ADwin

Treiber für LabVIEW ab Version 6

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

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 -verzeichnisse sind in spitzen Klammern und im 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	…	1	0
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 des ***ADwin***-LabVIEW-Treibers für LabVIEW® ab Version 6.

Der Treiber ist für den Einsatz unter früheren LabVIEW-Versionen (5 oder kleiner) nicht geeignet. Falls erforderlich, wenden Sie sich bitte an unsere Hotline.

Das Handbuch wird ergänzt durch

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

  Beginnen Sie die Installation mit diesem Handbuch.
- Falls Sie unter Linux arbeiten: das Handbuch „ADwin Linux“, das die Software-Installation und den ***ADbasic***-Compiler unter Linux beschreibt.
- das Handbuch ***ADbasic***, das die Entwicklungsumgebung und die Befehle des Compilers ***ADbasic*** beschreibt. Mit diesem komfortablen Echtzeit-Entwicklungstool programmieren Sie Ihr ***ADwin***-System.

  Die Online-Hilfe zu ***ADbasic*** enthält die gleichen Informationen wie das Handbuch.
- die Hardware-Handbücher für die von Ihnen verwendeten ***ADwin***-Systeme.

Es wird vorausgesetzt, dass Sie den Umgang mit der Entwicklungsumgebung LabVIEW® 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

Hotline-Adresse siehe vordere Umschlagseite, innen.

2 *ADwin*-LabVIEW-Treiber

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, Linux oder Mac, 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 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.

Die Abschnitte 2.1 und 2.2 schildern, wie LabVIEW und ***ADwin*** kommunizieren und vertiefen Ihr Verständnis für das ***ADwin***-Konzept.

In Kapitel 3 wird die Installation und Einbindung der neuen Befehle beschrieben.

Allgemeines zum LabVIEW-Treiber ist in Kapitel 4 erklärt, die einzelnen Funktionen in Form eines als Nachschlagewerks in Kapitel 5.

2.1 Schnittstelle zur Entwicklungsumgebung

Der ***ADwin***-LabVIEW-Treiber ist die Schnittstelle für die Entwicklungsumgebung LabVIEW® ab Version 6 zur Kommunikation mit ***ADwin***-Systemen.

Die Kombination von LabVIEW® mit einem ***ADwin***-System bietet Ihnen völlig neue Möglichkeiten. Die Intelligenz und Rechenleistung des ***ADwin***-Systems zum einen und die Funktionen zum Verwalten, Analysieren und Dokumentieren von Messdaten zum anderen bieten ein leistungsstarkes Konzept.

Typische Anwendungen sind:

- Steuerung schneller Prüfstände
- Signale generieren
- Intelligent messen, Daten mit komplexen Triggerbedingungen erfassen
- Regeln und Steuern
- Online-Verarbeitung, Datenreduzierung
- Hardware-in-the-Loop, Simulation von Sensordaten

2.2 Kommunikation mit dem *ADwin*-System

Aus der Entwicklungsumgebung LabVIEW® können Sie Prozesse im ***ADwin***-System steuern, sowie Daten von dort anfordern oder dorthin senden. Die Prozesse selbst programmieren Sie mit dem Echtzeit-Entwicklungstool ***ADbasic*** (siehe Handbuch oder Online-Hilfe ***ADbasic***).

Daten und Befehle zwischen LabVIEW® und dem ***ADwin***-System durchlaufen den nachfolgend dargestellten Weg.

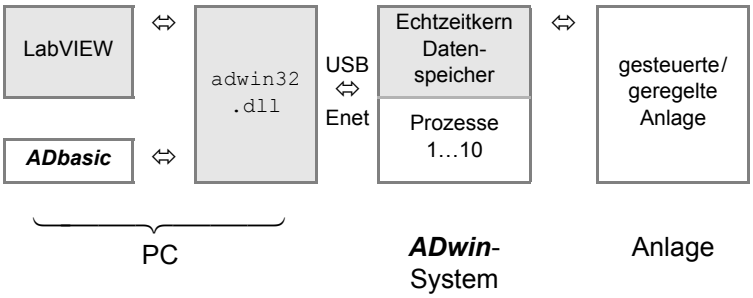


Abb. 1 – ***ADwin***-LabVIEW Schnittstelle

adwin32.dll

Die `adwin32.dll` ist die einzige Schnittstelle für Windows-Anwendungen zum ***ADwin***-System und wird daher auch vom ***ADwin***-LabVIEW-Treiber genutzt. Diese Schnittstelle ermöglicht, dass mehrere Windows-Programme gleichzeitig mit dem ***ADwin***-System kommunizieren: So können Entwicklungsumgebungen, ***ADbasic*** und ***ADtools*** parallel mit dem ***ADwin***-System arbeiten.

Echtzeitkern

Die Schnittstelle `adwin32.dll` kommuniziert mit dem Echtzeitkern des ***ADwin***-Systems, dem Betriebssystem. Deshalb müssen Sie nach jedem Einschalten des Systems zunächst das Betriebssystem (z. B. in Form einer Datei wie `<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-Variable (`PAR_1` … `PAR_80`), bereits vordefiniert
  - 80 Float-Variable (`FPAR_1` … `FPAR_80`), bereits vordefiniert
  - 200 Datenfelder (`DATA_1` … `DATA_200`), frei definierbare Länge

  Sie können die Werte dieser Variablen bzw. Datenfelder aus der Entwicklungsumgebung jederzeit lesen und ändern.

