ADbasic

Tutorial und Programmierbeispiele

ADbasic Tutorial Version 1.4, Juni 2003

Inhaltsverzeichnis

1 Wichtige Hinweise

Achtung! Bevor Sie mit ***ADbasic*** programmieren, müssen Sie Ihre Hard- und Software vollständig installiert haben. Beachten Sie hierzu bitte die Installationsanleitung „***ADwin*** Treiber-Installation“
(siehe auch Datei „`Driver-Installation_ger.pdf`“ im Verzeichnis `<C:\ADwin\Documents\Setup>`).

Die auf den folgenden Seiten aufgeführten Beispielprogramme sollen Sie auf einfache Weise in das Programmieren mit ***ADbasic*** einführen, damit Sie schon nach kurzer Zeit selbstständig Lösungen erstellen können.

Beachten Sie bitte, dass Sie den vollen Leistungsumfang von ***ADbasic*** erst im alltäglichen Umgang mit ***ADbasic*** unter Zuhilfenahme des ***ADbasic***-Handbuchs oder der Online-Hilfe erlernen werden.

Gebrauch der Beispielprogramme

Zu fast jedem Beispiel finden Sie den Programm-Quelltext in dem Verzeichnis `<C:\ADwin\ADbasic\Samples_ADwin>` (bei Standard-Installation).

Für das Testen eines Beispielprogramms benötigen Sie ein funktionsfähig angeschlossenes ***ADwin***-System, ***ADbasic*** zum Editieren der Programme und die auf ihrem Rechner installierten ***ADwin***-Treiber.

2 Erste Schritte mit *ADbasic*

Wir gehen davon aus, dass Sie Ihr ***ADwin***-System mit Hilfe des Installations-Handbuchs bereits erfolgreich installiert haben. Hierzu gehört auch, dass Sie dem ***ADwin***-System mit dem Programm *ADconfig* die passende Gerätenummer (Device No.; Standard: `150h`) zugewiesen haben.

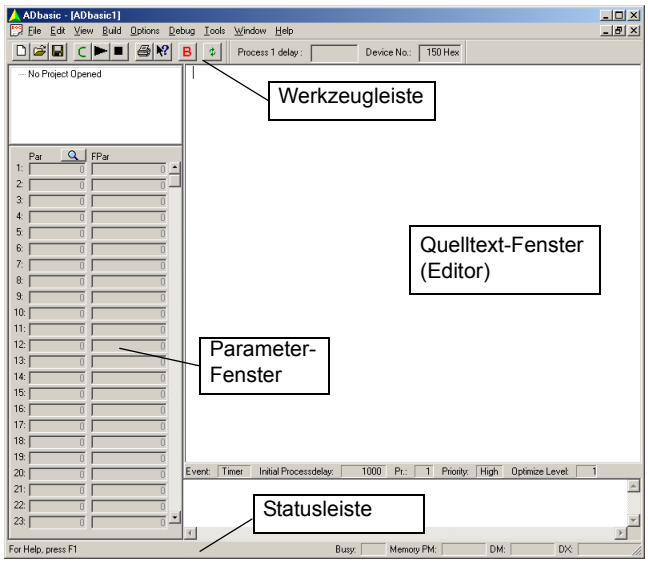
2.1 Kommunikation prüfen

Im ersten Schritt sollten Sie noch einmal die Kommunikation zwischen dem PC und Ihrem ***ADwin***-System überprüfen. Haben Sie seit der Installation keine Veränderungen vorgenommen, sind automatisch noch die richtigen Voreinstellungen gewählt.

➢ Starten Sie dazu die Entwicklungsumgebung ***ADbasic*** aus dem Windows Startmenü: `Programs` ▸ `ADwin` ▸ `ADbasic`.

Sie sehen nun die Standard-Oberfläche der Entwicklungsumgebung:

Einige Bedienelemente der Entwicklungsumgebung



➢ Klicken Sie mit der linken Maustaste auf die Schaltfläche B (Boot) in der Werkzeugleiste (am oberen Rand des Fensters).
Damit lösen Sie den Boot-Vorgang aus, der das Betriebssystem auf Ihr ***ADwin***-System überträgt und startet.

Mit dem Übertragen des Betriebssystems (= Booten) initialisieren Sie gleichzeitig Ihr ***ADwin***-System. Sie versetzen das System damit in einen definierten Anfangszustand, in dem der Speicher frei für neue Programme ist. Das Booten muss nach jedem Aus- und Einschalten des ***ADwin***-Systems wiederholt werden.

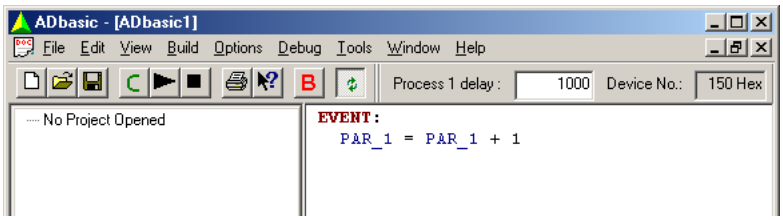
➢ Ob das Betriebssystem erfolgreich übertragen wurde, können Sie an jeder der folgenden Veränderungen im ***ADbasic***-Entwicklungssystem leicht erkennen:
- In der Statuszeile erscheint die Meldung `ADwin is booted`.
- Die Arrays im Parameterfenster werden hellgrau, d.h. sie sind aktiv.
- In der Werkzeugleiste wechselt die Schaltfläche in den aktiven Zustand („`Enable cyclic update`“, rechts neben der Boot-Schaltfläche B).
- Bei einem ***ADwin-Gold***-System oder bei dem Prozessor-Modul eines ***ADwin-Pro***-Systems blinkt die LED.

Klappt die Kommunikation mit Ihrem ***ADwin***-System, werden Sie im nächsten Abschnitt ein erstes Programmbeispiel kennen lernen. Ansonsten finden Sie im Anhang Hinweise zur Fehlersuche.

2.2 Das erste Programm

Das erste Programm besteht nur aus 2 Zeilen, mit denen die globale Variable PAR_1 periodisch erhöht wird.

➢ Geben Sie dazu das folgende Programm im Editorfenster ein:



Vergessen Sie nicht den Doppelpunkt hinter „**EVENT**“ einzugeben und den Unterstrich (Underscore) bei „PAR_1“.

➢ Klicken Sie mit der linken Maustaste auf die Schaltfläche (Compile) in der Werkzeugleiste.

Sie geben damit den Befehl, Ihr Programm zu kompilieren, es in das ***ADwin***-System zu übertragen und es dort zu starten.

