ADwin

Borland DELPHI Treiber

für ***ADwin*** Meßwerterfassungskarten

Version 1.5

Januar 1997

Meßwerterfassung mit ADwin-Karten in Borland DELPHI

1. Installation der ADwin Treiber

Legen Sie die Diskette mit der Aufschrift: ***ADwin*** Treiber in das Diskettenlaufwerk ihres Rechners und starten Sie das darauf befindliche **Programm: setup.exe**
Nachdem Sie die Sprache und das Zielverzeichnis (es wird empfohlen das Verzeichnis C:\ADBASIC2 zu bestätigen) gewählt haben, kopiert das setup Programm folgende Dateien in das gewählte Verzeichnis Ihres Rechners:

ADWIN2.BTL	Treiberprogramm für T225 Prozessor
ADWIN4.BTL	Treiberprogramm für T400 Prozessor
ADWIN5.BTL	Treiberprogramm für T450 Prozessor
ADWIN8.BTL	Treiberprogramm für T805 Prozessor
ISERVER.EXE	Programm zum Laden des Treibers unter DOS
RUN.BAT	Hilfsprogramm zum Laden des Treibers unter DOS
ADBLOAD.EXE	Programm zum Laden von kompilierten ADbasic Prozessen unter DOS
TESTVE16.EXE	Zeigt die Version der installierten 16-Bit ADwin DLLs an
TESTVE32.EXE	Zeigt die Version der installierten 32-Bit ADwin DLLs an
ADTEST.EXE	Windows Programm zum Testen der ***ADwin***-Karte
ADWINSET.EXE	Programm zum Anmelden der Linkadresse (nur für Windows NT)

Außerdem werden je nach verwendetem Betriebssystem folgende DLLs (Dynamic Link Libraries) in das Windows Verzeichnis Ihres Rechners kopiert:

Windows 3.X	**Windows 95**	**Windows NT**
ADWIN.DLL	ADWIN.DLL	ADWIN.DLL
ADWIN32.DLL	ADWIN32.DLL	ADWIN32.DLL
ADPOINT.DLL	ADPOINT.DLL	ADPOINT.DLL
	ADWIN95.DLL	ADWINNT.DLL

Nach ordnungsgemäßer Beendigung des setup Programms erscheint die Meldung: Das Setup wurde erfolgreich durchgeführt.

Falls Sie unter *Windows NT* arbeiten, muß der Rechner nach der Treiber-Installation neu gestartet werden

2. Laden des Treibers

Der PC kann erst mit der ***ADwin***-Karte kommunizieren, nachdem der ***ADwin*** Treiber geladen wurde. Der Ladevorgang löscht den gesamten Speicher der ***ADwin***-Karte, so daß aller bereits auf der Karte befindliche Programmcode dabei verloren geht. Der Treiber wird wahlweise durch eine der drei nachfolgenden Möglichkeiten geladen.

a) Unter Delphi

Mit der Funktion iserv('*Treiberdatei*', *Speicherausbau*) aus der Datei uadwin.pas. Je nach Typ und Ausführung der verwendeten ADwin-Karte gelten folgende Parameter:

Dwin-Typ	*Treiberdatei*
ADwin-2	*adwin2.btl*
ADwin-4	*adwin4.btl*
ADwin-5	*adwin5.btl*
ADwin-8	*adwin8.btl*

Ausführung	*Speicherausbau*
64 KB	*10000*
1 MB	*100000*
2 MB	*200000*
4 MB	*400000*
8 MB	*800000*
16 MB	*1000000*
32 MB	*2000000*

Beispiele:

ADwin-4, 1 MB Ausführung: erg := iserv('*adwin4.btl*', *100000*)

ADwin-8, 4 MB Ausführung: erg := iserv('*adwin8.btl*', *400000*)

b) Unter DOS (nicht für Windows NT !)

Mit der Anweisung ***run adwin****X* $X \in \{2,4,5,8\}$ je nach ADwin-Typ.

Wenn der Treiber richtig geladen wurde, erscheint die Meldung ***ADwin@ V2.0.0 started***.

c) Mit dem Windows Programm ADtest

Das Programms ***ADtest*** dient als Hilfe für die Funktionskontrolle der ***ADwin***-Karte und bietet ebenfalls eine Möglichkeit zum Laden des Treibers. Starten Sie dazu das Programm **ADtest.exe**. Falls der Treiber seit dem Einschalten des PCs noch nicht auf die Karte geladen wurde, erscheint die folgende Dialogbox:

Stimmen Sie alle Einstellungen auf die von Ihnen eingesetzte ***ADwin***-Karte ab, und betätigen Sie anschließend die OK-Taste. Falls keine Fehlermeldung ausgegeben wird, konnte der Treiber erfolgreich geladen werden. Das Programm ADtest zeigt dann die momentan an den analogen Eingängen 1-12 anliegenden Spannungen und die Pegel an den digitalen Eingängen an. Außerdem können die analogen Ausgänge und die digitalen Ausgänge gesetzt werden. Auf den analogen Ausgängen kann zum Testen ein Sinus- oder Dreiecksignal mit 10 Hz ausgegeben werden. Falls der Treiber nicht geladen werden konnte, läßt sich der Vorgang durch Betätigen der Settings-Taste mit geänderten Einstellungen wiederholen.

