JÄGER
Computergesteuerte
Messtechnik GmbH

# *ADwin*

## Visual-Basic-Treiber


**Ausgabe April 2000**

# Inhaltsverzeichnis

# 1 *ADwin*-Treiberkonzept

## 1.1 *ADwin* Betriebssystem

Das ***ADwin***-Betriebssystem befindet sich in der Datei mit der Bezeichnung ***ADwin9*.btl**[1] (ADSP bzw. T9). Nach jedem Einschalten der Spannungsversorgung muß zunächst das Betriebssystem auf das ***ADwin***-System geladen werden. Erst nach dessen erfolgreicher Übertragung ist das System in der Lage, Befehle vom PC entgegenzunehmen und Daten mit ihm auszutauschen.

Die Aufgaben des ***ADwin***-Betriebssystems sind:

- Verwaltung von bis zu 10 ***ADbasic***-Prozessen, wobei zwischen zwei frei wählbaren Prioritätsstufen unterschieden wird (siehe nachfolgende Tabelle).

| Priorität | Merkmal | Verwendungszweck |
|---|---|---|
| niedrig | kann von hoch priorisierten Prozessen unterbrochen werden | für zeitunkritische Berechnungen und langsame Messungen |
| hoch | kann von keinem anderen Prozeß unterbrochen werden | für zeitgenaue Messungen, Steuerungen und Regelungen |

- Bereitstellung von 80 vordefinierten ***ADbasic***-Integer-Variablen (PAR_1 bis PAR_80) und 80 vordefinierten ***ADbasic***-Float-Variablen (FPAR_1 bis FPAR_80). Außerdem werden 200 Datensätze mit frei definierbarer Länge bereitgestellt. Die Werte dieser Variablen bzw. Datensätze können vom PC jederzeit gelesen und geändert werden.

- Organisation und Durchführung der Kommunikation zwischen ***ADwin***-System und PC.

  Einen wesentlichen Bestandteil des ***ADwin***-Betriebssystems bildet der Kommunikationsprozeß. Dieser Prozeß läuft mit niedriger Priorität auf dem ***ADwin***-System und interpretiert bzw. bearbeitet alle Befehle, die der PC an das ***ADwin***-System richtet. Die wichtigsten Befehle lassen sich in zwei Gruppen unterteilen. In den folgenden Tabellen sind aus jeder Gruppe einige Beispiele aufgelistet.

| Steuerbefehle | |
|---|---|
| **ADBload** | überträgt einen ***ADbasic*** Prozeß auf das ***ADwin***-System |
| **ADBstart** | startet einen ***ADbasic***-Prozeß |
| | |

| Befehle für den Datenaustausch | |
|---|---|
| **get_par** | liefert den aktuellen Wert eines ***ADbasic***-Parameters |
| **set_par** | ändert den Wert eines ***ADbasic***-Parameters |
| **get_data** | liefert die Werte aus einem ***ADbasic***-Datensatz |

  Der Kommunikationsprozeß sendet niemals unaufgefordert Daten an den PC. Dadurch wird sichergestellt, daß nur dann Daten zum PC übertragen werden, wenn diese vorher explizit angefordert wurden.

[1] Bei Verwendung eines ***ADwin***-2, -4, -5 oder -8 Systems wird die Datei ***ADwin2***.btl, ***ADwin4***.btl, ***ADwin5***.btl bzw. ***ADwin8***.btl benötigt.

## 1.2 Windows-*ADwin*-Schnittstelle

Die Schnittstelle zwischen Windows und dem ***ADwin***-System bildet die **ADwin32.dll** (**D**ynamic **L**ink **L**ibrary). Alle Windows Programme, die in der Lage sind DLL-Funktionen aufzurufen, können mit dem ***ADwin***-System kommunizieren (siehe nachfolgende Abbildung).


***ADwin***-Kommunikationsschnittstelle zu Windows-Anwendungen

Die Schnittstelle bewältigt dabei folgende Aufgaben:

- Befehlsweiterleitung

  Alle Befehle, die an das ***ADwin***-System gerichtet sind, laufen über die ***ADwin32*.dll**. Dadurch ist die Schnittstelle für alle Windows Programme identisch. Die ***ADwin32*.dll** ist das einzige Modul, das mit dem ***ADwin***-System in Verbindung treten kann.

- Datentransfer

  Der Datentransfer zwischen ***ADwin***-System und PC wird in der ***ADwin32*.dll** durchgeführt. Unter Windows NT erfolgt der Zugriff über den ***ADwin***-Device-Treiber. Alle anderen Windows Versionen erlauben den direkten Zugriff auf das ***ADwin***-System.
  Da weder Interrupts noch DMA-Kanäle benötigt werden, ist ein zuverlässiger Betrieb des ***ADwin***-Systems in jedem PC problemlos möglich und aufwendige Konfigurationsprozeduren entfallen.