Kommunikation

– Kommunikation zwischen ***ADwin***-System und PC (`adwin32.dll`).

  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 aus LabVIEW (oder aus anderen Programmen) an das ***ADwin***-System richten: Steuerbefehle und Befehle für den Datenaustausch.

  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 LabVIEW-Treiber installieren

Bitte befolgen Sie die unten beschriebenen Installationsschritte, um auf einfache Weise Zugang aus LabVIEW® zum ***ADwin***-System zu bekommen.

3.1 „*ADwin* Installation“ ausführen

Für die Installation benötigen Sie die aktuelle ***ADwin***-CD.

3.1.1 Installation unter Linux oder Mac

Folgen Sie der Installationsanleitung im Handbuch „ADwin für Linux / Mac“. Achten Sie darauf, auch das Archiv `adwin-labview-x.y.tar.gz` zum Schluss zu installieren.

Nach erfolgreicher Installation befinden sich die VI-Sammlung und Beispiele für LabVIEW im Installationsverzeichnis:
`</opt/adwin/share/ADwin_v2.2.lib/>`

Fahren Sie fort mit Kapitel 3.2 „ADwin VIs einbinden“.

3.1.2 Installation unter Windows

Falls *ADwin* installiert ist

Wenn Sie bereits ein ***ADwin***-System installiert haben, stellen Sie bitte sicher, dass Sie das Programmpaket „`Developer-Software`“ von der ***ADwin***-CD mit Version 4.01.16.00 (oder höher) installiert haben. Wenn das der Fall ist, können Sie diesen Abschnitt überspringen und mit Kapitel 3.2 weiter arbeiten.

Sonst: Neue Installation

Falls ein ***ADwin***-System neu installiert werden soll, beginnen Sie bitte die Installation mit dem Handbuch „***ADwin*** Installation“, das mit der ***ADwin***-Hardware ausgeliefert wird. Es beschreibt, wie Sie

- die Software „`ADwin Driver and ADbasic`“ und „`Developer-Software`“ von der ***ADwin***-CD installieren.
- die Kommunikations-Treiber unter Windows installieren.
- die Hardware im PC einbauen (falls erforderlich) und die Hardware-Verbindung zwischen PC und ***ADwin***-System aufbauen.

Nach erfolgreicher Installation befinden sich im Installationsverzeichnis `<C:\ADwin\Developer\LabVIEW\ADwin_v2.2.lib>` die VI-Sammlung und Beispiele für LabVIEW.

Fahren Sie fort mit dem nächsten Abschnitt: „ADwin VIs einbinden“.

3.2 *ADwin* VIs einbinden

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 Schnittstelle (siehe Kapitel 2.2) implementiert sind.

Binden Sie die VIs mit folgenden Schritten in LabVIEW ein:

– Kopieren Sie den Ordner `<ADwin_v2.2.lib>` aus dem Installationsverzeichnis in das LabVIEW-Verzeichnis, unter Windows z.B. nach `<C:\Program Files\National Instruments\LabVIEW 6>`.

– Starten Sie LabVIEW und wählen die Schaltfläche „`Blank VI`“ (vor LabVIEW 8: „`New VI`“).

– Aktivieren Sie das weiße Diagramm-Fenster, entweder mit einem Mausklick auf das verdeckte weiße Fenster oder mit der Tastenkombination `[Strg]+[E]`.

Der folgende Ablauf ist für die LabVIEW-Version 8 anders als für die Versionen 6+7. Beachten Sie die verschiedenen Beschreibungen.

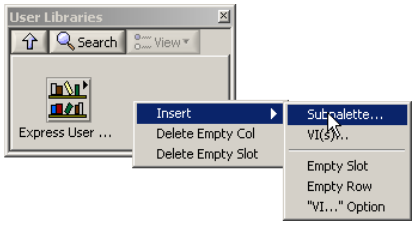
LabVIEW 8

– Wählen Sie im Menü des Blockdiagramm-Fensters die Option „`Tools` ▸ `Advanced` ▸ `Edit Palette Set ...`“.

Die Fenster „`Edit Controls and Functions Palettes`“ und „`Functions`“ öffnen sich.

– Klicken Sie im Fenster „`Functions`“ auf das Ordnersymbol „`User Libraries`“ und anschließend mit der rechten Maustaste (Mac: ⌘ + Mausklick) in das Fenster „`User Libraries`“.

– Klicken Sie mit der rechten Maustaste (Mac: ⌘ + Mausklick) auf die graue Fläche im Fenster „`User Libraries`“ und wählen im Kontextmenü den Menüeintrag „`Insert` ▸ `Subpalette`“.



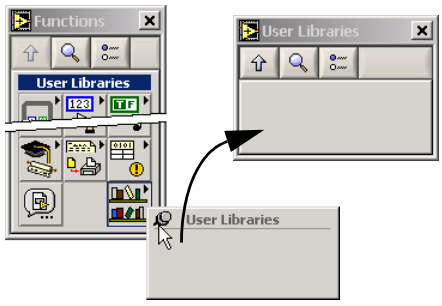
Es erscheint das Fenster „`Insert Subpalette`“.

Weiter mit „Alle LabVIEW-Versionen“.

LabVIEW 6+7

– Klicken Sie im Fenster „`Functions`“ mit der rechten Maustaste (Mac: ⌘ + Mausklick) auf das Symbol „`User Libraries`“.