➢ Beobachten Sie die Veränderung im Parameter-Fenster.

Sie sehen, dass im Eingabefeld oben links keine Null, sondern eine stetig wachsende Zahl angezeigt wird. Diese Zahl ist der jeweils aktuelle Wert der globalen Variablen PAR_1.

Hinweise

Das kontinuierliche Hochzählen der Variablen PAR_1 ist die Folge davon, dass das Programm automatisch in festen Zeitabständen, also zyklisch aufgerufen wird.

In dem Programm definiert die Zeile mit dem Schlüsselwort **EVENT:** den Beginn des Abschnitts, der zyklisch aufgerufen wird, in diesem Fall eine einzige Programmzeile

```
PAR_1 = PAR_1 + 1
```

die bei jedem Aufruf den Wert der Variablen PAR_1 um 1 erhöht.

Die Zykluszeit, mit der die einzelnen Aufrufe dieser Programmzeile erfolgen, können Sie im Eingabefeld Process 1 delay direkt eingeben und zwar in Einheiten von 25 ns (= 1 Taktzyklus des Prozessors). Bei einem kleineren Wert erfolgt die Abarbeitung – also hier das Hochzählen – schneller, bei einem größeren langsamer. Der Wert 40.000.000 bewirkt beispielsweise, dass die Zeile jede Sekunde ausgeführt wird.

Beachten Sie auch in der Statusleiste (am unteren Fensterrand) die Anzeige Busy. Er gibt Ihnen an, wie stark der Prozessor des ***ADwin***-Systems durch Ihren Prozess ausgelastet ist. Die Auslastung sollte in jedem Fall kleiner als 100 % bleiben, sonst kann das System in einen instabilen Zustand geraten.

Die in diesem Beispiel verwendete Variable besitzt den Namen PAR_1. Dadurch wird eine vordefinierte, globale Variable für ganze Zahlen gekennzeichnet, die für den bidirektionalen Datenverkehr zwischen parallel laufenden ***ADbasic***-Prozessen oder für den Austausch zwischen PC und ***ADbasic***-Prozessen verwendet wird. Sie wird auch als Parameter bezeichnet. Insgesamt stehen 80 solcher Variablen (PAR_1 bis PAR_80) zur Verfügung. Sie können sie an jeder beliebigen Stelle im Programm einsetzen

Auf lokale Variablen, Arrays und andere Datentypen gehen wir später ein.

3 A/D- und D/A-Wandlung

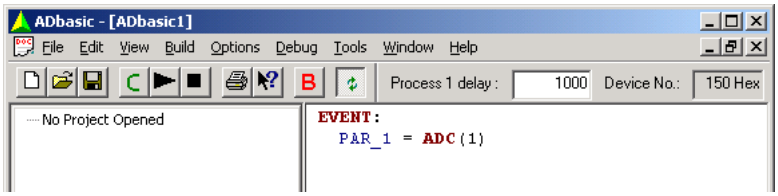
Wenn Sie ein ***ADwin-Pro***-System verwenden, fahren Sie bitte fort mit dem Abschnitt 3.2.

3.1 *ADwin-Gold*- und *ADwin-light-16*-Systeme

Um eine analoge Spannung mit dem A/D-Wandler des ***ADwin***-Systems in eine digitale Bitfolge umzusetzen, verwenden wir in diesem Beispiel den Befehl **ADC**; gleichzeitig wird der konvertierte Wert in die globale Variable PAR_2 geschrieben.

Analoge Spannung lesen

➢ Geben Sie das folgende Programm ein und kompilieren es:



Sie sollten jetzt im Parameterfenster in der Zeile PAR_1 einen Wert sehen, der um den Wert 32.768 schwankt. Dieser Wert entspricht einer gemessenen Spannung von 0V, was in diesem Fall korrekt ist, da der analoge Eingang 1 offen ist. Die Schwankungen resultieren aus Rauschen am Eingang (die Spannung wird ja bei jedem Prozesszyklus neu gemessen).

Sie können mit Hilfe des Befehls **ADC** Spannungen an den analogen Eingängen zwischen -10V und +10V messen. Dieses entspricht digitalisiert den Werten von 0 bis 65.535 (siehe auch Exkurs).

Im Unterschied zum ersten Programm können Sie an der Anzeige Busy erkennen, dass dieses Programm wesentlich mehr Prozessorzeit benötigt, obwohl bei jedem Prozesszyklus wieder nur eine Programmzeile ausgeführt wird. Die vergleichsweise lange Ausführungszeit des Befehls **ADC** kommt daher, dass für jede einzelne Messung die erforderliche Wandlungszeit abgewartet wird.

Analog-Signal ausgeben

Erweitern Sie nun Ihr Programm, indem Sie zusätzlich ein Analog-Signal über den D/A-Wandler ausgeben und dieses (wie vorher) mit dem A/D-Wandler zurück messen:

➢ Zu diesem Zweck verbinden Sie den analogen Ausgang 1 mit dem analogen Eingang 1 (+) sowie die analoge Masse (AGND) mit dem Eingang 1 (–).
Die Pinbelegung entnehmen Sie bitte der Hardware-Beschreibung Ihres ***ADwin***-Systems.

Das Ansprechen des Digital-/Analog-Wandlers erfolgt mit Hilfe des Befehls **DAC**, dem Sie die Nummer des Analog-Ausgangs und den zu wandelnden Wert übergeben müssen:

```
EVENT:
  DAC(1, PAR_2)
  PAR_1 = ADC(1)
```

Nach dem Übertragen des Programms auf das ***ADwin***-System können Sie den gemessenen Wert wieder in der globalen Variablen PAR_1 ablesen. Dieser schwankt jetzt um den Wert 0, weil der D/A-Wandler einen Wert von -10V ausgibt, entsprechend dem Wert 0 der globalen Variable PAR_2. Dass es tatsächlich -10V sind, die am analogen Ausgang 1 anliegen, können Sie mit einem externen Multimeter nachprüfen.

Den Wert, den der D/A-Wandler am analogen Ausgang ausgibt, können Sie sehr einfach verändern.

➢ Klicken Sie im Parameterfenster das Eingabefeld PAR_2 an und überschreiben den angezeigten Wert.
Nach der Bestätigung mit der Eingabetaste sehen Sie, dass sich auch der Wert in PAR_1 geändert hat. Auch die Anzeige eines angeschlossenen Multimeters sollte sich entsprechend ändern.

