ADwin

Treiber für LabVIEW ab Version 5

Oktober 2003

Inhaltsverzeichnis

1 Wie Sie mit LabVIEW und *ADwin* arbeiten

ADwin erweitert die Möglichkeiten von LabVIEW

Das ***ADwin***-System besteht aus einem eigenständigen Messrechner, der Mess- und Regelaufgaben extrem schnell und sicher erledigt, und einer Schnittstelle unter Windows, über die Sie mit LabVIEW das ***ADwin***-System steuern. Sie verlagern also alle zeitkritischen Prozesse in das ***ADwin***-System, haben aber die Steuerung der Prozesse und Verarbeitung der Daten weiterhin mit LabVIEW in der Hand.

Wie Sie das *ADwin*-System programmieren

ADwin-Systeme sind schnell, zuverlässig und flexibel. Um diese Vorteile optimal zu nutzen, verwenden Sie die einfache Programmiersprache ***ADbasic***. Bevor Sie die hier beschriebenen LabVIEW-Befehle anwenden können, empfehlen wir Ihnen eine Einarbeitung in ***ADbasic***. Hierzu verwenden Sie bitte das ***ADbasic***-Handbuch und die Programmieranleitung. Die Beschreibungen werden Ihnen auch das Verständnis des ***ADwin***-Systems erleichtern.

ADwin-Systeme mit LabVIEW steuern

Jetzt ist der Moment gekommen, mit diesem Handbuch den Schritt in die Praxis zu tun.

Kapitel 2 schildert, wie LabVIEW und ***ADwin*** kommunizieren und vertieft Ihr Verständnis für das ***ADwin***-Konzept.

In Kapitel 3 werden die Installation und Benutzung der LabVIEW-Treiber-Software sowie die Einbindung der neuen Befehle beschrieben.

Die Funktionen des LabVIEW-Treibers sind in Kapitel 4 erklärt, das auch als Nachschlagewerk angelegt ist.

Viel Erfolg beim Anwenden von ***ADwin***, dem mikrosekundengenauen Echtzeitsystem.

Dieses Handbuch beschreibt den Umgang mit dem ADwin-Treiber für **LabVIEW ab Version 5**.

Der Treiber ist nicht für den Einsatz unter früheren LabVIEW-Versionen geeignet!

2 *ADwin*-Treiberkonzept

2.1 *ADwin* Betriebssystem

Das Betriebssystem befindet sich in der Datei mit der Bezeichnung <ADwin9.btl>[1] (ADSP / T9) oder <ADwin10.btl> (ADSP / T10).

Nach jedem Einschalten der Spannungsversorgung müssen Sie zunächst das Betriebssystem auf das ***ADwin***-System laden (booten). Nach erfolgreicher Übertragung ist das ***ADwin***-System in der Lage, Befehle vom PC entgegenzunehmen und Daten mit ihm auszutauschen.

Die Aufgaben des Betriebssystems sind:

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

Priorität	Merkmal	Verwendungszweck
niedrig	kann von hoch priorisierten Prozessen unterbrochen werden	für nicht zeitkritische Berechnungen und langsame Messungen
hoch	kann von keinem anderen Prozess unterbrochen werden	für zeitgenaue Messungen, Steuerungen und Regelungen

- Bereitstellung von 80 vordefinierten Integer-Variablen (PAR_1 bis PAR_80) und 80 vordefinierten Float-Variablen (FPAR_1 bis FPAR_80). Außerdem werden 200 Datensätze mit frei zu definierender Länge (DATA_1 bis DATA_200) bereitgestellt. Die Werte dieser Variablen bzw. Datensätze können Sie vom PC aus lesen und ändern.

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

Ein wesentlicher Bestandteil des ***ADwin***-Betriebssystems ist der Kommunikationsprozess. Dieser Prozess läuft mit mittlerer Priorität auf dem ***ADwin***-System, d.h. er kann niederpriore Prozesse für kurze Zeit unterbrechen. Der Kommunikationsprozess interpretiert bzw. bearbeitet alle Befehle, die der PC an das ***ADwin***-System richtet. Die wichtigsten Befehle lassen sich in 2 Gruppen unterteilen: Steuerbefehle und Befehle für den Datenaustausch. In den folgenden Tabellen sind aus jeder Gruppe einige Beispiele aufgelistet.

Steuerbefehle	
Boot	überträgt einen Prozess auf das ***ADwin***-System
Start_process	startet einen Prozess

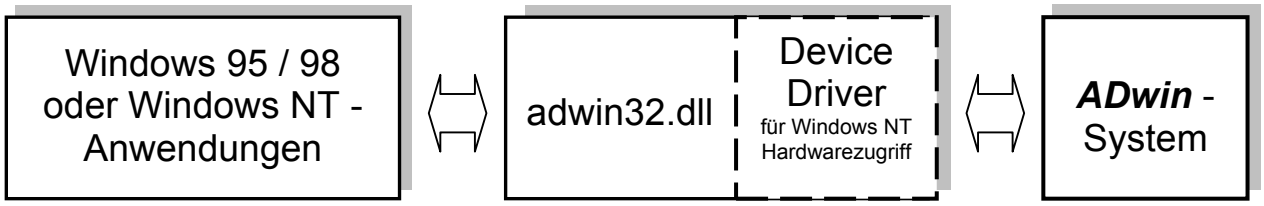
Befehle für den Datenaustausch	
get_par	liefert den aktuellen Wert eines Parameters
set_par	ändert den Wert eines Parameters
get_data	liefert die Werte aus einem Datensatz

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

[1] Bei Verwendung eines ADwin-2-, -4-, -5- oder -8-Systems verwenden Sie entsprechend die Datei <ADwin2.btl>, <ADwin4.btl>, <ADwin5.btl> oder <ADwin8.btl>.

