JÄGER
Computergesteuerte
Messtechnik GmbH

# *ADwin*

## C, C++, LabWindows/CVI-Treiber


**Ausgabe November 1999**

# 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 folgende 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 C Funktion Boot oder über ***ADbasic*** geladen. Zusätzlich bietet auch das Testprogramm ***ADtest*** die Möglichkeit das System zu laden.

### *a) Unter C bzw. LabWindows/CVI*

Mit der Funktion Boot ('*Systemdatei*', *Speicherausbau*) aus der Datei **adwin.c** bzw. **adwinCVI.h**. Nähere Informationen hierzu in Kapitel 3.

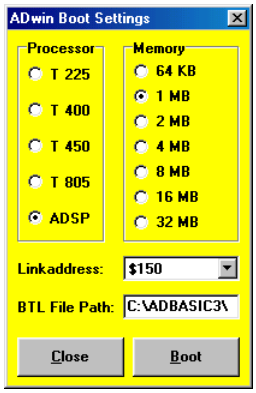
### *b) Über ADbasic*

- 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.

## 2.2 Einbinden der *ADwin* Funktionsbibliothek

Die C- bzw. LabWindows/CVI-Programme können, mit Hilfe der Funktionen aus der Datei **adwin.dll** (Windows 3.1) bzw. **adwin32.dll** (Windows '95, NT) Befehle an das ***ADwin***-System schicken und Daten abholen.

### 2.2.1 Mit C, C++

Diese Funktionen werden von der Datei **adwin.def** importiert. Die Prototypen der Funktionen sind in der Datei **adwin.h** zu finden. Um eine einfache und übersichtliche Programmierung der Meßaufgabe zu ermöglichen, wird auf die Funktionen aus der Datei **adwin.c** zurückgegriffen, welche auf den folgenden Seiten dieses Manuals dokumentiert sind.
Wenn Sie unter C, C++ mit dem ***ADwin***- System arbeiten wollen, müssen die folgenden Dateien zu Ihrem C Projekt hinzugefügt werden:

**adwin.h**
**adwin.c**
**adwin32.lib**

### 2.2.2 Mit LabWindows/CVI

Der Import der DLL-Funktionen, sowie die Deklaration der Prototypen, als auch die Bereitstellung der Funktionen zur einfachen Programmierung wird unter LabWindows/CVI von nur einer Datei übernommen. Diese Datei muß zu Ihrem LabWindows/CVI Projekt hinzugefügt werden und heißt:

**adwinCVI.h**

# 3 Funktionen aus der Datei adwin.c bzw. adwinCVI.h

## 3.3 Funktionen zur Steuerung von *ADbasic* Prozessen

### 3.3.1 Das ADwin-Betriebssystem laden

*long* ***Boot*** *(char *Systemdatei, long mem)**{** XE "**Boot** (char *Systemdatei, long mem)" **}***

***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)
```

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

*short* ***ADBPrLoad*** *(char *Dateiname)**{*** XE "***ADBPrLoad*** *(char *Dateiname)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

| | |
|---|---|
| 1 | OK |
| sonst | Fehler |

***Beispiel:***

```
erg = ADBPrLoad ("C:\\adbasic3\\samples\\bas_dmo1.T91")
```

### 3.3.3 ADbasic-Prozess starten

*short* ***ADBStart*** *(short nr)****{*** XE "***ADBStart*** *(short nr)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | OK |

***Beispiel:***

```
erg = ADBStart (1)          'startet ADbasic Prozess Nr.: 1
```

### 3.3.4 *ADbasic*-Prozess stoppen

*short* ***ADBStop*** *(short nr)****{*** XE "***ADBStop*** *(short nr)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | OK |

***Beispiel:***

```
erg = ADBStop (2)           'stoppt ADbasic Prozess Nr.: 2
```

### 3.3.5 Integer Parameter für *ADbasic* setzen

*short* ***SetPar*** *(short Parameternummer, long Wert)****{*** XE "***SetPar*** *(short Parameternummer, long 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

***Rückgabewert:***

255 Fehler
sonst OK

***Beispiel:***