3. Aufrufen von Meßfunktionen

Die Delphi Programme können, mit Hilfe der Funktionen aus der ***adwin.dll*** (Windows 3.1) bzw. ***adwin32.dll*** (Windows '95, NT) Befehle an die ***ADwin***-Karte schicken und Daten abholen. Diese Funktionen werden von der Datei ***uadwin.pas*** importiert. In dieser Datei befinden sich außerdem Funktionen, die eine einfache und übersichtliche Programmierung der Meßaufgabe ermöglichen.
Um die Funktionen aus der Datei ***Uadwin.pas*** Ihrem Programm zugänglich zu machen, sind zwei Maßnahmen erforderlich:

1) Die Datei ***Uadwin.pas*** muß in Ihr Projekt eingebunden werden.
2) Im Interface/Uses - Abschnitt Ihrer Unit muß der Name der Datei (Uadwin) hinzugefügt werden.

Die zur Verfügung stehenden Meßfunktionen und Parameter sind auf den folgenden Seiten beschrieben.

ACHTUNG !

Der ***ADwin***-Treiber muß nach dem Einschalten Ihres Rechners bzw. vor dem Starten von Meßprogrammen immer mit einer der drei unter 2. beschriebenen Anweisungen auf die ***ADwin***-Karte geladen werden. Anderenfalls kann Ihr Rechner nicht mit der ***ADwin***-Karte kommunizieren !

Funktionen des Meßprogramms auf der *ADwin*-Karte

1. Einfache I/O-Funktionen

Einen analogen Eingang messen

Meßwert := ADC(Nr.)

Der Transputer führt eine Messung am spezifizierten Eingang durch und übergibt das Meßergebniss

Einen analogen Ausgang setzen

Set_DAC(Ausgabewert)

Gibt auf dem spezifizierten DAC den Ausgabewert aus

Digitale Eingänge einlesen

Wert := Dig_In()

Liest den momentanen Zustand der digitalen Eingänge ein.

Digitale Ausgänge setzen

Set_Dig_Out(Ausgabewert)

Setzt die digitalen Ausgänge auf den gewünschten Wert.

Aktuellen Stand der digitalen Ausgänge rücklesen

Wert := Dig_Out()

Liest den momentanen Zustand des digitalen Ausgangsregisters ein.

Freien Speicher der ADwin-Karte abfragen

Wert := Freemen() */* Zusammenhängender freier Speicher der ADwin-Karte in kB */*

Auslastung der ADwin-Karte abfragen

Wert := Auslastung() */* Auslastung des Prozessors der ADwin-Karte in % */*

Testen, ob das richtige Programm auf die *ADwin*-Karte geladen ist

Wert := Test()

Wenn das richtige Programm geladen ist, wird die 0 zürckgegeben

Treiber auf die ADwin-Karte laden

iserv(Dateiname, Speicherausbau)

Wenn das Programm nicht geladen werden kann, wird die 1 zürckgegeben

2. Funktionen zur Steuerung zyklischer I/O-Prozesse

I/O-Prozess starten

Start_Zykl(Prozessnummer)

Prozessnummer 1 = Meßprozess
8 = Prozess zur analogen Ausgabe
16 = Prozess zum gleichzeitigen Signalerzeugen und Messen
32 = Digitaler PID-Regler

Startet den I/O-Prozess mit den vorher gewählten Parametern

I/O-Prozess stoppen

Stop_Zykl (Prozessnummer)

Prozessnummer Siehe I/O-Prozess starten

Parameter für den I/O-Prozess setzen

Set_Opt(Parameternummer, Wert)

Parameternummern für den Meßprozess :

1 = Delay zwischen den Messungen in Mikrosekunden
3 = Anzahl der Messschritte in einem Messzyklus
4 = Eingangs Auswahlregister (pro Eingang ein Bit setzen)
5 = Triggermode
(0 = sofort messen, 1 = falls U >Pegel , 2 = falls U < Pegel, 3 = Anst. Flanke, 4 = fallende Flanke)
6 = Triggerkanal
7 = Triggerschwelle
8 = Pretriggeranteil (in Meßpunkten)

Parameternummern für den analogen Ausgabeprozess :

32 = Delay zwischen den Ausgabepunkten in Mikrosekunden
34 = Anzahl der Ausgabepunkte in einem Zyklus
35 = Anzahl der auszugebenden Zyklen
36 = Ausgangs-Auswahlregister (pro benutzten Ausgang ein Bit setzen)

Parameter nummern für den PID-Reglerprozess :

128 = PID-Regler Sollwert
129 = PID-Regler P-Konstante
130 = PID-Regler T0
131 = PID-Regler Td
132 = PID-Regler Ti

Setzt den I/O-Parameter auf den gewünschten Wert

Ausgabeliste vorgeben

Set_DAC_Liste(Ausgangsnr., Anzahl, Werte-Array)*)*

Schickt ‘Anzahl’ Werte aus dem ‘Werte-Array’ in die Ausgabeliste für das gewünschte DAC.