- Multitasking-Verwaltung

  Mit dem ***ADwin***-System können mehrere Windows Programme gleichzeitig kommunizieren (Zeitscheiben-Verfahren). Wenn ein Programm Daten mit dem ***ADwin***-System austauscht, wird der Zugriff für alle anderen Programme gesperrt. Sofort nach der Beendigung des Austausches gibt die ***ADwin32*.dll** das ***ADwin***-System für andere Anforderungen frei. Der Zeitaufwand für den Datenaustausch liegt im Bereich von wenigen Millisekunden.

# 2 Software Installation

## 2.1 Installation der *ADwin*-Treiber

Wenn Sie die Treiber schon im Zuge der ***ADbasic*** Installation aufgespielt haben, ist eine erneute Installation nicht erforderlich.
Andernfalls legen Sie die mitgelieferte CD-ROM in das Laufwerk Ihres Rechners ein. Das Setup-Programm wird normalerweise automatisch gestartet.
Sollte dies nicht der Fall sein, dann wechseln Sie bitte in das Unterverzeichnis ...\Driver\Disk1 auf der CD-ROM und starten das dort befindliche Programm **setup.exe**.
Nachdem Sie die Sprache und das Zielverzeichnis - es wird empfohlen das angebotene Verzeichnis C:***ADbasic3*** zu bestätigen - ausgewählt haben, kopiert das Setup-Programm die folgenden Dateien in das Verzeichnis Ihres Rechners:

| | |
|---|---|
| ***ADWIN2*.BTL** | Betriebssystem für die ***ADwin***-Systeme mit T225-Prozessor |
| ***ADWIN4*.BTL** | Betriebssystem für die ***ADwin***-Systeme mit T400-Prozessor |
| ***ADWIN5*.BTL** | Betriebssystem für die ***ADwin***-Systeme mit T450-Prozessor |
| ***ADWIN8*.BTL** | Betriebssystem für die ***ADwin***-Systeme mit T805-Prozessor |
| ***ADWIN9*.BTL** | Betriebssystem für die ***ADwin***-Systeme mit ADSP-Prozessor |
| **TESTVE16.EXE** | zeigt die Version der installierten 16Bit-***ADwin***-DLLs an |
| **TESTVE32.EXE** | zeigt die Version der installierten 32Bit-***ADwin***-DLLs an |
| **ADTEST.EXE** | Programm zum Testen der ***ADwin***-PC-Einsteckkarten |
| **ADPRO.EXE** | Programm zum Testen des ***ADwin-Pro***-Systems |
| ***ADWINSET*.EXE** | Programm zum Anmelden der Linkadresse unter Windows NT.<br><br>Wenn Sie eine andere Linkadresse als $150 verwenden möchten, muß die neue Adresse mit diesem Programm angemeldet werden ! |

Falls Sie unter Windows NT arbeiten, müssen Sie bei der Installation über Administrator Rechte verfügen. Nach der Treiber-Installation muß Ihr Rechner neu gestartet werden.

## 2.2 Laden des *ADwin*-Betriebssystems

Der PC kann erst mit dem ***ADwin***-System kommunizieren, nachdem das Betriebssystem geladen wurde. Das Betriebssystem wird wahlweise durch die Visual-Basic Funktion Boot oder über ***ADbasic*** geladen. Zusätzlich bietet auch das Testprogramm ***ADtest*** die Möglichkeit, das System zu laden.

### *a) Unter Visual-Basic*

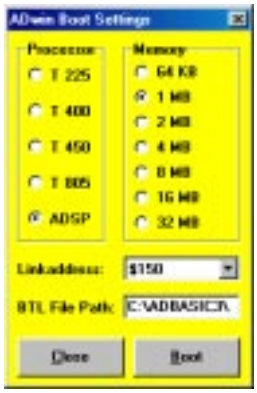
Mit der Funktion Boot ('*Systemdatei*', *Speicherausbau*) aus der Datei ***ADwin***.bas. Nähere Informationen hierzu in Kapitel 3.

### *b) Über ADbasic*

- Starten Sie ***ADbasic*** indem Sie im Windows-Menü
  Start ⇨ Programme ⇨ ***ADwin*** ⇨ **ADbasic 3.0** auswählen.
- Überprüfen Sie, ob die Einstellungen im ***ADbasic***-Menü Options ⇨ Compiler mit Ihrem ***ADwin***-System übereinstimmen.
- Klicken Sie in der ***ADbasic*** Toolbar auf das folgende Symbol: 
  (alternativ: Project ⇨ Boot ***ADwin***)

Das erfolgreiche Laden des Betriebssystems wird in der Statuszeile des ***ADbasic***-Editors durch die Meldung „***ADwin*** is booted" bestätigt.

### *c) Mit dem Windows Programm ADtest*




Das Programm ***ADtest*** dient als Funktionskontrolle für ***ADwin***-PC-Einsteckkarten und bietet ebenfalls eine Möglichkeit zum Laden des Betriebssystems. Starten Sie dazu das Programm **ADtest.exe**. Falls das Betriebssystem seit dem Einschalten des PCs noch nicht auf das ***ADwin*** System geladen wurde, erscheint die folgende Dialogbox:
Stimmen Sie alle Einstellungen auf dem von Ihnen eingesetzten ***ADwin***-System ab, und betätigen Sie anschließend die OK-Taste. Falls keine Fehlermeldung ausgegeben wird, konnte das Betriebssystem 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 das Betriebssystem nicht geladen werden konnte, läßt sich der Vorgang durch Betätigen der Settings-Taste mit geänderten Einstellungen wiederholen.

