ADwin

Visual-Basic Treiber

für ***ADwin*** Meßwerterfassungskarten

Version 1.3

März 1995

Meßwerterfassung mit ADwin-Karten in VISUAL BASIC

1. Das Programm ADwin8 auf die ADwin-Karte laden

Da die ADwin-Karten einen eigenen frei programmierbaren Prozessor enthalten, müssen die Karten vor dem Aufrufen von Meßfunktionen mit dem Meßprogramm **ADwin8** geladen werden. Das Programm ADwin8 stellt die auf den nächsten Seiten beschriebenen Meßfunktionen zur Verfügung. Die Meßfunktionen können bei Bedarf mit dem Echtzeit-Entwicklungstool **ADbasic** um anwendungsspezifische Funktionen erweitert werden.

Sie benötigen zum Laden die Dateien :

- **ADwin8.btl (bzw ADwin4.btl oder ADwin2.btl)**
- **iserver.exe**
- **run.bat**

Laden Sie mit Hilfe der Anweisung **run ADwin8** das Programm ADwin8.btl auf die ADwin-Karte.

Alternativ dazu kann das Treiberprogramm mit der Anweisung: **iserv('ADwin8.btl',*size*)** von Ihrem VB Programm geladen werden. Beim Laden des Treibers werden alle Daten auf der ***ADwin***-Karte gelöscht.

ACHTUNG !

Das Programm ADwin8.btl muß immer vor dem Starten von Meßprogrammen mit der DOS-Anweisung ***run ADwin8,*** oder der VB Prozedur **iserv('ADwin8.btl',*size*),** auf die ***ADwin***-Karte geladen werden.

Wenn das Programm richtig geladen wurde, gibt es die Meldung ***ADwin8 V1.4 geladen !*** aus.

2 Aufrufen von Meßfunktionen

Die VISUAL BASIC Programme können mit Hilfe der Funktionen **toLink** und **fromLink** Befehle an die ADwin-Karte schicken und Daten abholen. Diese Funktionen stehen in der DLL: **ADwin.dll.**

Die Datei **ADwin.dll** muß aus einem der beiden Ordner auf der mitgelieferten Diskette in Ihr Windows-Subdirectory kopiert werden. Je nachdem ob Sie unter **Windows 3.1 bzw. 3.11** oder ***Windows 95*** arbeiten, ist dies der Ordner **win31** oder ***win95***. Für ***Windows 95*** wird zusätzlich die DLL: **mut32.dll** benötigt.

Wenn Sie die Datei ADwin.bas in Ihr VB-Programm einbinden, können sie sofort auf diese Funktionen zugreifen. In ADwin.bas sind außerdem erweiterte Funktionen und Subroutinen wie. **adc%, Set_DAC, Set_Par, Start_Messen, stop_Messen, . . .** enthalten, die auf den toLink- und fromLink-Funktionen aufbauen und eine einfache und übersichtliche Programmierung der Meßaufgabe ermöglichen. Die zur Verfügung stehenden Meßfunktionen und Parameter sind auf den folgenden Seiten beschrieben.

Hinweis für Benutzer von ADwin-4 oder ADwin-2 Karten :

Die Meßprogramme auf den verschiedenen ADwin-Karten haben alle die gleiche Schnittstelle zum PC und werden mit den gleichen Befehlen angesprochen. Benutzer von ***ADwin-4*** oder ***ADwin-2*** Karten müssen nur anstatt des in diesem Text erwähnten Meßprogramms **ADwin8.btl** die mitgelieferten Meßprogramme **ADwin4.btl** oder **ADwin2.btl** benutzen.

Funktionen des Meßprogramms auf der *ADwin*-Karte

1. Einfache I/O-Funktionen

Funktion 1 Einen analogen Eingang messen

to_Link (1)
to_Link (ADC-Nr.)
Meßwert = from_Link2 ()
Der Transputer führt eine Messung am spezifizierten Eingang durch und übergibt das Meßergebniss

Funktion 2 Einen analogen Ausgang setzen

to_Link (2)
to_Link (DAC-Nr.)
to_Link2 (Ausgabewert)
Gibt auf dem spezifizierten DAC den einen Wert aus

Funktion 3 Digitale Eingänge einlesen

to_Link (3)
Dig_In = from_Link2 ()
Liest den momentanen Zustand der digitalen Eingänge ein.

Funktion 4 Digitale Ausgänge setzen

to_Link (4)
to_Link2 (Ausgabewert)
Setzt die digitalen Ausgänge auf den gewünschten Wert.

Funktion 5 Aktuellen Stand der digitalen Ausgänge rücklesen

to_Link (5)
Dig_Out = from_Link2 ()
Liest den momentanen Zustand des digitalen Ausgangsregisters ein.

Funktion 253 Freien Speicher der ADwin-Karte abfragen

to_Link (254)
Freemen = from_Link4 () / Zusammenhängender freier Speicher der ADwin-Karte in kB */*

Funktion 254 Auslastung der ADwin-Karte abfragen

to_Link (254)
Auslastung = from_Link2 () / Auslastung des Prozessors der ADwin-Karte in % */*

Funktion 255 Testen, ob das richtige Programm auf die *ADwin*-Karte geladen ist

to_Link (255)
Status = from_Link ()
Das aktuelle Meßprogramm für die ***ADwin***-Karte schickt den Wert **119** zürck

2. Funktionen zur Steuerung zyklischer I/O-Prozesse

Funktion 32 I/O-Prozess starten

to_link (32)
to_link (Prozessnummer)
Prozessnummer 1 = Meßprozess
8 = Prozess zur analogen Ausgabe
32 = Digitaler PID-Regler
Startet den I/O-Prozess mit den vorher gewählten Parametern

Funktion 33 I/O-Prozess stoppen

to_link (33)
to_link (Prozessnummer)
Prozessnummer 1 = Meßprozess
8 = Prozess zur analogen Ausgabe
32 = Digitaler PID-Regler
Stoppt den I/O-Prozess

Funktion 34 Parameter für den I/O-Prozess setzen

to_link (34)
to_link2 (Parameternummer)
to_link4 (Wert)
Parameter 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 = überschreiten, 2 = unterschreiten)
6 = Triggerkanal
7 = Triggerschwelle
8 = Pretriggeranteil (in Meßpunkten)
Parameter 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 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

Funktion 35 Ausgabeliste vorgeben

to_link (35)
to_link (DAC-Nummer)
to_link4 (Länge)
to_link2 (Ausgabewert)
. . . .Länge mal
Schreibt den übergebenen Werte in die Ausgabeliste für das gewünschte DAC

Funktion 36 Einen Wert in die Ausgabeliste schreiben

to_link (36)
to_link (DAC-Nummer)
to_link4 (Position)
to_link2 (Ausgabewert)
Schreibt den übergebenen Wert in die Ausgabeliste für das gewünschte DAC an die gewählte Position