Funktionsgenerator

Dieses zweizeilige Programm lässt sich sehr einfach zu einem Funktionsgenerator erweitern, der z.B. eine Rampenfunktion am Ausgang liefert. Dazu kombinieren wir das erste Beispiel zum Hochzählen mit dem letzten AD-DA-Beispiel:

```
EVENT:
  DAC(1, PAR_2)
  PAR_1 = ADC (1)
  PAR_2 = PAR_2 + 1
  IF (PAR_2 > 65535) THEN     ' 16 Bit Wandler
    PAR_2 = 0
  ENDIF
```

Wenn Sie dieses Programm kompilieren, sehen Sie, dass sich die Werte der globalen Variablen PAR_2 (und auch PAR_1) ändern.

➢ Schließen Sie ein Oszilloskop (ein Multimeter ist zu träge) am analogen Ausgang 1 an und beobachten Sie die Rampenfunktion der Spannung, entsprechend dem ansteigenden Wert von PAR_2.

Die Abfrage mit **IF** am Programmende dient dazu, den Zählerstand der globalen Variablen PAR_2 nach Erreichen des Maximalwerts von 65.535 wieder auf den Wert 0 zurückzusetzen. Dieses ist notwendig, weil die Funktion **DAC** Werte oberhalb des Maximalwerts auf 65.535 (bei 16 Bit-Wandlern) korrigiert.

Exkurs

Analog-Wandlung mit verschiedenen Wandlerbreiten

Die Eingangs-Wandler (ADC) der ***ADwin***-Systeme gibt es mit einer Wandlungsbreite von 12 Bit, 14 Bit oder 16 Bit, die Ausgangswandler (DAC) nur mit 16 Bit.

Ein 16 Bit-ADC wandelt die anliegende Spannung in einen Wert zwischen 0 und 65.535 Digits.
Hat der Wandler eine Breite von 12 Bit oder 14 Bit, dann wird die Spannung in entsprechend groberen Schritten „gerastert“. Damit Sie diese Werte mit denen des 16 Bit-ADC vergleichen können, werden zum 14 Bit-Wert von rechts her 2 Null-Bits hinzugefügt und beim 12 Bit-Wert 4 Null-Bits.

Daraus ergibt sich, dass die Messwerte bei verschiedenen Wandlerbreiten mit unterschiedlichen Schrittweiten zurückgegeben werden. Der maximale Messwert (Messbereich -10V…+10V) ist gerade eine Schrittweite kleiner als 2^{16} Digits:

	Schrittweite	max. Messwert
16 Bit-Wandler	1 Digit	65.535 Digits
14 Bit-Wandler	4 Digits	65.532 Digits
12 Bit-Wandler	16 Digits	65.520 Digits

3.2 *ADwin-Pro*-Systeme

Besonderheiten bei Pro-Systemen

Die flexible Ausrüstung der ***ADwin-Pro***-Systeme erfordert, dass Sie am Anfang jedes Programms bestimmte Dateien mit der Anweisung **#INCLUDE** einbinden. Vergessen Sie dieses, meldet der Compiler bei der Übersetzung einen Fehler. Außerdem müssen Sie bei allen Befehlen die Adresse des angesprochenen Pro-Moduls angeben.

Include-Datei

Jede dieser Include-Dateien enthält Definitionen von Befehlen, mit denen Sie auf die Hardware einer bestimmten Klasse von Pro-Modulen zugreifen können. Welche Datei Sie einbinden müssen, hängt also davon ab, welche Befehle (und welche Module) Sie verwenden. Diese Include-Dateien müssen immer am Anfang des Quelltextes eingebunden werden.
Die Zuordnung der Befehle zu den einzelnen Include-Dateien können Sie der Dokumentation mit dem Titel „***ADwin-Pro***: Systembeschreibung; Programmierung in ***ADbasic***" entnehmen. In dem nachfolgenden Beispiel ist für den Befehl **ADC** das Einbinden der Datei `ADWPAD.INC` erforderlich, für die Anweisung **DAC** die Datei `ADWPDA.INC`.

Geben Sie bitte immer den vollständigen Pfad zu den Include-Dateien an oder stellen Sie sicher, dass diese im Standardverzeichnis liegen. In diesem Beispiel werden die Dateien unter `C:\ADwin\ADbasic\Inc\` erwartet.
Sie können das Standardverzeichnis mit Hilfe des Menüpunktes `Options/Settings` anpassen; stellen Sie hierzu im Dialogfenster `Settings`, Blatt `Directory` die Option `Include-Directory` entsprechend ein (mit einem „\" als letztem Zeichen).

Moduladresse

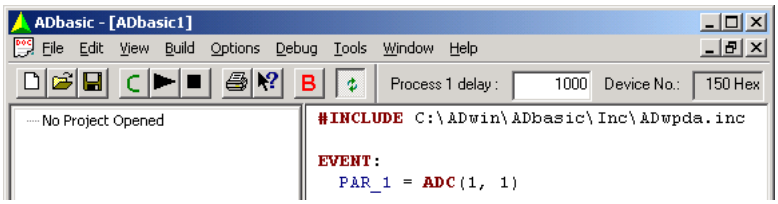
Um ein bestimmtes Pro-Modul anzusprechen, müssen Sie als erstes Argument eines Befehls immer die Moduladresse angeben. Daher haben Befehle für *Pro*-Systeme immer ein Argument mehr als funktionsgleiche Befehle für *Gold*- oder *light-16*-Systeme. Sie finden in der Pro-Hardware Handbuch Angaben, wie Sie die Adresse eines Moduls einstellen.
In diesem Beispiel werden ein A/D-Modul Pro-AIn-x (mit Multiplexer) mit der Adresse 1 und ein D/A-Modul Pro-AOut-x mit der Adresse 1 verwendet.

Das Beispiel beginnt

Um eine analoge Spannung mit dem A/D-Wandler des ***ADwin***-Systems in eine digitale Bitfolge umzusetzen, verwenden wir in diesem Beispiel den Befehl **ADC**, der den konvertierten Wert in die globale Variable `PAR_1` schreibt.

Analoge Spannung lesen

➢ Geben Sie das folgende Programm ein und kompilieren es:



Sie sollten jetzt im Parameterfenster in der Zeile `PAR_1` einen Wert sehen, der um einen Mittelwert von 32.768 schwankt. Dieser Mittelwert entspricht einem gemessenen Wert von 0V, was in diesem Fall korrekt ist, da der analoge Eingang 1 offen ist. Die Schwankungen um den Mittelwert resultieren aus Rauschen am Eingang (die Spannung wird ja bei jedem Prozesszyklus neu gemessen).

Sie können mit Hilfe des Befehls **ADC** Spannungen an den analogen Eingängen zwischen -10V und +10V messen. Dieses entspricht digitalisiert den Werten von 0 bis 65.535 (siehe auch Exkurs).