Prozessstatusparameter abfragen

Get_Opt(Parameternummer)

Parameternummer 32 = Meßprozess läuft (1 = läuft, 0 = läuft nicht)
34 = Nummer der letzten Messung
40 = Ausgabeprozess läuft (1 = läuft, 0 = läuft nicht)
48 = Meß/Ausgabeprozess läuft (1 = läuft, 0 = läuft nicht)
64 = PID-Reglerprozess läuft (1 = läuft, 0 = läuft nicht)

Messwerteliste abholen

Read_AD_Liste(Eingansnr., Anzahl, Ziel-Array)

Holt ‘Anzahl’ Meßwerte vom analogen Eingang ‘Eingangsnr.’ der ADwin-Karte ins ‘Ziel-Array’

Beispielprogramm Fgen zur schnellen Signalerzeugung

Das Programm Fgen gibt eine wählbare Signalform zyklisch aus.
Die Signalform kann mit dem Option-Button **SigSel** ausgewählt werden.
In den Textboxen **freq, amp** und **nsteps** können die Frequenz, die Amplitude (in ADC-Werten) und die Anzahl der Stützpunkte pro Zyklus eingegeben werden.
Der **Start-**Button übergibt die eingestellten Parameter an die ADwin-Karte und startet die Ausgabe.
Mit dem **Stop** Button kann die Ausgabe angehalten werden.


```
implementation
{$R *.DFM}

const
   btlfile     = 'ADWIN4.BTL';            {ADwin4-Karte Treiberprogramm}
   size        = 100000;                  {ADwin-Karte Speichergröße (1MB)}
var
  SigForm : ARRAY[1..4100] of INTEGER;

procedure TForm1.Start_Click(Sender: TObject);         {Start Knopf gedrückt}
var
  delay : LONGINT;
  min, max, mitte : INTEGER;
begin
  delay := 1000000 div
           (StrToInt(Freq.Text)*StrToInt(nSteps.Text));{Delay zwischen 2 Ausgabewerten}
  set_opt(32, delay);                                  {Delay an ADwin-Karte übergeben}
  set_opt(34, StrToInt(nSteps.Text));                  {Anzahl der Schritte/Zyklus}
  set_opt(35, 100000000);                              {Anzahl der Zyklen übergeben}
  set_opt(36, 1);                                      {Nummer des ausgebenden DACs}
  min := 2048 - StrToInt(Amp.Text);                    {Kleinster Ausgabewert}
  max := 2048 + StrToInt(Amp.Text);                    {Größter Ausgabewert}
  if SigSel.ItemIndex = 0 then                              {Falls Signalform Sinus}
    set_signalform_sinus(min, max, StrToInt(nSteps.Text));
  if SigSel.ItemIndex = 1 then                              {Falls Signalform Trapez}
    set_signalform_trapez(min, max, StrToInt(nSteps.Text));
  if SigSel.ItemIndex = 2 then                              {Falls Signalform Dreieck}
    set_signalform_dreieck(min, max, StrToInt(nSteps.Text));
  set_dac_liste(1, StrToInt(nSteps.Text), @SigForm[1]);{Signalform an ADwin-Karte übergeben}
  start_zykl(8);                                       {Ausgabe starten}
end;

procedure set_signalform_sinus(min, max, nsteps : INTEGER);  {Sinus Signalform berechnen}
var
  i: INTEGER;
  iamp, imitt, fh1, si : SINGLE;
begin
  iamp  := (max-min)/2 -1;
  imitt := (max+min)/2;
  for i := 1 to nsteps do begin
    fh1 := (i*6.2832)/nsteps;
    si  := sin(fh1)*iamp;
    SigForm[i] := INTEGER(round(imitt+si));
  end;
end;

procedure set_signalform_trapez(min, max, nsteps : INTEGER);  {Trapez Signalform berechnen}
var
  i, iws: INTEGER;
  id, ih, iw: LONGINT;
begin
  id := max-min;
  for i := 0 to (nsteps div 4 - 1) do begin
    ih := i * 4;
    iw := (ih * id) div nsteps;
    iws := iw + min;
    SigForm[i+1] := iws;
    SigForm[(nsteps*3) div 4 -i] := iws;
  end;
```

```
  for i := 0 to (nsteps div 4 - 1) do begin
    SigForm[i+1 + nsteps div 4] := max;
    SigForm[nsteps -i] := min;
  end;
end;

procedure set_signalform_dreieck(min, max, nsteps : INTEGER); {Dreieck Signalform berechnen}
var
  i, iws : INTEGER;
  id, ih, iw : LONGINT;
begin
  id := max-min;
  for i := 0 to (nsteps div 2 -1) do begin
    ih  := i*2;
    iw  := (ih*id) div nsteps;
    iws := INTEGER(iw+min);
    SigForm[i+1] := iws;
    SigForm[nsteps-i] := iws;
  end;
end;

procedure TForm1.Stop_Click(Sender: TObject);     {Stop Knopf gedrückt}
begin
  stop_zykl(8);                                   {Ausgabe stoppen}
end;

procedure TForm1.Exit_Click(Sender: TObject);     {Exit Knopf gedrückt}
begin
  Halt;                                           {Programm beenden}
end;

procedure TForm1.Amp_Exit(Sender: TObject);       {Ampliude wurde eingegeben}
var
  amplitude : INTEGER;
begin
  amplitude := StrToInt(Amp.Text);
  if amplitude < 0 then amplitude := 0;
  if amplitude > 2047 then amplitude := 2047;
  Amp.Text := IntToStr(amplitude);
end;

procedure TForm1.Freq_Exit(Sender: TObject);    {Frequenz wurde eingegeben}
var
  frequenz : LONGINT;
  schritte : INTEGER;
begin
  frequenz := StrToInt(Freq.Text);
  schritte := StrToInt(nSteps.Text);
  if frequenz < 1 then frequenz := 1;
  if (frequenz*schritte > 50000) then frequenz := 50000 div schritte;
  Freq.Text := IntToStr(frequenz);
end;

procedure TForm1.nSteps_Exit(Sender: TObject);  {Stützpunkte wurden eingegeben}
var
  frequenz : LONGINT;
  schritte : INTEGER;
begin
  frequenz := StrToInt(Freq.Text);
  schritte := StrToInt(nSteps.Text);
  if schritte < 2 then schritte := 2;
  if schritte > 4096 then schritte := 4096;
  if (frequenz*schritte > 50000) then schritte := 50000 div frequenz;
  nSteps.Text := IntToStr(schritte);
end;

initialization
  iserv(btlfile, size);       {Treiber auf die ADwin-Karte laden}
end.
```

