendung zeigt das Display beim Einschalten ein Firmenlogo oder eine Gerätebezeichnung. Diese Texte und Grafiken erfordern noch nicht die Mitarbeit des Mega664, sondern können durch Initialisierung bzw. direkte Programmierung des Displays festgelegt werden. Die erforderlichen Arbeitsschritte sollen hier an einem Beispiel erläutert werden. Das Gerät soll z.B. als Komponententester zur Prüfung von Transistoren und Dioden verwendet werden. Dazu kann ein entsprechendes Logo als Grafik eingesetzt werden. Zusätzlich gibt es eine Gerätebezeichnung als Textüberschrift. Textgrößen, Fonds und sonstige Eigenschaften können frei gewählt werden. Hier wird zunächst ein Schwarzweißbild im BMP-Format benötigt.



Die Grafik für den Schirm

Nach dem Muster der Beispielanwendungen kann nun der Startbildschirm programmiert werden. Das Ergebnis ist der Quelltext Transistor.kmc. Die Funktion kann mit dem Simulator überprüft werden.

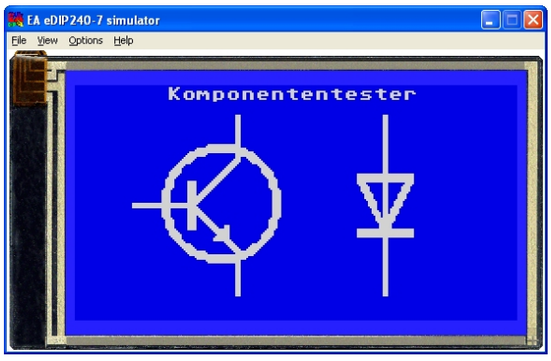
```
eDIP240-7   "Transistor"     ; define eDIP240-7, "Projectname" max. 30 character

SIMULATION  ; start simulation

;COM1: 115200     ; program eDIP on COMx with 115200 Baud
;USB: "eDIP Programmer" ; use EA 9777-1USB eDIP Programmer (only Windows 2000/XP)
; using USB under WIN98/ME search for COM port (Control Panel/Gerätemanager)

;VERIFY   ; verify after program

;------------------------------------------------------------
; Charactercodes für Zeichenketten definieren

'€' = 128
'äöüÄÖÜß' = $84,$94,$81, $8E,$99,$9A, $E1
'°' = 248

;------------------------------------------------------------
; allg. Konstanten definieren

ESC = 27
CR  = 13
LF  = 10
FF  = 12

AUS =  0
```

```
EIN =  1

XMAX = 239
YMAX = 127

; Verknüpfungsmodi
SET     = 1
DELETE  = 2
INVERS  = 3
REPLACE = 4
INVREPL = 5

; Blinkmodi
NOBLINK     = 0
BLINKONOFF  = 1
BLINKINVERS = 2

;-------------------------------------------------------
; Fonts einbinden  max. 15 eigene Fonts (FontNr. 1..15)

FONT8x8  = 0            ; interner Font, immer vorhanden

FONT4x6  = 1
FONT6x8  = 2
FONT7x12 = 3

GENEVA10  = 4
CHICAGO14 = 5
SWISS30B  = 6
BIGZIF57  = 7


;=======================================================
; Makro Nummern
MnPowerOn = 0
MnStart = 10

;=======================================================

; Bilder laden
      path <>
      BmpTransisitor = 100
      Picture:  BmpTransisitor     <Transistor.bmp>


;=======================================================

PowerOnMakro:           ; wird nach dem Einschalten ausgeführt
      #MN MnPowerOn

ResetMakro:       ; wird nach einem externen Reset ausgeführt
      #MN MnPowerOn

WatchdogMakro:    ; wird nach einem Fehlerfall/Timeout >500ms ausgeführt
      #MN MnPowerOn

BrownOutMakro:    ; wird nach einem Spannungseinbruch <4V ausgeführt
      #MN MnPowerOn

;=======================================================
MAKRO: MnPowerOn            ; Power-On/Reset Makro
      #TA               ; Terminal aus
      #MN MnStart       ; Hauptmenu anzeigen

;=======================================================
MAKRO: MnStart            ; Power-On/Reset Makro
      #TA               ; Terminal aus

      #UI 30,15,BmpTransisitor
```

```
#AI AUS           ; Touch Invertieren aus
#AS EIN           ; Summer ein

#ZF FONT8x8
#ZZ 1,1
#ZL 50,1 "Komponententester"
```

