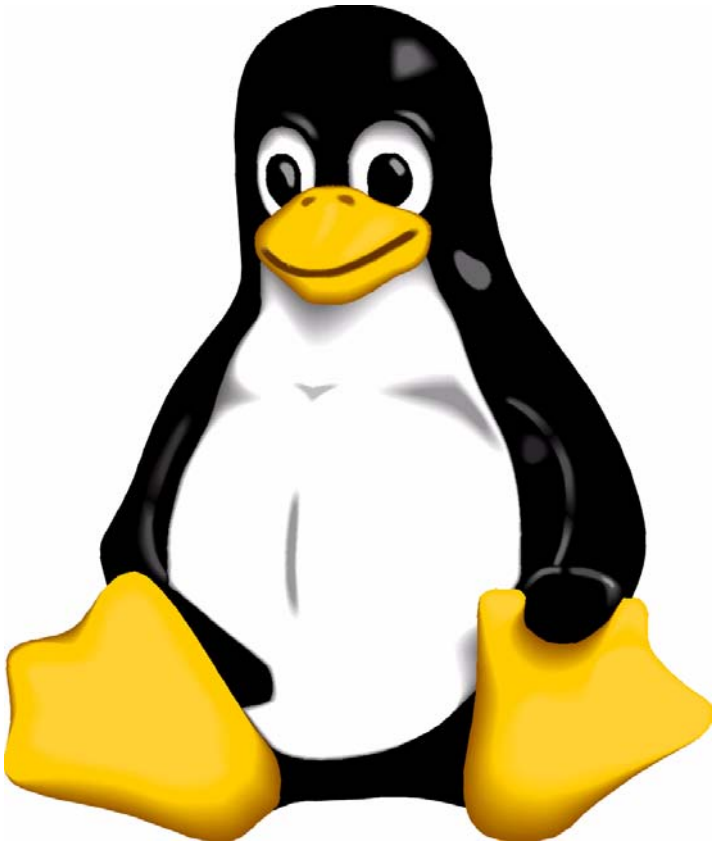
ADwin Linux

Handbuch



Linux-Treiber, Handbuch Version 2.1, Dezember 2004

Hier finden Sie immer einen Ansprechpartner für Ihre Fragen:

Hotline:	(0 62 51) 9 63 20
Fax:	(0 62 51) 5 68 19
E-Mail:	info@ADwin.de
Internet	www.ADwin.de

Jäger Computergesteuerte
Messtechnik GmbH
Rheinstraße 2-4
D-64653 Lorsch

Inhaltsverzeichnis

Typografische Konventionen IV

1 Zu diesem Handbuch 1

2 ***ADwin*** Linux 2

3 ***ADwin*** unter Linux installieren und einrichten 3
- 3.1 Software-Konfiguration und -Installation 3
- 3.2 Konfiguration des ***ADwin***-Systems 5
- 3.3 Bibliothek in C einbinden 9

4 ***ADbasic***-Quelltexte kompilieren 11
- 4.1 Compiler-Syntax. 11
- 4.2 Beispiele. 12

5 Methoden und Funktionen der ***ADwin*** Linux 15
- 5.1 Systemsteuerung und -information 15
- 5.2 Prozess-Steuerung 18
- 5.3 Übertragung von globalen Variablen 21
- 5.4 Übertragung von Datenfeldern (Arrays) 24
- 5.5 Fehlerbehandlung 32

Anhang A-1
- A-1 Fehlermeldungen. A-1
- A-2 Index der Methoden und Properties A-2

Typografische Konventionen

Das „Achtung“-Zeichen steht bei Informationen, die auf Folgeschäden durch Fehlbedienung an der Hard- oder Software, am Messaufbau oder an Personen hinweisen.

Einen „Hinweis“ finden Sie bei

- Informationen, die für einen fehlerfreien Betrieb unbedingt beachtet werden müssen.
- Tipps und Ratschlägen für einen effizienten Betrieb.

Das Zeichen „Information“ verweist auf weiterführende Informationen in dieser Dokumentation oder andere Quellen wie Handbücher, Datenblätter, Literatur etc.

<C:\ADwin\ …>

Dateinamen und -pfade sind in spitzen Klammern und dem Schrifttyp Courier New angeben.

Programmtext

Programmanweisungen und Benutzer-Eingaben sind durch den Schrifttyp Courier New gekennzeichnet.

Var_1

Elemente eines Quelltextes wie **BEFEHLE**, Variablen, Kommentar und sonstiger Text werden im Schrifttyp Courier New und farbig dargestellt (wie im Editor der Entwicklungsumgebung ***ADbasic***).

In einem Datenwort (hier: 16 Bit) werden die Bits wie folgt nummeriert:

Bit-Nr.	15	14	13	…	01	00
Wert des Bits	2^{15}	2^{14}	2^{13}	…	2^1=2	2^0=1
Bezeichnung	MSB	-	-	-	-	LSB

1 Zu diesem Handbuch

Dieses Handbuch enthält umfassende Informationen für den Einsatz der Schnittstellen-Bibliothek ***ADwin*** Linux und des ***ADbasic***-Compilers.

Es wird ergänzt durch

– das Handbuch „***ADwin*** Treiber-Installation“, das die Hardware-Schnittstellen-Installation zu allen ***ADwin***-Systemen beschreibt.

– das Handbuch „***ADbasic***“, das die Befehlsreferenz des Compilers enthält. Es beschreibt außerdem
  • den Aufbau von ***ADbasic***-Programmen,
  • die Optimierung von ***ADbasic***-Programmen,
  • das Laufzeitverhalten von Prozessen.

  Zum Programmieren steht Ihnen – bisher nur unter Windows – eine komfortable Bedienoberfläche als Echtzeit-Entwicklungstool zur Verfügung. Sie können stattdessen einen Editor Ihrer Wahl verwenden.

– nur bei ***ADwin-Pro***-Systemen: Das Software-Handbuch mit der Befehlsreferenz zur Programmierung der Pro-Module.

– das Hardware-Handbuch für Ihr ***ADwin***-System.

Es wird vorausgesetzt, dass Sie den Umgang mit der von Ihnen verwendeten Entwicklungsumgebung oder Programiersprache beherrschen.

Bitte beachten Sie folgende Hinweise

Damit Ihr ***ADwin***-System sicher arbeitet, halten Sie sich an die Informationen dieser und weiterführender Dokumentationen, auf die hier verwiesen wird.

Einschränkung der Anwendergruppe

Der Hersteller des in dieser Dokumentation beschriebenen Systems geht davon aus, dass an dem Gerät nur qualifiziertes Personal arbeitet.

> *Qualifiziertes Personal sind Personen, die aufgrund ihrer Ausbildung, Erfahrung und Unterweisung sowie ihrer Kenntnisse über einschlägige Normen, Bestimmungen, Unfallverhütungsvorschriften und Betriebsverhältnisse von dem für die Sicherheit der Anlage Verantwortlichen berechtigt worden sind, die jeweils erforderlichen Tätigkeiten auszuführen und die dabei mögliche Gefahren erkennen und vermeiden können. (Definition für Fachkräfte nach VDE 105 und ICE 60364).*

Verfügbarkeit der Unterlagen

Diese Produktdokumentation und Unterlagen, auf die verwiesen wird, müssen stets verfügbar sein und konsequent beachtet werden. Für Schäden, die durch Missachtung der Informationen in dieser bzw. der weiterführenden Dokumentation entstehen, übernimmt die Firma ***Jäger Computergesteuerte Messtechnik GmbH***, Lorsch, keine Haftung.

Rechtliche Grundlagen

Diese Dokumentation ist einschließlich aller Abbildungen urheberrechtlich geschützt. Reproduktion, Übersetzung sowie elektronische und fotografische Archivierung und Veränderung bedürfen der schriftlichen Genehmigung der Firma ***Jäger Computergesteuerte Messtechnik GmbH***, Lorsch.

Fremdprodukte werden ohne Vermerk auf mögliche Patentrechte genannt, deren Existenz nicht auszuschließen ist.

Änderungen vorbehalten.

Hotline-Adresse siehe vordere Umschlagseite, innen.

2 *ADwin* Linux

ADwin Linux ist eine Bibliothek. Sie stellt Schnittstellen-Funktionen zur Verfügung, damit Sie z.B. aus C, C++ oder LabView mit einem ***ADwin***-System kommunizieren können.

Die Bibliothek ***ADwin*** Linux definiert Methoden, Funktionen und Systemvariablen zur Prozesssteuerung und zur Datenübertragung vom und ins ***ADwin***-System. Die Definition umfasst jeweils den Namen einer Funktion sowie Anzahl, Datentyp und Reihenfolge der zu übergebenden Parameter.

Kommunikation mit dem *ADwin*-System

Vom PC aus können Sie Prozesse im ***ADwin***-System steuern, sowie Daten von dort anfordern oder dorthin senden. Die Prozesse selbst programmieren Sie in ***ADbasic*** (siehe gleichnamiges Handbuch) und kompilieren Sie mit dem mitgelieferten Compiler (siehe Kapitel 4).