```
erg = SetPar (1,2000)          'setzt den ADbasic Parameter PAR_1 auf 2000
```

### 3.3.6 Float Parameter für *ADbasic* setzen

*short* ***SetFPar*** *(short Parameternummer, float Wert)****{*** XE "***SetFPar*** *(short Parameternummer, float 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

***Rückgabewert:***

255 Fehler
sonst OK

***Beispiel:***

```
erg = SetFPar (6, 34.7)        'setzt den ADbasic Parameter FPAR_6 auf 34.7
```

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

*long* ***GetPar*** *(short Parameternummer)****{*** XE "***GetPar*** *(short Parameternummer)*" ***}***

***Aufrufparameter*:**

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

***Rückgabewert*:**

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

***Beispiel:***

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

### 3.3.8 ADbasic Activate_PC Flag abfragen

*short* ***GetActivate*** *(short nr)****{*** XE "***GetActivate*** *(short nr)*" ***}***

***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 = GetActivate(3)    'Stellt fest ob der ADbasic Prozess 3 den Befehl 'ACTIVATE_PC
                        ausgeführt hat
```

### 3.3.9 Float Parameter von *ADbasic* lesen

*float* ***GetFPar*** *(short Parameternummer)****{*** XE "***GetFPar*** *(short Parameternummer)*" ***}***

***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 = GetFPar (56)      'aktuellen Wert des ADbasic Parameters FPAR_56 lesen
```

### 3.3.10 Datensatz von *ADbasic* in ein Short, Integer oder Float Array übernehmen

*short* ***GetData*** *(short nr, long start, long anzahl, short ziel[ ])**{*** XE "***GetData*** *(short nr, long start, long anzahl, short ziel[ ])*" ***}***
*short* ***GetlData*** *(short nr, long start, long anzahl, long ziel[ ])**{*** XE "***GetlData*** *(short nr, long start, long anzahl, long ziel[ ])*" ***}***
*short* ***GetfData*** *(short nr, long start, long anzahl, float ziel[ ])**{*** XE "***GetfData*** *(short nr, long start, long anzahl, float ziel[ ])*" ***}***

***Aufrufparameter*:**

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

***Rückgabewert*:**

| | |
|---|---|
| 255 | Fehler |
| sonst | OK |

***Beispiel:***

```
erg = GetlData (2, 1, 100, ziel)        'die ersten 100 Elemente aus DATA_2 holen
```

### 3.3.11 Short, Integer oder Float Array als Datensatz an *ADbasic* schicken

*short* ***SetData*** *(short nr, long start, long anzahl, short quelle[ ])**{*** XE "***SetData*** *(short nr, long start, long anzahl, short quelle[ ])*" ***}***
*short* ***SetlData*** *(short nr, long start, long anzahl, long quelle[ ])**{*** XE "***SetlData*** *(short nr, long start, long anzahl, long quelle[ ])*" ***}***
*short* ***SetfData*** *(short nr, long start, long anzahl, float quelle[ ])**{*** XE "***SetfData*** *(short nr, long start, long anzahl, float quelle[ ])*" ***}***

***Aufrufparameter*:**

| | |
|---|---|
| *nr, start, anzahl* | Wie unter 3.1.10 |
| *quelle* | Array, dessen Werte in den **ADbasic** Datensatz übertragen werden |

***Beispiel:***