3. Erweiterte Funktionen zur Steuerung von ADbasic

Einen ADbasic-Prozess laden, der von ADbasic mit der Funktion 'Make Bin File' erzeugt wurde

ADbload(Dateiname)

Läd einen ADbasic-Prozess auf die ADwin-Karte

ADbasic-Prozess starten

Start(Prozessnummer)

Prozessnummer
1 = ADbasic Prozess 1 starten
2 = ADbasic Prozess 2 starten
3 = ADbasic Prozess 3 starten
4 = ADbasic Prozess 4 starten
. . . usw.

Startet einen ADbasic-Prozess

ADbasic-Prozess starten

Stop(Prozessnummer)

Prozessnummer
1 = ADbasic Prozess 1 stoppen
2 = ADbasic Prozess 2 stoppen
3 = ADbasic Prozess 3 stoppen
4 = ADbasic Prozess 4 stoppen
. . . usw.

Stoppt einen ADbasic Prozess

Integer Parameter für ADbasic setzen

Set_Par (Parameternummer, Wert)

Parameternummern für ADbasic :

Parameternummer
1 = Par_1
. . . .
80 = Par_80

-99 = Prozess_1 running
-98 = Prozess_2 running
-97 = Prozess_3 running
-96 = Prozess_4 running
. . .
-89 = Prozess_1 Delay (in Mikrosekunden)
-88 = Prozess_2 Delay (in Mikrosekunden)
. . .
-69 = Prozess_1 Activate
-68 = Prozess_2 Activate
. . .

Setzt den Parameter auf den gewünschten Wert

Float Parameter für ADbasic setzen

Set_Fpar (Parameternummer, Wert)

Parameternummern für ADbasic :

Parameternummer
1 = FPar_1
. . . .
80 = FPar_80

Integer Parameter von ADbasic lesen

Get_Par (Parameternummert)

Parameter für ADbasic :

Parameternummer 1 = Par_1
. . . . (Siehe Integer Parameter für ADbasic setzen)

Liest den gewünschten Parameter

Float Parameter von ADbasic lesen

Get_Fpar (Parameternummer)

Parameternummern für ADbasic :

Parameternummer 1 = FPar_1
. . . .
80 = FPar_80

Datensatz von ADbasic in ein Short, Integer o Float Array übernehmen

Get_Data, lGet_Data o. fGet_Data(DNr, Ind, Anzahl, Ziel-Array)

DNr: *ADbasic Datensatznummer*
Ind: *Index des 1. Zu übertragenen Elementes*
Antzahl: *Anzahl der Elemente die übertragen werden*
Ziel-Array: *Array für die übertragenen Werte*

Schreibt die Elemente aus dem angegebene ADbasic Datensatz in das 'Ziel-Array'.

Short, Integer o. Float Array als Datensatz an ADbasic schicken

Set_Data, lSet_Data o. fSet_Data(DNr, Anzahl, Werte-Array)

DNr: *ADbasic Datensatznummer*
Antzahl: *Anzahl der Elemente die übertragen werden*
Werte-Array: *Array das die zu übertragenen Werte enthält*

Schreibt die Elemente des übergebenen Arrays in den gewünschten *ADbasic* Datensatz

Die Werte aus einem ADbasic Datensatz direkt auf der Festplatte speichern (binär)

Save_Data(dateiname, Dnr; StartInd, Anzahl, mode)

Dateiname: *Dateiname unter dem die Werte gespeichert werden*
DNr: *ADbasic Datensatznummer*
StartInt: *Index des Elementes ab dem gespeichert wird*
Anzahl: *Anzahl der zu speichernden Elemente*
mode: *0: Bestehende Datei wird gelöscht 1: Daten werden angehängt*

Schreibt die Elemente aus einem *ADbasic* Datensatz direkt auf die Festplatte

Hinweis zum FIFO Datensatz

Die folgenden Funktionen greifen auf den ADbasic FIFO Datensatz zu. Unter ADbasic können Datensätze als FIFO (**F**irst **i**n **f**irst **o**ut) Ringspeicher deklariert werden. Die Elemente in einem FIFO Datensatz werden in der selben Reihenfolge ausgelesen, in der sie in das FIFO geschrieben wurden.
Die FIFO Datensätze müssen unter ADbasic deklariert werden. (Vgl. ADbasic Handbuch)

FIFO von ADbasic in ein Short, Integer o. Float Array übernehmen
Vor dem Aufruf dieser Funktion sollte mit der **Funktion GetFifoCount** überprüft werden, ob noch genügend Elemente im FIFO sind.