Im Unterschied zum ersten Programm können Sie an der Anzeige `Busy` erkennen, dass dieses Programm wesentlich mehr Prozessorzeit benötigt, obwohl bei jedem Prozesszyklus wieder nur eine Programmzeile ausgeführt wird. Die vergleichsweise lange Ausführungszeit des Befehls **ADC** kommt

daher, dass für jede einzelne Messung automatisch mehrere Vorgänge ablaufen.

Analog-Signal ausgeben

Erweitern Sie nun Ihr Programm, indem Sie zusätzlich ein Analog-Signal über den D/A-Wandler ausgeben und dieses wieder mit dem A/D-Wandler zurückmessen:

➢ Zu diesem Zweck verbinden Sie den analogen Ausgang 1 mit dem analogen Eingang 1(+) sowie die analoge Masse (AGND) mit dem Eingang 1(–).
Die Pinbelegung entnehmen Sie bitte der Hardware-Beschreibung Ihres ***ADwin-Pro***-Systems.

Das Ansprechen des Digital-/Analog-Wandlers erfolgt mit Hilfe des Befehls **DAC**, dem Sie die Moduladresse, die Nummer des Analog-Ausgangs und den zu wandelnden Wert übergeben müssen:

```
#INCLUDE ADwpad.inc
#INCLUDE ADwpda.inc

EVENT:
  DAC(1, 1, PAR_2)
  PAR_1 = ADC(1, 1)
```

Nach dem Übertragen des Programms auf das ***ADwin***-System können Sie den gemessenen Wert wieder in der globalen Variablen PAR_1 ablesen. Dieser schwankt jetzt um den Wert 0, weil der D/A-Wandler einen Wert von -10V ausgibt, entsprechend dem Wert 0 der globalen Variable PAR_2. Dass es tatsächlich -10V sind, die am analogen Ausgang 1 anliegen, können Sie mit einem externen Multimeter nachprüfen.

Den Wert, den der D/A-Wandler am analogen Ausgang ausgibt, können Sie sehr einfach verändern.

➢ Klicken Sie im Parameterfenster das Eingabefeld PAR_2 an und überschreiben den angezeigten Wert.
Nach der Bestätigung mit der Eingabetaste sehen Sie, dass sich auch der Wert in PAR_1 geändert hat. Auch die Anzeige eines angeschlossenen Multimeters sollte sich entsprechend ändern.

Funktionsgenerator

Dieses zweizeilige Programm lässt sich sehr einfach zu einem Funktionsgenerator erweitern, der z.B. eine Rampenfunktion am Ausgang liefert. Dazu kombinieren wir das erste Beispiel zum Hochzählen mit dem letzten AD-DA-Beispiel:

```
#INCLUDE ADwpad.inc
#INCLUDE ADwpda.inc

EVENT:
  DAC(1, 1, PAR_2)
  PAR_1 = ADC(1, 1)
  PAR_2 = PAR_2 + 1
  IF (PAR_2 > 65535) THEN     ' 16 Bit Wandler
    PAR_2 = 0
  ENDIF
```

Wenn Sie dieses Programm starten, sehen Sie, dass sich die Werte der globalen Variablen PAR_2 (und auch PAR_1) ändern.

➢ Schließen Sie ein Oszilloskop am analogen Ausgang 1 an (ein Multimeter ist zu träge) und beobachten Sie die Rampenfunktion der Spannung, entsprechend dem ansteigenden Wert von PAR_2.

Die Abfrage mit **IF** am Programmende dient dazu, den Zählerstand der globalen Variablen PAR_2 nach Erreichen des Maximalwerts von 65.535 wieder auf den Wert 0 zurückzusetzen. Dieses ist notwendig, weil die Funktion **DAC** Werte oberhalb des Maximalwerts auf 65.535 (bei 16 Bit-Wandlern) korrigiert, siehe auch den Exkurs auf Seite 5.

4 Speicherung von Messwerten

Große Datenmengen in Arrays

Wenn Sie mit aufeinanderfolgenden Messwerten, z.B. einer Messreihe arbeiten, so können Sie diese in ein Array einlesen. Dazu müssen Sie ein Array definieren, in das die Messwerte aufgenommen werden. Es gibt (wie bei den Variablen) globale und lokale Arrays; hier wird das globale Array DATA_1 verwendet.

```
DIM DATA_1[1000] AS LONG
DIM i AS LONG

INIT:
  i = 1

EVENT:
  DATA_1[i] = ADC (1)
  i = i + 1
  IF (i > 1000) THEN
    i = 1
  ENDIF
```

In diesem Beispiel wird als erstes das globale Array mit dem Namen DATA_1 mit 1.000 Array-Elementen angelegt. Diese Array-Elemente können Werte des Datentyps **LONG** aufnehmen, d.h. ganze Zahlen mit 32 Bit Datenlänge. Anschließend erfolgt die Definition der lokalen Variablen i, die als Index-Variable für den Zugriff auf das Datenelement verwendet werden soll.

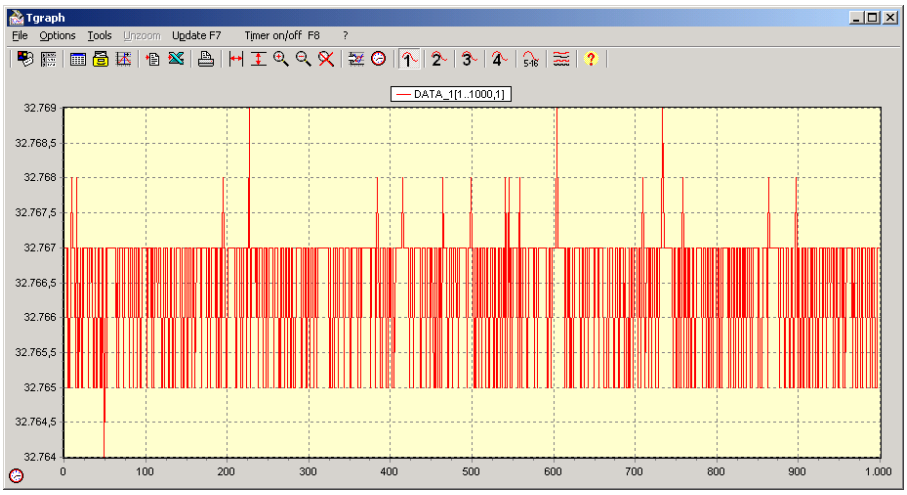
Im Abschnitt **INIT:** wird der Variablen i der Wert 1 zugeordnet. Dieser stellt den Index des ersten Datenelementes im globalen Array dar. Beachten Sie bitte, dass es kein Array-Element mit dem Index 0 gibt! Der Abschnitt **INIT:** wird bei der Abarbeitung des Programms nur genau einmal aufgerufen, direkt nach dem Start.