Echtzeitkern

Die Bibliotheks-Funktionen kommunizieren mit dem Echtzeitkern des ***ADwin***-Systems, dem Betriebssystem. Deshalb müssen Sie nach jedem Einschalten des Systems zunächst das Betriebssystem (in Form einer Datei, z.B. `<adwin9.btl>`) dorthin laden. Nach erfolgreicher Übertragung kann das System Prozesse empfangen und ausführen, Befehle vom PC entgegen nehmen und Daten mit ihm austauschen. Die in ***ADbasic*** programmierten Prozesse enthalten den Programmcode zur Messung, Steuerung oder Regelung Ihrer Applikation.

Die Aufgaben des Betriebssystems sind:

10 Prozesse

– Verwaltung von bis zu 10 Echtzeit-Prozessen mit niedriger oder hoher Priorität (frei wählbar). Niedrig priorisierte Prozesse können von hoch priorisierten Prozessen unterbrochen werden, letztere können nicht von anderen Prozessen unterbrochen werden.

Datenspeicher

– Bereitstellung von globalen Variablen:
  - 80 Integer-Variablen (`PAR_1` ... `PAR_80`), bereits vordefiniert
  - 80 Float-Variablen (`FPAR_1` ... `FPAR_80`), bereits vordefiniert
  - 200 Datenfelder (`DATA_1` ... `DATA_200`), frei definierbare Länge

  Sie können die Werte dieser Variablen bzw. Datenfelder vom PC aus jederzeit lesen und ändern. Außerdem ist die Definition lokaler Variablen und Felder möglich.

Kommunikation

– Kommunikation zwischen ***ADwin***-System und PC .

  Der Kommunikationsprozess läuft mit mittlerer Priorität auf dem ***ADwin***-System und kann niedrig priorisierte Prozesse für kurze Zeit unterbrechen. Er interpretiert oder bearbeitet alle Befehle, die Sie vom PC an das ***ADwin***-System richten: Steuerbefehle und Befehle für den Datenaustausch. Die folgende Tabelle zeigt Beispiele aus jeder Gruppe.

Steuerbefehle, z.B.	
`LOAD_PROCESS`	überträgt einen Prozess auf das System
`START_PROCESS`	startet einen Prozess
Befehle für den Datenaustausch, z.B.	
`GET_PAR`	liefert den aktuellen Wert eines Parameters
`SET_PAR`	ändert den Wert eines Parameters
`GETDATA_LONG`	liefert die Werte aus einem `DATA`-Feld

Der Kommunikationsprozess sendet niemals unaufgefordert Daten an den PC. Das stellt sicher, dass nur dann Daten zum PC übertragen werden, wenn Sie diese ausdrücklich angefordert haben.

3 *ADwin* unter Linux installieren und einrichten

Die Installation der Linux-Bibliothek wird mit folgenden Schritten durchgeführt:

– Software-Konfiguration und -Installation

– Konfiguration des ADwin-Systems

– Bibliothek in C einbinden

Teilweise benötigen Sie für die Installation „root“-Rechte. Wenn bei einer Eingabe „root“-Rechte erwartet werden, ist im folgenden das Zeichen `#` am Anfang der Eingabezeile angegeben (sonst das Zeichen `$`).

3.1 Software-Konfiguration und -Installation

Auf der ***ADwin***-CD finden Sie das Verzeichnis `./LINUX`, in der sich folgende tar-Archive befinden:

CD-Inhalt

– `adwin-lib-x.y.tar.gz`: Das Archiv (Version `x.y`) enthält:
  - Kernel-Modul für ***ADwin*** Linux
  - ***ADwin*** Linux-Bibliothek (shared library)
  - Konfigurations-Programm `adconfig`

– `adwin-compiler-x.y.tar.gz`: Der ***ADbasic***-Compiler.

– `adwin-labview-x.y.tar.gz`: Der Treiber für die Bedienoberfläche LabView (vollständige Installation siehe Handbuch „LabView-Treiber“).

– `adwin-doc-x.y.tar.gz`: Alle Dokumentationen für Software und Hardware im Format pdf.

Archive installieren

Installieren Sie die Archive in der Reihenfolge wie oben angegeben. Beginnen Sie mit `adwin-lib-x.y.tar.gz` (bitte Hinweise weiter unten beachten).

Sie installieren ein Archiv mit den folgenden Schritten:

1. Entpacken Sie das Archiv `<filename>.tar.gz`:
   ```
   $ cd ~
   $ tar -xvzf <cdrom-path>/<filename>.tar.gz
   $ cd <filename>
   ```
2. Führen Sie den üblichen Dreisatz aus:
   ```
   $ ./configure
   $ make
   # make install
   ```
   Die Dateien werden im Verzeichnis `/opt/adwin/` installiert (diese Einstellung wird im folgenden angenommen).
   Wenn Sie ein anderes Installationsverzeichnis verwenden möchten, führen Sie das `configure` Skript mit folgender Option aus.
   ```
   $ ./configure --prefix=/your-dir-here
   ```
3. Installieren Sie die übrigen Archive in der gleichen Weise.

Beachten Sie beim Installieren des Archivs `adwin-lib-x.y.tar.gz` folgende Hinweise:

– Das Kernel-Modul wird im Verzeichnis `/lib/modules/`, die Bibliothek und das Konfigurationsprogramm im Verzeichnis `/opt/adwin/` generiert und installiert. Die Voreinstellungen passen für Kernel-Version 2.4.

  Für Kernel 2.6 verwenden Sie zusätzlich die folgenden Optionen:

Kernel 2.6

```
$ ./configure --disable-link --disable-kernel-module
```

– Wenn Sie bisher noch keinen Linux-Kernel kompiliert haben, ist es möglich, dass Ihnen nach dem Aufruf von `make` das Fehlen von Dateien angezeigt wird. Versuchen Sie es in diesem Fall mit folgenden Zeilen:

```
$ cd /usr/src/linux
$ make oldconfig dep
```

Führen Sie nun den Dreisatz nochmals aus. Falls Sie wider Erwarten keinen Erfolg haben sollten, melden Sie sich bei uns.

Kernel-Version

– Das Kernel-Modul sollte mit allen Versionen des Kernels 2.4 stabil laufen. Die Kernel-Versionen sind beispielsweise unter `ftp.kernel.org` verfügbar.

Wenn Sie dennoch Probleme feststellen, können wir Ihnen am besten helfen, wenn Sie uns folgende Informationen schicken:

- Kurzbeschreibung des aufgetretenen Problems
- Verwendete Kernel-Version und Patches
- Die Datei `.config` des Kernels

Umgebungsvariable

In früheren Versionen der Linux-Bibliothek wurde in der Umgebungsvariablen `ADWINDIR` das Installationsverzeichnis abgelegt. Das ist auch weiter möglich, aber nicht erforderlich.
Um die Variable zu setzen, können Sie

- das Verzeichnis (aus dem Dreisatz) direkt angeben:

```
$ export ADWINDIR=/opt/adwin/
```

- oder eine (beim Dreisatz automatisch erzeugte) Datei benutzen:

```
$ export ADWINDIR=´cat /etc/adwin/ADWINDIR/´
```

Wenn Sie `bash` verwenden, ist vielleicht folgende Zeile zusätzlich sinnvoll:

```
$ export PATH=$PATH:$ADWINDIR/bin:$ADWINDIR/sbin
$ export MANPATH=$MANPATH:$ADWINDIR/man
```

3.2 Konfiguration des *ADwin*-Systems

Jedes ***ADwin***-System wird unter Linux über eine sogenannte „Device No.“ angesprochen. Zum Einrichten einer Device No. verwenden Sie das Programm `adconfig` im Verzeichnis `/opt/adwin/sbin/`:

– Melden Sie die Device No. des ***ADwin***-Systems unter Linux an (nach der Installation ist noch keine Device No. angemeldet). Der PC verwendet die Anmeldungdaten, um das ***ADwin***-System anzusprechen.

– Wenn das ***ADwin***-System eine Ethernet-Schnittstelle besitzt, müssen Sie das System zusätzlich konfigurieren. Sie finden im Handbuch „ADwin Treiber-Installation“ eine Beschreibung der Grundlagen zum Ethernet-Betrieb.

  Die Konfigurationsdaten werden nicht auf dem PC gespeichert, sondern an das ***ADwin***-System übertragen. Die Daten müssen mit den Anmeldungsdaten übereinstimmen.

Zum Aufruf des Programms `adconfig` benötigen Sie „root“-Rechte (`su`).