Festlegen eines Startbildschirms mit dem Kit-Editor

Im Quelltext findet sich der Eintrag `SIMULATION`. Deshalb startet nach dem Kompilieren der Simulator und zeigt den Bildschirmaufbau. So kann man leicht kleine Änderungen austesten.



Der Startbildschirm im Simulator

Wenn das Programm in das Display übertragen werden soll, muss die Zeile `;COM1: 115200` aktiviert werden, indem das Kommentarzeichen entfernt wird. Die Simulation muss entsprechen auskommentiert werden. Die Jumper auf der Steuerplatine müssen in Stellung „PC-eDip“ gebracht werden. Die serielle Schnitstellle ist dann über den MAX232 mit dem Display verbunden. Nun wird das Programm nach dem Kompilieren in das Display geladen. Der Startbildschirm erscheint nun nach jedem Neustart und beleibt so lange stehen, bis er durch einen Löschbefehl vom Anwenderprogramm im Mega644 gelöscht wird.

Man könnte nun mit dem Editor weitere Elemente ausprobieren und simulieren, die später in gleicher Weise vom BASCOM-Programm aus gesteuert werden. Tatsächlich hilft oft ein Blick in die vorhandenen Beispiele, wenn bestimmte Elemente des Displays genutzt werden sollen. Eine genaue Definition aller Funktionen und ihrer Parameter findet sich auch im Datenplatt des Displays auf der CD.

Hello World

Das erste Programmbeispiel soll einfach nur einen kurzen Text auf dem Display ausgeben. Dabei wird ein Datenpuffer mit Kommandos und Daten über die Sub Write_edip in das Dislpay übertragen. Die einzelnen Kommandos findet man im Datenblatt des eDIP240-7. Man kann wahlweise über die serielle Schnittstelle oder über den I2C-Bus (Adresse &HDC beim Schreiben bzw. &HDD beim Lesen) mit dem Display kommunizieren. Hier wird der I2C-Bus verwendet, um die serielle Schnittstelle des Mega644P für andere Aufgaben frei zu halten. Der I2C-Bus liegt an den Leitungen B2 (SDA) und B3 (SCL).

Eine Textausgabe

Die entscheidende Basisroutine zur Ansteuerung in Bascom ist die Sub Write_edip. Es wird jeweils zuerst ein Byte 17 übertragen. Dann folgt die Anzahl der zu übertragenen Bytes die in Buffer übergeben werden. Darauf folgen die einzelnen Nutzbytes und schließlich die Prüfsumme Bcc. Nach dem I2Cstop muss der Bus noch einmal in Leserichtung geöffnet werden, um ein Acknowledge-Byte vom Display zu lesen. Man könnte hier auswerten, ob die Übertragung erfolgreich war. Das ist jedoch im Normalfall nicht nötig, sondern man kann davon ausgehen, dass die Daten korrekt angekommen sind.