Funktion 38 Prozessstatusparameter abfragen

to_link (38)
to_link2 (Parameternummer)
Parameternummer 32 = Meßprozess läuft (1 = läuft, 0 = läuft nicht)
34 = Nummer der letzten Messung
40 = Ausgabeprozess läuft (1 = läuft, 0 = läuft nicht)
64 = PID-Reglerprozess läuft (1 = läuft, 0 = läuft nicht)
Parameterwert := from_link4()
Schreibt den übergebenen Werte in die Ausgabeliste für das gewünschte DAC

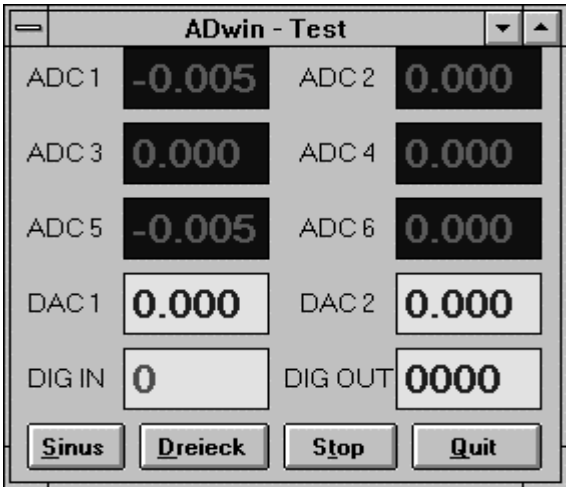
Funktion 41 Messwerteliste abholen

to_link (41)
to_link (Eingangsnummer)
to_link4 (Anzahl der Messwerte)
Messwert := from_link2()
. . . .Anzahl mal
Schickt die gewünschte Anzahl von Messwerten des ausgewählten analogen Eingangs, beginnend mit dem ersten gemessen Wert, an den PC.

Funktion 42 Einen Messwert abholen

to_link (42)
to_link (Eingangsnummer)
to_link4 (Messwertenummer)
Messwert := from_link2()
Schickt den gewünschten Messwert des ausgewählten analogen Eingangs an den PC.

Beispielprogramm ADtest zur Anzeige der analogen Eingänge



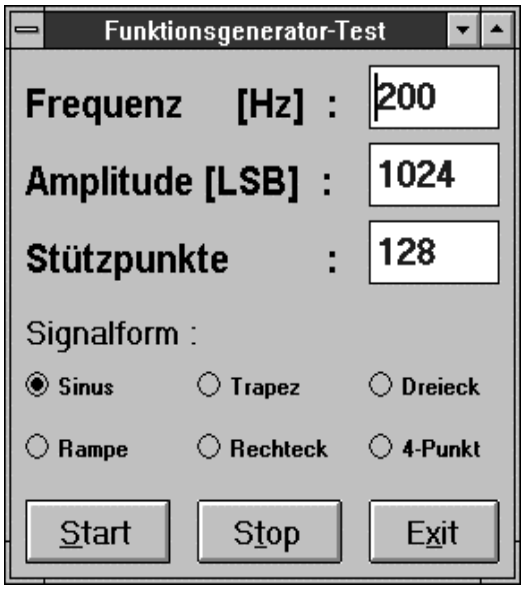
Das Programm ADtest mißt mit der Timerfunktion **Timer1** einmal pro Sekunde die analogen Eingänge 1 bis 6, rechnet die gemessenen ADC-Werte in Volt um zeigt die Spannungen an. In der Timerfunktion werden außerdem die digitalen Eingänge eingelesen und im Hexadezimalformat angezeigt.
Mit den Eingabefenstern **wDAC1** und **wDAC2** können die analogen Ausgänge gesetzt werden. Bei jeder Änderung des eingetragenen Wertes, wird die gewünschte Spannung in DAC-Werte umgerechnet und an die ADwin-Karte geschickt. Genauso können mit dem **digout**-Fenster die digitalen Ausgänge gesetzt werden.
Die Tasten **SINUS** und **Dreieck** starten die Ausgabe eines Sinus bzw. Dreieckssignals mit 10 Hz auf den analogen Ausgängen 1 und 2.

```
Sub digout_KeyPress (KeyAscii As Integer) : Rem Eingabe im DIG OUT Fenster
  If (KeyAscii = 13) Then ' falls  RETURN  gedrückt wurde
    ih1% = Val(digout.Text): Rem Den aktuellen Wert Lesen
    Call set_dig_out(ih1%): Rem und den Ausgabewert an die ADwin-Karte schicken
    End If
End Sub

Sub digout_LostFocus () :  Rem falls das DIG OUT Fenster verlassen wird
  ih1% = Val(digout.Text): Rem Den Wert lesen
  Call set_dig_out(ih1%):  Rem und den Ausgabewert an die ADwin-Karte schicken
  End Sub

Sub Dreieck_Click () : Rem Gibt auf den analogen Ausgängen ein Dreieck aus
  Call set_zykl_par(32, 100): Rem Alle 100 Mikrosekunden einen Wert ausgeben
  Call set_zykl_par(34, 1000): Rem Ein Ausgabezyklus hat 1000 Punkte
  Call set_zykl_par(35, 10000): Rem Der Zyklus soll 10000 mal wiederholt werden
  Call set_zykl_par(36, 3):Rem Auf beiden DACs soll ein Signal ausgegeben werden
  Call set_signalform_dreieck(1, 1000, 0, 4095): Rem Dreieckverlauf 1 berechnen
  Call set_signalform_dreieck(2, 1000, 0, 4095): Rem Dreieck für DAC2 berechnen
  Rem Es wird ein Dreieckverlauf mit 1000 Stützstellen, einem Minimalwert von 1
  Rem und einem Maximalwert von 4095 berechnet und an die ADwin-Karte geschickt
  Call Start_Out_Zykl: Rem Ausgabe Starten
End Sub

Sub Quit_Click () : Rem Beendet das Programm
  End
End Sub

Sub Sinus_Click () : Rem Gibt auf den analogen Ausgängen ein Sinussignal aus
  Call set_zykl_par(32, 100): Rem Alle 100 Mikrosekunden einen Wert ausgeben
  Call set_zykl_par(34, 1000): Rem Ein Ausgabezyklus hat 1000 Punkte
  Call set_zykl_par(35, 10000): Rem Der Zyklus soll 10000 mal wiederholt werden
  Call set_zykl_par(36, 3):Rem Auf beiden DACs soll ein Signal ausgegeben werden
  Call set_signalform_sinus(1, 1000, 0, 4095): Rem Dreieckverlauf 1 berechnen
  Call set_signalform_sinus(2, 1000, 0, 4095): Rem Dreieck für DAC2 berechnen
  Rem Es wird ein Sinusverlauf  mit 1000 Stützstellen , einem Minimalwert von 0
  Rem und einem Maximalwert von 4095 berechnet und an die ADwin-Karte geschickt
  Call Start_Out_Zykl: Rem Ausgabe Starten
End Sub
Sub StopSig_Click ():Rem Stoppt die zyklische Ausgabe auf den analogen Ausgängen
  Call stop_out_zykl
End Sub

Sub Timer1_Timer () : Rem Das Timerprogramm wird einmal pro Sekunde aufgerufen.
```

```
  Rem Es mißt alle analogen und Digitalen Eingänge und zeigt die Ergebnisse an.
  ih1% = ADC(1):                      Rem Den analogen Eingang 1 messen
  f1! = (ih1% - 2048) / 204.8:        Rem Das Ergebniss in Volt umrechnen
  wa00.Text = Format$(f1!, "0.000"):  Rem und in das Anzeigefenster schreiben
  ih1% = ADC(2)
  f1! = (ih1% - 2048) / 204.8:        Rem Die analogen Eingänge 2 bis 6 genauso
  wa10.Text = Format$(f1!, "0.000")
  ih1% = ADC(3)
  f1! = (ih1% - 2048) / 204.8
  wa01.Text = Format$(f1!, "0.000")
  ih1% = ADC(4)
  f1! = (ih1% - 2048) / 204.8
  wa11.Text = Format$(f1!, "0.000")
  ih1% = ADC(5)
  f1! = (ih1% - 2048) / 204.8
  wa02.Text = Format$(f1!, "0.000")
  ih1% = ADC(6)
  f1! = (ih1% - 2048) / 204.8
  wa12.Text = Format$(f1!, "0.000")
  ih1% = Dig_In():                    Rem Die digitalen Eingänge einlesen
  digin.Text = Hex$(ih1%):            Rem und im Hexadezimalformat anzeigen
End Sub

Sub wDAC1_KeyPress (KeyAscii As Integer)
  If (KeyAscii = 13) Then ' Falls RETURN gedrückt wurde
    fh1! = Val(wDAC1.Text): Rem Den neuen Wert auslesen
    f1! = fh1! * 204.8 + 2048: Rem Von Volt in DAC-Werte umrechnen
    ih1% = f1!
    Call set_dac(1, ih1%): Rem und auf DAC 1 ausgeben
    End If
End Sub

Sub wDAC1_LostFocus ()     ' Falls das wDAC1-Fenster verlassen wird
  ih1% = Val(wDAC1.Text):      Rem Den neuen Wert auslesen
  f1! = fh1! * 204.8 + 2048: Rem Von Volt in DAC-Werte umrechnen
  ih1% = f1!
  Call set_dac(1, ih1%):     Rem und auf DAC 1 ausgeben
End Sub

Sub wDAC2_KeyPress (KeyAscii As Integer)
  If (KeyAscii = 13) Then    ' Falls RETURN gedrückt wurde
    fh1! = Val(wDAC2.Text):    Rem Den neuen Wert auslesen
    f1! = fh1! * 204.8 + 2048: Rem Von Volt in DAC-Werte umrechnen
    ih1% = f1!
    Call set_dac(2, ih1%):     Rem und auf DAC 2 ausgeben
    End If
End Sub

Sub wDAC2_LostFocus ()     ' Falls das wDAC2-Fenster verlassen wird
  ih1% = Val(wDAC2.Text):    Rem Den neuen Wert auslesen
  f1! = fh1! * 204.8 + 2048: Rem Von Volt in DAC-Werte umrechnen
  ih1% = f1!
  Call set_dac(2, ih1%):     Rem und auf DAC 2 ausgeben
End Sub
```