Durch das Laden des Betriebssystems werden alle Prozesse auf dem ***ADwin***-System gelöscht und alle globalen Variablen auf den Wert 0 gesetzt. Der Wert für das Globaldelay ist nach dem Laden des Betriebssystems beim ADSP auf 25000 ns und bei allen anderen Prozessoren auf 1000 µs voreingestellt.

## 2.3 Einbinden der *ADwin*-Funktionsbibliothek

Mit Hilfe der Funktionen aus der Datei ***ADwin***.dll (Windows 3.1) bzw. ***ADwin32***.dll (Windows 95, 98, NT) können Visual Basic-Programme Befehle an ***ADwin***-Systeme schicken und Daten abholen.

Diese Funktionen werden von der Datei **win.bas** importiert.

In dieser Datei befinden sich außerdem Funktionen, die eine einfache und übersichtliche Programmierung der Regel- Steuer- oder Meßaufgabe ermöglichen. Sie brauchen nur die Datei adwin.bas in Ihr Visual-Basic Projekt einbinden und können dann alle Funktionen zur Steuerung des ***ADwin***-Systems verwenden.

Hinzufügen der Datei ***ADwin***.bas in:

Visual Basic Versionen 3.0 und 4.0 über den Menüpunkt ***Datei ⇨ Datei hinzufügen*** bzw.
***File ⇨ Add File***

Visual Basic Version 5.0 und 6.0 über den Menüpunkt ***Projekt ⇨ Datei hinzufügen*** bzw.
***Project ⇨ Add File***

Die zur Verfügung stehenden Funktionen sind auf den folgenden Seiten beschrieben.

# 3 Funktionen aus der Datei *ADwin*.bas

## 3.1 Funktionen zur Steuerung von *ADbasic* Prozessen

### 3.1.1 Das *ADwin*-Betriebssystem laden

*Function* ***Boot*** *(Systemdatei$, mem&) As Long*

***Aufrufparameter:***

| *ADwin*-Typ | *Systemdatei* |
|---|---|
| ***ADwin-2*** | ***Adwin2.btl*** |
| ***ADwin-4*** | ***ADwin4.btl*** |
| ***ADwin-5*** | ***ADwin5.btl*** |
| ***ADwin-8*** | ***ADwin8.btl*** |
| ***ADwin-9*** (ADSP) | ***ADwin9.btl*** |

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

***Rückgabewert:***

| | *ADwin-9* (ADSP) | *ADwin-2, 4, 5* u. *8* |
|---|---|---|
| **1** | Fehler | Fehler |
| **8000** | OK | |
| **erkannter Speicherausbau** | | OK |

***Beispiel:***

***ADwin-9***, 4 MB Ausführung: erg = Boot ("C:***ADbasic3****ADwin9**.btl", 400000)
***ADwin-4***, 1 MB Ausführung: erg = Boot ("C:***ADbasic3****ADwin4**.btl", 100000)

### 3.1.2 Einen Prozeß laden, der von *ADbasic* mit der Funktion 'Make Bin File' erzeugt wurde

*Function* ***ADBload*** *(Dateiname$) As Integer*

***Aufrufparameter:***

*Dateiname* Name der Binärdatei, die geladen werden soll.

***Rückgabewert:***

1 OK
sonst Fehler

***Beispiel:***

erg = ADBload ("C:***ADbasic3**\samples\bas_dmo1.T91")

### 3.1.3 *ADbasic*-Prozeß starten

*Sub* ***Start_proz*** *(nr%)*

***Aufrufparameter:***

*nr* Nummer des zu startenden Prozesses (1 – 10)

***Beispiel:***

Start_proz (1) ‘startet ***ADbasic*** Prozeß Nr.: 1

### 3.1.4 *ADbasic*-Prozeß stoppen

*Sub* ***Stop_proz*** *(nr%)*

***Aufrufparameter:***

*nr* Nummer des zu stoppenden Prozesses (1 – 10)

***Beispiel:***

Stop_proz (2) ‘stoppt ***ADbasic*** Prozeß Nr.: 2

### 3.1.5 Integer Parameter für *ADbasic* setzen

*Sub **Set_Par** (Parameternummer%, Wert&)*

***Aufrufparameter:***

| Parameternummer | Bedeutung |
|---|---|
| 1 | PAR_1 |
| 2 | PAR_2 |
| usw. bis 80 | PAR_80 |
| -89 | Prozeß 1 Delay |
| -88 | Prozeß 2 Delay |
| usw. bis –80 | Prozeß 10 Delay |
| -69 | Prozeß 1 Activate |
| -68 | Prozeß 2 Activate |
| usw. bis –60 | Prozeß 10 Activate |