Hier werden Beispiele für das Löschen des Displays (Kommando DL), die Einstellung der Hintergrundbeleuchtung (YH) und die Auswahl des Zeichensatzes (ZF) gezeigt. Das ZL-Kommando gibt einen Text an der Stelle x,y aus. Jedes Kommando wird mit einem ESC-Byte (27) eingeleitet und zusammen mit allen erforderlichen Parametern in den Textpuffer Buffer geschrieben, Im Beispiel wird ein Text in die Mitte des Displays geschrieben.

```
'-----------------------------------------------------
' eDIP240-7
' ATMEGA644P, 11,0592 MHz
' I2C-Display
' edip1.bas
'-----------------------------------------------------

$regfile = "m644Pdef.dat"
$crystal = 11059200
$hwstack = 64
$swstack = 64
$framesize = 64

Dim Dat As Byte
Dim Sbuf As Byte
Dim Rbuf As Byte
Dim Bcc As Byte
Dim C As String * 1
Dim N As Integer
Dim Buffer As String * 24

Declare Sub Write_edip

Portb = 255                                                 'Pullups

Config Sda = Portb.2
Config Scl = Portb.3

Const Slave = &HDC                 'I2C write DC
Const Slaverd = &HDD               'I2C read DD

I2cinit
  Buffer = Chr(27) + "YH" + Chr(         '80% brightness
  Write_edip
  Buffer = Chr(27) + "DL" + Chr(13)      'clear
  Write_edip
  Buffer = Chr(27) + "ZF" + Chr(2) + Chr(13)  '2 = FONT6x8
  Write_edip
  Buffer = Chr(27) + "ZL" + Chr(100) + Chr(60) + "eDIP240-7"
                                                + Chr(13)
  Write_edip

Do
Loop



'********************** eDIP240-7 ************************

Sub Write_edip
  I2cstart
  I2cwbyte Slave
  Dat = 17
  I2cwbyte Dat
  N = Len(buffer)
  Bcc = 17 + N
  I2cwbyte N
  For N = 1 To Len(buffer)
```

```
    C = Mid(buffer , N , 1)
    Dat = Asc(c)
    If Dat = 255 Then Dat = 1
    I2cwbyte Dat
    Bcc = Bcc + Asc(c)
  Next N
  I2cwbyte Bcc
  I2cstop
  Waitms 1
  N = 0
  I2cstart
  I2cwbyte Slaverd
  I2crbyte Dat , Nack
  I2cstop
  N = N + 1
  Waitms 1
End Sub
End
```

Ausgabe eines Textes (Edip1.bas)

8.4 Touch-Felder

Das eDIP240-7 ist mit einem Touch-Feld ausgestattet, das komplett unter der Kontrolle des Displaycontrollers steht. Der Programmierer kann Schaltflächen festlegen, die mit einem Rahmen und einen Test angezeigt werden. Wenn ein Anwender diese aktive Fläche mit dem Finger oder einem Zeigestift berührt, wird eine Nachricht generiert, auf die ein Programm reagieren kann.

Zwei aktive Schaltflächen

Mit dem AT-Kommando erzeugt man eine Schaltfläche. Als Parameter folgen die XY-Koordinaten für die aktive Fläche, ein Tastencode, ein Stoppcode und der gewünschte Text zur Beschriftung der Taste. Im Beispiel Edip2.bas werden zwei Tasten mit der Beschriftung „+1“ und „-1“ gebildet. Ihnen sind die Tastencodes 121 und 122 zugeordnet. Genau diese Bytes werden später empfangen, wenn jemand auf die entsprechenden Tasten drückt. Hier soll ein Zählerstand in der Variablen Count entsprechend erhöht und verkleinert werden, um ihn dann mit dem ZL-Kommando anzuzeigen.

```
'-----------------------------------------------------------
' eDIP240-7
' ATMEGA644P, 11,0592 MHz
' I2C-Display
' Edip2.bas
'-----------------------------------------------------------

$regfile = "m644Pdef.dat"
$crystal = 11059200
$hwstack = 64
$swstack = 64
$framesize = 64

Dim Dat As Byte
Dim Sbuf As Byte
Dim Rbuf As Byte
Dim Bcc As Byte
Dim Touchcode As Byte
Dim Endcode As Byte
```

```
Dim C As String * 1
Dim N As Integer
Dim Buffer As String * 24

Dim Count As Integer

Declare Sub Write_edip
Declare Sub Read_edip
Declare Sub Buffer_edip

…

  Buffer = Chr(27) + "DL" + Chr(13)        'clear
  Write_edip
  'Touch button at 10,10;60,30, Touchcode 121, Endcode = 250
  Buffer = Chr(27) + "AT" + Chr(10) + Chr(10) + Chr(60)
       + Chr(30) + Chr(121) + Chr(250) + "+ 1" + Chr(13)
  Write_edip
  'Touch button at 70,10;120,30, Touchcode 122, Endcode = 250
  Buffer = Chr(27) + "AT" + Chr(70) + Chr(10) + Chr(120)
            + Chr(30) + Chr(122) + Chr(250) + "- 1" + Chr(13)
  Write_edip
  Buffer = Chr(27) + "ZF" + Chr(2) + Chr(13)    '2 = FONT6x8
  Write_edip
  Buffer = Chr(27) + "ZL" + Chr(100) + Chr(60) + "eDIP240-7"
                                                  + Chr(13)
  Write_edip


Do
  Read_edip
  If Touchcode = 121 Then Count = Count + 1
  If Touchcode = 122 Then Count = Count - 1
  Buffer = Chr(27) + "ZL" + Chr(180) + Chr(15) + Str(count)
                                          + "  " + Chr(13)
  Write_edip
  Waitms 300
Loop


…

Sub Buffer_edip
  I2cstart
  I2cwbyte Slave
  Dat = 18                                    'DC2
  I2cwbyte Dat
  N = 1                                       '1 Byte
  Bcc = 18 + N
  I2cwbyte N
  'Waitms 1
  C = "I"
  Dat = Asc(c)
  Bcc = Bcc + Dat
  I2cwbyte Dat                                '"I"
  I2cwbyte Bcc
  I2cstop
  Waitms 1
  I2cstart
  I2cwbyte Slaverd
  I2crbyte Dat , Ack                          '06=ACK
  I2crbyte Dat , Ack                          '18=DC2
  I2crbyte Dat , Ack                          '2=2 Bytes
  I2crbyte Sbuf , Ack                         'Send Buffer
  I2crbyte Rbuf , Ack                         'Receive Buffer
  I2crbyte Dat , Nack                         'BCC
  I2cstop
  Waitms 1
End Sub
```