Get_Fifo, lGet_Fifo o. fGet_Fifo(FNr, Anzahl, Ziel-Array)
FNr: ADbasic Fifo Datensatznummer
Antzahl: Anzahl der Elemente die übertragen werden
Ziel-Array: Array für die übertragenen Werte
Übernimmt die Elemente des gewünschten ADbasic FIFO in das 'Ziel-Array'

Short, Integer o. Float Array als FIFO an ADbasic schicken
Vor dem Aufruf dieser Funktion sollte mit der **Funktion GetFifoEmpty** überprüft werden, ob noch genügend Platz im FIFO ist.

Se_Fifo, lSet_Fifo o. fSet_Fifo(FNr, Anzahl, Werte-Array)
FNr: ADbasic Fifo Datensatznummer
Antzahl: Anzahl der Elemente die übertragen werden
Werte-Array: Array, das die zu übertragenden Werte enthält
Schreibt die Elemente des übergebenen Arrays in das gewünschte ADbasic FIFO

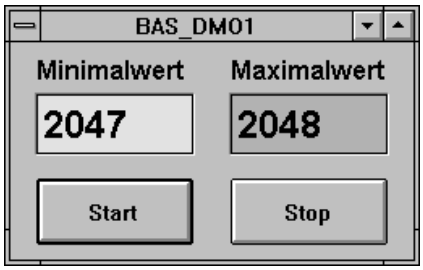
Anzahl der Elemente im FIFO ermitteln
Get_Fifo_Count(FIFO-Nummer)
Gibt die Anzahl der Elemente im ADbasic FIFO zurück

Anzahl der freien FIFO Positionen ermitteln
Get_Fifo_Empty(FIFO-Nummer)
Gibt den freien Speicherplatz im ADbasic FIFO zurück

Inhalt des FIFO löschen
Clear_Fifo(FIFO-Nummer)
Löscht den Inhalt des spezifizierten ADbasic FIFOs

Beispiele zum Datenaustausch mit ADbasic Prozessen

Beispielprogramm BAS_DMO1.PAS Online-Auswertung von Messwerten

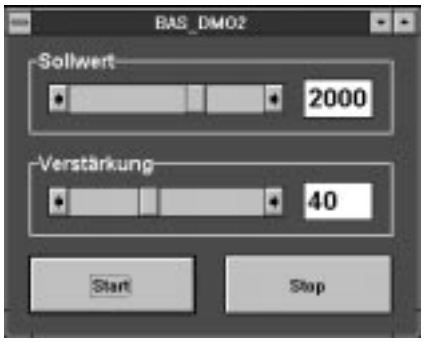


Das Programm BAS_DMO1 prüft mit der Timerfunktion **Timer1** einmal pro Sekunde, ob der ***ADbasic***-Prozess1 das ACTIVATE_PC Flag gesetzt hat. Ist dies der Fall, so holt sich das Programm die beiden ***ADbasic***-Parameter PAR_1 sowie PAR_2 und zeigt sie in den Fenstern für Minimal- und Maximalwert an.
Der ***ADbasic***-Prozess1 setzt das ACTIVATE_PC Flag, nachdem er aus 1000 Messungen den Minimal- und Maximalwert ermittelt hat.
Mit den Tasten **Start** und **Stop** wird der Prozess1 gestartet bzw. angehalten. Zusätzlich aktivieren bzw. deaktivieren die Tasten die Timerfunktion **Timer1**.

```
implementation

{$R *.DFM}

var
  count: INTEGER;

const
   btlfile     = 'ADWIN8.BTL';             {ADwin8-Karte Treiberprogramm}
  {btlfile     = 'ADWIN4.BTL';             {ADwin4-Karte Treiberprogramm}
  {btlfile     = 'ADWIN2.BTL';             {ADwin2-Karte Treiberprogramm}
   size        = 100000;                   {ADwin-Karte Speichergröße (1MB)}
  {size        = 400000;                   {ADwin-Karte Speichergröße (4MB)}
  {size        = 800000;                   {ADwin-Karte Speichergröße (8MB)}
   adbasicfile = 'BAS_DMO1.T81';  {ADwin-Karte ADbasic Prozess}

procedure TForm1.Start_Click(Sender: TObject);
begin
  start(1);                                  { ADbasic-Prozess1 starten }
  Timer1.Enabled := TRUE                     { Timer1 aktivieren        }
end;

procedure TForm1.Stop_Click(Sender: TObject);
begin
  Timer1.Enabled := FALSE;                   { Timer1 deaktivieren     }
  stop(1)                                    { ADbasic-Prozess1 stoppen}
end;

procedure TForm1.Timer1_Timer(Sender: TObject);
var par1, par2: LongInt;
begin
  IF get_activate(1) = 1 THEN BEGIN          {ACTIVATE_PC (ADbasic Prozess1) gesetzt ?}
    par1 := get_par(1);                      {ADbasic Parameter1 holen  }
    par2 := get_par(2);                      {ADbasic Parameter2 holen  }
    Edit1.Text := IntToStr(par1);            {ADbasic Parameter1 anzeigen  }
    Edit2.Text := IntToStr(par2)             {ADbasic Parameter1 anzeigen  }
  END;
end;

initialization
  iserv(btlfile, size);   {Treiber auf ADwin-Karte laden}
  ADBload(adbasicfile);   {ADbasic Prozess auf ADwin-Karte laden}
end.
```