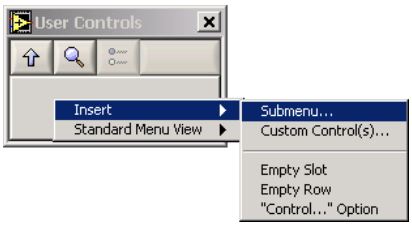
Verankern Sie das Fenster „`User Libraries`“ auf dem Desktop, indem Sie die Reißzwecke anklicken (siehe rechts).



– Öffnen Sie das Fenster „`Edit Controls and Functions Palettes`“ auf folgende Weise:
  - LabVIEW 7: Wählen Sie im Menü des Blockdiagramm-Fensters die Option „`Tools` ▸ `Advanced` ▸ `Edit Palette Views`“.
  - LabVIEW 6:Klicken Sie im Fenster „`User Controls`“ mit der linken Maustaste auf das Symbol „`Options`“. Wählen Sie in dem erscheinenden Fenster die Taste „`Edit Palettes...`“.

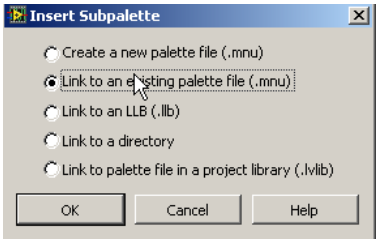
Alle LabVIEW-Versionen

– Klicken Sie mit der rechten Maustaste (Mac: Ctrl+Klick) auf die graue Fläche im Fenster „User Libraries“ und wählen im Kontextmenü den Menüeintrag „Insert ▸ Submenu“.



Es erscheint das Fenster „Insert Subpalette“.

– Wählen Sie die Option „Link to an existing menu file (.mnu)“ und bestätigen mit „OK“.

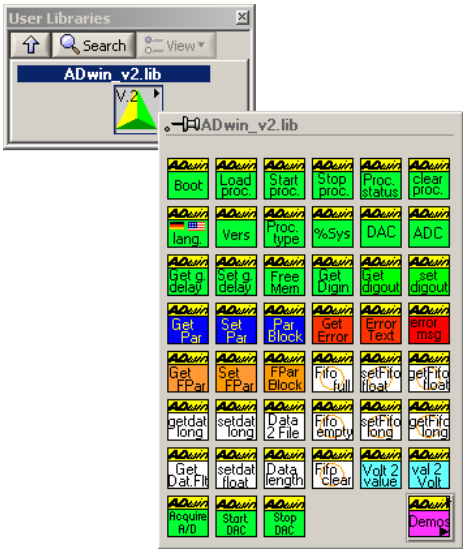


Wählen Sie in der Dateiauswahlbox – im LabVIEW-Verzeichnis – das Verzeichnis <ADwin_v2.2.lib> und die Datei <ADwin_v2.2.mnu>.

Die ***ADwin***-VIs sind nun in die Palette eingebunden. Speichern Sie die Änderung, indem Sie im Fenster „Edit Controls and Functions Palette Set“ die Schaltfläche „Save Changes“ anklicken.

– Aktivieren Sie nochmals das weiße Diagramm-Fenster.

Wenn Sie in der Gruppe „User Libraries“ die Maus über das Icon „ADwin_v2.2.lib“ bewegen, erscheinen alle VIs in einem neuen Fenster (siehe rechts).



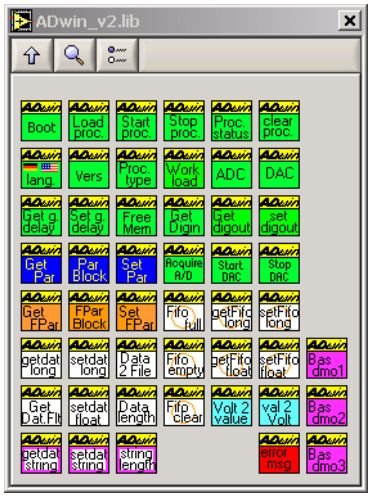
Sie können die VIs aus der Gruppe direkt in ein LabVIEW-Diagramm ziehen. Die einzelnen VIs sind in Kapitel 5 beschrieben.

4 Allgemeines zu *ADwin*-VIs

4.1 Grundlagen

Die Farben der VI-Symbole (Icons) zeigen die Funktion der VIs an; die Farben orientieren sich teilweise am LabVIEW-Standard:

Grün:	Systeminformationen und Steuerung von Prozessen
Blau:	Übertragung von globalen Long-Variablen
Orange:	Übertragung von globalen Float-Variablen
Rot:	Fehlerfunktion
Weiß:	Übertragung von DATA-Feldern
Weiß mit Ring:	Übertragung von FIFO-Feldern (Ringspeicher)
Weiß mit Rahmen:	Übertragung von Zeichenfolgen
Hellblau:	Hilfsfunktionen
Pink:	Beispielprogramme



Beachten Sie bitte für die Verwendung der ***ADwin***-VIs folgende Hinweise:

Fehlerverdrahtung

– Verbinden Sie alle ***ADwin***-VIs mit „`error in`“ und „`error out`“ durch eine Fehlerverdrahtung (siehe Kapitel 4.3).

Device No.

– Übergeben Sie eine „Device No.“ (siehe Kapitel 4.4) als Eingangswert an die VIs. Nur wenige VIs kommen ohne „Device No.“ aus.

Kontext-Hilfe

– Zu allen VIs gibt es eine Kontext-Hilfe, die Sie durch `[Strg]+[H]` starten.