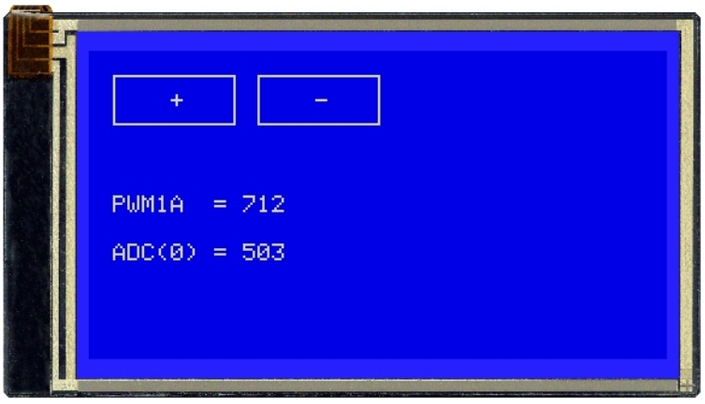
```
Sub Read_edip
  Touchcode = 0
  Buffer_edip
  If Sbuf > 0 Then
    I2cstart
    I2cwbyte Slave
    Dat = 18                                    'DC2
    I2cwbyte Dat
    N = 1                                       '1 Byte
    Bcc = 18 + N
    I2cwbyte N
    C = "S"
    Dat = Asc(c)
    Bcc = Bcc + Dat
    I2cwbyte Dat                                 '"I"
    I2cwbyte Bcc
    I2cstop
    I2cstart
    I2cwbyte Slaverd
    I2crbyte Dat , Ack                          '06=ACK
    I2crbyte Dat , Ack                          '17=DC1
    I2crbyte Dat , Ack                          '4...
    I2crbyte Dat , Ack                          'bufferdata
    I2crbyte Dat , Ack                          'bufferdata
    I2crbyte Dat , Ack                          'bufferdata
    I2crbyte Dat , Ack                          'bufferdata
    If Dat > 100 Then Touchcode = Dat       'Touchcode > 100
    I2crbyte Dat , Nack                         'BCC
    I2cstop
  End If
End Sub
End
```

Schaltflächen (Edip2.bas)

Die entscheidende Sub zum Auslesen eines eventuell vorhandenen Tastencodes ist Read_edip. Am Anfang wird über die Sub Buffer_edip abgefragt, wie viele Zeichen zum Abholen bereitstehen. Nur wenn der Display-Buffer gefüllt ist, wird er ausgelesen. An der siebten Stelle findet man den gesuchten Tastencode. Die globale Variable Touchcode liefert diese Information an das Hauptprogramm.