2.2 Windows-*ADwin*-Schnittstelle

Die Schnittstelle zwischen Windows und dem ***ADwin***-System bildet die <ADwin32.dll> (**D**ynamic **L**ink **L**ibrary). Alle Windows-Programme, die DLL-Funktionen aufrufen können, können auch mit dem ***ADwin***-System kommunizieren (siehe nachfolgende Abbildung). Dazu gehört auch LabVIEW.



***ADwin32*.dll** –Schnittstelle zu Windows-Anwendungen

Die Schnittstelle bewältigt folgende Aufgaben:

- Befehlsweiterleitung

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

- Datentransfer

  Der Datentransfer zwischen ***ADwin***-System und PC-Anwendungen erfolgt über die <ADwin32.dll>.
  Unter Windows NT greift die <ADwin32.dll> über den zusätzlich installierten Device-Treiber auf das ***ADwin***-System zu, alle anderen Windows-Versionen erlauben den direkten Zugriff.
  Da die <ADwin32.dll> weder Interrupts noch DMA-Kanäle benötigt, ist ein zuverlässiger Betrieb des ***ADwin***-Systems in jedem PC problemlos möglich und aufwendige Konfigurationsprozeduren entfallen.

- Multitasking-Verwaltung

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

3 Software Installation

3.1 Installation der *ADwin*-Treiber

Legen Sie die mitgelieferte CD-ROM in das Laufwerk Ihres Rechners ein. Das Setup-Programm wird normalerweise automatisch gestartet.

Sollte dies nicht der Fall sein, dann starten Sie das dort befindliche Programm <`setup.exe`>.

Beachten Sie die Hinweise in <`readme.txt`> und installieren Sie die Software entsprechend.

Nach erfolgreicher Installation werden sich auf Ihrer Festplatte unter anderem die nachstehenden Dateien befinden, die Sie später benötigen werden. Bei der Standardinstallation finden Sie die Dateien im Verzeichnis `C:\ADwin\`.

`..\Developer\LabVIEW\ADwin_v2.lib\*.VI`	Treiber (VI-Sammlung) für LabVIEW 5/6/7
`..\Developer\LabVIEW\ADwin_v2.lib\Bas_dmo*.VI`	Beispiele zum LabVIEW 5/6/7 - Treiber
`..\Developer\LabVIEW\ADwin.lib\*.VI`	Treiber (VI-Sammlung) für LabVIEW 4
`..\Developer\LabVIEW\ADwin.lib\Bas_dmo*.VI`	Beispiele zum LabVIEW 4 - Treiber
`..\Tools\Test\ADTEST.EXE`	Programm zum Testen der ***ADwin***-PC-Einsteckkarten, ***ADwin***-Gold und ***ADwin***-L16
`..\Tools\Test\ADPRO.EXE`	Programm zum Testen des ***ADwin-Pro***-Systems

3.2 Laden des *ADwin*-Betriebssystems

Der PC kann erst mit dem ***ADwin***-System kommunizieren, nachdem das Betriebssystem geladen wurde. Zum Laden des Betriebssystems haben Sie folgende 3 Möglichkeiten:

a) Unter LabVIEW

Mit dem ***ADwin-***VI 'Boot'. Nähere Informationen hierzu in Kapitel 4.

b) Über ADbasic

- Starten Sie ***ADbasic***, indem Sie im Windows-Menü

  `Start` ► `Programme` ► `ADwin` ► `ADbasic 3.2` auswählen.

- Überprüfen Sie, ob die Einstellungen im Menü `Options` ► `Compiler` mit Ihrem ***ADwin***-System übereinstimmen.

- Klicken Sie in der ***ADbasic***-Toolbar auf das folgende Symbol: 
  (alternativ: `Project` ► `Boot ADwin`)

Das erfolgreiche Laden des Betriebssystems wird in der Statuszeile des Editors durch die Meldung „ADwin is booted“ bestätigt. Bei Problemen finden Sie weitere Informationen im Handbuch und in der Programmieranleitung.

c) Mit dem Windows Programm ADtest

Das Programm ***ADtest*** dient als Funktionskontrolle für ADwin-PC-Einsteckkarten und bietet ebenfalls eine Möglichkeit zum Laden des Betriebssystems. Starten Sie dazu das Programm <`ADtest.exe`>. Falls das Betriebssystem seit dem Einschalten des PCs noch nicht auf das ***ADwin***-System geladen wurde, erscheint eine Dialogbox.

Stimmen Sie alle Einstellungen auf das von Ihnen eingesetzte ***ADwin***-System ab, und betätigen Sie anschließend die OK-Taste. Wenn keine Fehlermeldung ausgegeben wird, haben Sie das Betriebssystem erfolgreich geladen. Falls das Betriebssystem nicht geladen werden konnte, lässt sich der Vorgang durch Betätigen der Settings-Taste mit geänderten Einstellungen wiederholen.

Das Programm ***ADtest*** zeigt Ihnen die momentan an den analogen Eingängen 1-12 anliegenden Spannungen und die Pegel an den digitalen Eingängen an. Außerdem können die analogen und digitalen Ausgänge gesetzt werden. Auf den analogen Ausgängen kann zum Testen ein Sinus- oder Dreiecksignal mit 10 Hz ausgegeben werden.

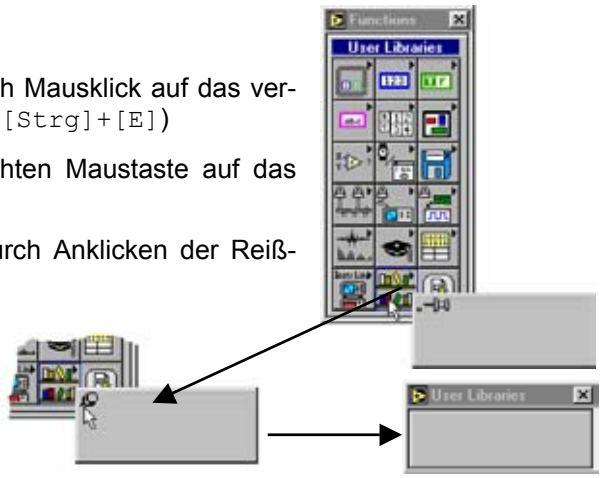
Durch das Laden des Betriebssystems werden alle Prozesse auf dem ***ADwin***-System gelöscht und alle globalen Variablen auf den Wert 0 gesetzt. Der Wert für das Global delay ist nach dem Laden des Betriebssystems beim ADSP auf 25 µs und bei allen anderen Prozessoren auf 1000 µs voreingestellt.