  Wenn Sie die Maus bei geöffneter Hilfe auf ein VI-Symbol bewegen, werden ein (englischer) Hilfetext und eine Funktionsskizze des VI angezeigt. Dieses Handbuch enthält nur den Hilfetext, nicht aber die Funktionsskizze.

4.2 Treiber für LabVIEW 4, 5

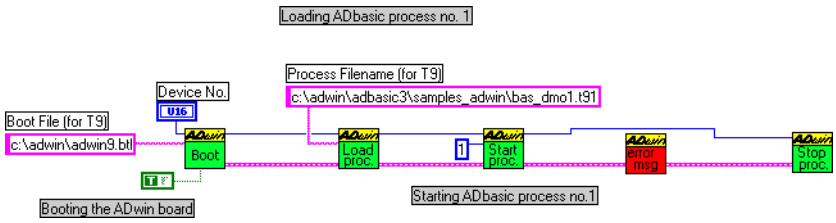
Wenn Sie einen älteren Treiber für LabVIEW 4 or 5 besitzen, können Sie VIs des älteren Treibers weiterhin verwenden, wir empfehlen es jedoch nicht.

Der aktuelle Treiber hat im Vergleich dazu eine überarbeitete Namensgebung für die VI-Funktionen, ist besser auf LabVIEW 6 / 7 abgestimmt und ermöglicht eine Fehlerbehandlung.

4.3 Fehlerbehandlung

Der LabVIEW-Treiber stellt Ihnen mit der „Fehlerverdrahtung“ eine Möglichkeit der Fehlerbehandlung zur Verfügung. Die Fehlerverdrahtung legt außerdem eindeutig fest, in welcher Reihenfolge LabVIEW die VIs bearbeitet.

In dem Beispiel unten verbindet eine gestrichelte Linie (pink) alle VIs: Dies ist die Fehlerverdrahtung. Der Fehlereingang des ersten VI bleibt offen.



Sobald bei einem ***ADwin***-VI ein Fehler auftritt, wird über die Verdahtung ein Fehlersignal an alle folgenden VI weiter geleitet, die daraufhin das ***ADwin***-System nicht mehr ansprechen. Mit dem VI `Error_msg` kann eine Fehlermeldung generiert werden.

Wenden Sie die Fehlerverdrahtung konsequent an:

– Verbinden Sie alle VIs miteinander über die Anschlüsse „`error in`“ und „`error out`“.
– Benutzen Sie die Fehlermeldungen zur Fehlerbehandlung, indem Sie die Meldungen mit dem VI `error_msg` abfragen (beispielsweise einmal pro Schleifendurchlauf).

4.4 Die „DeviceNo.“

Eine „Device No.“ kennzeichnet ein ***ADwin***-System eindeutig und wird für die korrekte Kommunikation mit dem ***ADwin***-System benötigt.

Sie stellen die „Device No.“ eines ***ADwin***-Systems mit dem Programm ADconfig ein.

In LabVIEW benötigt fast jedes ***ADwin***-VI die „Device No.“. Schließen Sie an das erste VI die „Device No.“ an. Verbinden Sie den Ausgang „Device No. out“ mit dem nächsten VI und erstellen so eine durchgehende Verdrahtung (im Beispiel im Abschnitt „Fehlerbehandlung“ die obere Linie in blau).

4.5 Beispielprogramme

Die Treiberinstallation enthält einige Beispielprogramme, die das Zusammenspiel von LabVIEW mit dem ***ADwin***-System verdeutlichen. Die Programme sind in folgendem Verzeichnis abgelegt:

– Windows: `<C:\ADwin\ADbasic\samples_ADwin\>`
– Linux: `</opt/adwin/share/samples_ADwin/>`

Jedes Beispiel besteht jeweils aus einem LabVIEW-Diagramm und einem ***ADbasic***-Programm:

– Das LabVIEW-Diagramm steuert den ***ADwin***-Prozess und zeigt die übertragenen Daten an.
– Das ***ADbasic***-Programm definiert den Ablauf des Prozesses auf dem ***ADwin***-System.

Zum besseren Verständnis der Beispiele betrachten Sie bitte die ***ADbasic***-Quelltexte und die zugehörigen LabVIEW-Diagramme.

5 Beschreibung der *ADwin*-VIs

Die Beschreibung der VIs ist in folgende Abschnitte unterteilt:

Beachten Sie auf jeden Fall das Kapitel 4, in dem allgemeine Hinweise zur Verwendung der ***ADwin***-VIs beschrieben sind.

5.1 Systemsteuerung und -information

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

Boot.VI

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

Eingänge

Device No.	Device No. des Systems
Filename	Name der Betriebssystem-Datei mit vollständigem Pfad, abhängig vom Prozessortyp: T2: `ADwin2.btl` T4: `ADwin4.btl` T5: `ADwin5.btl` T8: `ADwin8.btl` T9: `ADwin9.btl` T10: `ADwin10.btl` T11: `ADwin11.btl` Voreinstellung: `C:\ADwin\ADwin9.btl`.
Memsize	Nur für Prozessoren T2…T8: Eingebaute Speichergröße (64 kB…32 MB)
error in	Anschluss des Fehlersignals.
showError	true: bei anstehendem Fehlersignal die Fehlermeldung in einem Meldungsfenster ausgeben. false: keine Fehlermeldung ausgeben.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Memory	Statusmeldung: <1000: Fehler beim Booten 8000: Booten o.k.; nur für T9, T10, T11 >8000: Booten o.k. und MemSize ist korrekt; nur für T2…T8. Werte für Speichergröße MemSize: 10000: 64 kB 100000: 1 MB 200000: 2 MB 400000: 4 MB 800000: 8 MB 1000000: 16 MB 2000000: 32 MB
error out	Fehlersignal

Bemerkungen

Die Initialisierung löscht alle Prozesse auf dem System und setzt alle globalen Variablen auf den Wert 0. Der Parameter `Processdelay` wird auf 1000 voreingestellt.