8.5 Berührungsdauer auswerten

Im den meisten Fällen wird jeweils eine Berührung der Touch-Fläche einem Ereignis zuordnen. Es kann jedoch auch sinnvoll sein, die Dauer eines Tastendrucks auszuwerten. Hier soll eine PWM-Ausgangsspannung über zwei Tasten eingestellt werden. Solange jemand auf eine Taste drückt, wird der Ausgangswert kontinuierlich erhöht bzw. verkleinert. Ein längerer Tastendruck führt zusätzlich zu einer Vergrößerung der Änderungsgeschwindigkeit. So kann der gesamte PWM-Bereich zwischen 0 und 1023 bequem eingestellt werden. Auf dem Display erscheint neben dem PWM-Ausgabewert auch ein Analogwert vom Eingang ADC0.



Verstellen eines analogen Ausgangswerts

Die Ansteuerung verwendet die bereits vorhandenen Display-Subs. Neu hinzugekommen ist eine erweiterte Abfrage mit der Sub Read_edip_repeating. Bei jedem Aufruf wird der gleiche Tastencode zurückgeliefert, solange die Taste noch gedrückt ist. Erst beim Loslassen einer Taste ändert sich der Code auf den Wert Null. Die Tastencodes müssen hierbei so festgelegt werden, dass der Tastencode über 100 und der Endecode über 200 liegt.

Bei einem längeren Tastendruck wird die Änderungsgeschwindigkeit des Ausgabewerts kontinuierlich gesteigert. Dazu wird die Wartezeit bis zur nächsten Anzeige und Abfrage von anfangs 200 ms allmählich bis auf 30 ms reduziert. Außerdem ändert sich nach einiger Zeit die Schrittweite von anfangs 1 auf 5 und schließlich auf 10. So kann man sowohl eine Feineinstellung durchführen als auch in überschaubarer Zeit den gesamten Bereich zwischen 0 und 1023 durchfahren.

```
'-----------------------------------------------------------
' eDIP240-7
' ATMEGA644P, 11,0592 MHz
' I2C-Display
' Edip3.bas
'-----------------------------------------------------------

$regfile = "m644Pdef.dat"
$crystal = 11059200
$hwstack = 64
$swstack = 64
$framesize = 64


Dim Dat As Byte
Dim Sbuf As Byte
Dim Rbuf As Byte
Dim Bcc As Byte
Dim Touchcode As Byte
Dim Endcode As Byte
Dim C As String * 1
Dim N As Integer
Dim Buffer As String * 24
Dim Time_repeat As Integer

Dim Count As Integer
Dim U As Integer


Declare Sub Write_edip
Declare Sub Read_edip
Declare Sub Read_edip_repeating
Declare Sub Buffer_edip


Portb = 255                                 'Pullups

Config Sda = Portb.2
Config Scl = Portb.3

Const Slave = &HDC                          'I2C write DC
Const Slaverd = &HDD                        'I2C read DD
```