Die eigentlichen Programmzeilen finden Sie nach dem Schlüsselwort **EVENT:**. Darin wird als erstes der Wert des analogen Kanals 1 eingelesen, der gleich dem Element des globalen Arrays mit dem Index i zugewiesen wird. Danach wird der Wert der Index-Variablen um 1 erhöht. In der sich anschließenden **IF**-Abfrage wird die Index-Variable auf den Startwert zurückgesetzt, sobald sie den Wert 1.000 erreicht hat, um ein Überlaufen des Arrays zu verhindern. Damit werden immer nur die letzten 1.000 Messwerte im globalen Array gespeichert.

Array-Inhalte anzeigen

Wenn Sie das Programm kompilieren und starten, werden Sie im Parameterfenster keine Veränderungen feststellen. Das ist aber auch nicht weiter verwunderlich, da keine der globalen Variablen (PAR_1 … PAR_80 oder FPAR_1 … FPAR_80) im Programm eingesetzt worden sind. Dennoch können Sie sich den Inhalt des Arrays mit dem Zusatzprogramm TGraph der ***ADtools*** anzeigen lassen.
Die ***ADtools*** sind eine Sammlung kleiner nützlicher Windows-Programm, die direkt auf die globalen Variablen und Arrays (PAR, FPAR und DATA) des ***ADwin***-Systems zugreifen können, um sie anzuzeigen oder zu verändern.

➢ Das Programm mit dem Namen TGraph rufen Sie im Startmenü unter Programs ▸ ADwin ▸ ADtools ▸ TGraph auf.



In diesem Hilfsprogramm werden Ihnen die Werte des Arrays `DATA_1` grafisch dargestellt. Im Bild sehen Sie die eingelesenen Werte des AD-Wandlers bei offenem Eingang.

Wenn Sie `TGraph` das erste Mal starten, sollten Sie sich auf jeden Fall die Optionen näher anschauen, die über den Menüpunkt `Options` aufgerufen werden. Wichtig dabei ist, dass Sie die entsprechende Device No. und die Nummer des gewünschten Daten-Arrays angeben (bei `DATA_1` in Ihrem Programm eine 1, bei `DATA_10` eine 10).
Den Wertebereich der Anzeige können Sie am besten über die verschiedenen Schaltflächen anpassen, aber zum ersten Testen reicht die automatische Skalierung (Option `Autoscale` = `Yes`) aus.

5 Online-Auswertung von Messwerten

In diesem Abschnitt soll Ihnen aufgezeigt werden, wie Sie innerhalb eines Programms Messwerte einlesen und schnell weiter verarbeiten können. Sie benötigen dazu ein ***ADwin***-System mit einem analogen Eingang und ein externes, analoges Signal im Bereich -10V…+10V.

➢ Verbinden Sie das externe analoge Signal mit dem analogen Eingang 1(+) sowie die analoge Masse (AGND) mit dem Eingang 1(-).

Die Pinbelegung entnehmen Sie bitte der Hardware-Beschreibung Ihres ***ADwin***-Systems. Beachten Sie dort bitte auch die Hinweise zum Messbereich und zur Erdung der Geräts.

Das Programm liest 1.000 Messwerte am analogen Eingang 1 ein und bestimmt daraus den größten und den kleinsten Wert. Der Minimalwert wird in der Variablen `PAR_1`, der Maximalwert in `PAR_2` gespeichert.

Ab hier müssen Sie die Programmbeispiele nicht mehr komplett eintippen. Sie finden diese im Verzeichnis `C:\ADwin\ADbasic\samples_ADwin`. Wenn Sie mit einem ***ADwin-Pro***-System arbeiten, beachten Sie bitte die Hinweise in Kapitel 3.2 auf Seite 6.

Das erste Beispiel ist unter dem Namen `BAS_DMO1.BAS` gespeichert. Nach dem Übersetzen und Starten des Programms wird Ihnen der ermittelte Minimalwert bei der Variablen `PAR_1` und der Maximalwert bei der Variablen `PAR_2` angezeigt.
Statt der Minimal-/Maximalberechnung lassen sich auf einfache Weise auch andere Berechnungen ausführen.

BAS_DMO1.BAS

```
'#########################################################
'# The process BAS_DMO1 is searching for the maximum
'# and minimum value out of 1000 samples from ADC-#1
'# and writes the result to PAR_1 (min.) and PAR_2 (max).
'# After 1000 samples a flag (= PAR_10) is set.
'# PAR_1 = minimum value
'# PAR_2 = maximum value
'# Platform: ADwin-Gold or ADwin-light-16
'#########################################################

#DEFINE limit 65535          'max. 16 bit ADC-value

DIM i1, iw, max, min AS LONG

INIT:
  i1 = 1                     'reset sample counter
  max = 0                    'initial maximum value
  min = limit                'initial minimum value
  PAR_10 = 0                 'init End-Flag
  GLOBALDELAY = 40000        'cycle-time of 1ms (ADSP)

EVENT:
  iw = ADC(1)                'get sample
  IF (iw > max) THEN max = iw 'new maximum sample?
  IF (iw < min) THEN min = iw 'new minimum sample?
  i1 = i1 + 1                'increment index
  IF (i1 > 1000) THEN        '1000 samples done?
    i1 = 1                   'reset index
    PAR_1 = min              'write minimum value
    PAR_2 = max              'write maximum value
    min = limit              'reset minimum value
    max = 0                  'reset maximum value
    ACTIVATE_PC              'only for use with TestPoint
    PAR_10 = 1               'set End-Flag
  ENDIF
```

6 Digitaler P-Regler

Für das nachfolgende Beispiel, einen digitalen P-Regler, benötigen Sie ein ***ADwin***-System mit einem analogen Eingang und einem analogen Ausgang sowie ein extern zu regelndes System.

Mit der globalen Variablen `PAR_1` geben Sie den Sollwert vor und mit `PAR_2` die Verstärkung. Am A/D-Wandlereingang 1 wird der Istwert Ihres zu regelnden Systems gemessen. Das Programm berechnet den Stellwert und gibt diesen am D/A-Wandlerausgang 1 aus. Die Verstärkung muss an Ihr zu regelndes System angepasst werden.