3.2.1 Beispiel: Standard-Konfiguration

Mit den folgenden Zeilen richten Sie die Device No. 0x150 für ein ***ADwin***-System mit Ethernet-Schnittstelle ein:

1. Melden Sie die Device No. 0x150 (dezimal:336) mit einer festen IP-Adresse (hier: 10.100.100.83) unter Linux an:
   ```
   # adconfig add 0x150 TYPE net IP 10.100.100.83
   ```
2. Lassen Sie sich den angemeldeten Eintrag anzeigen
   ```
   # adconfig
   # DeviceNo=0x150 Type=net IP=10.100.100.83 Port=6543
    Timeout=1000 PackageCount=5
   ```
3. Konfigurieren Sie das ***ADwin***-System mit Mac-Adresse, IP-Adresse und Subnet Mask:
   ```
   # adconfig config 00:50:C2:0A:22:DD IP 10.100.100.83
    MASK 255.255.255.00
   ```

Die Device No. 0x150 ist nun eingerichtet und Sie können mit dem ***ADwin***-System arbeiten.

3.2.2 Konfiguration ausführlich

Mit dem Programm `adconfig` können Sie folgende Funktionen durchführen:

– Liste aller unter Linux angemeldeten Device No. anzeigen:
  ```
  $ adconfig
  ```

– Neue Device No. unter Linux anmelden:
  ```
  # adconfig add <deviceno> [OPTIONS] TYPE link|net
  ```

– ***ADwin***-System mit Ethernet-Schnittstelle konfigurieren:
  ```
  # adconfig config <deviceno> [OPTIONS]
  ```

  Verwenden Sie die gleichen Daten, die Sie mit `add` angemeldet haben!

– Eine angemeldete Device No. löschen:
  ```
  # adconfig del <deviceno>
  ```

– Hilfe zum Programm `adconfig` anzeigen:
  ```
  $ adconfig --help
  ```

– Versionsnummer des Programms anzeigen:
  ```
  $ adconfig --version
  ```

Der Parameter `<deviceno>` kann in dezimaler oder hexadezimaler Schreibweise angegeben werden, z. B. 336 oder 0x150.

Die Details zu den Funktionen des Programms sind nachfolgend beschrieben oder können mit `man 8 adconfig` angezeigt werden.

Device No. anzeigen

Liste aller Device No. anzeigen

```
$ adconfig
```

Es wird eine Liste aller angemeldeten Device No. mit deren Parametern angezeigt. Die Device No. wird hexadezimal angezeigt, z.B. `0x150` (dezimal: 336).

Device No. anmelden: Link

Neue Link-Device No. unter Linux anmelden

Diese Funktion ist nur mit Kernel 2.4 verfügbar.

```
# adconfig add <deviceno> [UID <uid|username>]
  [GID <gid|groupname>] [MODE <mode>] [DESC <"">]
  <TYPE link BASE <#>>
```

`<deviceno>`	Die anzumeldende Device No.
`UID <uid|username>`	Name des Nutzers, der das ***ADwin***-System nutzen darf (Voreinstellung: root)
`GID <gid|groupname>`	Name der Gruppe, die das ***ADwin***-System nutzen darf (Voreinstellung: root)
`MODE <mode>`	Zugriffsrechte der Nutzer auf das ***ADwin***-System (Voreinstellung: 0666)
`DESC <"">`	Beschreibung (bis 32 Zeichen, Voreinstellung: "")
`TYPE link BASE <#>`	***ADwin***-System mit Link-Datenverbindung. `<#>`: Link-Basisadresse (z.B. 0x150)

Sie müssen die Device No. auf der ***ADwin***-Karte als Link-Adresse einstellen. Die Beschreibung hierzu finden Sie im Installations-Handbuch.

Geben Sie nur die Parameter an, die keine Voreinstellung haben oder deren Voreinstellung sie ändern möchten.

Device No. anmelden: Ethernet

Neue Ethernet-Device No. unter Linux anmelden

```
# adconfig add <deviceno> [UID <uid|username>]
  [GID <gid|groupname>] [MODE <mode>] [DESC <"">]
  <TYPE IP <addr> [PW <"">] [PORT <#>] [TIMEOUT <#>]>
```

`<deviceno>`	Die anzumeldende Device No.
`UID <uid|username>`	Name des Nutzers, der das ***ADwin***-System nutzen darf (Voreinstellung: root)
`GID <gid|groupname>`	Name der Gruppe, die das ***ADwin***-System nutzen darf (Voreinstellung: root)
`MODE <mode>`	Zugriffsrechte der Nutzer auf das ***ADwin***-System (Voreinstellung: 0666)
`DESC <"">`	Beschreibungstext (bis 32 Zeichen, Voreinstellung: "")
`TYPE net IP <addr>`	***ADwin***-System mit Ethernet-Schnittstelle. `<addr>`: IP-Adresse im Ethernet-Netzwerk
`PW <"">`	Passwort (Voreinstellung: "")
`PORT <#>`	Netzwerk Port Nummer (Voreinstellung: 6543)
`TIMEOUT <#>`	Timeout des Netzwerk-Protokolls (Voreinstellung: 1000 ms)

Geben Sie nur die Parameter an, die keine Voreinstellung haben oder deren Voreinstellung sie ändern möchten.

Die mit `UID`, `GID` und `MODE` eingestellten Zugriffsrechte stellen bei Ethernet kaum mehr als einen Schutz vor versehentlicher Fehlbedienung dar, weil die Einstellungen umgangen werden können.

Die Einrichtung eines ***ADwin***-Systems in einem Ethernet-Netzwerk ist im Installations-Handbuch beschrieben. Dort finden Sie auch eine genauere Beschreibung der Netzwerk-spezifischen Parameter.

Achten Sie darauf, dass eine IP-Adresse in einem Ethernet-Netzwerk nicht doppelt vergeben werden darf. Anderenfalls können erhebliche Kommunikations-Probleme auftreten.

Beispiel

```
# adconfig add 0x3A TYPE net IP 10.100.100.83
```

Lassen Sie sich anschließend den definierten Eintrag anzeigen:

```
# adconfig
# DeviceNo=0x3A Type=net IP=10.100.100.83 Port=6543
 Timeout=1000 PackageCount=5
```

Device No. anmelden – *ADwin*-System am Host-Rechner

Device No. anmelden: ADwin-System am Host-Rechner

In einem IP-Netzwerk (z.B. Ethernet) kann ein Client-PC ein ***ADwin***-System an einem anderen Rechner (Host-PC) auch dann ansprechen, wenn das System eine Link- oder USB-Schnittstelle hat, oder wenn es in einem anderen Netzwerk betrieben wird.

Voraussetzungen hierfür:

- Der Host-PC hat Zugriff auf das ***ADwin***-System.
- Der Host-PC kann vom Client-PC über das Netzwerk angesprochen werden.
- Das Programm `ADwinTcpipServer` läuft auf dem Host-PC (derzeit nur unter Windows).

Wenn die Voraussetzungen erfüllt sind, müssen die Anmeldungs-Parameter wie folgt verwendet werden:

```
# adconfig add <deviceno> …
 TYPE net IP <addr> [PW <"">][PORT <#>]
 [TIMEOUT <#>] HOSTLINK <#>
```

`net IP <addr>`	IP-Adresse des Host-Rechners

`PW <"">`
`PORT <#>`
`TIMEOUT <#>`
	Die Parameter müssen hier identisch sein mit den Einstellungen, die am Host-PC im Programm `ADwinTcpipServer` vorgenommen wurden (siehe Online-Hilfe des Programms).

`HOSTLINK <#>`	Zusätzlich: Device No. des ***ADwin***-Systems, die am Host-PC mit `adconfig` eingestellt ist.

Ethernet-Gerät ohne DHCP konfigurieren

ADwin-System mit Ethernet-Schnittstelle konfigurieren (ohne DHCP)

```
# adconfig config <hwaddr> IP <addr> MASK <addr>
  [GATEWAY <addr>] [PORT <#>] [PW <"">]
```

`<hwaddr>`	Die MAC-Adresse des ***ADwin***-Systems im Format `00:50:C2:0A:nn:nn`.
`IP <addr>`	IP-Adresse des ***ADwin***-Systems im Ethernet-Netzwerk im Format `nnn.nnn.nnn.nnn`.
`MASK <addr>`	Subnet mask im Format `nnn.nnn.nnn.nnn`.
`GATEWAY <addr>`	IP-Adresse für Default gateway (Voreinstellung: 0.0.0.0, d.h. kein gateway)
`PORT <#>`	Netzwerk Port-Nummer (Voreinstellung: 6543)
`PW <"">`	Neues Passwort (Voreinstellung: "", d.h. das existierende Passwort wird entfernt!)

Geben Sie nur die Parameter an, die keine Voreinstellung haben oder deren Voreinstellung sie ändern möchten. Verwenden Sie die gleichen Werte, die Sie mit `add` angemeldet haben!

Beachten Sie bitte, dass beim Konfigurieren immer ein Passwort übermittelt wird. Wenn Sie ein vorher eingestelltes Passwort beibehalten möchten, müssen Sie das Passwort also angeben, sonst wird es gelöscht.