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

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

Stellen Sie die Speichergröße Memsize ein, indem Sie mit der rechten Maustaste auf den Eingang klicken und „Create ▸ Constant“ auswählen. In dem erscheinenden Auswahlfenster wählen Sie die passende Speichergröße.

Test_Version.VI

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

Eingänge

Device No.	Device No. des Systems.
error in	Anschluss des Fehlersignals.
showError	true: bei anstehendem Fehlersignal die Fehlermeldung in einem Meldungsfenster ausgeben. false: keine Fehlermeldung ausgeben.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Version	0: OK ≠0: Betriebssystem hat falsche Version oder antwortet nicht.
error out	Fehlersignal

Processor_Type.VI

Processor_Type.VI gibt den Prozessortyp des Systems zurück.

Eingänge

Device No.	Device No. des Systems.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Processor type	Prozessortyp: 0: Fehler 2: T2 4: T4 5: T5 8: T8 9: T9 1010: T10 1011: T11
error out	Fehlersignal

Workload.VI

`Workload.VI` gibt die Prozessor-Auslastung zurück.

Eingänge

Device No.	Device No. des Systems.
priority	0: Gesamtauslastung des Prozessors ≠0: keine Funktion
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Workload	≠255: Prozessorauslastung in % 255: Fehler
error out	Fehlersignal

Free_Mem.VI

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

Eingänge

Device No.	Device No. des Systems.
Processor	Prozessor- oder Speichertyp: 0: T2…T8 1: Interner Programmspeicher (PM_LOCAL); ab T9 2: Interner Datenspeicher (DM_LOCAL); ab T9 3: Externer Datenspeicher (DRAM_EXTERN); ab T9 4: Interner Zusatzspeicher (EM_LOCAL); ab T11
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Memory	≠255: Zusammenhängender freier Speicher in Byte 255: Fehler
error out	Fehlersignal

5.2 Prozess-Steuerung

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

Load_Process.VI

`Load_Process.VI` lädt eine Binärdatei als Prozess ins ***ADwin***-System.

Eingänge

Device No.	Device No. des Systems.
Filename	Name der Binärdatei mit vollständigem Pfad.
error in	Anschluss des Fehlersignals.
showError	true: bei anstehendem Fehlersignal die Fehlermeldung in einem Meldungsfenster ausgeben. false: keine Fehlermeldung ausgeben.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	1: OK ≠1: Fehler
error out	Fehlersignal

Bemerkungen

Eine Binärdatei wird mit dem Programm ***ADbasic*** erzeugt.

Der letzte Buchstabe im Dateinamen der Binärdatei (eine Ziffer) gibt an, mit welcher Nummer der Prozess im ***ADwin***-System angesprochen wird. Die „0“ steht für Prozess 10.

Start_Process.VI

`Start_Process.VI` startet einen Prozess.

Eingänge

Device No.	Device No. des Systems.
ProcessNo	Nummer (1…10) des Prozesses.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
error out	Fehlersignal

Bemerkungen

Der Prozess startet mit der Priorität, mit der er in ***ADbasic*** kompiliert wurde.

Stop_Process.VI

`Stop_Process.VI` stoppt einen Prozess.

Eingänge

Device No.	Device No. des Systems.
ProcessNo	Nummer (1…10) des Prozesses.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
error out	Fehlersignal

Clear_Process.VI

`Clear_Process.VI` löscht einen Prozess aus dem Speicher.

Eingänge

Device No.	Device No. des Systems.
ProcessNo	Nummer (1…10, 15) des Prozesses.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠1:OK 1: Fehler
error out	Fehlersignal

Bemerkungen

Diese Funktion gibt es nur bei Systemen mit einem Prozessor T9 oder T10 und ist unter Linux nicht verfügbar.

Das Löschen eines Prozesses gibt den zuvor belegten Programmspeicher frei. Um einen Prozess zu löschen, führen Sie die folgenden Schritte aus:

1. Beenden Sie den Prozess mit `Stop_Process.VI`.

2. Überprüfen Sie mit `Process_Status.VI`, ob der Prozess tatsächlich beendet ist.

3. Löschen Sie den Prozess mit `Clear_Process.VI`.

Der Prozess 15 erzeugt das Blinken der LED beim ***ADwin-Gold***- und beim ***ADwin-Pro***-System und belegt nur wenig Programmspeicher.

Löschen Sie Prozesse in umgekehrter Reihenfolge zum Laden der Prozesse, den zuletzt geladenen Prozess also zuerst. Sie vermeiden damit eine Fragmentierung des Programmspeichers.

Beachten Sie bitte, dass das Löschen eines Prozesses die im Prozess deklarierten Felder NICHT löscht.

Process_Status.VI

`Process_Status.VI` gibt den Status eines Prozesses zurück.