3.3 Einbinden der *ADwin* VIs

In LabVIEW-Programmen können Sie mit Hilfe vorgefertigter VIs Befehle und Daten an das ***ADwin***-System schicken und Daten vom System abrufen. Die VIs nutzen hierzu Funktionen, die in der <`ADwin32.dll`> implementiert sind.

Zum Einbinden der VIs in LabVIEW befolgen Sie bitte die folgenden Anweisungen:

- Kopieren Sie den Ordner <`..\ADwin_v2.lib\...`> mit dem gesamten Inhalt in Ihr LabVIEW-Verzeichnis
  (z.B. <`C:\Programme\National Instruments\LabVIEW 6`>)
- Starten Sie LabVIEW
- Klicken Sie auf '`New VI`'
- aktivieren Sie das weiße Diagramm-Fenster (durch Mausklick auf das verdeckte weiße Fenster oder die Tastenkombination `[Strg]+[E]`)
- Drücken Sie im Fenster: 'Functions' mit der rechten Maustaste auf das Symbol '`User Libraries`'
- Verankern Sie das neu erscheinende Fenster durch Anklicken der Reißzwecke auf dem Desktop (s. rechts)



- Unter LabVIEW **7**

  Wählen Sie im Menü des Blockdiagramm Fensters die Option '`Tools` ▶ `Advanced` ▶ `Edit Palette Views`'

- Unter LabVIEW **5**

  Wählen Sie im Menü des Diagram Fensters die Option '`Edit` ▶ `Edit Control & Function Palettes`'

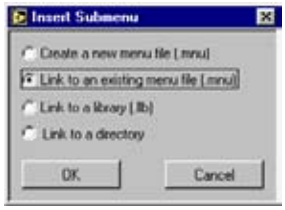
- Unter LabVIEW **6**

  Klicken Sie im Fenster '`User Controls`' mit der linken Maustaste auf das Symbol '`Options`'.

  Im daraufhin erscheinenden Fenster betätigen Sie die Taste '`Edit Palettes ..`'

- Drücken Sie mit der rechten Maustaste auf die graue Fläche im Fenster '`User Libraries`' und wählen im daraufhin erscheinenden Menü den Punkt: '`Insert` ▶ `Submenu`'

- Im Fenster "Insert Submenu" wählen Sie die Einstellung "Link to an existing menu file" und betätigen anschließend die Taste: OK

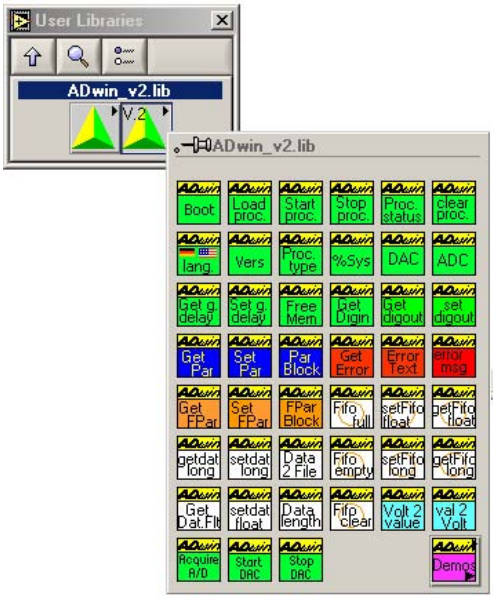


- Wählen Sie in der Dateiauswahlbox im Verzeichnis ADwin_v2.lib die Datei ADwin.mnu.

- Speichern Sie die Änderung durch die Betätigung der Taste "Save Changes" im Fenster "Edit Control and Functions Palettes".

- Aktivieren Sie nochmals das weiße Diagramm-Fenster

- Alle VIs erscheinen jetzt in der Gruppe "User Libraries" (siehe rechts). Sie können sie direkt von dort in Ihre LabVIEW Diagramme ziehen. Die Beschreibung der einzelnen VIs folgt im nächsten Kapitel.



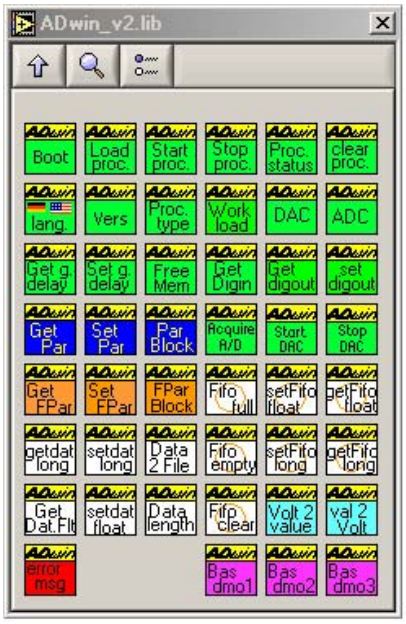
3.4 Portierung der VIs zwischen MS Windows und Linux

Um die VIs von einer LabVIEW-Windows-Version auf Linux bzw. umgekehrt zu portieren, müssen Sie in den Einstellungen der VIs den Pfadnamen auf die ***ADwin***-Bibliothek entsprechend anpassen.