Beispielprogramm Fgen zur schnellen Signalerzeugung

Das Programm Fgen gibt eine wählbare Signalform zyklisch aus.
Die Signalform kann mit dem Option-Buttom **fgsig** ausgewählt werden.
In den Textboxen **wfreq, wamp** und **wsteps** können die Amplitude (in ADC-Werten), die Frequenz 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.




```
Dim freq!
Dim nsteps%, amp%, isig%

Sub fgexit_Click () : Rem Programmende
  End
End Sub

Sub fgsig_Click (index As Integer) :        Rem Auswahl der Signalform
  isig% = index
  If index = 4 Then
    nsteps% = 2
    wsteps.Text = Format$(nsteps%, "####"): Rem neue Schrittzahl
    End If:                                 Rem auf den Schirm schreiben
  If index = 5 Then
    nsteps% = 4
    wsteps.Text = Format$(nsteps%, "####")
    End If
End Sub

Sub fgstart_Click () :       Rem Unterprogramm zum Starten der Ausgabe
  delay& = 1000000 / (freq! * nsteps%):  Rem Delay zwischen 2 Ausgaben
  Call set_zykl_lpar(32, delay&):        Rem Delay übergeben
  Call set_zykl_par(34, nsteps%):        Rem Anzahl der Schritte/Zyklus
  Call set_zykl_lpar(35, 100000000):     Rem Anzahl der Zyklen setzen
  Call set_zykl_par(36, 1):              Rem Nummer des ausgebenden DACs
  mi% = 2048 - amp%
  ma% = 2048 + amp%
  If isig% = 0 Then                       ' falls Signalform Sinus
    Call set_signalform_sinus(1, nsteps%, mi%, ma%): Rem Signalform
    End If
  If isig% = 1 Then                      'falls Signalform Trapez
    Call set_signalform_trapez(1, nsteps%, mi%, ma%)
    End If
  If isig% = 2 Then                      ' falls Signalform Dreieck
    Call set_signalform_dreieck(1, nsteps%, mi%, ma%)
    End If
  If isig% = 3 Then                      'falls Signalform Rampe
    Call set_signalform_rampef(1, nsteps%, mi%, ma%)
    End If
  If isig% = 4 Then                      ' falls Signalform Rechteck
    delay& = 1000000 / (freq! * 2)
    Call set_zykl_lpar(32, delay&)
    Call set_zykl_par(34, 2)
    Call set_zykl_lpar(35, 100000000)
    Call set_zykl_par(36, 1)
    Call Set_DAC_List(1, 0, mi%)
    Call Set_DAC_List(1, 1, ma%)
```

```
    End If
  If isig% = 5 Then                      ' falls Signalform 4-Punkt
    delay& = 1000000 / (freq! * 4)
    Call set_zykl_lpar(32, delay&)
    Call set_zykl_par(34, 4)
    Call set_zykl_lpar(35, 100000000)
    Call set_zykl_par(36, 1)
    mh% = (ma% + mi%) / 2
    Call Set_DAC_List(1, 0, mi%)
    Call Set_DAC_List(1, 1, mh%)
    Call Set_DAC_List(1, 2, ma%)
    Call Set_DAC_List(1, 3, mh%)
    End If
  Call Start_Out_Zykl:                    Rem Ausgabe Starten
End Sub

Sub fgstop_Click () : Rem Ausgabe Abbrechen
  Call Stop_Out_Zykl
End Sub

Sub Form_Load () : Rem Wird bei Programmstart ausgeführt und
  freq! = 200    : Rem initialisiert die Variablen
  nsteps% = 128
  amp% = 1024
  isig% = 0
  wfreq.Text = Format$(freq!, "#####")
  wamp.Text = Format$(amp%, "####")
  wsteps.Text = Format$(nsteps%, "####")
  fgsig(isig%).Value = True
End Sub

Sub wamp_Change () : Rem Wird ausgeführt wenn ein neuer Amplitudenwert
                     Rem eingegeben wurde
  amp% = Val(wamp.Text)
  If amp% < 0 Then amp% = 0
  If amp% > 2047 Then amp% = 2047
  wamp.Text = Format$(amp%, "####")
End Sub

Sub wfreq_LostFocus () : Rem wenn eine neue Frequenz eingegebn wurde
  freq! = Val(wfreq.Text)
  If freq! < 1 Then freq! = 1
  If (freq! * nsteps%) > 50000 Then freq! = 50000 / nsteps%
  wfreq.Text = Format$(freq!, "#####")
End Sub

Sub wsteps_LostFocus () : Rem wenn eine neue Schrittzahl eingegeben wurde
  nsteps% = Val(wsteps.Text)
  If nsteps% < 2 Then nsteps% = 2
  If nsteps% > 4096 Then nsteps% = 4096
  If (freq! * nsteps%) > 50000 Then nsteps% = 50000 / freq!
  wsteps.Text = Format$(nsteps%, "####")
End Sub
```