*Wert* Neuer Wert für den gewählten ***ADbasic*** Integer Parameter

***Beispiel:***

```
Set_Par (1,2000)        'setzt den ADbasic Parameter PAR_1 auf 2000
```

### 3.1.6 Float Parameter für *ADbasic* setzen

*Sub **Set_FPar** (Parameternummer%, Wert!)*

***Aufrufparameter:***

| Parameternummer | Bedeutung |
|---|---|
| 1 | FPAR_1 |
| 2 | FPAR_2 |
| 3 | FPAR_3 |
| usw. bis 80 | PAR_80 |

*Wert* Neuer Wert für den gewählten ***ADbasic*** Float Parameter

***Beispiel:***

```
Set_FPar (6, 34.7)          'setzt den ADbasic Parameter FPAR_6 auf 34.7
```

### 3.1.7 Den aktuellen Wert eines *ADbasic* Integer Parameters lesen

*Function* ***Get_Par*** *(nr%) As Long*

***Aufrufparameter***:

| | |
|---|---|
| nr | Siehe unter SetPar |

***Rückgabewert***:

| | |
|---|---|
| 255 | Fehler |
| sonst | aktueller Wert des gewählten Integer Parameters |

***Beispiel:***

```
erg = Get_Par (1)        'aktuellen Wert des ADbasic Parameters PAR_1 lesen
```

### 3.1.8 *ADbasic* Activate_PC Flag abfragen

*Function* ***Activate*** *(nr%) As Long*

***Aufrufparameter***:

| | |
|---|---|
| *nr* | Nummer des Prozesses (1 – 10) von dem das Flag abgefragt wird |

***Rückgabewert***:

| | |
|---|---|
| 255 | Fehler |
| 1 | Flag gesetzt |
| 0 | Flag nicht gesetzt |

***Beispiel:***

```
erg = Activate(3)        'Stellt fest ob der ADbasic Prozeß 3 den Befehl 'ACTIVATE_PC
                         ausgeführt hat
```

### 3.1.9 Float Parameter von *ADbasic* lesen

*Function* ***Get_FPar*** *(Parameternummer%) As Single*

***Aufrufparameter***:

| Parameternummer | Bedeutung |
|---|---|
| 1 | FPAR_1 |
| 2 | FPAR_2 |
| 3 | FPAR_3 |
| usw. bis 80 | PAR_80 |

***Rückgabewert***:

| | |
|---|---|
| 255 | Fehler |
| sonst | aktueller Wert des gewählten Float Parameters |

***Beispiel:***

```
erg = Get_FPar (56)      'aktuellen Wert des ADbasic Parameters FPAR_56 lesen
```

### 3.1.10 Datensatz von *ADbasic* in ein Long oder Float Array übernehmen

*Sub **Get_Data_Long** (nr%, start&, anzahl&, ziel&())*
*Sub **Get_Data_Float** (nr%, start&, anzahl&, ziel!())*

***Aufrufparameter*:**

| | |
|---|---|
| *nr* | ***ADbasic*** Datensatznummer |
| *start* | Index des 1. zu übertragenden Elementes |
| *anzahl* | Anzahl der Elemente die übertragen werden |
| *ziel* | Array für die übertragenen Werte |

***Beispiel:***

```
Get_Data_Long (2, 1, 100, ziel())     'die ersten 100 Elemente aus DATA_2 holen
```

### 3.1.11 Long oder Float Array als Datensatz an *ADbasic* schicken

*Sub **Set_Data_Long** (nr%, start&, anzahl&, quelle&())*
*Sub **Set_Data_Float** (nr%, start&, anzahl&, quelle!())*

***Aufrufparameter*:**

| | |
|---|---|
| *nr, start, anzahl* | Wie unter 4.1.10 |
| *quelle* | Array, dessen Werte in den ***ADbasic*** Datensatz übertragen werden |

***Beispiel:***

```
E rg = Set_Data_Long (3, 10, 100, quelle())     'DATA_3 Elemente 10-109 mit dem Inhalt
                                                von  'quelle belegen
```

### 3.1.12 Daten aus einem *ADbasic* Datensatz direkt auf der Festplatte speichern

*Function **Data2File** (dateiname$, nr%, start&, anzahl&, mode%) As Integer*

***Aufrufparameter:***

| mode | Bedeutung |
|---|---|
| 0 | Vorhandene Datei wird überschrieben |
| 1 | Falls die Datei bereits existiert, werden die Daten angehängt |

| | |
|---|---|
| *dateiname* | Name der Datei, in der die Daten gespeichert werden |
| *nr* | ***ADbasic*** Datensatz Nummer |
| *start* | Index des 1. zu speichernden Elementes |
| *anzahl* | Anzahl der Elemente die gespeichert werden |

***Rückgabewert:***

| | |
|---|---|
| 0 | Fehler |
| sonst | OK |

***Beispiel:***

```
erg = Data2File("Test.dat", 1, 1, 1000, 0)   'Speichert Element 1-1000 aus ADbasic
                                             'Datensatz DATA_1 in der Datei Test.dat
```

## 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)

### 3.1.13 FIFO von *ADbasic* in ein Long oder Float Array übernehmen

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

*Sub **Get_Fifo_Long** (Fnr%, anzahl&, ziel&())*
*Sub **Get_Fifo_Float** (Fnr%, anzahl&, ziel!())*