Beispiel

```
# adconfig config 00:50:C2:0A:22:DD IP 10.100.100.83
  MASK 255.255.255.00
```

Ethernet-Gerät mit DHCP konfigurieren

ADwin-System mit Ethernet-Schnittstelle für DHCP konfigurieren

Es ist möglich, aber keineswegs empfohlen, dass ein ***ADwin***-System seine IP-Adresse von einem DHCP-Server dynamisch zugewiesen bekommt. Hierzu aktivieren Sie die Option `DHCP`.

```
# adconfig config <hwaddr> [GATEWAY <addr>] [PORT <#>]
  DHCP yes [PW <"">]
```

`<hwaddr>`	Die MAC-Adresse des ***ADwin***-Systems im Format `00:50:C2:0A:nn:nn`.
`GATEWAY <addr>`	IP-Adresse für Default gateway (Voreinstellung: 0.0.0.0, d.h. kein gateway)
`PORT <#>`	Netzwerk Port-Nummer (Voreinstellung: 6543)
`PW <"">`	Neues Passwort (Voreinstellung: "", d.h. das existierende Passwort wird entfernt!)
`DHCP <no | yes>`	Festlegung, ob ein DHCP-Server verwendet wird (Voreinstellung: `no`); siehe unten.

Wir empfehlen, die Option `DHCP` nicht zu aktivieren!
Bei aktiver Option können oft Kommunikationsprobleme auftreten.

Geben Sie nur die Parameter an, die keine Voreinstellung haben oder deren Voreinstellung sie ändern möchten. Verwenden Sie die gleichen Werte, die Sie mit `add` angemeldet haben!

Device No. löschen

Angemeldete Device No. löschen

```
# adconfig del <deviceno>
```

Die Device No. wird unter Linux abgemeldet und kann nicht mehr angesprochen werden.

3.3 Bibliothek in C einbinden

Um die Funktionen der Bibliothek ***ADwin*** Linux in C oder C++ zu verwenden, gehen Sie vor wie folgt:

– Fügen Sie folgende Zeilen in Ihr Programm ein:
  - Funktionen zum Zugriff auf das ***ADwin***-System:
    `#include <libadwin.h>`
  - Fehlernummern, die beim Zugriff auf das ***ADwin***-System auftreten können:
    `#include <libadwin/errno.h>`

– Fügen Sie folgende Flags in Ihr Makefile für den C-Compiler ein, um Ihr Programm mit der Bibliothek ***ADwin*** Linux zu kompilieren:

```
$ CFLAGS += -I$(ADWINDIR)/include -L$(ADWINDIR)/lib
–ladwin
$ CFLAGS += -Wl,--rpath -Wl,$(ADWINDIR)/lib
```

  Beachten Sie, dass hier die Umgebungsvariable `ADWINDIR` verwendet wird (Setzen der Variablen siehe Seite 4).

  Wenn Sie die Threads benutzen, müssen Sie zusätzlich hinzufügen:

```
$ CFLAGS += -D_REENTRANT
```

Portierung von Windows

Die Funktionen der Bibliothek ***ADwin*** Linux sind kompatibel zu den Funktionen der Schnittstelle unter Windows (`adwin32.dll`). Sie können daher Programm-Quelltexte, die Sie in einer Programmiersprache (hier: C / C++) unter Windows erstellt haben, in der Regel ohne Änderung in Linux verwenden (natürlich nur, soweit es um den Zugriff auf ***ADwin***-Systeme geht).

Änderungen im Programm sind nur dort erforderlich, wo Pfadangaben oder Dateinamen verwendet werden. Hier machen sich die typischen Unterschiede zwischen Windows und Linux bemerkbar: Syntax und Groß- und Kleinschreibung der Pfadnamen.

Für ***ADbasic***-Quelltexte ist eine Portierung von (und nach) Windows in gleicher Weise möglich.

4 *ADbasic*-Quelltexte kompilieren

In einem ***ADbasic***-Quelltext programmieren Sie die Funktion eines Prozesses, der auf dem ***ADwin***-System abläuft. Sie erzeugen einen ***ADbasic***-Quelltext mit einem Editor Ihrer Wahl. Unter Windows steht Ihnen eine komfortable Bedienoberfläche als Echtzeit-Entwicklungstool zur Verfügung.

Der ***ADbasic***-Compiler wird in Linux aus der Kommandozeile (Shell) aufgerufen. Er übersetzt eine Quelltext-Datei und erzeugt – je nach Schalter und Parameter – entweder eine Binär-Datei oder eine Library-Datei. Die Funktion von Binär- und Library-Dateien ist im ***ADbasic***-Handbuch erläutert.

Vor dem ersten Compiler-Aufruf müssen Sie über den Schalter `-k` Ihren License key eingeben. Erst danach können Sie Binär- oder Library-Dateien erzeugen.
Sie finden den License key auf dem Deckblatt Ihres ***ADbasic***-Handbuchs.

Ein oder mehrere Kommandozeilen-Aufrufe können in einem Shell-Skript oder Makefile zusammengefasst werden, um mehrere Quelltexte eines Projekts mit nur einem Aufruf zu kompilieren.

4.1 Compiler-Syntax

Es gibt 5 Hauptschalter beim Aufruf des ***ADbasic***-Compilers:

1.	`adbasic -v`	Versionsnummer des Compilers anzeigen
2.	`adbasic -h`	Hilfe aufrufen
3.	`adbasic -k <license>`	License key eingeben (siehe Deckblatt des ***ADbasic***-Handbuchs). Sie müssen die Schreibberechtigung für `/opt/adwin/var/license/` besitzen.
4.	`adbasic [OPTIONS] <filename>`	Einen ***ADbasic***-Quelltext kompilieren und eine Binär-Datei erzeugen.
5.	`adbasic -l [OPTIONS] <filename>`	Einen ***ADbasic***-Quelltext kompilieren und eine Library-Datei erzeugen.

Die Schalter `[OPTIONS]` für das Kompilieren eines Quelltexts sind nachfolgend beschrieben.

Syntax

```
[path/]adbasic [-p <ptype>] [-s <system>]
       [-n <#>] [-e <event>] [-g <cycle>]
       [-r <prio>] [-l] [-L <path/>] [-I <path>]
       [-a [path/]<fout>] [path/]infile.bas
```

Schalter

`[path/]`	Verzeichnis, in dem sich das Programm `adbasic` befindet, bei Standardinstallation: `/opt/adwin/bin/`
`-p<ptype>`	Prozessortyp, für den die Datei kompiliert werden soll:
`-p2`	Prozessor T2
`-p4`	Prozessor T4
`-p5`	Prozessor T5
`-p8`	Prozessor T8
`-p9`	Prozessor T9; Default
`-p10`	Prozessor T10

`-s<system>`	***ADwin***-System, für das die Datei kompiliert werden soll: `-sc` Karten (ISA-Bus) `-sl` Light-16 `-sg` Gold; Default `-sp` Pro
`-n<#>`	Setzt die Prozessnummer (1…10) der zu kompilierenden Datei; Default = 1.
`-e<event>`	Ereignisquelle für den Prozess wählen: `-et` Timer; Default `-ee` Extern
`-g<cycle>`	Setzt das 'Initial Processdelay'; Default = 1000.
`-r<prio>`	Setzt die Priorität des Prozesses auf: `-rh` hoch (high); Default `-rl` niedrig (low) `-rln` niedrig, Level n (1…255); Default: n=1. Level 255 hat die höchste Priorität.
`-l`	Quelltext kompilieren und eine Library-Datei (Bibliothek) erzeugen mit der Endung „`.liX`“. Bei nicht gesetztem Schalter wird eine Binärdatei mit der Endung „`.tXY`“ erzeugt. `X` = 2, 4, 5, 8, 9, a (entspricht dem Prozessortyp `-p`) . `Y` = 1…9, 0 (entspricht der Prozessnummer `-n`: 1…10) .
`-L<path>`	Setzt den Suchpfad für Library-Dateien; Default `/opt/adwin/share/lib/`
`-I<path>`	Setzt den Suchpfad für Include-Dateien; Default `/opt/adwin/share/inc/`
`-a<fout>`	Optional: Name der zu erzeugenden Binär- oder Library-Datei, ohne Dateiendung.
`infile.bas`	Dateiname des zu kompilierenden Quelltextes. Der Pfad ist optional.

Bemerkungen

Optionale Angaben sind in eckige Klammern gesetzt. Die Reihenfolge der Schalter ist beliebig. In Kommandozeilen wird zwischen Groß- und Kleinschreibung unterschieden.

Wenn der Schalter `-a` nicht verwendet wird, wird die erzeugte Binär- oder Library-Datei in dem Verzeichnis gespeichert, in dem sich der Quelltext befindet.