3. Erweiterte Funktionen zur Steuerung von ADbasic

Funktion 32 ADbasic-Prozess starten

to_link (32)
to_link (Prozessnummer)

Prozessnummer	101	=	ADbasic Prozess 1 mit hoher Priorität starten
	102	=	ADbasic Prozess 1 mit niedriger Priorität starten
	103	=	ADbasic Prozess 2 mit hoher Priorität starten
	104	=	ADbasic Prozess 2 mit niedriger Priorität starten
	. . .		usw.

Startet einen ADbasic-Prozess

Funktion 33 ADbasic-Prozess stoppen

to_link (33)
to_link (Prozessnummer)

Prozessnummer	101	oder	102	=	ADbasic Prozess 1 stoppen
	103	oder	104	=	ADbasic Prozess 2 stoppen
	105	oder	106	=	ADbasic Prozess 3 stoppen
	107	oder	108	=	ADbasic Prozess 4 stoppen

Stoppt einen ADbasic Prozess

Funktion 34 Parameter für ADbasic setzen

to_link (34)
to_link2 (Parameternummer)
to_link4 (Wert)
Parameter für ADbasic :

Parameternummer	1001	=	Par_1
	. . . .		
	1080	=	Par_80
	1101	=	Fpar_1
	. . . .		} nur bei Verwendung des T805 Prozessors
	1180	=	Fpar_80
	901	=	Prozess_1 running
	902	=	Prozess_2 running
	903	=	Prozess_3 running
	904	=	Prozess_4 running
	911	=	Prozess_1 Delay (in Mikrosekunden)
	912	=	Prozess_2 Delay (in Mikrosekunden)
	. . .		
	931	=	Prozess_1 Activate
	932	=	Prozess_2 Activate

Setzt den Parameter auf den gewünschten Wert

Funktion 38 Parameter von ADbasic lesen

to_link (38)
to_link2 (Parameternummer)
Parameter für ADbasic :
Parameternummer 1001 = Par_1
. . . . (Siehe Funktion 34)
Parameterwert := from_link4()
Liest den gewünschten Parameter

Funktion 105 Array als Datensatz an ADbasic schicken

to_Link (105)
to_Link (Datensatznummer)
to_Link4 (Anzahl der Elemente)
from_Link (Datentyp)
Datentyp 2 = SHORT
3 = LONG
4 = LONG
5 = FLOAT
falls Datentyp = 2:
to_Link2 (Element)
. . . .Anzahl mal
sonst:
to_Link4 (Element)
. . . .Anzahl mal
Schreibt die Elemente des übergeben Arrays in den gewünschten ADbasic Datensatz

Funktion 107 Datensatz von ADbasic in ein Array übernehmen

to_Link (107)
to_Link (Datensatznummer)
to_Link4 (Position des 1. zu übertrageden Elementes)
to_Link4 (Anzahl der Elemente)
from_Link (Datentyp)
Datentyp 2 = SHORT
3 = LONG
4 = LONG
5 = FLOAT
falls Datentyp = 2
From_Link2 (Element)
. . . .Anzahl mal
sonst
From_Link4 (Element)
. . . .Anzahl mal
Schreibt die Elemente des spezifizierten ADbasic Datensatzes in ein Array

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, Seite 22)

Funktion 110 FIFO von ADbasic in ein Array übernehmen

Vor dem Aufruf dieser Funktion sollte mit der **Funktion 112** überprüft werden, ob noch genügend Elemente im FIFO sind.

to_Link (110)
to_Link (FIFO-Nummer)
to_Link4 (Anzahl der Elemente)
from_Link (Datentyp)
Datentyp 2 = SHORT
3 = LONG
4 = LONG
5 = FLOAT
falls Datentyp = 2
From_Link2 (Element)
. . . .Anzahl mal
sonst
From_Link4 (Element)
. . . .Anzahl mal

Übernimmt die Elemente des gewünschten ADbasic FIFO in ein Array

Funktion 111 Array als FIFO an ADbasic schicken

Vor dem Aufruf dieser Funktion sollte mit der **Funktion 113** überprüft werden, ob noch genügend Platz im FIFO ist.

to_Link (111)
to_Link (FIFO-Nummer)
to_Link4 (Anzahl der Elemente)
from_Link (Datentyp)
Datentyp 2 = SHORT
3 = LONG
4 = LONG
5 = FLOAT
falls Datentyp = 2:
to_Link2 (Element)
. . . .Anzahl mal
sonst:
to_Link4 (Element)
. . . .Anzahl mal

Schreibt die Elemente des übergebenen Arrays in das gewünschte ADbasic FIFO

Funktion 112 Anzahl der Elemente im FIFO ermitteln

to_Link (112)
to_Link (FIFO-Nummer)
from_Link4 (Belegt)

Gibt die Anzahl der Elemente im ADbasic FIFO zurück

Funktion 113 Anzahl der freien FIFO Positionen ermitteln

to_Link (113)
to_Link (FIFO-Nummer)
from_Link4 (Frei)
Gibt den freien Speicherplatz im ADbasic FIFO zurück

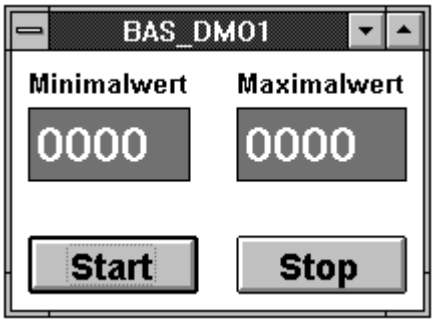
Funktion 114 Inhalt des FIFO löschen

to_Link (114)
to_Link (FIFO-Nummer)
Löscht den Inhalt des spezifizierten ADbasic FIFOs