***Aufrufparameter*:**

| | |
|---|---|
| *Fnr* | ***ADbasic*** Fifo Datensatznummer |
| *anzahl* | Anzahl der Elemente die übertragen werden |
| *ziel* | Array für die übertragenen Werte |

***Beispiel:***

```
erg = Get_Fifo_Long (2, 200, ziel()) '100 Elemente aus DATA_2 (AS FIFO)  holen
```

### 3.1.14 Long oder Float Array als FIFO an *ADbasic* schicken

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

*Sub **Set_Fifo_Long** (nr%, start&, anzahl&, quelle&())*
*Sub **Set_Fifo_Float** (nr%, start&, anzahl&, quelle!())*

***Aufrufparameter:***

| | |
|---|---|
| *Fnr* | ***ADbasic*** Fifo Datensatznummer |
| *anzahl* | Anzahl der Elemente die übertragen werden |
| *quelle* | Array, dessen Werte in den ***ADbasic*** FIFO übertragen werden |

***Beispiel:***

```
erg = Set_Fifo_Float (12, 1000, quelle())  '1000 Elemente aus quelle an ADbasic DATA_12
                                           '(AS FIFO)  übertragen
```

### 3.1.15 Anzahl der Elemente im FIFO ermitteln

*Function **Get_Fifo_Count** (DATAnr%) As Long*

***Aufrufparameter:***

| | |
|---|---|
| *DATAnr* | Nummer des als FIFO deklarierten ***ADbasic*** DATAs |

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | Anzahl der im FIFO befindlichen Elemente |

***Beispiel:***

```
erg = Get_Fifo_Count (2)      'Ermittelt die Anzahl der Elemente in DATA_2 (AS FIFO)
```

### 3.1.16 Anzahl der freien FIFO Positionen ermitteln

*Function **Get_Fifo_Empty** (DATAnr%) As Long*

***Aufrufparameter:***

| | |
|---|---|
| *DATAnr* | Nummer des als FIFO deklarierten ***ADbasic*** DATAs |

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | Anzahl der freien Positionen im FIFO |

***Beispiel:***

```
erg = Get_Fifo_Empty (5)      'Ermittelt die freien Positionen in DATA_5 (AS FIFO)
```

### 3.1.17 Inhalt des FIFO löschen

*Sub **Clear_Fifo** (DATAnr%)*

***Aufrufparameter:***

| | |
|---|---|
| *DATAnr* | Nummer des als FIFO deklarierten ***ADbasic*** DATAs |

***Beispiel:***

```
erg = Clear_Fifo (45)     'Löscht den Inhalt von DATA_45 (AS FIFO)
```

## 3.2 System- und einfache I/O-Funktionen

### 3.2.1 Einen analogen Eingang messen

*Function **ADC** (ADCnr%) As Long*

***Aufrufparameter**:*

| | |
|---|---|
| *ADCnr* | Nummer des ADCs, der einen Meßwert liefern soll |

***Rückgabewert**:*

| | |
|---|---|
| 255 | Fehler |
| sonst | Meßwert (12-Bit: 0-4095; 16-Bit: 0-65535) vom gewählten ADC |

***Beispiel:***

```
erg = ADC (1)     'Liefert einen Meßwert von ADC Kanal 1
```

### 3.2.2 Einen analogen Ausgang setzen

*Sub **Set_DAC** (DACnr%, neu&)*

***Aufrufparameter:***

| | |
|---|---|
| *neu* | *Auszugebender Wert* (12-Bit: 0-4095; 16-Bit: 0-65535) |
| *DACnr* | Nummer des DACs, der den Wert *neu* ausgeben soll |

***Beispiel:***

```
erg = Set_DAC (2,32768)      'Setzt analog Ausgang 2 auf 32768
```

### 3.2.3 Digitale Eingänge einlesen

*Function **Dig_In** () As Long*

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | aktueller Zustand aller digitalen Eingänge |

***Beispiel:***

```
erg = Dig_In ()   'Liefert den Wert der dig. Eingänge (Bit0-Bit15 ⇒ Eing.0-Eing.15)
```

### 3.2.4 Digitale Ausgänge setzen

*Sub **Set_Dig_Out** (neu&)*

***Aufrufparameter:***

| | |
|---|---|
| *neu* | *Auszugebender Wert* (Bit0-Bit15 ⇒ Ausg.0-Ausg.15) |

***Beispiel:***

```
erg = Set_Dig_Out (7)   'Ausg. 0-2 setzen und 3-15 rücksetzen
```

### 3.2.5 Aktuellen Stand der digitalen Ausgänge rücklesen

*Function **Dig_Out** () As Long*

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | aktueller Zustand aller digitalen Ausgänge |

***Beispiel:***

```
erg = Dig_Out ()  'Liefert den Wert der dig. Ausgänge (Bit0-Bit15 ⇒ Ausg.0-Ausg.15)
```

### 3.2.6 Auslastung des *ADwin*-Systems abfragen

*Function **Auslast** () As Integer*

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | aktuelle Prozessorauslastung in % |