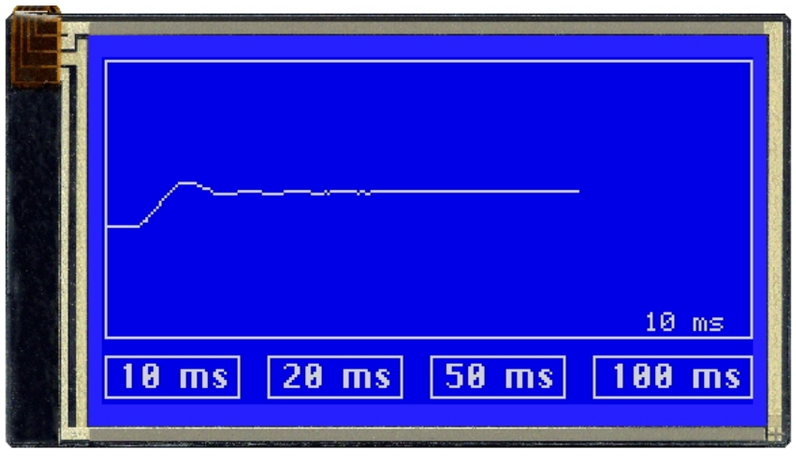
```
  I2cinit

  Config Adc = Single , Prescaler = 32 , Reference = Off
  Start Adc
  Config Timer1 = Pwm , Prescale = 8 , Pwm = 10 ,
                                    Compare A Pwm
                       = Clear Down , Compare B Pwm
                                     = Clear Down
  Start Timer1


  Buffer = Chr(27) + "DL" + Chr(13)           'clear
  Write_edip
  Buffer = Chr(27) + "ZF" + Chr(2) + Chr(13)  '2 = FONT6x8
  Write_edip

  'Touch button at 10,10;60,30, Touchcode 121, Endcode = 250
  Buffer = Chr(27) + "AT" + Chr(10) + Chr(10) + Chr(60)
        + Chr(30) + Chr(121) + Chr(250) + "+" + Chr(13)
  Write_edip
  'Touch button at 70,10;120,30, Touchcode 122, Endcode = 250
  Buffer = Chr(27) + "AT" + Chr(70) + Chr(10) + Chr(120)
        +  Chr(30) + Chr(122) + Chr(250) + "-" + Chr(13)
  Write_edip

   Time_repeat = 200
Do
  If Touchcode = 0 Then Read_edip Else Read_edip_repeating
  If Touchcode = 121 Then                          '+
    If Time_repeat > 90 Then Count = Count + 1
    If Time_repeat < 91 Then Count = Count + 5
    If Time_repeat < 11 Then Count = Count + 10
    If Count > 1023 Then Count = 1023
  End If
  If Touchcode = 122 Then                          '+
    If Time_repeat > 90 Then Count = Count - 1
    If Time_repeat < 91 Then Count = Count - 5
    If Time_repeat < 11 Then Count = Count - 10
    If Count < 0 Then Count = 0
  End If

  If Touchcode > 100 Then Time_repeat = Time_repeat - 2
  If Time_repeat < 30 Then Time_repeat = 30
  If Touchcode = 0 Then Time_repeat = 200
  Waitms Time_repeat

  Pwm1a = Count
  Buffer = Chr(27) + "ZL" + Chr(10) + Chr(60) + "PWM1A = "
                            + Str(count) + "  " + Chr(13)
  Write_edip
  U = Getadc(0)
  Buffer = Chr(27) + "ZL" + Chr(10) + Chr(80) + "ADC(0) = "
                                + Str(u) + "  " + Chr(13)
  Write_edip


Loop


'*************** eDIP240-7 *************

…

Sub Read_edip_repeating   ' last touchcode remaining
  Endcode = 0
  Buffer_edip
  If Sbuf > 0 Then
    I2cstart
```

```
    I2cwbyte Slave
    Dat = 18                                        'DC2
    I2cwbyte D
    N = 1                                           '1 Byte
    Bcc = 18 + N
    I2cwbyte N
    C = "S"
    Dat = Asc(c)
    Bcc = Bcc + Dat
    I2cwbyte Dat                                     '"I"
    I2cwbyte Bcc
    I2cstop
    I2cstart
    I2cwbyte Slaverd
    I2crbyte Dat , Ack
    I2crbyte Dat , Ack
    I2crbyte Dat , Ack
    I2crbyte Dat , Ack
    I2crbyte Dat , Ack
    I2crbyte Dat , Ack
    I2crbyte Dat , Ack
    If Dat > 100 Then Touchcode = Dat
    'Touchcode > 100
    If Dat > 200 Then Endcode = Dat
    'Tastencode > 200 = Endcode
    If Endcode > 200 Then Touchcode = 0
    I2crbyte Dat , Nack
    I2cstop
  End If
End Sub
End
```

Auswertung der Tastendauer

8.4 Grafische Anzeige

Das Display kennt zahlreiche Grafik-Befehle, die in einem Bascom-Programm einfach angewandt werden können. Damit eröffnen sich zahlreiche Möglichkeiten für die Darstellung von Messergebnissen. Möglich ist z.B. ein Datenplotter mit einem übersichtlichen XY-Diagramm.