Folgende Schalter schließen sich gegenseitig aus:

Schalter	dadurch ausgeschlossen:
`-sg,-sl`	`-p2, -p4, -p5, -p8`
`-sp`	`-p2`
`-l`	`-n, -e, -g, -r, -d, -o`

Wenn während des Kompilierens ein Fehler auftritt, bricht der Compiler seine Arbeit ab und erzeugt keine Binär- oder Library-Datei.

4.2 Beispiele

```
$ /opt/adwin/bin/adbasic -l /home/user/test.bas
```

Der Aufruf kompiliert die Quelltextdatei `test.bas` und erzeugt im Verzeichnis `/home/user/` die Library-Datei `test.li9`.

Da außer `-l` keine weiteren Schalter angegeben sind, werden die Standardeinstellungen verwendet:

- Prozessor: T9
- System: Gold
- Prozessnummer: 1
- Event: Timer
- Initial Processdelay: 1000
- Priorität: Hoch (high)
- Include-Pfad: `/opt/adwin/share/inc/`
- Library-Pfad: `/opt/adwin/share/lib/`

Wenn Sie das Verzeichnis, in dem sich der Compiler `adbasic` befindet, in Ihren Suchpfad aufnehmen (siehe Installation, Seite 3) , können Sie obige Zeile kürzer schreiben als:

```
$ adbasic -l /home/user/test.bas
```

In den folgenden Beispielen nehmen wir an, dass sich der Pfad von `adbasic` im Suchpfad befindet.

```
$ adbasic -l /string.bas -sp
```

Der Kommandozeilenaufruf kompiliert die Quelltextdatei `string.bas` in eine Library-Datei für ein Pro-System mit Prozessor T9.
Der gleiche Aufruf, nur für den Prozessor T10, sieht so aus:

```
adbasic -l /test.bas -p10 -sl
```

```
$ adbasic /home/user/test.bas
/opt/adwin/share/example/bas_dmo6f -p9 -sg
```

Kompiliert die Demo-Datei `bas_dmo6f.bas` in eine Binär-Datei für ein ***ADwin-Gold***-System mit T9-Prozessor.

```
$ adbasic /opt/adwin/share/example/bas_dmo6 -p8 -sc
```

Kompiliert die Demo-Datei `bas_dmo6.bas` in eine Binär-Datei für eine *ADwin*-Karte mit dem Prozessor T8.

```
$ adbasic /home/user/my_file.bas -p4 -sc -ayour_file
```

Die Anweisung kompiliert die Datei `my_file.bas` für eine ***ADwin***-Karte mit Prozessor T4. Die erzeugte Binärdatei hat den Namen `your_file.t41` und befindet sich im aktuellen Verzeichnis.

```
$ adbasic /home/user/my_file.bas -a/somewhere/your_file
```

Die Binärdatei heißt nun `your_file.t91` und befindet sich im Verzeichnis `/somewhere`.

5 Methoden und Funktionen der *ADwin* Linux

Die Syntax der Methoden und Funktionen aus der Bibliothek ***ADwin*** Linux ist abhängig von der jeweiligen Programmiersprache.
Aus diesem Grund haben wir im folgenden eine allgemeine Syntax zur Beschreibung verwendet. Verwenden Sie die für Ihre Programmiersprache jeweils richtige Syntax.

Die Befehlssyntax ist in folgender Weise dargestellt:

```
Datentyp Befehl(Datentyp Var1, Datentyp Var2, …)
```

Bedeutung

`Datentyp`	Bei einem Befehl: Datentyp des Rückgabewerts. Bei einem Parameter: Datentyp des Parameters. Mögliche Datentypen sind: `void`, `long`, `float`, `double`, `char` Beachten Sie, dass der Datentyp `long` hier gleich gesetzt wird mit `signed int 32bit`.
`Befehl`	In manchen Programmiersprachen (C, C++) müssen die Befehlsnamen zwingend in der angegebenen Groß- und Kleinschreibung verwendet werden.
`Var1`, *`Var2`*	Variablen und Rückgabewerte werden unter „Parameter“ nach der Befehlssyntax beschrieben.

Befehle zum Ansprechen analoger und digitaler Ein- und Ausgänge (und anderer Hardware-Funktionalitäten) sind in der Bibliothek ***ADwin*** Linux nicht enthalten. Sie können solche Anwendungen in ***ADbasic*** programmieren.

5.1 Systemsteuerung und -information

Initialisierung des ***ADwin***-Systems und Informationen über den Betriebszustand.

Set_DeviceNo

Die Funktion `Set_DeviceNo` setzt die „Device No.“ des ***ADwin***-Systems für alle folgenden Funktionen.

```
void Set_DeviceNo(long DeviceNo)
```

Parameter

`DeviceNo`	Device No. des Systems. Typische Device No. sind 336 (= 0x150 hexadezimal) oder 400 (= 0x190 hexadezimal).

Bemerkungen

ADwin-Systeme werden vom PC über die sogenannte „Device No.“ unterschieden und angesprochen. Systeme mit Link-Adapter werden mit hardware-seitig (auf 0x150) eingestellter Device No. geliefert.

Die Funktion `Set_DeviceNo` kommuniziert nicht mit dem ***ADwin***-System und hat daher keinen Einfluss auf die Fehlerbehandlung.

Beispiel

```
Set_DeviceNo(3)
```

Die Device No. ist auf 0x3 gesetzt. Alle weiteren Befehle (bis zu einem neuen `Set_DeviceNo`) beziehen sich auf dieses Gerät.

Boot

`Boot` intialisiert das ***ADwin***-System und lädt die Betriebssystem-Datei dorthin.

```
void Boot(char pFilename())
```

Parameter

Filename Zeiger auf den Dateinamen der Betriebssystem-Datei

Bemerkungen

Die Initialisierung löscht alle Prozesse auf dem System und setzt alle globalen Variablen auf den Wert 0.

Das zu ladende Betriebssystem ist prozessorabhängig. Die folgende Tabelle zeigt die Dateinamen für die verschiedenen Prozessoren.

Prozessor	Kurzname	Betriebssystem-Datei
T225	T2	`ADwin2.btl`
T400	T4	`ADwin4.btl`
T450	T5	`ADwin5.btl`
T805	T8	`ADwin8.btl`
ADSP 21062	T9	`ADwin9.btl`
ADSP 21162	T10	`ADwin10.btl`

Der PC kann erst mit dem ***ADwin***-System kommunizieren, nachdem Sie das Betriebssystem geladen haben. Laden Sie das Betriebssystem nach jedem Aus- und Einschalten des ***ADwin***-Systems neu.

Das erfolgreiche Laden des Betriebssystems mit `Boot` nimmt ca. 1 Sekunde in Anspruch.

Test_Version

`Test_Version` prüft, ob Treiberversion und Betriebssystem des Prozessors zueinander passen, und ob der Prozessor ansprechbar ist.

```
long Test_Version()
```

Parameter

Rückgabewert 0 : OK
1 : Falsche Treiberversion, Prozessor läuft weiter
2 : Falsche Treiberversion, Prozessor wird gestoppt
3 : Keine Antwort vom ***ADwin***-System

Processor_Type

`Processor_Type` gibt den Prozessortyp des Systems zurück.

```
long Processor_Type()
```

Parameter

Rückgabewert Prozessor-Kennziffer des Systems

Bemerkungen

Die Funktion liefert für Prüfzwecke den verwendeten Prozessortyp entsprechend der nachstehenden Tabelle.

Rückgabewert	Prozessortyp
0	Fehler
2	T200
4	T400

Rückgabewert	Prozessortyp
5	T450
8	T800
9	T9 (ADSP 21062)
1010	T10 (ADSP 21162)

Workload

`Workload` gibt die Prozessor-Auslastung zurück.

```
long Workload(long Priority)
```

Parameter

Priority	0 (Null): aktuelle Gesamtauslastung des Prozessors ≠0 : wird derzeit noch nicht unterstützt
Rückgabewert	Prozessor-Auslastung (in Prozent)

Free_Mem

`Free_Mem` ermittelt den auf dem System verfügbaren freien Speicher für verschiedene Speicherarten.

```
long Free_Mem(long Mem_Spec)
```

Parameter

Mem_Spec	Speicherart 0 : alle Speicherarten gemeinsam; nur für Prozessoren T2, T4, T5, T8 1 : interner Programmspeicher (PM_LOCAL); nur für Prozessoren T9 und T10 3 : interner Datenspeicher (DM_LOCAL); nur für Prozessoren T9 und T10 4 : externer DRAM-Speicher (DRAM_EXTERN); nur für Prozessoren T9 und T10
Rückgabewert	Momentaner freier Speicher (in Byte)

Bemerkungen

Die verschiedenen Speicherbereiche sind im ***ADbasic***-Handbuch näher erläutert.

5.2 Prozess-Steuerung

Befehle zur Steuerung einzelner Prozesse auf dem ***ADwin***-System.