***Beispiel:***

```
erg = Auslast ()        'Liefert die Prozessorauslastung (0% - 100%)
```

### 3.2.7 Testen, ob das *ADwin*-Betriebssystem korrekt geladen ist

*Function **Test** () As Integer*

***Rückgabewert:***

| | |
|---|---|
| 0 | OK |
| sonst | System nicht geladen |

***Beispiel:***

```
erg = Test ()       'Testet ob das Betriebssystem geladen ist
```

### 3.2.8 Freien Speicher des *ADwin*-2, 4, 5 o. 8 Systems abfragen

*Function **Freemem** () As Long*

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | Zusammenhängender freier Speicher |

***Beispiel:***

```
erg = Freemem ()        'Liefert den freien Speicher in kByte
```

### 3.2.9 Freien Speicher des *ADwin*-9 Systems abfragen

*Function **Freemem_T9** (typ%) As Long*

***Aufrufparameter:***

| Typ | Speicherart |
|---|---|
| 1 | Prozessorinterner Programmspeicher (PM) |
| 3 | Prozessorinterner Datenspeicher (DM) |
| 4 | Externer Datenspeicher (DRAM) |

***Rückgabewert:***

255 Fehler
sonst Zusammenhängender freier Speicher

***Beispiel:***

```
erg = Freemem_T9 (4)  'Liefert den freien Datenspeicher (DRAM) in Byte
```

### 3.2.10 Fehlermeldungen aktivieren/deaktivieren

*Sub **Err_Message** (onoff%)*

***Aufrufparameter:***

| Onoff | Bedeutung |
|---|---|
| 0 | Keine Fehlermeldungen von den aufgelisteten Funktionen |
| 1 | Die Funktionen: Boot, Test, ADBLoad u. Net_Connect geben Fehlermeldungen aus |

***Rückgabewert:***

Nicht vorhanden

***Beispiel:***

```
Err_Message (1)        'Fehlermeldungen werden angezeigt
```

## Hinweis zur Netzwerkverbindung:

Die Funktion *Net_Connect* baut über Netzwerk eine Verbindung zu einem Rechner (Wirtsrechner) auf, an dem ein ***ADwin***-System angeschlossen ist.
Auf dem Wirtsrechner muß vorher das Programm ***ADServer.exe*** gestartet werden. Nach erfolgreichem Verbindungsaufbau gehen alle Zugriffe auf das ***ADwin***-System automatisch über das Netzwerk zum Wirtsrechner.

### 3.2.11 Netzwerkverbindung zu einem *ADwin*-System aufbauen

*Function* ***Net_Connect****(pro$, wirt$, endp$, pass$) As Integer*

***Aufrufparameter*:**

| pro | Protokoll |
|---|---|
| ncacn_dnet_nsp | DECnet |
| ncacn_ip_tcp | TCP/IP |
| ncacn_nb_nb | NetBIOS über NetBEUI |
| ncacn_nb_tcp | NetBIOS über TCP/IP |
| ncacn_np | Named pipes |
| ncacn_spx | SPX |

*wirt* Name oder Adresse des Wirtsrechners
*endp* Endpunkt (Siehe ADserver.exe bzw. ***ADbasic*** Handbuch)
*pass*Passwort (Siehe ADserver.exe bzw. ***ADbasic*** Handbuch)

***Rückgabewert:***

1 Verbindung erfolgreich aufgebaut
sonst Fehler

***Beispiel:***

```
erg = ("ncacn_ip_tcp", "pc_oliver", "200", "")     'baut eine Verbindung zum Rechner
                                                   'pc_oliver mit Endpunkt 200 über das
                                                   'Netzwerkprotokoll TCP/IP ohne
                                                   'Passwort auf
```

### 3.2.12 Netzwerkverbindung beenden

*Function* ***Net_Disconnect*** *() As Integer*

***Rückgabewert*:**

*1* OK
sonst Fehler

***Beispiel:***

```
erg = Net_Disconnect ()      'Beendet eine Netzwerkverbindung, die mit der Funktion
                             '*Net_Connect* aufgebaut wurde
```

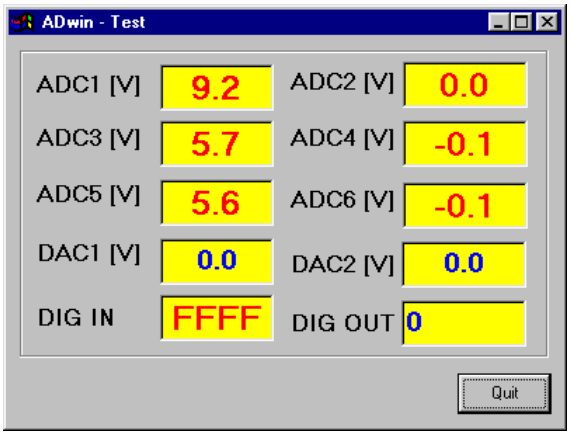
# 4 Programmbeispiele

Alle in diesem Abschnitt beschriebenen Visual-Basic Programme befinden sich inklusive Source code auf der ***ADwin***-CD-ROM. Da jedes der Programme bei Programmstart das ***ADwin***-Betriebssystem (***ADwin9.btl***) lädt, werden alle eventuell laufenden Prozesse beim Start beendet und gelöscht.