Betriebssystem	Windows	Linux
Bibliothek	C:\Windows\ADwin32.dll oder C:\WinNT\ADwin32.dll	/usr/lib/libADwin.so

4 LabVIEW VIs für *ADwin*-Systeme

4.1 Grundlagen



Die VI-Symbole (Icons) haben unterschiedliche Farben, die sich teilweise am LabVIEW-Standard orientieren. Die Farben im Einzelnen:

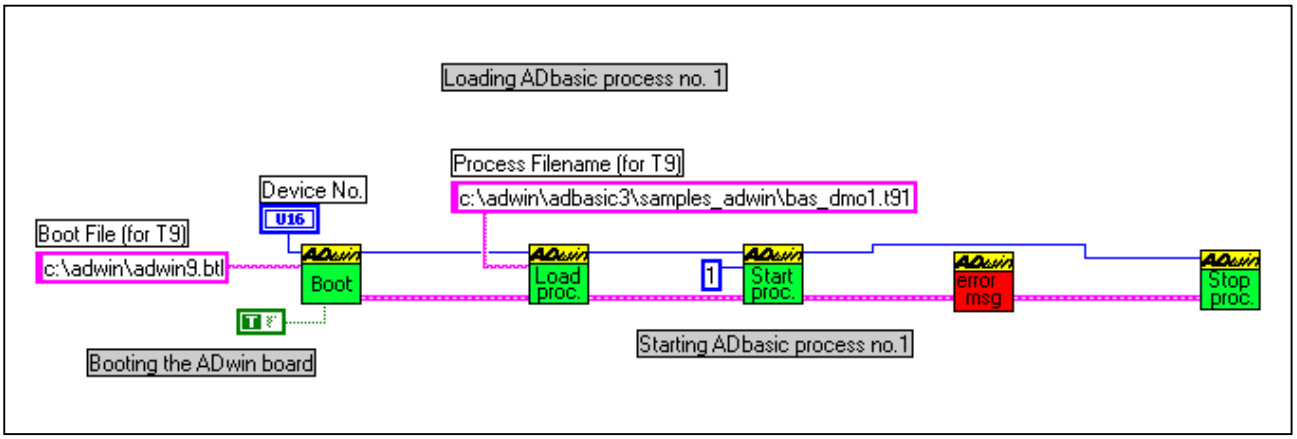
Grün	:	System-/ und "Low level"-Funktionen
Blau	:	Integer-Parameterfunktionen
Orange	:	Floating point-Parameterfunktionen
Rot	:	Fehlerfunktionen
Weiß	:	DATA-Funktionen
Weiß mit Ring	:	FIFO-Funktionen (Ringspeicher)
Hellblau	:	Tools
Pink	:	Demos

Das unten dargestellte Diagramm zeigt ein Programmier-Beispiel für LabVIEW, das Ihnen die nachfolgenden Hinweise verdeutlicht. Die Funktion des Programms ist hier nicht von Bedeutung (das Beispielprogramm bootet ein ***ADwin***-System (hier: ADSP/T9), lädt und startet einen Prozess, überprüft Fehler und stoppt anschließend den Prozess wieder).

Sie müssen alle *ADwin*-VIs mit "error in" und "error out" durch eine Fehlerverdrahtung verbinden. Betrachten Sie die gestrichelte Linie (pink), die alle VIs verbindet: Dies ist die Fehlerverdrahtung. Anderenfalls kann LabVIEW bei einem evtl. auftretenden Fehler in eine Endlosschleife geraten und Sie können nicht mehr darauf zugreifen. Mit Fehlerverdrahtung bleibt LabVIEW bedienbar und Sie erhalten zusätzlich eine Fehlermeldung (mit dem VI *error_msg*). Mit der Verdrahtung legen Sie außerdem eine eindeutige Reihenfolge bei der VI-Bearbeitung fest.

Die "Device No." kennzeichnet ***ADwin***-Systeme eindeutig und wird für die korrekte Kommunikation mit dem ***ADwin***-System benötigt. Deswegen müssen Sie an fast alle VIs die “Device No.” als Eingangswert übergeben. Zur Vereinfachung besitzen die entsprechenden VIs einen Ausgang "Device No. out“, den Sie mit dem nächsten VI-Eingang "Device No. in“ verbinden können. Dadurch entsteht eine durchgehende Verkabelung (im Beispiel die obere blaue Linie). Anderenfalls müssten Sie jedem VI die “Device No.” separat übergeben.

Zu allen VIs gibt es eine Kontext-Hilfe, die Sie durch `[Strg]+[H]` starten. Wenn diese Hilfe geöffnet ist und Sie die Maus auf ein VI-Symbol bewegen, wird die entsprechende Hilfe angezeigt (englisch), bestehend aus der grafischen Darstellung des VI und einem Hilfstext (der Hilfstext ist auch in diesem Handbuch enthalten, die Grafiken nicht).



4.2 Treiber für LabVIEW kleiner Version 5.0

Wenn Sie einen Treiber für LabVIEW kleiner Version 5.0 besitzen, können Sie die bisherigen VIs weiterhin verwenden, wir empfehlen es jedoch nicht. Der aktuelle Treiber im Vergleich dazu hat eine überarbeitete Namensgebung für die VI-Funktionen, ist besser auf LabVIEW 5/6/7 abgestimmt und ermöglicht eine einfache Fehlerbehandlung.

Der Vollständigkeit halber finden Sie auf der CD sowohl die bisherige als auch die neue VI-Sammlung (siehe Installation).

In der folgenden Beschreibung ist nur die aktuelle Namensgebung der VI-Funktionen berücksichtigt.

4.3 Fehlerbehandlung

Diese Treiberversion stellt Ihnen mit der "Fehlerverdrahtung" eine neue Möglichkeit der Fehlerbehandlung zur Verfügung.