Load_Process

`Load_Process` lädt die Binärdatei eines Prozesses ins ***ADwin***-System.

```
void Load_Process(char pFilename())
```

Parameter

Filename	Zeiger auf den Dateinamen der zu ladenden Binärdatei

Bemerkungen

Sie erzeugen eine Binärdatei unter Linux, indem Sie den Compiler `adbasic` aufrufen (ohne den Schalter `-l`, siehe Kapitel 4.1 auf Seite 9).

Das Aus- und Einschalten eines Systems löscht geladene Prozesse. Sie müssen deshalb nach dem Einschalten die von Ihnen benötigten Prozesse erneut laden.

Start_Process

`Start_Process` startet einen Prozess.

```
void Start_Process(long ProcessNo)
```

Parameter

ProcessNo	Nummer des Prozesses (1...10)

Stop_Process

`Stop_Process` stoppt einen Prozess.

```
void Stop_Process(long ProcessNo)
```

Parameter

ProcessNo	Nummer des Prozesses (1...12, 15)

Bemerkungen

Es kann vorkommen, dass ein Prozess nicht sofort nach Durchführen des Befehls gestoppt wird, sondern etwas später. Sie können den Status des Prozesses mit `Process_Status` abfragen.

Clear_Process

`Clear_Process` löscht einen Prozess aus dem Speicher.

```
void Clear_Process(long ProcessNo)
```

Parameter

ProcessNo	Nummer des Prozesses (1...12, 15)

Bemerkungen

Geladene Prozesse belegen Speicherplatz im System. Sie können mit `Clear_Process` Prozesse aus dem Speicher entfernen, um für andere Prozesse mehr Platz zu erhalten.

Ein Prozess, der entfernt werden soll, darf nicht laufen. Stoppen Sie einen laufenden Prozess zuerst mit `Stop_Process` (und stellen Sie ggf. auch das Beenden mit `Process_Status` fest), bevor Sie ihn mit `Clear_Process` aus dem Speicher entfernen.

Auf einem ***ADwin***-System sind nach dem Booten des Betriebssystems die internen Prozesse 11, 12 und 15 verfügbar. Wenn Sie diese entfernen möchten, tun Sie dies in folgender Reihenfolge: 15, 12, 11.
Auf Gold- und Pro-Systemen sorgt der Prozess 15 für das Blinken der LED; nach dem Entfernen blinkt die LED nicht mehr.

Process_Status

`Process_Status` liefert den Status eines Prozesses.

```
long Process_Status(ProcessNo)
```

Parameter

ProcessNo	Nummer des Prozesses (1...12, 15)
Rückgabewert	Status des Prozesses 1 : Prozess läuft. 0 : Prozess läuft nicht, d.h. er ist nicht geladen, nicht gestartet oder gestoppt. -1: Prozess wird gestoppt, d.h. er hat ein `Stop_Process` erhalten, wartet aber noch auf den letzten Event.

Set_Globaldelay

`Set_Globaldelay` stellt den Parameter `Globaldelay` für einen Prozess ein.

```
void Set_Globaldelay(long ProcessNo,
       long Globaldelay)
```

Parameter

ProcessNo	Nummer des Prozesses (1...10).
Globaldelay	Einzustellender Wert für den Parameter `Globaldelay`.

Bemerkungen

Der Parameter *Globaldelay* steuert die Zeitspanne zwischen zwei Event-Aufrufen eines zeitgesteuerten Prozesses (siehe ***ADbasic***-Handbuch).

Die Zeitspanne wird in Zeiteinheiten angegeben, die abhängig sind vom Prozessortyp und der Priorität eines Prozesses.

Prozessortyp	Prozess-Priorität	
	hoch	niedrig
T2, T4, T5, T8	1µs	64µs
ADSP 21062 (T9)	0,025µs (=25ns)	100µs
ADSP 21162 (T10)	0,025µs (=25ns)	50µs

Beispiel (C)

```
Set_Globaldelay(1,2000);
// Bei Prozess 1 wird das Globaldelay auf 2000 gesetzt
```

Bei einem hochpriorisierten, zeitgesteuerten Prozess und einem T9-Prozessor wird der Prozess alle 50µs (= 2000 · 25ns) aufgerufen.

Get_Globaldelay

`Get_Globaldelay` gibt den Parameter `Globaldelay` eines Prozesses zurück.

```
long Get_Globaldelay(long ProcessNo)
```

Parameter

ProcessNo	Nummer des Prozesses (1...10)
Rückgabewert	Aktuell eingestellter Wert für *Globaldelay*; im Fehlerfall: 255

Bemerkungen

Der Parameter *Globaldelay* steuert die Zeitspanne zwischen zwei Event-Aufrufen eines zeitgesteuerten Prozesses (siehe *Set_Globaldelay* und ***ADbasic***-Handbuch).

Beispiel (C)

```
long erg;
erg = Get_Globaldelay(1);
// Globaldelay des ADbasic-Prozesses 1 abfragen
```

5.3 Übertragung von globalen Variablen

Befehle zur Datenübertragung zwischen PC und ***ADwin***-System mit den vordefinierten globalen Variablen `PAR_1` ... `PAR_80` und `FPAR_1` ... `FPAR_80`.

5.3.1 Globale Long-Variablen (`PAR_1` … `PAR_80`)

Set_Par

`Set_Par` setzt eine globale Long-Variable auf den gewünschten Wert.

```
void Set_Par(long Index, long Value)
```

Parameter

Index	Nummer der globalen Long-Variablen (1 … 80)
Value	Zu setzender Wert für die Long-Variable

Get_Par

`Get_Par` gibt den Wert einer globalen Long-Variablen zurück.

```
long Get_Par(long Index)
```

Parameter

Index	Nummer der globalen Long-Variablen (1 … 80)
Rückgabewert	Aktueller Wert der Variablen

Get_Par_Block

`Get_Par_Block` überträgt eine anzugebende Anzahl an globalen Long-Variablen in ein Feld.

```
void Get_Par_Block(long Array(), long StartIndex,
       long Count)
```

Parameter

Array	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
StartIndex	Nummer der ersten zu übertragenden Variablen (1...80)
Count	Anzahl der zu übertragenden Variablen (max. 80)

Get_Par_All

`Get_Par_All` überträgt alle globalen Long-Variablen in ein Feld.

```
void Get_Par_All(long Array())
```

Parameter

Array	Zeiger auf ein Feld (Feldlänge ≥ 80)

5.3.2 Globale Float-Variablen (FPAR_1...FPAR_80)

Set_FPar

Set_FPar setzt eine globale Float-Variable auf den gewünschten Wert.

```
void Set_FPar(long Index, long Value)
```

Parameter

Index	Nummer der globalen Float-Variablen (1...80)
Value	Zu setzender Wert für die Float-Variable

Get_FPar

Get_FPar gibt den Wert einer globalen Float-Variablen zurück.

```
float Get_FPar(long Index)
```

Parameter

Index	Nummer der globalen Float-Variablen (1...80)
Rückgabewert	Aktueller Wert der Variablen

Get_FPar_Block

Get_FPar_Block überträgt eine anzugebende Anzahl an globalen Float-Variablen in ein Feld.

```
void Get_FPar_Block(float Array(), long StartIndex,
       long Count)
```

Parameter

Array	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
StartIndex	Nummer der ersten zu übertragenden Variablen (1...80)
Count	Anzahl der zu übertragenden Variablen (max. 80)

Get_FPar_All

Get_FPar_All überträgt alle globalen Float-Variablen in ein Feld.

```
void Get_FPar_All(float Array())
```

Parameter

Array	Zeiger auf ein Feld (Feldlänge ≥ 80)

Get_FPar_Block_Double

`Get_FPar_Block_Double` überträgt eine anzugebende Anzahl an globalen Float-Variablen in ein Double-Feld.

```
void Get_FPar_Block_Double(double Array(),
       long StartIndex, long Count)
```

Parameter

Array	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
StartIndex	Nummer der ersten zu übertragenden Variablen (1...80)
Count	Anzahl der zu übertragenden Variablen (max. 80)

Bemerkungen

Diese Funktion wurde für Applikationen entwickelt, die Fließkomma-Datensätze nur mit doppelter Genauigkeit (double precision arrays) verarbeiten können.

Bitte beachten Sie, dass die Daten auf dem ***ADwin***-System immer mit einfacher Genauigkeit (single precision) verarbeitet werden. Sie sollten daher Daten aus dem Feld *Array()* nur mit einfacher Genauigkeit anzeigen lassen, um Missverständnisse bezüglich der Genauigkeit zu vermeiden.

Get_FPar_All_Double

`Get_FPar_All_Double` überträgt alle globalen Float-Variablen in ein Double-Feld.

```
void Get_FPar_All_Double(double Array())
```

Parameter

Array	Zeiger auf ein Feld (Feldlänge ≥ 80)

Bemerkungen

Diese Methode wurde für Applikationen entwickelt, die Fließkomma-Datensätze nur mit doppelter Genauigkeit (double precision arrays) verarbeiten können.