Eingänge

Device No.	Device No. des Systems.
ProcessNo	Nummer (1…10, 15) des Prozesses.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	0: Prozess ist gestoppt. ≠0: Prozess läuft.
error out	Fehlersignal

Set_Processdelay.VI

`Set_Processdelay.VI` stellt den Parameter `Processdelay` für einen Prozess ein.

Eingänge

Device No.	Device No. des Systems.
ProcessNo	Nummer (1…10) des Prozesses.
Processdelay	Einzustellender Processdelay-Wert.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
error out	Fehlersignal

Bemerkungen

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

Die Zeitspanne wird in einer Zeiteinheit angegeben, die vom Prozessortyp und von der Priorität des Prozesses abhängt:

Prozessortyp	Prozesspriorität	
	hoch	niedrig
T2, T4, T5, T8	1000 ns	64 µs
T9	25 ns	100 µs
T10	25 ns	50 µs
T11	3,3 ns	0,003 µs = 3,3 ns

Get_Processdelay.VI

`Get_Processdelay.VI` gibt den Wert des Parameter `Processdelay` für einen Prozess zurück.

Eingänge

Device No.	Device No. des Systems.
ProcessNo	Nummer (1…10) des Prozesses.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: Wert des Parameters `Processdelay`. 255: Fehler
error out	Fehlersignal

Bemerkungen

Weitere Informationen unter `Set_Processdelay.VI` oder im Handbuch ***ADbasic***.

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.VI

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

Eingänge

Device No.	Device No. des Systems.
Index	Nummer (1…80) der globalen Long-Variablen `PAR_1` ... `PAR_80`.
Value	Neuer Wert für die Variable.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255 OK 255 Fehler
error out	Fehlersignal

Get_Par.VI

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

Eingänge

Device No.	Device No. des Systems.
Index	Nummer (1…80) der globalen Long-Variablen `PAR_1` ... `PAR_80`.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Parameter value	≠255: Wert der gewählten Variable. 255: Fehler
error out	Fehlersignal

Get_Par_Block.VI

`Get_Par_Block.VI` liest eine anzugebende Anzahl an globalen Long-Variablen.

Eingänge

Device No.	Device No. des Systems.
Startindex	Nummer (1…80) der ersten gelesenen globalen Long-Variablen `PAR_1` ... `PAR_80`.
Count	Anzahl der Variablen, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	0: OK ≠0: Fehler
Data	Feld mit den ausgelesenen Werten.
error out	Fehlersignal

Bemerkungen

Die gelesenen Werte werden im Feld Data ab dem Element 0 abgelegt. Wenn beispielsweise Startindex = 5 ist, ist der Wert von `PAR_5` im Element 0 des Felds Data abgelegt.

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

Set_FPar.VI

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

Eingänge

Device No.	Device No. des Systems.
Index	Nummer (1…80) der globalen Float-Variablen FPAR_1 … FPAR_80.
Value	Neuer Wert für die Variable.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255 OK 255 Fehler
error out	Fehlersignal

Get_FPar.VI

Get_FPar.VI gibt den Wert einer globalen Float-Variablen zurück.

Eingänge

Device No.	Device No. des Systems
Index	Nummer (1…80) der globalen Float-Variablen FPAR_1 … FPAR_80.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Parameter value	≠255: Wert des gewählten Parameters. 255: Fehler
error out	Fehlersignal

Get_FPar_Block.VI

`Get_FPar_Block.VI` liest eine anzugebende Anzahl an globalen Float-Variablen.

Eingänge

Device No.	Device No. des Systems.
Startindex	Nummer (1…80) der ersten gelesenen globalen Float-Variablen `FPAR_1` ... `FPAR_80`.
Count	Anzahl der Variablen, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	0: OK ≠0: Fehler
Data	Feld mit den ausgelesenen Werten.
error out	Fehlersignal

Bemerkungen

Die gelesenen Werte werden im Feld Data ab dem Element 0 abgelegt. Wenn beispielsweise Startindex = 9 ist, ist der Wert von `FPAR_9` im Element 0 von Data abgelegt.

5.4 Übertragung von Datenfeldern (Arrays)

Befehle zur Datenübertragung zwischen PC und ***ADwin***-System mit globalen DATA-Feldern (DATA_1...DATA_200):

- Einfache DATA-Felder
- FIFO-Felder
- Data-Felder mit Zeichenfolgen

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

5.4.1 Einfache DATA-Felder

Sie müssen jedes DATA-Feld vor seiner Verwendung unter ***ADbasic*** deklarieren (vgl. Handbuch „***ADbasic***“): DIM DATA_x[n] AS *LONG/FLOAT*.

Data_Length.VI

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

Eingänge

Device No.	Device No. des Systems.
DataNo	Nummer (1…200) des DATA-Felds DATA_1...DATA_200.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠0: Deklarierte Länge des DATA-Felds. 0: Fehler oder DATA-Feld ist nicht deklariert.
error out	Fehlersignal

SetData_Long.VI

SetData_Long.VI überträgt Daten von LabVIEW als Long-Daten in ein DATA-Feld im ***ADwin***-System.

Eingänge

Device No.	Device No. des Systems.
DataNo	Nummer (1…200) des DATA-Felds DATA_1...DATA_200, in das die Werte übertragen werden.
Data	Feld mit den zu übertragenden Werten.
Startindex	Index des ersten übertragenen Feldelements im Feld Data.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
error out	Fehlersignal

Bemerkungen

Es werden alle Werte aus dem LabVIEW-Feld übertragen (ab dem Startelement). Das DATA-Feld muss größer sein als die Anzahl der übertragenen Werte.