Als zu regelndes System können Sie beispielsweise einen einfachen Tiefpass verwenden. Damit der Regler funktionieren kann, müssen das Signal „Istwert“ des zu regelnden Systems mit dem analogen Eingang 1(+) sowie die analoge Masse (AGND) mit dem Eingang 1(-) verbunden werden. Außerdem ist eine Verbindung des analogen Ausgangs mit dem Stellwert-Eingang des zu regelnden Systems herzustellen.

Die Pinbelegung entnehmen Sie bitte der Hardware-Beschreibung Ihres ***ADwin***-Systems. Beachten Sie dort bitte auch die Hinweise zum Messbereich und zur Erdung der Geräts.

Den Quelltext des Reglers finden Sie unter dem Dateinamen `BAS_DMO2.BAS` (im Verzeichnis `C:\ADwin\ADbasic\samples_ADwin`). Nachdem Sie das Programm kompiliert und gestartet haben, sehen Sie im Parameterfenster neben der `1:` den Sollwert und neben der `2:` den Verstärkungsfaktor. Durch Änderung des Wertes von `Par_1` können Sie die Sollposition variieren und durch Änderung des Wertes von `Par_2` können Sie den Verstärkungsfaktor des Reglers einstellen.

BAS_DMO2.BAS

```
'#####################################################
'# The process BAS_DMO2 is a digital P-controller.
'# PAR_1 = setpoint
'# PAR_2 = gain
'# Platform: ADwin-Gold or ADwin-light-16
'#####################################################

#DEFINE offset 32768          '0V for 16 bit ADC/DAC-systems

DIM cd, av AS LONG

INIT:
  PAR_1 = offset              'initial setpoint
  PAR_2 = 10                  'initial gain
  GLOBALDELAY = 40000         'cycle-time of 1ms (ADSP)

EVENT:
  cd = PAR_1 - ADC(1)         'compute control deviation (cd)
  av = cd * PAR_2 + offset    'compute actuating value (av)
  DAC(1, av)                  'output actuating value on DAC#1
```

7 Datenaustausch mit einem globalen Array

Wollen Sie große Datenmengen auf einfache Weise austauschen, so funktioniert dies am einfachsten mit globalen Arrays. Für dieses Beispiel benötigen Sie ein ***ADwin***-System mit einem analogen Eingang und ein externes, analoges Signal im Bereich -10V … +10V. Dieses verbinden Sie mit dem analogen Eingang 1(+). Zusätzlich müssen Sie eine Verbindung zwischen der analogen Masse (AGND) und dem Eingang 1(-) herstellen. Die Pinbelegung entnehmen Sie bitte der Hardware-Beschreibung Ihres ***ADwin***-Systems. Beachten Sie bitte auch die Hinweise zum Messbereich des Systems.

Das Programm misst den analogen Eingang 1 und informiert nach 1.000 Messungen den PC, der dann die Messwerte abholen kann. Die Daten werden mit Hilfe eines globalen Arrays übertragen. Das ***ADbasic***-Programm beendet sich nach den 1000 Messungen selbstständig.

Der zugehörige Quelltext ist unter dem Namen `BAS_DMO3.BAS` gespeichert. Das Programm enthält einen Befehl zur Kommunikation mit TestPoint. Wenn TestPoint nicht auf Ihrem Rechner installiert ist, dann löschen Sie diese Zeile aus dem Beispielprogramm.

Um das ***ADbasic***-Programm zu testen, können Sie mit Hilfe des Programms `TGraph` die Daten des globalen Arrays `DATA_1` auslesen und auf dem PC anzeigen.

BAS_DMO3.BAS

```
'########################################################
'# The process BAS_DMO3 samples ADC-#1 1000 times,
'# then stops and informs the PC to fetch the samples.
'# Data is transferred using a standard DATA array.
'# DATA_1 = series of samples
'# PAR_10 = End-flag
'# Platform: ADwin-Gold or ADwin-light-16
'########################################################

DIM DATA_1[1000] AS LONG
DIM index AS LONG

INIT:
  index = 0                     'reset array pointer
  PAR_10 = 0                    'init End-flag
  GLOBALDELAY = 40000           'cycle-time of 1ms (ADSP)

EVENT:
  index = index + 1             'increment array pointer
  IF (index > 1000) THEN        '1000 samples done?
    ACTIVATE_PC                 'only for use with TestPoint
    PAR_10 = 1                  'set End-Flag
    END                         'terminate process
  ENDIF
  DATA_1[index] = ADC(1)        'acquire sample and save in array
```

8 Kontinuierliche Datenübertragung mit FIFO

Wenn Sie kontinuierlich große Datenmengen übertragen wollen, können Sie dies mit der Datenstruktur FIFO erledigen. Dazu benötigen Sie ein ***ADwin***-System mit einem analogen Eingang und ein externes, analoges Signal im Bereich -10V…+10V.

Die Datenstruktur FIFO ist ein globales Array, in dem die Daten automatisch und auf einfache Weise verwaltet werden. Damit wird sicher gestellt, dass diejenigen Daten, die zuerst in das Array geschrieben werden, auch zuerst wieder ausgelesen werden: First In, First Out = FIFO.

Das Programm misst den analogen Eingang 1 kontinuierlich. Ein Anwendungsprogramm auf dem PC kann kontinuierlich die Daten aus dem FIFO-Array abholen. Dabei muss aber sichergestellt sein, dass die Daten schneller abgeholt werden, als sie vom ***ADwin***-System in den FIFO geschrieben werden. Ist dies nicht der Fall, so kann der FIFO „überlaufen“ und es gehen Daten verloren.

Für den Einsatz müssen Sie das externe Signal mit dem analogen Eingang 1(+) sowie die analoge Masse (AGND) mit dem Eingang 1(-) verbinden.

Die Pinbelegung entnehmen Sie bitte der Hardware-Beschreibung Ihres ***ADwin***-Systems. Beachten Sie dort bitte auch die Hinweise zum Messbereich und zur Erdung der Geräts.

Der Quelltext ist unter dem Namen `BAS_DMO4.BAS` zu finden. Das Programm enthält einen Befehl zur Kommunikation mit TestPoint. Wenn TestPoint nicht auf Ihrem Rechner installiert ist, dann löschen Sie diese Zeile aus dem Beispielprogramm.

Starten Sie vor dem Kompilieren das Hilfsprogramm `TFifo` aus den ***ADtools***; das Programm wartet nun darauf, dass Daten in das FIFO-Array geschrieben werden. Nach dem Kompilieren und Übertragen des ***ADbasic***-Programms liest `TFifo` die Daten aus und speichert sie für eine weitere Bearbeitung auf der Festplatte. Den Umgang mit `TFifo` finden Sie in der integrierten Online-Hilfe erklärt.