Bitte beachten Sie, dass die Genauigkeit der übertragenen Daten einfach (single precision) ist. Sie sollten daher Daten aus dem Feld *Array()* nur mit einfacher Genauigkeit anzeigen lassen, um Missverständnisse bezüglich der Genauigkeit zu vermeiden.

5.4 Übertragung von Datenfeldern (Arrays)

Befehle zur Datenübertragung zwischen PC und ***ADwin***-System mit globalen DATA-Feldern (DATA_1…DATA_200), darunter auch FIFO-Felder.

Achten Sie darauf, dass Sie jedes Feld vor seiner Verwendung im ***ADbasic***-Programm mit DIM deklarieren müssen (vgl. Handbuch „***ADbasic***“).

5.4.1 Einfache Datenfelder

Data_Length

Data_Length gibt die in ***ADbasic*** deklarierte Länge eines Felds zurück, d. h. die Anzahl der Elemente.

```
long Data_Length(long DataNo)
```

Parameter

DataNo	Nummer des Felds (1...200)
Rückgabewert	Deklarierte Länge des Felds (=Anzahl der Elemente)

SetData_Long

SetData_Long überträgt Long-Daten vom PC in ein DATA-Feld des ***ADwin***-Systems.

```
void SetData_Long(long DataNo, long Data(),
       long Startindex, long Count)
```

Parameter

DataNo	Feldnummer (1...200)
Data()	Zeiger auf ein Feld
StartIndex	Nummer der ersten zu übertragenden Variablen
Count	Anzahl der zu übertragenden Variablen

GetData_Long

GetData_Long überträgt Long-Daten aus einem DATA-Feld vom ***ADwin***-System in ein Feld.

```
void GetData_Long(long DataNo, long Data(),
       long Startindex, long Count)
```

Parameter

DataNo	Feldnummer (1...200)
Data()	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
StartIndex	Nummer der ersten zu übertragenden Variablen
Count	Anzahl der zu übertragenden Variablen

SetData_Float

SetData_Float überträgt Float-Daten vom PC in ein DATA-Feld des ***ADwin***-Systems.

```
void SetData_Float(long DataNo, float Data(),
       long Startindex, long Count)
```

Parameter

DataNo	Feldnummer (1...200)
Data()	Zeiger auf ein Feld
StartIndex	Nummer der ersten zu übertragenden Variablen
Count	Anzahl der zu übertragenden Variablen

GetData_Float

GetData_Float überträgt Float-Daten aus einem DATA-Feld vom ***ADwin***-System in ein Feld.

```
void GetData_Float(long DataNo, float Data(),
       long Startindex, long Count)
```

Parameter

DataNo	Feldnummer (1...200)
Data()	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
StartIndex	Nummer der ersten zu übertragenden Variablen
Count	Anzahl der zu übertragenden Variablen

SetData_Double

SetData_Double überträgt Double-Daten vom PC mit einfacher Genauigkeit in ein DATA-Feld des ***ADwin***-Systems.

```
void SetData_Double(long DataNo, double Data(),
       long Startindex, long Count)
```

Parameter

DataNo	Feldnummer (1...200)
Data()	Zeiger auf ein Feld
StartIndex	Nummer der ersten zu übertragenden Variablen
Count	Anzahl der zu übertragenden Variablen

Bemerkungen

Diese Methode wurde für Applikationen entwickelt, die Fließkomma-Datensätze nur mit doppelter Genauigkeit (double precision arrays) verarbeiten können.

Bitte beachten Sie, dass die Daten auf dem ***ADwin***-System immer mit einfacher Genauigkeit (single precision) verarbeitet werden. Sie sollten daher Daten aus dem Feld *Data()* nur mit einfacher Genauigkeit anzeigen lassen, um Missverständnisse bezüglich der Genauigkeit zu vermeiden.

GetData_Double

`GetData_Double` überträgt Float-Daten aus einem `DATA`-Feld vom ***ADwin***-System in ein Double-Feld.

```
void GetData_Double(long DataNo, double Data(),
       long Startindex, long Count)
```

Parameter

DataNo	Feldnummer (1...200)
Data()	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
StartIndex	Nummer der ersten zu übertragenden Variablen
Count	Anzahl der zu übertragenden Variablen

Bemerkungen

Diese Methode wurde für Applikationen entwickelt, die Fließkomma-Datensätze nur mit doppelter Genauigkeit (double precision arrays) verarbeiten können.

Bitte beachten Sie, dass die Genauigkeit der übertragenen Daten einfach (single precision) ist. Sie sollten daher Daten aus dem Feld *Data()* nur mit einfacher Genauigkeit anzeigen lassen, um Missverständnisse bezüglich der Genauigkeit zu vermeiden.

Data2File

`Data2File` speichert Long- oder Float-Daten aus einem `DATA`-Feld des ***ADwin***-Systems in einer Datei (auf der Festplatte).

```
void Data2File(char pFilename(), long DataNo,
       long Startindex, long Count, long Mode)
```

Parameter

Filename	Zeiger auf den Dateinamen
DataNo	Nummer des `DATA`-Felds (1...200)
Startindex	Nummer der ersten zu übertragenden Variablen
Count	Anzahl der zu übertragenden Elemente
Mode	0 : Datei wird überschrieben (falls vorhanden) 1 : Daten werden an eine vorhandene Datei angehängt

Bemerkungen

Das `DATA`-Feld darf nicht als `FIFO` definiert sein. Die Daten werden binär gespeichert.

Beispiel (C)

```
Data2File("Test.dat", 1, 1, 1000, 0);
// Speichert die Elemente 1…1000 aus dem ADbasic-
// Feld DATA_1 in der Datei Test.dat
```

5.4.2 FIFO-Felder

Befehle zur Datenübertragung zwischen PC und ***ADwin***-System mit globalen DATA-Feldern (DATA_1…DATA_200), die als FIFO deklariert sind.

Sie müssen jedes FIFO-Feld vor seiner Verwendung unter ***ADbasic*** deklarieren (vgl. Handbuch „***ADbasic***“): `DIM DATA_x[n] as TYPE as FIFO`

Beim Arbeiten mit FIFO-Feldern sollten Sie als Faustregel nicht mehr Elemente beschreiben als deklariert sind. Auf keinen Fall sollten Sie mehr Elemente auslesen als vorher beschrieben wurden. Vermeiden Sie dies wie folgt:

– Prüfen Sie mit der Funktion `Fifo_Full`, ob ein FIFO-Feld mindestens so viele Elemente enthält, wie Sie mit `GetFifo_X` auslesen möchten.

– Prüfen Sie mit der Funktion `Fifo_Empty`, ob ein FIFO-Feld mindestens so viele freie Elemente enthält, wie mit `SetFifo_X` beschreiben möchten.

Fifo_Empty

`Fifo_Empty` liefert die Anzahl der freien Elemente eines Fifo-Felds.

```
long Fifo_Empty(long FifoNo)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)
Rückgabewert	Freie Elemente im Fifo-Feld

Fifo_Full

`Fifo_Full` liefert die Anzahl der belegten Elemente eines Fifo-Felds.

```
long Fifo_Full(long FifoNo)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)
Rückgabewert	Belegte Elemente im Fifo-Feld

Fifo_Clear

`Fifo_Clear` initialisiert den Schreib- und den Lesezeiger eines Fifo-Felds.

```
void Fifo_Clear(long FifoNo)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)

Bemerkungen

Beim Deklarieren eines FIFO-Felds (im ***ADbasic***-Programm) werden die FIFO-Zeiger nicht automatisch initialisiert. Rufen Sie deshalb `Fifo_Clear` gleich zu Beginn Ihres Programms auf, entweder in ***ADbasic*** oder mit dieser Funktion.

Das Initialisieren der FIFO-Zeiger im Programmablauf ist sinnvoll, wenn alle beschriebenen Elemente (z.B. wegen eines Fehlers) verworfen werden sollen.

SetFifo_Long

`SetFifo_Long` überträgt Long-Daten aus dem PC in ein Fifo-Feld des ***ADwin***-Systems.

```
void SetFifo_Long(long FifoNo, long Data(),
       long Count)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)
Data()	Zeiger auf ein Feld
Count	Anzahl der zu übertragenden Elemente

GetFifo_Long

`GetFifo_Long` überträgt Long-Daten aus einem Fifo-Feld vom ***ADwin***-System in den PC.

```
void GetFifo_Long(long FifoNo, long Data(),
       long Count)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)
Data()	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
Count	Anzahl der zu übertragenden Elemente

SetFifo_Float

`SetFifo_Float` überträgt Float-Daten aus dem PC in ein Fifo-Feld des ***ADwin***-Systems.

```
void SetFifo_Float(long FifoNo, float Data(),
       long Count)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)
Data()	Zeiger auf ein Feld
Count	Anzahl der zu übertragenden Elemente

GetFifo_Float

`GetFifo_Float` überträgt Float-Daten aus einem Fifo-Feld vom ***ADwin***-System in den PC.

```
void GetFifo_Float(long FifoNo, float Data(),
       long Count)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)
Data()	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
Count	Anzahl der zu übertragenden Elemente

SetFifo_Double

`SetFifo_Double` überträgt Double-Daten aus dem PC mit einfacher Genauigkeit in ein Fifo-Feld des ***ADwin***-Systems.