Beispiele zum Datenaustausch mit ADbasic Prozessen

Beispielprogramm BAS_DMO1 Online-Auswertung von Meßwerten

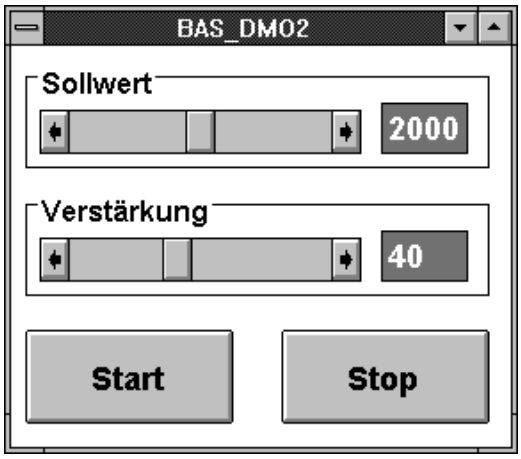
Das Programm BAS_DMO1 prüft mit der Timerfunktion **Timer1** einmal pro Sekunde, ob der ***ADbasic***-Prozess1 das ACTIVATE_PC Flag gesetzt hat. Ist dies der Fall, so holt sich das Programm die beiden ***ADbasic***-Parameter PAR_1 sowie PAR_2 und zeigt sie in den Fenstern für Minimal- und Maximalwert 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**.




```
Sub Form_Load ()
Rem Beim Starten des Programms werden automatisch ADwin8.btl und der
Rem Basic-Prozess geladen
  ih% = iserver("c:\adwin\adwin8.btl", 336, 400000)
  ih% = starte("c:\adwin\bas_dmo1.t81", 336)
                      rem Achtung!! Der Pfad muß stimmen !
End Sub

Sub Start_Click ()
  Call Start_Proz(1, 0):     Rem ADbasic-Prozess1 starten
  Timer1.Enabled = True:     Rem Timer1 aktivieren
End Sub


Sub Stop_Click ()
  Timer1.Enabled = False:    Rem Timer1 deaktivieren
  Call Stop_Proz(1):         Rem ADbasic-Prozess1 anhalten
End Sub


Sub Timer1_Timer ()
  If Activate(1) Then '      Rem falls der ADbasic-Prozess1 das ACTIVATE_PC Flag
                             Rem gesetzt hat
    Min.Text = get_par(1):   Rem ADbasic-Parameter1 abholen und anzeigen
    Max.Text = get_par(2):   Rem ADbasic-Parameter2 abholen und anzeigen
  End If
End Sub
```

Beispielprogramm BAS_DMO2 Online-Vorgabe von Regelgrößen

Das Programm BAS_DMO2 wird verwendet um einem auf der ADwin-Karte als Prozess1 laufenden digitalen P-Regler den Sollwert, sowie die Verstärkung vorzugeben.
Der digitale P-Regler wurde unter Adbasic implementiert und muß vor dem Start von BAS_DMO2 auf die ADwin-Karte geladen werden.
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.



ACHTUNG !

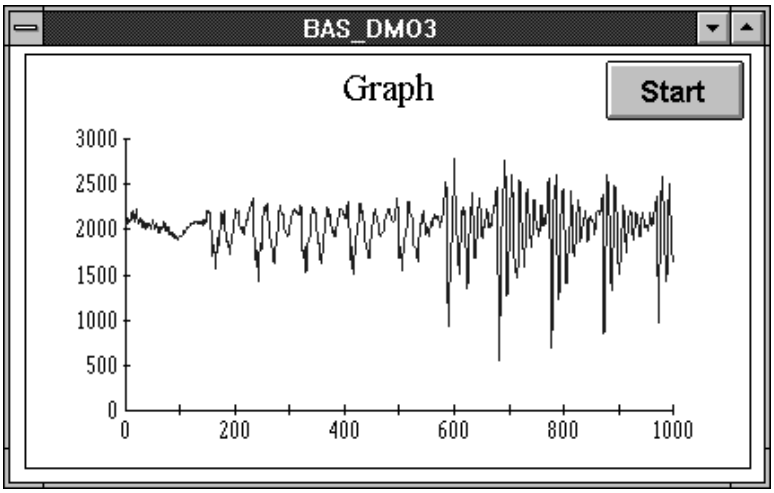
Vor dem Start des Programms BAS_DMO2 muß der ***ADbasic***-Prozess BAS_DMO2 auf die ***ADwin***-Karte geladen werden. Dies geschieht mit der folgenden DOS Anweisung:

adbload *BAS_DMO2.t81* (bzw. *BAS_DMO2.t41* o. *BAS_DMO2.t21*, je nach Prozessortyp)

```
Sub Start_Click ()
  SollHS.Enabled = True:     Rem Sollwert Scrollbar aktivieren
  VerstHS.Enabled = True:    Rem Verstärkung Scrollbar aktivieren
  Call Start_Proz(1, 0):     Rem ADbasic-Prozess1 starten
End Sub

Sub Stop_Click ()
  Call Stop_Proz(1):         Rem ADbasic-Prozess1 starten
  SollHS.Enabled = False:    Rem Sollwert Scrollbar deaktivieren
  VerstHS.Enabled = False:   Rem Verstärkung Scrollbar deaktivieren
End Sub

Sub VerstHS_Change ()
  Verst& = VerstHS.Value
  VerstAnz.Text = Verst&:    Rem Scrollbar-Wert anzeigen
  Call set_par(2, Verst&):   Rem ADbasic PAR_2 auf Scrollbar-Wert setzen
End Sub

Sub SollHS_Change ()
  Soll& = SollHS.Value
  SollAnz.Text = Soll&:      Rem Scrollbar-Wert anzeigen
  Call set_par(1, Soll&):    Rem ADbasic PAR_1 auf Scrollbar-Wert setzen
End Sub
```

Beispielprogramm BAS_DMO3 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.



ACHTUNG !