BAS_DMO4.BAS

```
'#######################################################
'# The process BAS_DMO4 continuously samples ADC-#1.
'# After 1000 samples the PC will be informed to
'# fetch the samples.
'# Data is transferred using a FIFO array.
'# DATA_1 = series of samples
'# Platform: ADwin-Gold or ADwin-light-16
'#######################################################

DIM DATA_1[4000] AS LONG AS FIFO
DIM index AS LONG

INIT:
  index = 0                     'reset sample counter
  PAR_10 = 0                    'init End-flag
  GLOBALDELAY = 40000           'cycle-time of 1ms (ADSP)

EVENT:
  index = index + 1             'increment sample counter
  IF (index > 1000) THEN        '1000 samples done?
    ACTIVATE_PC                 'only for use with TestPoint
    PAR_10 = 1                  'set End-flag
    index = 0                   'reset sample counter
  ENDIF
  DATA_1 = ADC(1)               'acquire sample and save in FIFO
```

9 Digitaler PID-Regler

Für dieses Programm, das einen digitalen PID-Regler darstellt, benötigen Sie ein ***ADwin***-System mit einem analogen Eingang und einem analogen Ausgang sowie ein extern zu regelndes System.

Die Einstellungen des PID-Reglers werden mit mehreren Variablen vorgegeben. Am A/D-Wandlereingang 1 wird der Istwert Ihres zu regelnden Systems gemessen. Das Programm berechnet den Stellwert und gibt diesen am D/A-Wandlerausgang 1 aus. Die Verstärkung muss an Ihr zu regelndes System angepasst werden.

Zum Testen des Reglers müssen Sie das externe Signal mit dem analogen Eingang 1(+) sowie die analoge Masse (AGND) mit dem Eingang 1(-) verbinden.

Die Pinbelegung entnehmen Sie bitte der Hardware-Beschreibung Ihres ***ADwin***-Systems. Beachten Sie dort bitte auch die Hinweise zum Messbereich und zur Erdung der Geräts.

Der Quelltext ist gespeichert unter dem Namen `BAS_DMO6.BAS`. Das Programm enthält einen Befehl zur Kommunikation mit TestPoint. Wenn TestPoint nicht auf Ihrem Rechner installiert ist, dann löschen Sie diese Zeile aus dem Beispielprogramm.

Nachdem Sie das Programm auf Ihr ***ADwin***-System übertragen haben, können Sie die Regler-Einstellungen mit Hilfe der globalen Variablen im Parameterfenster ändern.

Die berechnete Regelabweichung wird zusätzlich kontinuierlich in das Array `DATA_1` geschrieben. Durch Visualisierung dieser Daten mit dem Programm `Tgraph` können Sie die Regelparameter des Reglers optimieren.

BAS_DMO6.BAS

```
'#######################################################
'# The process BAS_DMO6 is a digital PID-controller.
'# The controller operates with float variables whose
'# coefficients have to be computed on the PC and then
'# transferred to the ADwin-system.
'# PAR_1  = setpoint in digits
'# FPAR_2 = P (gain, proportional factor)
'# FPAR_3 = I (integration time)
'# FPAR_4 = D (derivative time)
'# PAR_5  = buffer index for the control deviation
'# PAR_6  = cycle-time in 25ns steps
'# PAR_9  = flag for new setpoint definition by PC
'# Attention: Prior to start the program the controller
'#            parameters have to be set to the correct value!
'#            Make sure that FPAR_3 is not zero!!!
'# Platform: ADwin-Gold or ADwin-light-16
'#######################################################

#DEFINE offset 32768          '0V output

DIM DATA_1[4000] AS LONG
DIM av, cd, cdo, sum AS LONG
DIM diff AS FLOAT

INIT:
  sum = 0                     'initial value of integral part
  cd = ADC(1)                 'initial value of control
                              'deviation (cd) & MUX to Ch-#1
  PAR_5 = 1                   'reset array index
  IF (FPAR_3 < 1E3) THEN FPAR_3 = 1E3 'check min. of
                              'integration time
  IF (PAR_6 < 4E4) THEN PAR_6 = 4E4 'allow cycle-times >=1ms
  GLOBALDELAY = PAR_6         'set cycle-time

EVENT:
                              'compute actuating value
  av = FPAR_2*(cd + sum/FPAR_3 + diff*FPAR_4)
  START_CONV(1)               'start conversion ADC-#1
  DAC(1, av + offset)         'output actuating value at DAC#1
  cdo = cd                    'keep control deviation in mind
  WAIT_EOC(1)                 'wait for ADC's end of conversion
  cd = PAR_1 - READADC(1)     'compute control deviation
  FPAR_9 = FPAR_9*0.99+cd*0.01'mean value of control deviation
  sum = sum + cd              'calculate integral
  IF (sum > 2E6) THEN sum = 2E6 'positive limit of integral
  IF (sum < -2E6) THEN sum = -2E6 'negative limit of integral
  diff = (cd - cdo)           'calculate deviation difference
  DATA_1[PAR_5] = cd          'write control deviation
                              'in a buffer
  INC PAR_5                   'increment buffer index
  IF (PAR_5 >= 4000) THEN     '4000 samples done?
    ACTIVATE_PC               'only for use with TestPoint
    PAR_10 = 1                'set End-Flag
    PAR_5 = 1                 'reset buffer index
  ENDIF

FINISH:
  DAC(1,offset)               'analog output #1 to 0V
```

Abhilfe bei Fehlern

Im folgenden finden Sie Hinweise, wie Sie mit den folgenden Fehlern umgehen:

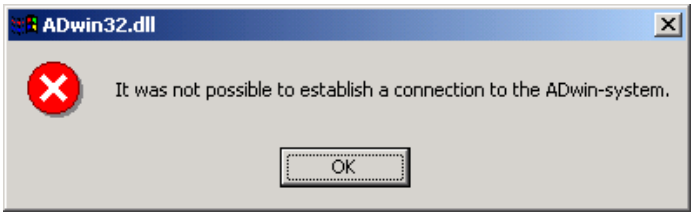
– Fehler beim Booten

– Kommunikation ist unterbrochen

A-1 Fehler beim Booten

Fehlermeldung

Wenn das ***ADwin***-Betriebssystem nicht auf Ihr System geladen wurde, erhalten Sie maximal zwei Fehlermeldungen. Beide Meldungen müssen Sie mit `OK` quittieren, um in ***ADbasic*** weiter arbeiten zu können.



Diese Fehlermeldung besagt, dass das ***ADwin***-System nicht auf Verbindungsanfragen reagiert.