```
void SetFifo_Double(long FifoNo, double Data(),
       long Count)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)
Data()	Zeiger auf ein Feld
Count	Anzahl der zu übertragenden Elemente

Bemerkungen

Diese Methode wurde für Applikationen entwickelt, die Fließkomma-Datensätze nur mit doppelter Genauigkeit (double precision arrays) verarbeiten können.

Bitte beachten Sie, dass die Daten auf dem ***ADwin***-System immer mit einfacher Genauigkeit (single precision) verarbeitet werden. Sie sollten daher Daten aus dem Feld *Data()* nur mit einfacher Genauigkeit anzeigen lassen, um Missverständnisse bezüglich der Genauigkeit zu vermeiden.

GetFifo_Double

`GetFifo_Double` überträgt Float-Daten aus einem Fifo-Feld vom ***ADwin***-System in ein Double-Feld im PC.

```
void GetFifo_Double(long FifoNo, double Data(),
       long Count)
```

Parameter

FifoNo	Nummer des Fifo-Felds (1...200)
Data()	Zeiger auf ein Feld (Feldlänge ≥ *Count*)
Count	Anzahl der zu übertragenden Elemente

Bemerkungen

Diese Methode wurde für Applikationen entwickelt, die Fließkomma-Datensätze nur mit doppelter Genauigkeit verarbeiten.

Bitte beachten Sie, dass die Genauigkeit der übertragenen Daten einfach ist. Sie sollten daher Daten aus dem Feld *Data()* mit einfacher Genauigkeit anzeigen, um Missverständnisse bzgl. der Genauigkeit zu vermeiden.

5.4.3 Datenfelder mit String-Daten

Befehle zur Datenübertragung zwischen PC und ***ADwin***-System mit globalen DATA-Feldern (DATA_1…DATA_200), die String-Daten enthalten. Das DATA-Feld muss in ***ADbasic*** mit DIM … AS STRING deklariert werden.

SetData_String

SetData_String überträgt einen String in ein DATA-Feld.

```
void SetData_String(long DataNo, char pData())
```

Parameter

DataNo	Nummer des DATA-Felds (1...200)
pData	Zeiger auf den zu übertragenden String

Bemerkungen

Jedem übertragenen String wird zusätzlich als letztes Zeichen die Endekennung (ASCII-Nummer 0) angehängt.

Beachten Sie den unterschiedlichen Umgang der Programmiersprachen mit der Endekennung, wenn diese im zu übertragenden String enthalten ist.

Beispielsweise übertragen ***ADbasic*** und C den String „Hello\0 world" nur bis zur Endekennung, also „Hello“. Dagegen überträgt beispielsweise Delphi (Kylix) die vollständige Zeichenfolge.

Beispiel (C)

```
SetData_String (2, "Dies ist ein Text" );
// Der String "Dies ist ein Text" wird in das Feld
// DATA_2 geschrieben und die Endekennung angehängt.
```

GetData_String

GetData_String überträgt einen String aus einem DATA-Feld in ein Feld.

```
long GetData_String(long DataNo, char pData(),
       long MaxCount)
```

Parameter

DataNo	Nummer des DATA-Felds (1...200)
pData	Zeiger auf das Feld, in das der String geschrieben werden soll (Feldlänge ≥ *MaxCount*+1)
MaxCount	Max. Anzahl der zu übertragenden Zeichen
Rückgabewert	Anzahl der gelesenen Zeichen (ohne Endekennung)

Bemerkungen

Nach dem Schreiben des Strings in das Feld *pData* wird zusätzlich als letztes Zeichen die Endekennung (ASCII-Nummer 0) angehängt, so dass der String insgesamt *MaxCount*+1 Zeichen enthält.

Wenn der zu übertragende String eine Endekennung beinhaltet, stoppt die Übertragung genau dort. Die Anzahl der bis dahin gelesenen Zeichen ohne die Endekennung ist der Rückgabewert.

Beispiel (C)

```
char ArrayString[101];
GetData_String (2, ArrayString, 100);
// einen String mit 100 Elementen aus DATA_2 holen
```

Enthält das DATA-Feld im ***ADwin***-System z. B. an Position 9 eine Endekennung, so werden 8 Zeichen gelesen.

String_Length

String_Length gibt die Länge eines Datenstrings in einem DATA-Feld zurück.

```
long String_Length(long DataNo)
```

Parameter

DataNo	Nummer des DATA-Felds (1...200)
Rückgabewert	Länge des Strings

Bemerkungen

String_Length zählt die Zeichen im DATA-Feld bis zum ersten Auftreten der Endekennung (ASCII-Nummer 0). Die Endekennung selbst wird nicht als Zeichen gezählt.

Beispiel (C)

```
long erg;
erg = String_Length (2);
// Ermittelt die Länge des Strings in DATA_2.
```

5.5 Fehlerbehandlung

Get_Last_Error

`Get_Last_Error` gibt die Fehlernummer zurück, die sich auf den zuletzt an das ***ADwin***-System gesendeten Befehl bezieht.

```
long Get_Last_Error()
```

Parameter

Rückgabewert ≠0: Nummer des zuletzt aufgetretenen Fehlers (siehe Anhang A-1).
0 : kein Fehler aufgetreten

Bemerkungen

Die zurückgegebene Fehlernummer bezieht sich immer auf

- den letzten Befehl, der an das ADwin-System gesendet wurde (die meisten, aber nicht alle Befehle gehören zu dieser Gruppe).
- das Programm, das den Befehl gesendet hat.

Nachdem die Funktion ausgeführt wurde, wird die Fehlernummer auf 0 (Null) zurückgesetzt.

Beispiel (C)

```
long Error;
Error = Get_Last_Error();
```

Get_Last_Error_Text

`Get_Last_Error_Text` gibt einen Fehlertext zu einer vorhandenen Fehlernummer zurück.

```
char Get_Last_Error_Text(long Errno)
```

Parameter

Errno Fehlernummer

Rückgabewert Zeiger auf Fehlertext

Bemerkungen

Diese Funktion wird im Allgemeinen in Verbindung mit der Funktion `Get_Last_Error` aufgerufen.

Beispiel (C)

```
long Error = Get_Last_Error();
printf(„Get_Last_Error:%d\n", Error);
printf(„Last_Error_Text:%s\n",
 Get_Last_Error_Text(Error));
```

Anhang

A-1 Fehlermeldungen

Fehler-Nr.	Fehlermeldung (nur englisch möglich)
0	No Error.
1	Timeout error on writing to the ADwin-system.
2	Timeout error on reading from the ADwin-system.
3	Timeout error on fast writing to the ADwin-system.
4	Timeout error on fast reading from the ADwin-system.
10	The device No. is not allowed.
11	The device No. is not known.
15	Function for this device not allowed.
20	Incompatible versions of ADwin operating system (*.btl), driver and/or ADbasic binary-file.
100	The Data is too small.
101	The Fifo is too small or not enough values.
102	The Fifo has not enough values.
200	File not found.
201	A temporary file could not be created.
202	The file is not an ADBasic binary-file.
203	The file is not valid.
204	The file is not a BTL.
1000	Network error (RPC).
1001	Network error (Bind).
1002	Network error (String bind).
1003	Wrong password (RPC).
2000	Network error (TcpIp).
2001	Network timeout.
2002	Wrong password.
3001	Device is unknown.

A-2 Index der Methoden und Properties

B
Boot · 16

C
Clear_Process · 18

D
Data_Length · 24
Data2File · 26

F
Fifo_Clear · 27
Fifo_Empty · 27
Fifo_Full · 27
Free_Mem · 17

G
Get_FPar · 22
Get_FPar_All · 22
Get_FPar_All_Double · 23
Get_FPar_Block · 22
Get_FPar_Block_Double · 23
Get_Globaldelay · 20
Get_Last_Error · 32
Get_Last_Error_Text · 32
Get_Par · 21
Get_Par_All · 21
Get_Par_Block · 21
GetData_Double · 26
GetData_Float · 25
GetData_Long · 24
GetData_String · 31
GetFifo_Double · 29
GetFifo_Float · 28
GetFifo_Long · 28

L
Load_Process · 18

P
Process_Status · 19
Processor_Type · 16

S
Set_DeviceNo · 15
Set_FPar · 22
Set_Globaldelay · 19
Set_Par · 21
SetData_Double · 25
SetData_Float · 25
SetData_Long · 24
SetData_String · 30
SetFifo_Double · 29
SetFifo_Float · 28
SetFifo_Long · 28
Start_Process · 18
Stop_Process · 18
String_Length · 31

T
Test_Version · 16

W
Workload · 17