## 4.1 Funktionsprüfung der Analog- und Digital-I/Os

**Programmbeschreibung:**

Starten Sie das Programm ***ADtest.exe*** in dem Visual Basic-Verzeichnis. Das Programm mißt (mit der Timerfunktion **Timer1)** zehnmal pro Sekunde die analogen Eingänge 1 bis 6, rechnet die gemessenen ADC-Werte in Volt um und zeigt die Spannungen an. In der Timerfunktion werden außerdem die digitalen Eingänge eingelesen und im Hexadezimalformat angezeigt.
Mit den Eingabefenstern **DAC1** und **DAC2** 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 und gelöscht werden.

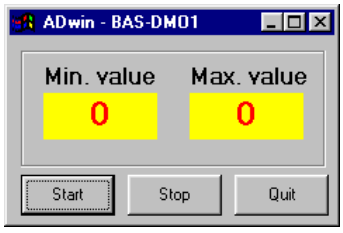



## 4.2 Online-Auswertung von Meßwerten

Das Visual-Basic Programm zur Online-Auswertung von Meßwerten trägt den Namen BAS_DMO1. Es lädt beim Programmstart das ***ADwin***-Betriebssystem für den Prozessor T9 (***ADwin9.btl***). Wenn Sie in Ihrem System einen anderen Prozessor verwenden, müssen Sie an der entsprechenden Stelle im Visual-Basic Programm den Namen der Datei ändern.
Außerdem wird die Binärdatei 'BAS_DMO1.T91' als Prozeß1 für den ADSP geladen. Auch hier müssen Sie die Angaben an Ihr System anpassen. Die Binärdatei wird vom ***ADbasic*** Compiler aus der Datei BAS_DMO1.bas mit der Funktion *Project ⇒ Make Bin File* erzeugt.

**Programmbeschreibung:**

Mit den Tasten **Start** und **Stop** wird der Prozeß1 gestartet bzw. angehalten. Zusätzlich aktivieren bzw. deaktivieren die Tasten die Timerfunktion **Timer1.** Das Programm BAS_DMO1 prüft mit der Timerfunktion **Timer1** zehnmal pro Sekunde, ob der ***ADbasic***-Prozeß1 das ACTIVATE_PC-Flag gesetzt hat. Der ***ADbasic***-Prozeß1 setzt das ACTIVATE_PC-Flag, nachdem er aus 1000 Messungen den Minimal- und Maximalwert ermittelt hat. Ist dies der Fall, so holt sich das Programm die ***ADbasic***-Parameter PAR_1 und PAR_2 und zeigt sie in den Fenstern für Minimalwert bzw. Maximalwert an.

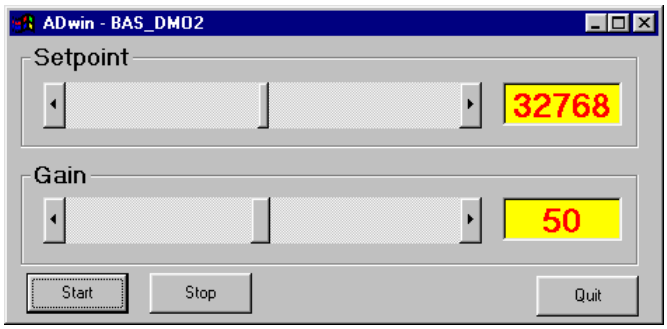



## 4.3 Online-Vorgaben von Regelgrößen

Das Visual-Basic Programm zur Online-Vorgabe von Regelgrößen trägt den Namen BAS_DMO2. Es lädt beim Programmstart das ***ADwin***-Betriebssystem für den Prozessor T9 (***ADwin9.btl***). Wenn Sie in Ihrem System einen anderen Prozessor verwenden, müssen Sie an der entsprechenden Stelle im Visual-Basic Programm den Namen der Datei ändern.
Außerdem wird die Binärdatei 'BAS_DMO2.T91' als Prozeß1 für den ADSP geladen. Auch hier müssen Sie die Angaben an Ihr System anpassen. Die Binärdatei wird vom ***ADbasic*** Compiler aus der Datei BAS_DMO2.bas mit der Funktion *Project ⇒ Make Bin File* erzeugt.

**Programmbeschreibung:**

Mit dem Programm BAS_DMO2 wird der Sollwert und die Verstärkung eines digitalen P-Reglers vorgegeben. Sollwert und Verstärkung können über zwei Schieberegler eingestellt werden. Bei jeder Bewegung eines Schiebereglers wird der neu eingestellte Wert an den ***ADbasic***-Prozeß1 übergeben. 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.