Wenn das Feld Data FLOAT-Werte enthält, werden die Werte bei der Übertragung in ein 32-Bit LONG-Format gewandelt.

GetData_Long.VI

GetData_Long.VI überträgt Daten aus einem DATA-Feld vom ***ADwin***-System als Long-Daten nach LabVIEW.

Eingänge

Device No.	Device No. des Systems.
DataNo	Nummer (1…200) des DATA-Felds DATA_1...DATA_200.
Startindex	Index des ersten Feldelements im gewählten Feld.
Count	Anzahl der Feldelemente, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
Data	Feld mit den übertragenen Werten.
error out	Fehlersignal

Bemerkungen

Wenn das angegebene DATA-Feld FLOAT-Werte enthält, werden die Werte nach der Übertragung in ein 32-Bit LONG-Format gewandelt.

SetData_Float.VI

SetData_Float.VI überträgt Daten von LabVIEW als Float-Daten in ein DATA-Feld im ***ADwin***-System.

Eingänge

Device No.	Device No. des Systems.
DataNo	Nummer (1…200) des DATA-Felds DATA_1…DATA_200, in das die Werte übertragen werden.
Data	Feld mit den zu übertragenden Werten.
Startindex	Index des ersten übertragenen Feldelements im Feld Data.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
error out	Fehlersignal

Bemerkungen

Wenn das Feld Data LONG-Werte enthält, werden die Werte bei der Übertragung in ein 32-Bit FLOAT-Format gewandelt.

GetData_Float.VI

GetData_Float.VI überträgt Daten aus einem DATA-Feld vom ***ADwin***-System als Float-Daten nach LabVIEW.

Eingänge

Device No.	Device No. des Systems.
DataNo	Nummer (1…200) des DATA-Felds DATA_1…DATA_200.
Startindex	Index des ersten Feldelements im gewählten Feld.
Count	Anzahl der Feldelemente, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
Data	Feld mit den übertragenen Werten.
error out	Fehlersignal

Bemerkungen

Wenn das angegebene DATA-Feld LONG-Werte enthält, werden die Werte nach der Übertragung in ein 32-Bit FLOAT-Format gewandelt.

Data2File.VI

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

Eingänge

Device No.	Device No. des Systems.
Mode	Schreibmodus: 0: Datei überschreiben. 1: Daten an die Datei anhängen.
Filename	Name der Speicherdatei mit vollständigem Pfad.
DataNo	Nummer (1…200) des DATA-Felds DATA_1...DATA_200, dessen Werte übertragen werden.
Startindex	Index des ersten übertragenen Feldelements.
Count	Anzahl der Feldelemente, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	0: OK ≠0:Fehler
error out	Fehlersignal

Bemerkungen

Die Funktion kann nicht mit DATA-Feldern benutzt werden, die als FIFO-Feld definiert sind.

Die Daten werden binär gespeichert.

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.VI

Fifo_Empty.VI gibt die Anzahl der freien Elemente eines FIFO-Felds zurück.

Eingänge

Device No.	Device No. des Systems.
FifoNo	Nummer (1…200) des FIFO-Felds DATA_1…DATA_200.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Count	≠255: Anzahl der freien Elemente im FIFO-Feld. 255: Fehler
error out	Fehlersignal

Fifo_Full.VI

Fifo_Full.VI gibt die Anzahl der belegten Elemente eines FIFO-Felds zurück.

Eingänge

Device No.	Device No. des Systems.
FifoNo	Nummer (1…200) des FIFO-Felds DATA_1…DATA_200.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
Count	≠255: Anzahl der belegten Elemente im FIFO-Feld. 255: Fehler
error out	Fehlersignal

Fifo_Clear.VI

`Fifo_Clear.VI` initialisiert den Schreib- und den Lesezeiger eines FIFO-Felds.

Eingänge

Device No.	Device No. des Systems.
FifoNo	Nummer (1…200) des FIFO-Felds `DATA_1`…`DATA_200`.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
error out	Fehlersignal

Bemerkungen

Ein FIFO-Feld muss vor der Verwendung initialisiert werden, entweder im ***ADbasic***-Programm oder mit `Fifo_Clear.VI`.

Bei laufender Anwendung kann eine Initialisierung der FIFO-Zeiger sinnvollsein, wenn alle im FIFO-Feld gespeicherten Werte verworfen werden sollen (beispielsweise wegen eines Fehlers).

SetFifo_Long.VI

`SetFifo_Long.VI` überträgt Daten aus einem Feld in LabVIEW als Long-Werte in ein Fifo-Feld im ***ADwin***-System.

Eingänge

Device No.	Device No. des Systems.
FifoNo	Nummer (1…200) des FIFO-Felds `DATA_1`…`DATA_200`.
Data	Feld mit den zu übertragenden Werten.
Count	Anzahl der Feldelemente, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
error out	Fehlersignal

Bemerkungen

Prüfen Sie zuerst mit der Funktion `Fifo_Empty.VI`, ob noch genügend Platz im FIFO ist.
Übertragen Sie erst dann Werte mit `SetFifo_Long.VI`.

Wenn das FIFO-Feld FLOAT-Werte enthält, werden die Werte bei der Übertragung in ein 32-Bit LONG-Format gewandelt.

GetFifo_Long.VI

`GetFifo_Long.VI` überträgt Daten aus einem Fifo-Feld im ***ADwin***-System als Float-Werte nach LabVIEW.