Vor dem Start des Programms BAS_DMO3 muß der ***ADbasic***-Prozess BAS_DMO3 auf die ***ADwin***-Karte geladen werden. Dies geschieht mit der folgenden DOS Anweisung:

adbload *BAS_DMO3.t81* (bzw. *BAS_DMO3.t41* o. *BAS_DMO3.t21*, je nach Prozessortyp)

```
Dim Adbdata(999)              Rem Visual-Basic Array für die Meßwertreihe


Sub Start_Click ()
  Call Start_Proz(1, 0):      Rem ADbasic-Prozess1 starten
  Timer1.Enabled = True:      Rem Timer1 aktivieren
End Sub


Sub Timer1_Timer ()
  If Activate(1) Then '                     Falls der ADbasic Prozess1 das
                                            Rem ACTIVATE_PC Flag gesetzt hat
    Call get_data(1, 1, 1000, Adbdata()):   Rem Die ersten 1000 Elemente des
                                            Rem ADbasic Datensatzes data1 holen
    For i% = 0 To 999
      graph1.GraphData = Adbdata(i%):       Rem Meßwert an Graph übergeben
    Next i%
    graph1.DrawMode = 2:                    Rem Graph darstellen
  End If
End Sub
```

Funktionen zur Steuerung der ADwin-Karte mit VISUAL BASIC

1. Die Funktionen von ADwin.dll zur Kommunikation mit der ADwin-Karte

Die DLL **ADwin.dll** enthält Funktionen zum Austausch von Daten mit dem Meßprogramm auf der ADwin-Karte. Diese Funktionen werden in der Datei **ADwin-bas** deklariert.
Die Subroutine Funktion **tolink** schickt ein Byte über die Linkschnittstelle an den Transputer auf der ADwin-Karte. Dazu wird zunächst das Statusregister des Linkadapters ausgelesen. Sobald im niederwertigsten Bit des Statusregisters eine 1 steht wird kann ein Byte ins Datenregister geschrieben werden und wird dann an den Transputer geschickt. Die Subroutinen **tolink2** und **tolink4** schicken nach dem gleichen Verfahren 2 oder 4 Byte an die ADwin-Karte.

```
Rem 1. Linkkommunikationsfunktionen der DLL ADwin
Rem ============================================
Rem Achtung! die Datei "ADWIN.DLL" muß in einem Subdirectory stehen, das
Rem Windows findet ( am besten in " \windows")
Rem Um diese Funktionen nutzen zu können muß_ vorher das Programm
Rem "ADwin8.btl" in den Transputer geladen werden!

Rem Die DLL ADwin.dll enthält die Funktionen zum Datenaustausch zwischen dem
Rem PC und dem Transputer auf der ADwin-Karte.
Rem To_link sendet 1 Byte an den Transputer.To_link2 sendet 2 und to_link4 4.
Rem Entspechend holen die from_link Funktionen Daten vom Transputer ab.
Declare Function to_link Lib "adwin.dll" (ByVal ladr%, ByVal iw%) As Integer
Declare Function to_link2 Lib "adwin.dll" (ByVal ladr%, ByVal iw%) As Integer
Declare Function to_link4 Lib "adwin.dll" (ByVal ladr%, ByVal iw&) As Integer
Declare Function to_linkf Lib "adwin.dll" (ByVal ladr%, ByVal iw!) As Integer
Declare Function from_link Lib "adwin.dll" (ByVal ladr%) As Integer
Declare Function from_link2 Lib "adwin.dll" (ByVal ladr%) As Integer
Declare Function from_link4 Lib "adwin.dll" (ByVal ladr%) As Long
Declare Function from_linkf Lib "adwin.dll" (ByVal ladr%) As Single
Declare Function starte Lib "adwin.dll" (ByVal dateiname As String, ByVal
Linkaddress As Integer) As Integer
Declare Function iserver Lib "adwin.dll" (ByVal dateiname As String, ByVal
Linkaddress As Integer, ByVal iboardsize As Long) As Integer
Declare Function starte Lib "adwin.dll" (ByVal dateiname As String, ByVal
Linkaddress As Integer) As Integer

Global Const linkadr% = 336: Rem Default Linkadresse auf 0x150 setzen
```