Beispielprogramm BAS_DMO2.PAS P-Regler

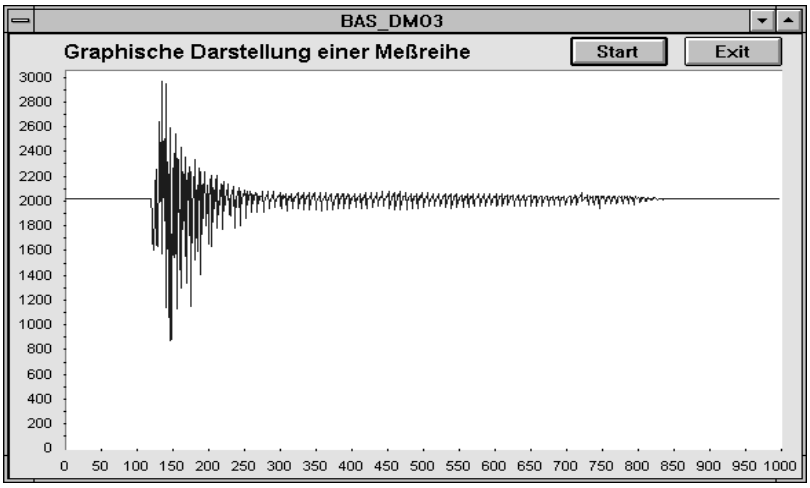
Das Programm BAS_DMO2 wird verwendet um einem auf der ADwin-Karte als Prozess1 laufenden digitalen P-Regler den Sollwert, sowie die Verstärkung vorzugeben.
Die Zuweisung von Sollwert und Verstärkung erfolgt über die beiden ADbasic-Parameter PAR_1 und PAR_2.
Mit den Tasten **Start** und **Stop** wird der Prozess1 gestartet bzw. angehalten. Sollwert und Verstärkung können über zwei Schieberegler eingestellt werden. Bei jeder Bewegung eines Schiebereglers wird der neu eingestellte Wert an den ***ADbasic***-Prozess1 übergeben.




```
implementation

{$R *.DFM}

const
   btlfile     = 'ADWIN8.BTL';             {ADwin8-Karte Treiberprogramm}
  {btlfile     = 'ADWIN4.BTL';             {ADwin4-Karte Treiberprogramm}
  {btlfile     = 'ADWIN2.BTL';             {ADwin2-Karte Treiberprogramm}
   size        = 100000;                   {ADwin-Karte Speichergröße (1MB)}
  {size        = 400000;                   {ADwin-Karte Speichergröße (4MB)}
  {size        = 800000;                   {ADwin-Karte Speichergröße (8MB)}
   adbasicfile = 'BAS_DMO2.T81';  {ADwin-Karte ADbasic Prozess}

procedure TForm1.Soll_Change(Sender: TObject);
begin
  Edit1.Text:= IntToStr(ScrollBar1.Position);
  set_par(1, ScrollBar1.Position);                {ADbasic Parameter1 auf den
                                                   Wert des Scrollbalken setzen}
end;

procedure TForm1.Ver_Change(Sender: TObject);
begin
  Edit2.Text:= IntToStr(ScrollBar2.Position);
  set_par(2, ScrollBar2.Position);                {ADbasic Parameter2 auf den
                                                   Wert des Scrollbalken setzen}
end;

procedure TForm1.Start_Click(Sender: TObject);
begin
  ScrollBar1.Enabled:= TRUE;                 {Sollwert Scrollbar aktivieren}
  ScrollBar2.Enabled:= TRUE;                 {Verstärkung Scrollbar aktivieren}
  start(1);                                  {ADbasic Prozess1 starten}
end;

procedure TForm1.Stop_Click(Sender: TObject);
begin
  stop(1);                                 {ADbasic Prozess1 stoppen}
  ScrollBar1.Enabled:= FALSE;                {Sollwert Scrollbar deaktivieren}
  ScrollBar2.Enabled:= FALSE;                {Verstärkung Scrollbar deaktivieren}
end;

initialization
  iserv(btlfile, size);                {Treiber auf ADwin-Karte laden}
  ADBload(adbasicfile);                {ADbasic Prozess auf ADwin-Karte laden}

end.
```