Alle neuen ***ADwin***-VIs haben die Eigenschaft, ein evtl. anstehendes Fehlersignal am Eingang "error in" aufzunehmen und es an den Ausgang "error out" weiterzureichen, ohne das ***ADwin***-System anzusprechen. Dadurch wird vermieden, dass LabVIEW ggf. vergeblich den Kontakt zum ***ADwin***-System sucht.

Wenden Sie die Fehlerverdrahtung konsequent an: Verbinden Sie alle VIs miteinander über die Error-in und Error-out Anschlüsse. Benutzen Sie die Fehlermeldungen zur Fehlerbehandlung, indem Sie diese mit *error_msg* abfragen (beispielsweise einmal pro Schleifendurchlauf).

4.4 Die “DeviceNo.”

Alle ***ADwin***-Systeme werden über die sogenannte “Device No.“ angesprochen. Mit dem Programm ***ADconfig*** stellen Sie die “Device No.” für Ihr ***ADwin***-System ein.

In LabVIEW muss jedes VI die “Device No.” ‘kennen‘. Hierzu genügt es, an das erste VI die “Device No.” anzuschließen und die Verdrahtung jeweils ‘durchzuschleifen‘ (analog zur Fehlerverdrahtung).

4.5 Beispielprogramme

Die Beispielprogramme verdeutlichen das Zusammenspiel von LabVIEW mit dem ***ADwin***-System. Sie bestehen jeweils aus einem LabVIEW- und einem ***ADbasic***-Programm. Das LabVIEW-Programm ermöglicht Ihnen die Steuerung des ***ADwin***-Prozesses und die Darstellung der von dort übertragenen Daten, während das ***ADbasic***-Programm den Ablauf des Prozesses auf dem ***ADwin***-System definiert.

Der Quelltext der ***ADbasic***-Programme ist unter <`C:\ADwin\ADbasic3\samples_ADwin\*.BAS`> abgelegt, die LabVIEW-Programme im gleichen Pfad unter <`..\bas_dmo.VI`>.

Zum besseren Verständnis der Beispiele betrachten Sie bitte die Quelltexte sowie die zugehörigen LabVIEW-Diagramme.

4.6 Funktionen (VIs) zur Steuerung von *ADbasic* Prozessen

4.6.1 Das Betriebssystem laden

Boot.VI

Parameter:

ADwin-Typ	Filename
ADwin-2	ADwin2.btl
ADwin-4	ADwin4.btl
ADwin-5	ADwin5.btl
ADwin-8	ADwin8.btl
T9 (*ADwin*-ADSP)	ADwin9.btl
T10 (*ADwin*-ADSP)	ADwin10.btl

Speicherausbau	Memsize
64 KB	*0*
1 MB	*1*
2 MB	*2*
4 MB	*3*
8 MB	*4*
16 MB	*5*
32 MB	*6*

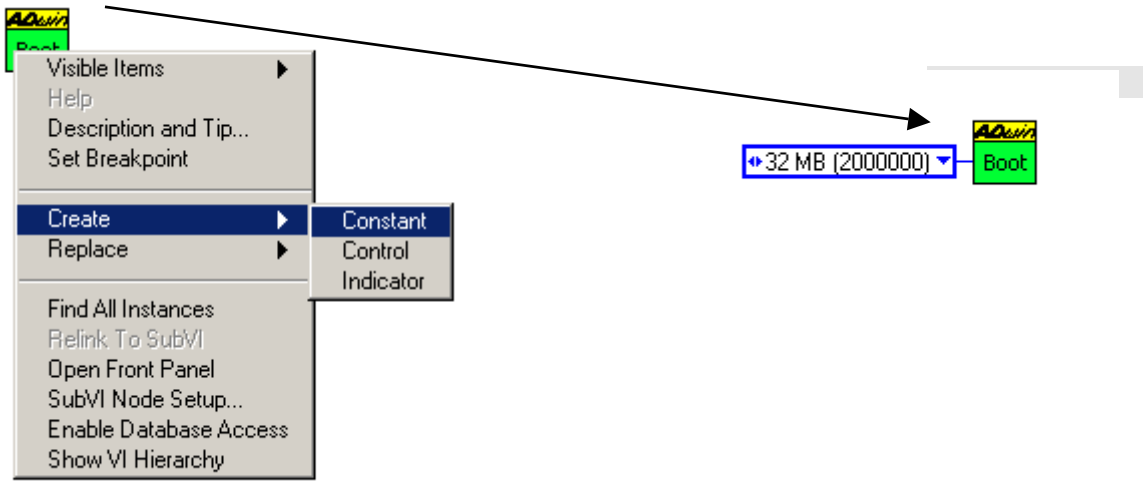
Return value memory:

	ADwin-ADSP (T9, T10)	*ADwin-2*, -4, -5 u. -8
<1000	Fehler	Fehler
8000	OK	
Erkannter Speicherausbau		OK

Showerror *true*: bei Fehler Meldungsfenster ausgeben

Hinweise:

- Beim T9 und T10 (ADSP) wird der Parameter „*MemSize*“ ignoriert
- Die Dateien <ADwin2.btl> … <ADwin10.btl> sind bei Standardinstallationen auf <C:\ADwin\..> vorhanden.
- Die Speichergröße (*Memsize*) sollten Sie einstellen, indem Sie mit der rechten Maustaste auf den Anschluss klicken und "Create ► Constant" auswählen. Daraufhin erscheint ein Auswahlfenster, bei dem Sie von Hand den Speicherausbau auswählen können (s.u.).



4.6.2 Prozess laden

Load_Process.VI

Parameter:

Filename	Name der Binärdatei, die geladen werden soll.
Showerror	*true*: bei Fehler Meldungsfenster ausgeben

Return value:

1	OK
ungleich 1	Fehler

Das VI lädt einen Prozess (Binärdatei), der von ***ADbasic*** mit der Funktion '*Make Bin File*' erzeugt wurde.