Ein Messergebnis des Plotters ((Anzeige4.jpg))

Das Programm Edip4.bas bildet einen analogen Einkanal-Plotter mit grafischer Ausgabe und Umschaltmöglichkeit für die Messgeschwindigkeit. Die folgenden Grafikbefehle werden eingesetzt:

GR: Zeichnen eines Rechtecks
GD: Zeichnen einer Linie von x1, y1 nach x2, y2
GW: Fortführung einer Linie nach x, y

RL: Löschen eines rechteckigen Bereichs

Messwerte im Bereich 0 … 1023 werden hier durch zehn geteilt und in einem 102 Punkte hohen Bereich geplottet. Die Messung wird jeweils nach fünf Sekunden automatisch wiederholt. Zuvor wird jeweils der innere Bereich des Rahmens gelöscht um ein neues Bild aufzubauen. Vor jedem neuen Plot wird abgefragt, ob eine der Auswahltasten gedrückt wurde. In dem Fall wird die Intervallzeit zwischen zwei Messpunkten entsprechend verändert.

```
  Buffer = Chr(27) + "DL" + Chr(13)          'clear
  Write_edip
  Buffer = Chr(27) + "ZF" + Chr(2) + Chr(13)
  Write_edip


  Buffer = Chr(27) + "AT" + Chr(1) + Chr(110) + Chr(50) +
       Chr(125) + Chr(121) + Chr(250) + "10 ms" + Chr(13)
  Write_edip

  Buffer = Chr(27) + "AT" + Chr(61) + Chr(110) + Chr(110) +
       Chr(125) + Chr(122) + Chr(250) + "20 ms" + Chr(13)
  Write_edip

  Buffer = Chr(27) + "AT" + Chr(121) + Chr(110) + Chr(170) +
        Chr(125) + Chr(123) + Chr(250) + "50 ms" + Chr(13)
  Write_edip
  Buffer = Chr(27) + "AT" + Chr(181) + Chr(110) + Chr(239) +
       Chr(125) + Chr(124) + Chr(250) + "100 ms" + Chr(13)
  Write_edip

  Buffer = Chr(27) + "GR" + Chr(1) + Chr(1) + Chr(239) +
                                    Chr(103) + Chr(13)
  Write_edip

  T = 10
Do
  Read_edip
  If Touchcode = 121 Then T = 10
  If Touchcode = 122 Then T = 20
  If Touchcode = 123 Then T = 50
  If Touchcode = 124 Then T = 100
  Read_edip
  Read_edip
  Read_edip
  Buffer = Chr(27) + "ZL" + Chr(200) + Chr(94) + Str(t)
                                     + " ms" + Chr(13)
  Write_edip
  U = Getadc(0)
  U = U / 10
  U = 102 - U
  Buffer = Chr(27) + "GD" + Chr(1) + Chr(u) + Chr(2)
                                  + Chr(u) + Chr(13)
  Write_edip
  For X = 2 To 238
    Waitms T
    U = Getadc(0)
    U = U / 10
    U = 102 - U
    Buffer = Chr(27) + "GW" + Chr(x) + Chr(u) + Chr(13)
    Write_edip
  Next X
  Wait 5
  Buffer = Chr(27) + "RL" + Chr(2) + Chr(2) + Chr(238)
                                  + Chr(102) + Chr(13)
  Write_edip
Loop
```

Grafische Elemente (Edip4.bas)

Nach einer Messung bleibt das Bild noch fünf Sekunden lang stehen. Wenn man in dieser Zeit auf eine Zeit-Taste drückt, wird das Ergebnis erst mit einiger Verzögerung sichtbar. Daher ist es nicht ausgeschlossen, dass man unbewusst zwei oder mehr Touch-Ereignisse auslöst. Damit diese nicht im Puffer stehen bleiben, wird das Display nach der Auswertung zur Sicherheit noch einige Male ausgelesen.