Beispielprogramm BAS_DMO3.PAS Darstellung einer Meßreihe

Das Programm BAS_DMO3 prüft mit der Timerfunktion **Timer1** einmal pro Sekunde, ob der ***ADbasic***-Prozess1 das ACTIVATE_PC Flag gesetzt hat. Ist dies der Fall, so holt sich das Programm die ersten 1000 Werte aus dem ***ADbasic***-Array Data1 und stellt diese graphisch dar.
Mit der **Start** Taste wird der ***ADbasic***-Prozess1 gestartet. Der Prozess BAS_DMO3 nimmt 1000 Meßwerte auf und setzt anschließend das ACTIVATE_PC Flag.




```
implementation
{$R *.DFM}

const
  {btlfile     = 'ADWIN8.BTL';             {ADwin8-Karte Treiberprogramm}
  {btlfile     = 'ADWIN4.BTL';             {ADwin4-Karte Treiberprogramm}
   btlfile     = 'ADWIN2.BTL';             {ADwin2-Karte Treiberprogramm}
   size        = 100000;                   {ADwin-Karte Speichergröße (1MB)}
  {size        = 400000;                   {ADwin-Karte Speichergröße (4MB)}
  {size        = 800000;                   {ADwin-Karte Speichergröße (8MB)}
   adbasicfile = 'BAS_DMO3.T21';  {ADwin-Karte ADbasic Prozess}

procedure TForm1.Start_Click(Sender: TObject);
begin
  start(1);                                {ADbasic-Prozess1 starten}
  Timer1.Enabled := TRUE                   {Timer1 aktivieren}
end;

procedure TForm1.Timer1_Timer(Sender: TObject);
var
  index : INTEGER;
  messwerte : ARRAY[1..1000] of INTEGER;
begin
  if get_activate(1)= 1 then begin      {ACTIVATE_PC (ADbasic Prozess1) gesetzt}
    get_data(1, 1, 1000, @messwerte[1]); {ADbasic Datensatz1 (1000 Werte) holen}
    graph1.OpenData[MAKELONG(1,999)] := 0;
    for index := 0 to 998 do
      graph1.Value[index] := messwerte[index+1]; {Messwert an Graph übergeben}
    graph1.CloseData[MAKELONG(1,999)] := 0;
    graph1.RGB2DBk := clWhite;
    Timer1.Enabled := False;                    {Timer1 deaktivieren}
  end;
end;

procedure TForm1.Exit_Click(Sender: TObject);
begin
  halt;                                         {Programm beenden}
end;

initialization
  iserv(btlfile, size);   {Treiber auf ADwin-Karte laden}
  ADBload(adbasicfile);   {ADbasic Prozess auf ADwin-Karte laden}

end.
```