2. Oft benötigte Meß- und Steuerungsfunktionen

Die Datei **ADwin.bas** enthält außerdem noch vorgeferetigte Meßfunktionen, die eine einfache und übersichtliche Programmierung der Meßaufgabe ermöglichen. Diese Funktionen bauen auf den DLL-Funktionen fromlink und tolink auf. Sie senden die in der Auflistung der Meßfunktionen beschriebenen Funktionsaufrufe und Parameter an die ADwin-Karte und lesen die Ergebnisse ein

```
Rem Überprüft ob der Prozess "PNo%" den Adbasic-Befehl Activate_PC
Rem ausgeführt hat (Activate_PC Flag gesetzt)
Function Activate (PNo%) As Long
  id% = to_link(linkadr%, 38)
  id% = to_link2(linkadr%, PNo% + 930)
  Activate = from_link4(linkadr%)
  id% = to_link(linkadr%, 34)
  id% = to_link2(linkadr%, PNo% + 930)
  id% = to_link4(linkadr%, 0)
End Function


Rem Einmaliges Messen des analogen Eingangs mit der Nummer "NADC%"
Rem der Meßwert steht in den niedrigsten 12 Bits des Rückgabewerts
Function ADC (NADC%) As Integer
  id% = to_link(linkadr%, 1)
  id% = to_link(linkadr%, NADC%)
  ADC = from_link2(linkadr%)
End Function


Rem Fragt die momentane Auslastung des Transputers ab.
Rem Rückgabewert ist die Auslastung in Prozent.'
Function auslast () As Integer
  id% = to_link(linkadr%, 254)
  auslast = from_link2(linkadr%)
End Function


Rem Abfragen der digitalen Eingänge
Rem Die Eingänge stehen bitweise in einem 16-Bit-Wort'
Function Dig_In () As Integer
  id% = to_link(linkadr%, 3)
  Dig_In = from_link2(linkadr%)
End Function


Rem Rücklesen des auf den digitalen Ausgängen ausgegbenen Wertes
Function Dig_Out () As Integer
  id% = to_link(linkadr%, 5)
  Dig_Out = from_link2(linkadr%)
End Function
```

```
Rem Fragt den momentane freien Speicher des Transputers ab.
Rem Rückgabewert ist der freie Speicher in Byte.
Function freemem () As Long
  id% = to_link(linkadr%, 253)
  freemem = from_link4(linkadr%)
End Function
Rem Liest aus dem Adbasic Datensatz mit der Nummer "DNo%"
Rem "count&" Elemente (ab der Position "first&"
Sub get_data (DNo%, first&, count&, Dest())
id% = to_link(linkadr%, 107)
  id% = to_link(linkadr%, DNo%)
  id% = to_link4(linkadr%, first&)
  id% = to_link4(linkadr%, count&)
  typ% = from_link(linkadr%)
  If typ% = 2 Then
    For i1& = 0 To (count& - 1)
      Dest(i1&) = from_link2(linkadr%)
    Next i1&
  ElseIf typ% = 4 Then
    For i1& = 0 To (count& - 1)
      Dest(i1&) = from_link4(linkadr%)
    Next i1&
  ElseIf typ% = 5 Then
    For i1& = 0 To (count& - 1)
      Dest(i1&) = from_linkf(linkadr%)
    Next i1&
  End If
End Sub


Rem Liefert den Wert des Adbasic Integer Parameters mit der Nummer "PNo%"
Function get_par (PNo%) As Long
  id% = to_link(linkadr%, 38)
  id% = to_link2(linkadr%, PNo% + 1000)
  get_par = from_link4(linkadr%)
End Function


Rem Liefert den Wert des ADbasic Float Parameters mit der Nummer: PNo%
Function get_fpar (PNo%) As Single
  id% = to_link(linkadr%, 38)
  id% = to_link2(linkadr%, PNo% + 1100)
  get_fpar = from_linkf(linkadr%)
End Function


Rem Setzen des DACs Nr. "NDAC%" auf den Wert "iw%"
Rem Der auszugebende Wert muß in den niedrigsten 12 Bits von "iw%" stehen
Sub set_dac (NDAC%, iw%)
  id% = to_link(linkadr%, 2)
  id% = to_link(linkadr%, NDAC%)
  id% = to_link2(linkadr%, iw%)
End Sub
```

```
Rem Setzt in der Liste für die zyklische Ausgabe für den DAC-Nr. "NADC%"
Rem das Element Nummer "Nstep%" auf den Wert "iw%"
Sub Set_DAC_List (NDAC%, nstep%, iw%)
  id% = to_link(linkadr%, 36)
  id% = to_link(linkadr%, NDAC%)
  nst& = nstep%
  id% = to_link4(linkadr%, nst&)
  id% = to_link2(linkadr%, iw%)
End Sub


Rem Weist dem ADbasic Datensatz mit der Nummer "DNo%" die
Rem 1. "count&" Elemente aus dem Visual-Basic Array
Rem "source()" zu
Sub set_data (DNo%, count&, source())
id% = to_link(linkadr%, 105)
  id% = to_link(linkadr%, DNo%)
  id% = to_link4(linkadr%, count&)
  typ% = from_link(linkadr%)
  If typ% = 2 Then
    For i1& = 0 To (count& - 1)
      id% = to_link2(linkadr%, source(i1&))
    Next i1&
  ElseIf typ% = 4 Then
    For i1& = 0 To (count& - 1)
      id% = to_link4(linkadr%, source(i1&))
    Next i1&
  ElseIf typ% = 5 Then
    For i1& = 0 To (count& - 1)
      id% = to_linkf(linkadr%, source(i1&))
    Next i1&
  End If
End Sub


Rem gibt den Wert der in "iw%" steht auf die digitalen Ausgänge aus
Sub set_dig_out (iw%)
  id% = to_link(linkadr%, 4)
  id% = to_link2(linkadr%, iw%)
End Sub


Rem Setzt den ADbasic Integer-Parameter mit der Nummer: PNo%
Rem auf den Wert: iw%
Sub set_par (PNo%, iw&)
  id% = to_link(linkadr%, 34)
  id% = to_link2(linkadr%, PNo% + 1000)
  id% = to_link4(linkadr%, iw&)
End Sub


Rem Setzt den ADbasic Float-Parameter mit der Nummer: PNo%
Rem auf den Wert: iw%
Sub set_fpar (PNo%, iw!)
  id% = to_link(linkadr%, 34)
  id% = to_link2(linkadr%, PNo% + 1100)
  id% = to_linkf(linkadr%, iw!)
End Sub
```

```
Rem Erzeugt in der Augabeliste für den DAC-Nr. "NDAC%" einen Dreieckverlauf
Rem mit "nsteps%" Schritten , dem Minmalwert "imin%" und dem Maximalwert
Rem "imax%", zur späteren zyklischen Ausgabe.
Sub set_signalform_dreieck (NDAC%, nsteps%, imin%, imax%)
  For i1% = 0 To (nsteps% / 2 - 1)
    id& = imax% - imin%
    ih& = i1% * 2
    nst& = nsteps%
    iw& = (ih& * id&) / nst&
    iws% = iw& + imin%
    Call Set_DAC_List(NDAC%, i1%, iws%)
    Call Set_DAC_List(NDAC%, (nsteps% - i1% - 1), iws%)
    Next i1%
End Sub


Rem Erzeugt in der AUsgabeliste für den DAC-Nr. "NDAC%" einen Sinusverlauf
Rem mit "nsteps%" Schritten , dem Minmalwert "imin%" und dem Maximalwert
Rem "imax%", zur späteren zyklischen Ausgabe.
Sub set_signalform_sinus (NDAC%, nsteps%, imin%, imax%)
  For i1% = 0 To (nsteps% - 1)
    iamp! = (imax% - imin%) / 2 - 1
    imitt! = (imax% + imin%) / 2
    fh1! = (i1% * 6.2832) / nsteps%
    si! = Sin(fh1!) * iamp!
    ih1% = si!
    Call Set_DAC_List(NDAC%, i1%, (imitt! + ih1%))
    Next i1%
End Sub


Rem Setzt den Parameter "Npar%"für die zyklische I/O-Funktionen auf "iw&"
Rem (S. Funktionen des Meßprogramms: Funktion 34)
Sub set_zykl_lpar (Npar%, iw&)
  id% = to_link(linkadr%, 34)
  id% = to_link2(linkadr%, Npar%)
  id% = to_link4(linkadr%, iw&)
End Sub


Rem Setzt den Parameter "Npar%"für die zyklische I/o-Funktionen auf "iw%"
Rem Gleiche Funktion Wwie 'set_zykl_lpar' nur mit 16 Wert
Sub set_zykl_par (Npar%, iw%)
  iwh& = iw%
  id% = to_link(linkadr%, 34)
  id% = to_link2(linkadr%, Npar%)
  id% = to_link4(linkadr%, iwh&)
End Sub


Rem Startet eine Zyklische Messung
Rem Erst aufrufen wenn alle Parameter richtig gesetzt sind.
Sub Start_Mess_Zykl ()
  id% = to_link(linkadr%, 32)
  id% = to_link(linkadr%, 1)
End Sub
```

```
Rem Startet den ADbasic Prozess mit der Nummer "PNo%"
Rem Prioritätsfestlegung:   "Pri%"  Priorität
Rem                           0       High
Rem                           1       Low
Sub Start_Proz (PNo%, Pri%)
  id% = to_link(linkadr%, 32)
  id% = to_link(linkadr%, 100 + PNo% + Pri%)
End Sub


Rem Hält den ADbasic Prozess mit der Nummer "PNo%" an
Sub Stop_Proz (PNo%)
  id% = to_link(linkadr%, 33)
  id% = to_link(linkadr%, 100 + PNo%)
End Sub


Rem Startet eine Zyklische Ausgabe
Rem Erst aufrufen wenn alle Parameter und Ausgabelisten richtig gesetzt sind.
Sub Start_Out_Zykl ()
  id% = to_link(linkadr%, 32)
  id% = to_link(linkadr%, 8)
End Sub


Rem Startet einen Pid-Regler
Rem Erst aufrufen wenn alle Parameter richtig gesetzt sind.'
Sub Start_Pid ()
  id% = to_link(linkadr%, 32)
  id% = to_link(linkadr%, 32)
End Sub


Rem Bricht eine zyklische Messung ab.
Sub Stop_Mess_Zykl ()
  id% = to_link(linkadr%, 33)
  id% = to_link(linkadr%, 1)
End Sub


Rem Bricht eine zyklische Ausgabe ab.
Sub Stop_out_Zykl ()
  id% = to_link(linkadr%, 33)
  id% = to_link(linkadr%, 8)
End Sub


Rem Bricht einen PID-Regler ab
Sub Stop_Pid ()
  id% = to_link(linkadr%, 33)
  id% = to_link(linkadr%, 32)
End Sub
```

```
Rem Auslesen des zyklisch gemessenen Messpunkts Nr. "Nstep%" vom analogen
Rem Eingang Nr. "NADC%".
Function Zykl_Wert (NADC%, nstep%) As integer
  id% = to_link(linkadr%, 42)
  id% = to_link(linkadr%, NADC%)
  nste& = nstep%
  id% = to_link4(linkadr%, nste&)
  Zykl_Wert = from_link2(linkadr%)
End Function


Rem Anzahl der Elemente im FIFO ermitteln
Function get_fifo_count (FNo%)
  id% = to_link(linkadr%, 112)
  id% = to_link(linkadr%, FNo%)
  get_fifo_count = from_link4()
End Function


Rem Anzahl der freien Positionen im FIFO ermitteln
Function get_fifo_empty (FNo%)
  id% = to_link(linkadr%, 113)
  id% = to_link(linkadr%, FNo%)
  get_fifo_empty = from_link4()
End Function


Rem Inhalt des FIFO löschen
Sub clear_fifo (FNo%)
  id% = to_link(linkadr%, 114)
  id% = to_link(linkadr%, FNo%)
End Sub


Rem Element/e aus einem ADbasic FIFO vom Transputer abholen
Sub get_fifo (FNo%, count&, Dest())
  id% = to_link(linkadr%, 110)
  id% = to_link(linkadr%, FNo%)
  id% = to_link4(linkadr%, count&)
  typ% = from_link(linkadr%)
  If typ% = 2 Then
    For i1& = 0 To (count& - 1)
      Dest(i1&) = from_link2(linkadr%)
    Next i1&
  ElseIf typ% = 4 Then
    For i1& = 0 To (count& - 1)
      Dest(i1&) = from_link4(linkadr%)
    Next i1&
  ElseIf typ% = 5 Then
    For i1& = 0 To (count& - 1)
      Dest(i1&) = from_linkf(linkadr%)
    Next i1&
  End If
End Sub
```

```
Rem Einem ADbasic FIFO die Werte aus einem Array zuweisen
Sub set_fifo (FNo%, count%, source())
  id% = to_link(linkadr%, 111)
  id% = to_link(linkadr%, FNo%)
  id% = to_link4(linkadr%, count&)
  typ% = from_link(linkadr%)
  If typ% = 2 Then
    For i1& = 0 To (count& - 1)
      id% = to_link2(linkadr%, source(i1&))
    Next i1&
  ElseIf typ% = 4 Then
    For i1& = 0 To (count& - 1)
      id% = to_link4(linkadr%, source(i1&))
    Next i1&
  ElseIf typ% = 5 Then
    For i1& = 0 To (count& - 1)
      id% = to_linkf(linkadr%, source(i1&))
    Next i1&
  End If
End Sub

Rem Läd den Treiber auf die ADwin-Karte
Rem Je nach Prozessor wird eine der folgenden Treiberdateien benötigt
Rem    T225 Prozessor -> btlfile$ = "adwin2.btl"
Rem    T400 Prozessor -> btlfile$ = "adwin4.btl"
Rem    T805 Prozessor -> btlfile$ = "adwin8.btl"
Rem Die Speichergröße (size&) wird in hexadezimaler Schreibweise angegeben
Rem Beispiele: ADwin-Karte mit 1MB -> size& = 100000
Rem              "     "       4MB -> size& = 400000
Rem              "     "       8MB -> size& = 800000
Sub iserv (btlfile$, size&)
  erg% = iserver(btlfile$, linkadr%, size&)
End Sub

Rem Lädt einen ADbasic-Prozess (von ADbasic erzeugtes Bin File) auf die ADwin-
Karte
Sub adbload (adbasicfile$)
  erg% = starte(adbasicfile$, linkadr%)
End Sub
```