```
erg = SetlData (3, 10, 100, quelle)     'DATA_3 Elemente 10-110 mit dem Inhalt von
                                        'quelle belegen
```

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

*short* ***Data2File*** *(char *dateiname, short nr, long start, long anzahl, short mode)**{*** XE "***Data2File*** *(char *dateiname, short nr, long start, long anzahl, short mode)*" ***}***

***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)

### 3.3.13 FIFO von *ADbasic* in ein Short, Integer oder Float Array übernehmen

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

*short **GetFifo** (short Fnr, long anzahl, short ziel[ ])**{*** XE "***GetFifo** (short Fnr, long anzahl, short ziel[ ])*" ***}***
*short **GetlFifo** (short Fnr, long anzahl, long ziel[ ])**{*** XE "***GetlFifo** (short Fnr, long anzahl, long ziel[ ])*" ***}***
*short **GetfFifo** (short Fnr, long anzahl, float ziel[ ])**{*** XE "***GetfFifo** (short Fnr, long anzahl, float ziel[ ])*" ***}***

***Aufrufparameter*:**

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

***Rückgabewert*:**

| | |
|---|---|
| 255 | Fehler |
| sonst | OK |

***Beispiel:***

erg = ***GetfFifo** (2, 200, ziel)* ‘100 Elemente aus DATA_2 (AS FIFO) holen

### 3.3.14 Short, Integer oder 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.

*short **SetFifo** (short Fnr, long anzahl, short quelle[ ])**{*** XE "***SetFifo** (short Fnr, long anzahl, short quelle[ ])*" ***}***
*short **SetlFifo** (short Fnr, long anzahl, long quelle[ ])**{*** XE "***SetlFifo** (short Fnr, long anzahl, long quelle[ ])*" ***}***
*short **SetfFifo** (short Fnr, long anzahl, float quelle[ ])**{*** XE "***SetfFifo** (short Fnr, long anzahl, float quelle[ ])*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | OK |

***Beispiel:***

erg = ***SetfFifo** (12, 1000, quelle)* ‘1000 Elemente aus quelle an ***ADbasic*** ‘DATA_12 (AS FIFO) übertragen

### 3.3.15 Anzahl der Elemente im FIFO ermitteln

*long* ***GetFifoCount*** *(DATAnr)**{*** XE "***GetFifoCount*** *(DATAnr)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

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

***Beispiel:***

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

### 3.3.16 Anzahl der freien FIFO Positionen ermitteln

*long* ***GetFifoEmpty*** *(DATAnr)**{*** XE "***GetFifoEmpty*** *(DATAnr)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

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

***Beispiel:***

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

### 3.3.17 Inhalt des FIFO löschen

*short* ***ClearFifo*** *(DATAnr)**{*** XE "***ClearFifo*** *(DATAnr)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert*:**

| | |
|---|---|
| 255 | Fehler |
| sonst | OK |

***Beispiel:***

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

# 3.4 System- und einfache I/O-Funktionen

## 3.4.1 Einen analogen Eingang messen

*unsigned short* ***ADC*** *(short ADCnr)****{*** XE "***ADC*** *(short ADCnr)*" ***}***

***Aufrufparameter*:**

| | |
|---|---|
| *ADCnr* | Nummer des ADCs, der eine 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 auf
```

## 3.4.2 Einen analogen Ausgang setzen

*short* ***SetDAC*** *(short DACnr, unsigned short neu)****{*** XE "***SetDAC*** *(short DACnr, unsigned short neu)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | OK |

***Beispiel:***

```
erg = SetDAC (2,2048)  'Setzt analog Ausgang 2 auf 2048
```

## 3.4.3 Digitale Eingänge einlesen

*long* ***DigIn*** *()****{*** XE "***DigIn*** *()*" ***}***

***Rückgabewert:***

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

***Beispiel:***

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

### 3.4.4 Digitale Ausgänge setzen

*short* ***SetDigOut*** *(short neu)****{*** XE "***SetDigOut*** *(short neu)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

| | |
|---|---|
| 255 | Fehler |
| sonst | OK |

***Beispiel:***

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

### 3.4.5 Aktuellen Stand der digitalen Ausgänge rücklesen

*long* ***DigOut*** *()****{*** XE "***DigOut*** *()*" ***}***

***Rückgabewert:***

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

***Beispiel:***

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

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

*short* ***Auslast*** *()****{*** XE "***Auslast*** *()*" ***}***

***Rückgabewert:***

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

***Beispiel:***

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

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

*short* ***Test*** *()****{*** XE "***Test*** *()*" ***}***

***Rückgabewert:***

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

***Beispiel:***

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

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

*long* ***Freemem*** *()****{*** XE "***Freemem*** *()*" ***}***

***Rückgabewert:***

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

***Beispiel:***

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

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

*long* ***Freemem_T9*** *(short typ)****{*** XE "***Freemem_T9*** *(short typ)*" ***}***

***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
```

### 3.4.10 Fehlermeldungen aktivieren/deaktivieren

*void* ***ErrMessage*** *(short onoff)****{*** XE "***ErrMessage*** *(short onoff)*" ***}***

***Aufrufparameter:***

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

***Rückgabewert:***

Nicht vorhanden

***Beispiel:***

```
erg = ErrMessage (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.