2. Oft benötigte Meß- und Steuerungsfunktionen

Die folgenden Funktionen bauen auf den Funktionen aus der Datei adwin.dll (Windows 3.1) oder adwin32.dll (Windows '95, NT) auf. Der Source-Code dieser Funktionen, sowie diverse Beispiele zu deren Aufruf, befinden sich in der Datei: Uadwin.pas.

```
{ Den analogen Eingang "nr" messen und das Ergebnis zurückgeben                 }
{ ============================================================================}
FUNCTION adc(nr: INTEGER) : INTEGER;


{ Liefert die momentane Auslastung des Transputers                              }
{ ============================================================================}
FUNCTION auslast : INTEGER;


{ Abfragen der digitalen Eingänge. Die Eingänge stehen bitweise in einem 16-Bit Wort}
{ ============================================================================}
FUNCTION dig_in : INTEGER;


{ Rücklesen des auf den digitalen Ausgängen ausgegebenen Wertes                 }
{ ============================================================================}
FUNCTION dig_out : INTEGER;


{ Liefert den momentanen freien Speicher des Transputers (Ergebnis in Bytes)    }
{ ============================================================================}
FUNCTION freemem : LongInt;


{ Den analogen Ausgang "ndac" auf den Wert "iw" setzen                          }
{ ============================================================================}
PROCEDURE set_dac (ndac, iw : INTEGER);


{ Das INTEGER Array auf welches iw zeigt als Ausgabesignalform auf ADwin-Karte ausgeben }
{ ============================================================================}
PROCEDURE set_dac_liste (ndac : INTEGER;i1 : LongInt;iw : POINTER);


{ Gibt den Wert "iw" auf den digitalen Ausgängen aus                            }
{ ============================================================================}
PROCEDURE set_dig_out (iw : INTEGER);


{ Einen Parameter auf der ADwin-Karte auf den Wert "il" setzen                  }
{ ============================================================================}
PROCEDURE set_par (npar : INTEGER; il : LongINT);


{ Options auf der ADwin-Karte auf den Wert "il" setzen                          }
{ ============================================================================}
PROCEDURE set_opt (npar : INTEGER; il : LongINT);


{ Einen Float Parameter auf der ADwin-Karte auf den Wert "iw" setzen            }
{ ============================================================================}
PROCEDURE set_fpar (npar : INTEGER; iw : SINGLE);


{ Einen ADbasic Prozess auf der ADwin-Karte starten                             }
{ ============================================================================}
PROCEDURE start (nproz : INTEGER );
```

```
{ Einen ADbasic Prozess auf der ADwin-Karte stoppen                          }
{ ===========================================================================}
PROCEDURE stop (nproz : INTEGER );


{ Einen I/O-Zyklus auf der ADwin-Karte starten                               }
{ ===========================================================================}
PROCEDURE start_zykl (nproz : INTEGER );


{ Einen I/O-Zyklus auf der ADwin-Karte stoppen                               }
{ ===========================================================================}
PROCEDURE stop_zykl (nproz : INTEGER );


{ Einen Parameter von der ADwin-Karte lesen                                  }
{ ===========================================================================}
FUNCTION get_par(npar: INTEGER) : LongInt;


{ Options von der ADwin-Karte lesen                                          }
{ ===========================================================================}
FUNCTION get_opt(npar: INTEGER) : LongInt;


{ Einen Float Parameter von der ADwin-Karte lesen                            }
{ ===========================================================================}
FUNCTION get_fpar(npar: INTEGER) : SINGLE;


{ Einen einzelnen Wert aus einer Messwertliste abfragen                      }
{ ===========================================================================}
FUNCTION read_ad_wert(nadc: INTEGER;ind: LongInt):Integer;


{ Einen Messwertliste abfragen und in ein INTEGER Array schreiben            }
{ ===========================================================================}
PROCEDURE read_ad_liste(nadc: INTEGER;il: LongInt;iw : POINTER);


{ Element/e aus einem ADbasic Datensatz vom Transputer holen. Übergabe in INTEGER ARRAY}
{ ===========================================================================}
PROCEDURE  Get_Data(ndata : INTEGER; nstart,il :LongInt;iw : POINTER);


{ Element/e aus einem ADbasic Datensatz vom Transputer holen. Übergabe in LONGINT ARRAY}
{ ===========================================================================}
PROCEDURE lGet_Data(ndata : INTEGER; nstart,il :LongInt;iw : POINTER);


{ Element/e aus einem ADbasic Datensatz vom Transputer holen. Übergabe in SINGLE ARRAY}
{ ===========================================================================}
PROCEDURE fGet_Data(ndata : INTEGER; nstart,il :LongInt;iw : POINTER);


{ Einem ADbasic Datensatz die Werte aus einem INTEGER ARRAY zuweisen         }
{ ===========================================================================}
PROCEDURE  Set_Data(ndata : INTEGER; il :LongInt;iw : POINTER);


{ Einem ADbasic Datensatz die Werte aus einem LONGINT ARRAY zuweisen         }
{ ===========================================================================}
PROCEDURE lSet_Data(ndata : INTEGER; il :LongInt;iw : POINTER);


{ Einem ADbasic Datensatz die Werte aus einem SINGLE ARRAY zuweisen          }
{ ===========================================================================}
PROCEDURE fSet_Data(ndata : INTEGER; il :LongInt;iw : POINTER);
```

```
{ ADbasic ACTIVATE_PC-Flag abfragen und das Ergebnis zurückgeben                }
{ ==============================================================================}
FUNCTION get_activate(nr: INTEGER) : LONGINT;


{ ANzahl der Elemente im FIFO ermitteln                                         }
{ ==============================================================================}
FUNCTION get_fifo_count(nfifo : INTEGER):LONGINT;


{ ANzahl der freien Positionen im FIFO ermitteln                                }
{ ==============================================================================}
FUNCTION get_fifo_empty(nfifo : INTEGER):LONGINT;


{ Inhalt des FIFO löschen                                                       }
{ ==============================================================================}
PROCEDURE clear_fifo (nfifo : INTEGER );


{ Element/e aus einem FIFO vom Transputer holen. Übergabe in INTEGER ARRAY      }
{ ==============================================================================}
PROCEDURE  Get_FIFO(nfifo : INTEGER; il :LongInt;iw : POINTER);


{ Element/e aus einem FIFO vom Transputer holen. Übergabe in LONGINT ARRAY      }
{ ==============================================================================}
PROCEDURE lGet_FIFO(nfifo : INTEGER; il :LongInt;iw : POINTER);


{ Element/e aus einem FIFO vom Transputer holen. Übergabe in SINGLE ARRAY       }
{ ==============================================================================}
PROCEDURE fGet_FIFO(nfifo : INTEGER; il :LongInt;iw : POINTER);


{ Einem ADbasic FIFO die Werte aus einem INTEGER ARRAY zuweisen                 }
{ ==============================================================================}
PROCEDURE  Set_FIFO(nfifo : INTEGER; il :LongInt;iw : POINTER);


{ Einem ADbasic FIFO die Werte aus einem LONGINT ARRAY zuweisen                 }
{ ==============================================================================}
PROCEDURE lSet_FIFO(nfifo : INTEGER; il :LongInt;iw : POINTER);


{ Einem ADbasic FIFO die Werte aus einem SINGLE ARRAY zuweisen                  }
{ ==============================================================================}
PROCEDURE fSet_FIFO(nfifo : INTEGER; il :LongInt;iw : POINTER);


{ Testet, ob Transputer vorhanden und richtig initialisiert ist                 }
{ ==============================================================================}
FUNCTION Test : BOOLEAN;


{ Läd einen ADbasic Prozess auf die ADwin-Karte                                 }
{ ==============================================================================}
PROCEDURE ADBload(dateiname:STRING);


{ Läd den Treiber der ADwin-Karte                                               }
{ ==============================================================================}
PROCEDURE ISERV(dateiname:STRING ; IBOARDSIZE : LONGINT);


{ Element/e aus einem ADbasic Datensatz vom Transputer holen und auf der Festplatte
ablegen  overwrite oder append Modus möglich,  mode:0 : Bestehende Datei wird
gelöscht    1: Daten werden angehängt }
{ ==============================================================================}
PROCEDURE Save_Data(dateiname:string, ndata : INTEGER; nstart,il :LongInt;mode :integer);
```