4.6.3 Prozess starten

Start_Process.VI

Parameter:

ProcessNo	Nummer des zu startenden Prozesses (1...10)

Return value:

255	Fehler
ungleich 255	OK

4.6.4 Prozess stoppen (beenden)

Stop_Process.VI

Parameter:

ProcessNo	Nummer des zu beendenden Prozesses (1...10)

Return value:

255	Fehler
ungleich 255	OK

4.6.5 Prozess löschen

Diese Funktion gibt es nur bei ADwin-Systemen mit ADSP-Prozessor (T9, T10) und ist unter Linux nicht verfügbar.

Clear_Process.VI

Parameter:

ProcessNo	Nummer des zu löschenden Prozesses: 1…12, 15

Return value:

1	Fehler
ungleich 1	OK

Mit Hilfe dieser Funktion können die Standardprozesse 11, 12 und 15 sowie die Prozesse 1 bis 10 gelöscht werden. Die Standardprozesse 11 und 12 sind Messprozesse, die von den Entwicklungsumgebungen wie z.B. TestPoint und LabVIEW benutzt werden. Der Prozess 15 erzeugt das Blinken der LED beim ***ADwin-Gold***- und beim ***ADwin-Pro***-System. Durch Löschen der Prozesse 11, 12 und 15 wird der zur Verfügung stehende Programmspeicher vergrößert.

Bevor Sie einen Prozess löschen können, müssen Sie

1. diesen Prozess beenden (siehe: ***Stop_Process***)
2. überprüfen, ob der Prozess auch tatsächlich beendet ist (siehe: ***Process_Status***).

Erst jetzt dürfen Sie den Prozess aus dem Speicher löschen!

4.6.6 Prozess-Status auslesen

Process_Status.VI

Parameter:

ProcessNo	Nummer des Prozesses (1...10), dessen Status ermittelt werden soll

Return value:

0	Prozess ist inaktiv
ungleich 0	Prozess ist aktiv

4.6.7 Parameter Globaldelay setzen

Set_Globaldelay.VI

Parameter:

ProcessNo	Nummer des Prozesses (1...10) bei dem das *Globaldelay* gesetzt werden soll.
Globaldelay	Einzustellender *Globaldelay*-Wert (siehe unten)

Return value:

255	Fehler
ungleich 255	OK

Mit *Globaldelay* legen Sie die Zeitspanne zwischen zwei Event-Aufrufen eines zeitgesteuerten Prozesses fest (siehe ***ADbasic***-Handbuch).

Bei den Prozessoren T9 und T10 wird das *Globaldelay* bei hoch priorisierten Prozessen in Schritten zu 25 ns und bei niedrig priorisierten Prozessen in Schritten zu 100 µs (T9) oder 50 µs (T10) eingestellt.

Bei den Prozessoren T2, T4, T5 und T8 wird das *Globaldelay* bei hoch priorisierten Prozessen in Schritten zu 1 µs und bei niedrig priorisierten Prozessen in Schritten zu 64 µs eingestellt.

4.6.8 Parameter Globaldelay lesen

Get_Globaldelay.VI

Parameter:

ProcessNo	Nummer des Prozesses (1...10), von dem das *Globaldelay* ausgelesen werden soll

Return value:

Der aktuell eingestellte *Globaldelay*-Wert (im Fehlerfall 255)

Weitere Informationen siehe ***Set_Globaldelay*** und ***ADbasic***-Handbuch.

4.6.9 Integer-Parameter setzen

Set_Par.VI

Parameter:

Index	Bedeutung
1	PAR_1
2	PAR_2
usw. bis 80	PAR_80

Value	Neuer Wert für den gewählten Integer-Parameter

Return value:

255	Fehler
ungleich 255	OK

4.6.10 Float-Parameter setzen

Set_Fpar.VI

Parameter:

Index	Bedeutung
1	FPAR_1
2	FPAR_2
usw. bis 80	FPAR_80

Value Neuer Wert für den gewählten Float-Parameter

Return value:

255 Fehler
ungleich 255 OK

4.6.11 Integer-Parameter lesen

Get_Par.VI

Parameter:

Index	Bedeutung
1	PAR_1
2	PAR_2
usw. bis 80	PAR_80

***Parameter value*:**

Aktueller Wert des gewählten Integer-Parameters (im Fehlerfall 255)

4.6.12 Mehrere Integer-Parameter lesen

Get_Par_Block.VI

Parameter:

Startindex Parameter, ab dem der Block ausgelesen wird
Count Anzahl der Parameter, die gelesen werden sollen

Data Array der gewünschten Parameter

***Return value*:**

0 OK
ungleich 0 Fehler

Startindex bestimmt das Element 0 des Arrays: Wenn Sie mit *Startindex = 1* beginnen, ist PAR_1 das Element 0 des DATA-Arrays; bei *Startindex = 5* ist PAR_5 das Element 0 des Arrays.

4.6.13 Float-Parameter lesen

Get_Fpar.VI

Parameter:

Index	Bedeutung
1	FPAR_1
2	FPAR_2
usw. bis 80	FPAR_80

***Return value*:**

aktueller Wert des gewählten FLOAT-Parameters (im Fehlerfall 255)

4.6.14 Mehrere Float-Parameter lesen

Get_FPar_Block.VI

Parameter:

Startindex	Parameter, ab dem der Block ausgelesen wird
Count	Anzahl der Parameter, die gelesen werden sollen
Data	Array der gewünschten Parameter

***Return value*:**

0	OK
ungleich 0	Fehler

Startindex bestimmt das Element 0 des Arrays: Wenn Sie mit *Startindex = 1* beginnen, ist PAR_1 das Element 0 des DATA-Arrays; bei *Startindex = 5* ist PAR_5 das Element 0 des Arrays.