## 4.4 Darstellung einer Meßreihe

Das Visual Basic-Programm zur Darstellung einer Meßreihe trägt den Namen BAS_DMO3. Es lädt beim Programmstart das ***ADwin***-Betriebssystem für den Prozessor T9 (***ADwin9.btl***). Wenn Sie in Ihrem System einen anderen Prozessor verwenden, müssen Sie an der entsprechenden Stelle im Visual-Basic Programm den Namen der Datei ändern.
Außerdem wird die Binärdatei 'BAS_DMO3.T91' als Prozeß1 für den ADSP geladen. Auch hier müssen Sie die Angaben an Ihr System anpassen. Die Binärdatei wird vom ***ADbasic*** Compiler aus der Datei BAS_DMO3.bas mit der Funktion *Project ⇒ Make Bin File* erzeugt.

**Programmbeschreibung:**

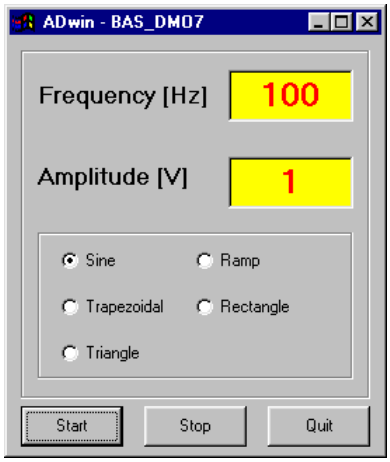
Mit der **Start** Taste wird der ***ADbasic***-Prozeß1 gestartet. Der Prozeß BAS_DMO3 nimmt 1000 Meßwerte auf und setzt anschließend das ACTIVATE_PC Flag. Das Programm BAS_DMO3 prüft mit der Timerfunktion **Timer1** zehnmal pro Sekunde, ob der ***ADbasic***-Prozeß1 das ACTIVATE_PC Flag gesetzt hat. Ist dies der Fall, so holt sich das Programm die ersten 1000 Werte aus dem ***ADbasic***-Array Data1 und stellt diese dar.

## 4.5 Schnelle Signalerzeugung

Das Visual Basic-Programm zur Signalerzeugung trägt den Namen BAS_DMO7. Es lädt beim Programmstart das ***ADwin***-Betriebssystem für den Prozessor T9 (***ADwin9.btl***). Wenn Sie in Ihrem System einen anderen Prozessor verwenden, müssen Sie an der entsprechenden Stelle im Visual-Basic Programm den Namen der Systemdatei ändern.
Außerdem wird die Binärdatei 'BAS_DMO7.T91' als Prozeß1 für den ADSP geladen. Auch hier müssen Sie die Angaben an Ihr System anpassen. Die Binärdatei wird vom ***ADbasic*** Compiler aus der Datei BAS_DMO7.bas mit der Funktion *Project ⇒ Make Bin File* erzeugt.

**Programmbeschreibung:**

Durch die Betätigung der **Start** Taste wird die gewählte Signalform mit der gewünschten Frequenz und Amplitude ausgegeben. Die **Stop** Taste beendet die Signalausgabe. Die aktuell eingestellte Frequenz und Amplitude wird an das ***ADwin***-System über die ***ADbasic*** Parameter PAR_6 und PAR_7 übergeben.

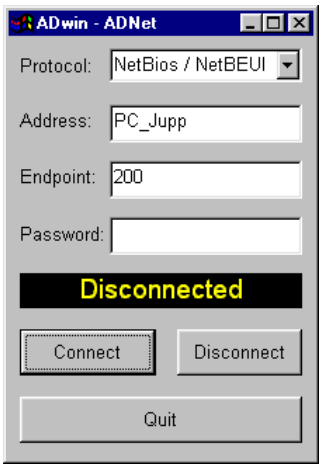


## 4.6 Netzwerkverbindung

Das Visual Basic-Programm zum Aufbau einer Netzwerkverbindung trägt den Namen ADNet. Das Programm kann die Netzwerkverbindung zu jedem Rechner aufbauen auf dem das Programm ***ADServer.exe*** gestartet wurde, welches im Lieferumfang von ***ADbasic*** enthalten ist. Nach erfolgreichem Verbindungsaufbau laufen alle Befehle, die an das ***ADwin***-System gerichtet sind, automatisch über das Netzwerk und adressieren das ***ADwin***-System, das an den Rechner angeschlossen ist, zu dem die Verbindung besteht. Damit ist jedes ***ADwin***-System über ein beliebiges Netzwerk komplett fernsteuerbar.

**Programmbeschreibung:**

Bei Betätigung der Taste **Connect** versucht das Programm über das Netzwerk eine Verbindung zu dem unter **Addreß** angegebenen Rechner (Hostrechner) herzustellen. Auf dem Hostrechner muß zuvor das Programm ***ADServer*** gestartet und aktiviert werden. Wenn die Verbindung erfolgreich hergestellt wurde, erhalten Sie in der schwarz unterlegten Statusinformationszeile die Meldung: Connection OK. Anderenfalls erscheint dort die Nachricht: Connection failed. In diesem Fall sollten Sie die Einstellungen prüfen und nach deren Korrektur einen erneuten Versuch starten.
Die Taste Disconnect beendet die Netzwerkverbindung die mit der Taste Connect aufgebaut wurde.



# 5 Befehlsindex