Das Programm "Auslast" zum Anzeigen der Auslastung.

Das Programm Auslast zeigt die momentane Auslastung des Prozessors auf der ADwin-Karte und den freien Speicher an. Vor allem beim Entwickeln von VISUAL BASIC Anwendungen, bei denen mehrere Prozesse gleichzeitig auf der ADwin-Karte laufen sollen, ist es sehr nützlich dieses kleine Programm mitlaufen zu lassen. Damit ist immer sofort zu sehen, wie stark jedes Object die Adwin-Karte mit der momentanen Abtastrate und Buffergröße belastet.
Der **Auslastungswert** gibt an wieviel Prozent der Rechenzeit des Prozessors auf der ADwin-Karte benötigt wird. Die benötigte Rechenleistung steigt bei allen Meßprozessen linear mit der Abtastrate an.
Der Wert **Freemem** gibt an, wieviele kB zusammenhängender Datenspeicher auf der Karte noch maximal frei sind. Der benötigte Speicher wird jeweils beim Starten eines Objects allociert. Da der gesamte freie Speicher von mehreren Prozessen zerstückelt werden kann, ist der freie Speicher des Systems sehr schwer vorhersehbar. Falls bei ADwin-2 Karten nicht genügend zusammenhängender Speicher frei ist, kann die Messung nicht korrekt starten. In diesem Fall sollte man das Programm **ADwin2.btl** neu laden (geht auch im DOS-Fenster von Windows) und die Anwendung mit einem kleineren Buffer neu starten. Falls der freie Speicher für Ihre Anwendungen öfter nicht ausreichen sollte, empfehlen wir Ihnen eine **ADwin-4** oder **ADwin-8** Karte bei denen der größte Teil der 4 MB als Datenspeicher zur Verfügung stehen, so daß auch lange Messungen mit vielen Eingängen problemlos möglich sind.

ACHTUNG !

Wie bei VISUAL BASIC muß auch vor dem Starten von **Auslast** das Programm **ADwin8.btl** mit der Anweisung **run ADwin8** auf die ADwin-Karte geladen worden sein.
Das Programm Auslast benötigt die DLLs **VBRUN200.dll** und **ADwin.dll**.

ACHTUNG !

Da das Programm **Auslast** ständig auf die ADwin-Karte zugreift, muß es vor dem erneuten Laden der ADwin-Karte mit **ADwin8.btl** unbedingt abgebrochen werden. Sonst wird der Ladevorgang gestört.