Eine Fehlermeldung dieser Art beschreibt genauer, welcher Art der Fehler war. Die Meldung ist abhängig von der gerade verwendeten Datenverbindung zwischen PC und ***ADwin***-System; hier sehen Sie die Meldung eines Ethernet-Netzwerks.

Fehler beheben

Beheben Sie den Fehler, indem Sie folgende Punkte prüfen und anschließend erneut die Kommunikation zum ***ADwin***-System aufnehmen:

Mechanische Kontrolle

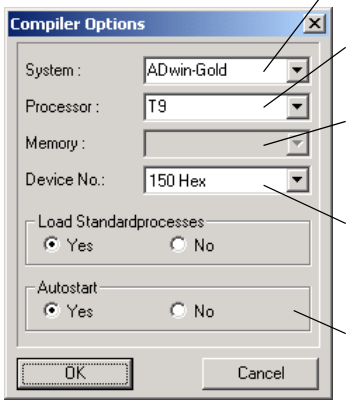
Überprüfung	Abhilfe
Ist die Stromversorgung angeschlossen und ist das System eingeschaltet?	Stellen Sie die notwendigen Verbindungen her und schalten Sie das System ein.
Ist die Datenleitung zwischen System und PC richtig angeschlossen?	Schließen Sie das System entweder mittels eines USB-Kabels an den PC an oder verbinden Sie es mit dem Ethernet-Netzwerk.

Gerätenummer / Device No.

Überprüfung	Abhilfe
Ist die Gerätenummer (Device No.) in *ADconfig* korrekt eingerichtet? Rufen Sie dazu das Programm aus dem Windows-Startmenü heraus auf.	Stellen Sie in *ADconfig* die für ihr System passende Device No. (Standard `150h`) ein.

Konfiguration in ADbasic

Überprüfung	Abhilfe
Wurde das System in ***ADbasic*** richtig konfiguriert? Rufen Sie in ***ADbasic*** den Menüpunkt `Options` ▸ `Compiler` auf. Prüfen Sie im angezeigten Dialog, ob die Einstellungen mit Ihrem System korrespondieren (Erklärung siehe unten).	Stellen Sie die passenden Daten im Dialog ein. Achten Sie dabei darauf, dass die Optionen `Autostart` auf `Yes` eingestellt sein sollte.
Ist in ***ADbasic*** der korrekte Pfad für die Betriebssystem-Datei angegeben? Wählen Sie zur Überprüfung den Menüpunkt `Options` ▸ `Setting` und dort das Blatt `Directory` aus.	Geben Sie im Feld `BTL-Directory` den entsprechenden Pfad-Namen ein (mit einem „`\`“ am Ende). Bei einer Standardinstallation muss hier der Pfad `C:\ADwin\` zu finden sein.



Die Bauart Ihres ***ADwin***-Systems.

Der Prozessortyp Ihres ***ADwin***-Systems:
T9 oder T10.

Die Speichergröße ist für die Prozessoren T9 und T10 ohne Bedeutung.

Die Gerätenummer Ihres ***ADwin***-Systems, die Sie mit *ADconfig* eingestellt haben.

Diese Option sollte zunächst auf `Yes` eingestellt sein.

Sollten Sie den Fehler auf diese Weise nicht beheben können, ist möglicherweise ein Fehler bei der Installation aufgetreten. Versuchen Sie bitte, mit Hilfe des Handbuchs „***ADwin*** Treiber-Installation“ die Installation nachzuvollziehen.

Wenn Sie den Fehler dennoch einmal nicht beheben können, rufen Sie bitte unseren Support an: Tel. +49 (6251) 96320.

A-2 Kommunikation ist unterbrochen

Fehler erkennen

Es kann vorkommen, dass die Kommunikation zwischen dem ***ADwin***-System und dem PC abbricht. Dieses können Sie daran erkennen, dass

- verschiedene Arrays dunkelgrau werden: im Parameterfenster, im Prozessfenster sowie das Eingabefeld `Process n delay`,
- die Werte im Parameter- und Prozessfenster unverändert stehen bleiben,
- in der Statusleiste keine Werte angezeigt werden
- die Schaltfläche `Enable cyclic update` in der Werkzeugleiste deaktiviert wird  .

Eine unterbrochene Kommunikation bedeutet meistens, dass der Prozessor auf dem ***ADwin***-System nicht genügend Zeit hat, um mit dem PC zu kommunizieren. Das heißt dann aber nicht, dass Ihr Programm auf dem System nicht mehr läuft, sondern dass der PC die Kommunikation abgebrochen hat.

Kommunikation wieder herstellen

Abhilfe können Sie in diesem Fall nur schaffen, indem Sie das ***ADwin***-System neu booten. Dabei ist aber auch Ihr Programm im Systemspeicher gelöscht worden; Sie müssen es also neu kompilieren und starten (Schaltfläche).

Durch das Booten und Kompilieren haben Sie den aufgetretenen Fehler noch nicht endgültig beseitigt. und er tritt erneut auf. Beseitigen Sie also bitte erst die Fehlerursache, bevor Sie weiter arbeiten.

Fehler beheben

Ein Kommunikationsabbruch kann viele Gründe haben; meistens besteht die Ursache darin, dass der Prozessor durch einen hochprioren Prozess so sehr ausgelastet ist, dass er Kommunikations-Anfragen des PC nicht rechtzeitig beantworten kann. Im folgenden sind einige Hinweise zur Fehlersuche angegeben.

Überprüfung	Abhilfe
Gab es ein Problem in der Datenleitung? Wenn die Datenverbindung inzwischen wieder funktioniert, kann die Kommunikation wieder aufgenommen werden.	Klicken Sie auf die Schaltfläche `Enable cyclic update` in der Werkzeugleiste. Falls dies nicht hilft, booten Sie Ihr System.
Startet der hochpriore Prozess mit einem zu kleinen Processdelay? Rufen Sie in ***ADbasic*** den Menüpunkt `Options` ▸ `Process` auf und prüfen Sie die Einstellung `Initial Processdelay`.	Stellen Sie einen größeren Wert für das `Initial Processdelay` ein und starten das Programm erneut.
Haben Sie im Programm Array-Elemente angesprochen, die außerhalb des deklarierten Bereichs liegen?	Korrigieren Sie Ihr Programm und starten es erneut.
Ist die Ausführungszeit des Programmabschnitts **`EVENT`**: immer (!) kleiner als das Processdelay?	Vergrößern Sie das Processdelay oder verlagern Sie nicht zeitkritische Programmteile in einen niederprioren Prozess.
Sind mehrere hochpriore Prozesse aktiv, die sich gegenseitig behindern?	Passen Sie die Ausführungszeiten und das Processdelay der Prozesse aneinander an.