4.6.15 Datensatz in ein Long-, oder Float-Array übernehmen

GetData_Long.VI
GetData_Float.VI

Parameter:

DataNo	Datensatznummer (1...200)
Startindex	Index des ersten zu übertragenden Elementes
Count	Anzahl der Elemente, die übertragen werden
Data	Daten-Array

***Return value*:**

255	Fehler
ungleich 255	OK

Mit dem FLOAT-Befehl können auch LONG-Datas vom ***ADwin***-System gelesen werden, die dann in ein 32-Bit FLOAT-Format gewandelt werden.

Mit dem LONG-Befehl können auch FLOAT-Datas vom ***ADwin***-System gelesen werden, die dann in ein 32-Bit LONG-Format gewandelt werden.

4.6.16 Long- oder Float-Array als Datensatz senden

SetData_Long.VI
SetData_Float.VI

Parameter:

DataNo	Datensatznummer (1...200)
Startindex	Index des ersten zu übertragenen Elementes
Count	Anzahl der Elemente die übertragen werden
Data	Array, das übertragen wird

***Return value*:**

255	Fehler
ungleich 255	OK

Mit dem FLOAT-Befehl können auch LONG-Datas zum ***ADwin***-System geschrieben werden, die dann in ein 32-Bit FLOAT-Format gewandelt werden.

Mit dem LONG-Befehl können auch FLOAT-Datas zum ***ADwin***-System geschrieben werden, die dann in ein 32-Bit LONG-Format gewandelt werden.

4.6.17 Datensatz in einer Datei speichern

Data2File.VI

Parameter:

Mode	Bedeutung
0	Vorhandene Datei wird überschrieben
1	Falls die Datei bereits existiert, werden die Daten angehängt

Filename	Name der Datei, in der die Daten gespeichert werden
DataNo	Datensatz-Nummer
Startindex	Index des 1. zu speichernden Elementes
Count	Anzahl der Elemente, die gespeichert werden

Return value:

0	OK
ungleich 0	Fehler

4.6.18 Länge eines Datensatzes ermitteln

Data_Length.VI

Parameter:

DataNo	Datensatznummer

length:

0	Fehler oder Datensatz existiert nicht
ungleich -1	Länge des Datensatzes

Hinweis zum FIFO Datensatz:

Die folgenden Funktionen greifen auf FIFO Datensätze zu. Unter ***ADbasic*** können Datensätze (DATA) als FIFO-(**F**irst **I**n **F**irst **O**ut)-Ringspeicher deklariert werden. Die Elemente in einem FIFO-Datensatz werden in der selben Reihenfolge ausgelesen, in der sie in das FIFO geschrieben wurden.
FIFO-Datensätze müssen unter ***ADbasic*** deklariert werden. (Vgl. ***ADbasic*** Handbuch)

4.6.19 FIFO-Daten in Integer- oder Float- Array übernehmen

Vor dem Aufruf dieser Funktion sollten Sie mit der Funktion FIFO_FULL überprüfen, ob noch genügend Elemente im FIFO sind.

GetFifo_Long.VI
GetFifo_Float.VI

Parameter:

FifoNo	FIFO Datensatznummer
Count	Anzahl der Elemente, die übertragen werden
Data	gelesenes Array

***Return value*:**

255	Fehler
ungleich 255	OK

Mit dem FLOAT- Befehl können auch LONG-Datas vom ***ADwin***-System gelesen werden, die dann in ein 32-Bit FLOAT-Format gewandelt werden.

Mit dem LONG-Befehl können auch FLOAT-Datas vom ***ADwin***-System gelesen werden, die dann in ein 32-Bit LONG-Format gewandelt werden.

4.6.20 Integer oder Float Array-Daten in FIFO schreiben

Vor dem Aufruf dieser Funktion sollten Sie mit der Funktion FIFO_EMPTY überprüfen, ob noch genügend Platz im FIFO ist.

SetFifo_Long.VI
SetFifo_Float.VI

Parameter:

FifoNo	FIFO Datensatznummer
Count	Array, dessen Werte in den FIFO übertragen werden
Data	Anzahl der Elemente, die übertragen werden

Return value:

255	Fehler
ungleich 255	OK

Mit dem FLOAT-Befehl können auch LONG-Datas zum ***ADwin***-System geschrieben werden, die dann in ein 32-Bit FLOAT-Format gewandelt werden.

Mit dem LONG-Befehl können auch FLOAT-Datas zum ***ADwin***-System geschrieben werden, die dann in ein 32-Bit LONG-Format gewandelt werden.

4.6.21 Anzahl der Elemente im FIFO ermitteln

Fifo_Full.VI

Parameter:

FifoNo	Nummer des als FIFO deklarierten Datensatzes

Count (= return value):

255	Fehler
ungleich 255	Anzahl der im FIFO befindlichen Elemente

4.6.22 Anzahl der freien FIFO Elemente ermitteln

Fifo_Empty.VI

Parameter:

FifoNo	Nummer des als FIFO deklarierten Datensatzes

Return value:

255	Fehler
ungleich 255	Anzahl der freien Elemente im FIFO

4.6.23 Inhalt des FIFO löschen

Fifo_Clear.VI

Parameter:

FifoNo	Nummer des als FIFO deklarierten Datensatzes

***Return value*:**

255	Fehler
ungleich 255	OK

4.7 Systemfunktionen

4.7.1 Auslastung des *ADwin*-Systems abfragen

Workload.VI

Parameter:

Priority	
0	aktuelle Gesamtauslastung des Prozessors
ungleich 0	wird derzeit noch nicht unterstützt

Workload (= return value):

aktuelle Auslastung des Prozessors in % (im Fehlerfall 255)

4.7.2 Korrektes Laden des Betriebssystems prüfen

Test_Version.VI

Return value:

0	OK
ungleich 0	System nicht geladen

4.7.3 Freien Speicher abfragen

Free_Mem.VI

Parameter:

Mem_Spec	Speicherart
0	Programm- und Datenspeicher bei T2, T4, T5 und T8
1	Prozessorinterner Programmspeicher (PM) beim ADSP (T9, T10)
2	Prozessorinterner Datenspeicher (DM) beim ADSP (T9, T10)
3	Externer Datenspeicher (SDRAM) beim ADSP (T9, T10)

Return value:

255	Fehler
ungleich 255	Zusammenhängender freier Speicher in Byte

4.7.4 Ausgabe einer Fehlermeldung

Error_msg.VI

Parameter:

Show Error Dialog ?	bei *true* wird Fehlermeldung ausgegeben, bei *false* nicht

Return value:

Error ?	wird *true* bei Fehlern

Dieses VI kann gut zum Anzeigen von Fehlern und z.B. Stoppen einer Schleife verwendet werden.