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

*short* ***Net_Connect****(char *pro, char * wirt, char *endp, char *pass)****{*** XE "***Net_Connect****(char *pro, char * wirt, char *endp, char *pass)*" ***}***

***Aufrufparameter*:**

| pro | Protokoll |
|---|---|
| ncacn_dnet_nsp | DECnet |
| ncacn_ip_tcp | TCP / IP |
| ncacn_nb_nb | NetBIOS / NetBEUI |
| ncacn_nb_tcp | NetBIOS / TCP |
| Ncacn_nb_ipx | IPX / NetBEUI |
| 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
```

### 3.4.12 Netzwerkverbindung beenden

*short* ***Net_Disconnect ()****{* XE "***Net_Disconnect ()***" ***}***

***Rückgabewert*:**

*1* OK
sonst Fehler

***Beispiel:***

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

# 4 Programmbeispiele

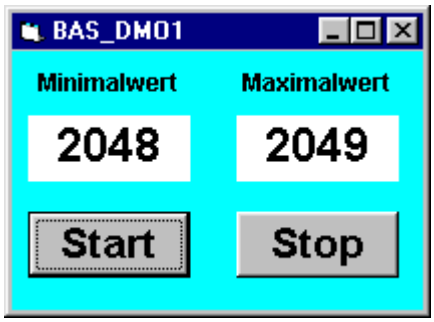
Alle in diesem Abschnitt beschriebenen C-Programme befinden sich inklusive Source code auf der ***ADwin***-CD-ROM. Die Programme laden jeweils das ***ADwin***-Betriebssystem (adwin9.btl) und einen binären ***ADbasic*** Prozess, der anschließend gestartet wird.

## 4.5 Online-Auswertung von Messwerten mit C/C++

Das C-Programm zur Online-Auswertung von Meßwerten trägt den Namen BAS_DMO1. Es lädt beim Programmstart das ***ADwin***-Betriebssystem für den Prozessor T9 (ADSP). Wenn Sie in Ihrem System einen anderen Prozessor verwenden, müssen Sie an der entsprechenden Stelle im C-Programm den Systemdatei Name 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.

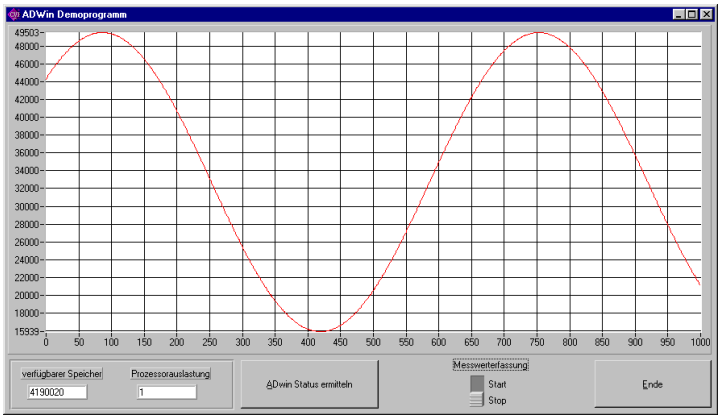



## 4.6 Darstellung einer Meßreihe mit LabWindows/CVI

Das LabWindows/CVI-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 LabWindows/CVI-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:**

Bei Umstellung des Schalter Messwert-erfassung in die Position **Start,** 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 einer Timerfunktion 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.
Der Taster ***ADwin*** **Status ermitteln** zeigt in den entsprechenden Fenstern die aktuelle Prozessorauslastung und den freien DRAM Speicher an.



# 5 Befehlsindex