4.7.5 Sprache der Fehlermeldungen einstellen

Set_Language.VI

Parameter:

0	Übernimmt die Windows-Spracheinstellung. Bisher stehen Fehlermeldungen nur in Deutsch und Englisch zur Verfügung, daher wird bei anderen Sprachen Englisch verwendet.
1	Sprache englisch
2	Sprache deutsch

4.7.6 Prozessortyp abfragen

Processor_Type.VI

Return value:

Wert	Bedeutung
0	Fehler
2	T2
4	T4
5	T5
8	T8
9	T9
1010	T10

4.8 Nur in LabVIEW vorhandene Funktionen

4.8.1 Spannung in *value*-Wert umrechnen

Volt2Value.VI

Das VI berechnet aus einem Spannungswert in [Volt] denjenigen *value*-Wert, der an den DAC zu übergeben ist, damit er diese Spannung am Analogausgang erzeugt.

Je nach Bedarf können Sie Einzelwert, 1D-Array oder 2D-Array am Ein- und Ausgang anschließen. *Converter resolution* und *Voltage range* stellen Sie je nach ***ADwin***-System ein.

4.8.2 *Value*-Wert in Spannung umrechnen

Value2Volt.VI

Das VI berechnet aus einem vom ADC gelesenen *value*-Wert die Spannung in [Volt], die dort anliegt.

Je nach Bedarf können Sie Einzelwert, 1D-Array oder 2D-Array am Ein- und Ausgang anschließen. *Converter resolution* und *Voltage range* stellen Sie je nach ***ADwin***-System ein.

4.8.3 Mehrere Analogwerte einlesen

Diese Funktion sollte nur zum schnellen Testen und nicht zusammen mit einem ***ADbasic***-Prozess verwendet werden.

Acquire.VI

Parameter:

Sampling rate	Abtastrate in Hz (1 Hz...30000 Hz)
Processnumber	Messprozessnummer (1...4)
No. of samples	Anzahl der zu messenden Werte, 2...32000
Channel List	Array für die zu messenden Kanäle, 1 = Kanal messen, Array-Index+1: Inputkanalnummer (1...12)

4.8.4 Zyklische Analogausgabe eines Puffers starten

Diese Funktion sollte nur zum schnellen Testen und nicht zusammen mit einem ***ADbasic***-Prozess verwendet werden.

StartDAC.VI

Parameter:

Sample Rate	Ausgabetaktrate in Hz (1 Hz ... 100 000 Hz)
Process	D/A Prozessnummer (1 oder 2)
Values	Array der auszugebenden Werte
Channel List	Array für die Ausgabekanäle, 1 = Kanal ausgeben, Array-Index+1: Ausgangskanalnummer (1...6)
Repeat	Anzahl der Wiederholungen von *Values*

4.8.5 Zyklische Analogausgabe von *StartDAC* stoppen

Diese Funktion sollte nur zum schnellen Testen und nicht zusammen mit einem ***ADbasic***-Prozess verwendet werden.

StopDAC.VI

Parameter:

Process	D/A Prozessnummer (1 oder 2)

5 Befehlsindex

Revisionen

Date/Version	Name	Revision	Info
Bis 01 04	MH		
01 05 03	HR	neu: Kap- 2.4 Portierung MS WIN und Linux	MB
01 05 04	MH	Kap. 2.1 ersetzt Kap. 2.3 überarbeitet (Differenzierung zw. LabVIEW6 u. allen vorigen Versionen) Bei allen Abbildungen Linkadresse durch Device No. ersetzt Neue VIs: Get_Par_Block Get_FPar_Block GetiFiFoPart GetfFiFoPart Fifo_Full Get_Last_Error Get_Last_Error_Text	MH
28 05 01 **Version 5/01**	HR	Hinweis am Ende des Kap 3.5 entfällt	MH; PDF
19 02 02	TN	Neue Treiberversion v2, in Anlehnung an Delphi-Treiber VIs gelöscht: Get_Last_Error, Get_Last_Error_Text	MH
22.02.02 Version 1.6	GC	Konzept angepasst, Sprache und Format überarbeitet	TN
25.02.02	RRa	Formatierungen angepasst; Fehler bereinigt	
27.03.02	GC	Workload: Derzeit nur Unterstützung von Priorität 0 (Gesamtauslastung Prozessor)	PDF
9.4.02	GC	Versionsbezeichnung geändert in Version 4/02	PDF
9.7.02	GC	Befehle DAC, ADC, Get_Digin, Set_digout, Get_digout entfernt, da sie nur noch über ADbasic angesprochen werden sollen.	PDF Juli 02
11.7.02	GC	Kap. 4.3: Ersetze "neue VI-Funktionsnamen" sinngemäß durch "Umgang mit älteren Treiberversionen"	
27.8.02	GC	Data_Length: Rückgabewert von "-1" in "0" korrigiert	
20.10.03	GC	Clear_Process: Neuer Hinweis „nicht verfügbar unter Linux“	
22.10.03	GC	Informationen für T10 ergänzt. Einbinden der VIs für Version 7 eingefügt. Eignung für LabView 7 ergänzt.	PDF Okt. 03