Eingänge

Device No.	Device No. des Systems.
FifoNo	Nummer (1…200) des FIFO-Felds `DATA_1…DATA_200`.
Count	Anzahl der Feldelemente, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
Data	Feld mit den übertragenen Werten.
error out	Fehlersignal

Bemerkungen

Prüfen Sie zuerst mit der Funktion `Fifo_Full.VI`, ob noch genügend Elemente im FIFO sind.
Lesen Sie erst dann Werte mit `GetFifo_Long.VI` aus.

Wenn das DATA-Feld LONG-Werte enthält, werden die Werte bei der Übertragung in ein 32-Bit FLOAT-Format gewandelt.

SetFifo_Float.VI

`SetFifo_Float.VI` überträgt Daten aus einem Feld in LabVIEW als Float-Werte in ein Fifo-Feld im ***ADwin***-System.

Eingänge

Device No.	Device No. des Systems.
FifoNo	Nummer (1…200) des FIFO-Felds `DATA_1…DATA_200`.
Data	Feld mit den zu übertragenden Werten.
Count	Anzahl der Feldelemente, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
error out	Fehlersignal

Bemerkungen

Prüfen Sie zuerst mit der Funktion `Fifo_Empty.VI`, ob noch genügend Platz im FIFO ist.
Übertragen Sie erst dann Werte mit `SetFifo_Float.VI`.

Wenn das FIFO-Feld LONG-Werte enthält, werden die Werte bei der Übertragung in ein 32-Bit FLOAT-Format gewandelt.

GetFifo_Float.VI

`GetFifo_Float.VI` überträgt Daten aus einem Fifo-Feld im ***ADwin***-System als Float-Werte nach LabVIEW.

Eingänge

Device No.	Device No. des Systems.
FifoNo	Nummer (1…200) des FIFO-Felds `DATA_1…DATA_200`.
Count	Anzahl der Feldelemente, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠255: OK 255: Fehler
Data	Feld mit den übertragenen Werten.
error out	Fehlersignal

Bemerkungen

Prüfen Sie zuerst mit der Funktion `Fifo_Full.VI`, ob noch genügend Elemente im FIFO sind.
Lesen Sie erst dann Werte mit `GetFifo_Float.VI` aus.

Wenn das DATA-Feld LONG-Werte enthält, werden die Werte bei der Übertragung in ein 32-Bit FLOAT-Format gewandelt.

5.4.3 Data-Felder mit Zeichenfolgen

Befehle zur Übertragung von Zeichenfolgen mit globalen DATA-Feldern (DATA_1...DATA_200).

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

SetData_String.VI

SetData_String.VI überträgt eine Zeichenfolge von LabVIEW in ein DATA-Feld im ***ADwin***-System.

Eingänge

Device No.	Device No. des Systems.
DataNo	Nummer (1…200) des Felds DATA_1...DATA_200.
Data	Feld mit der zu übertragenden Zeichenfolge.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠-1: Anzahl der übertragenen Zeichen. -1: Fehler
error out	Fehlersignal

GetData_String.VI

GetData_String.VI überträgt eine Zeichenfolge von einem DATA-Feld im ***ADwin***-System nach LabVIEW.

Eingänge

Device No.	Device No. des Systems.
DataNo	Nummer (1…200) des Felds DATA_1...DATA_200.
MaxCount	Max. Anzahl der Zeichen, die übertragen werden.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠-1: Anzahl der übertragenen Zeichen. -1: Fehler
Data	Feld mit der übertragenen Zeichenfolge.
error out	Fehlersignal

Bemerkungen

Eine Zeichenfolge in einem DATA-Feld wird mit einer Ende-Kennung beendet (ASCII-Wert 0). Das Auslesen der Zeichenfolge endet genau an dieser Stelle. Der Rückgabewert ist die Anzahl der gelesenen Zeichen ohne Ende-Kennung.

String_Length.VI

`String_Length.VI` gibt die Länge einer Zeichenfolge in einem `DATA`-Feld im ***ADwin***-System zurück.

Eingänge

Device No.	Device No. des Systems.
DataNo	Nummer (1…200) des Felds `DATA_1…DATA_200`.
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
return value	≠-1: Länge der Zeichenfolge im DATA-Feld. -1: Fehler
error out	Fehlersignal

Bemerkungen

`String_Length` zählt die Zeichen in einerm `DATA`-Feld bis zum Auftreten der ersten Endekennung (ASCII-Wert 0). Die Endekennung wird nicht als Zeichen gezählt.

5.5 Fehlerbehandlung

Error_msg.VI

`Error_msg.VI` gibt eine Fehlermeldung zu einem Fehlersignal aus.

Eingänge

show error dialog?	true: Fehlermeldung ausgeben false: Keine Funktion
error in	Anschluss des Fehlersignals.

Ausgänge

error ?	true: Fehler liegt an false: kein Fehler
error out	Fehlersignal

Bemerkungen

Der Ausgang „error ?“ ist geeignet, um im Falle eines Fehlers das Programm zu stoppen, z. B. in einer Schleife.

Sie finden eine Liste aller Fehlermeldungen im Abschnitt A.2 im Anhang.

Set_Language.VI

`Set_Language.VI` stellt ein, in welcher Sprache Fehlermeldungen angezeigt werden.

Eingänge

Device No.	Device No. des Systems.
Language	gewählte Sprache: 0: Windows-Spracheinstellung übernehmen. 1: englisch 2: deutsch
error in	Anschluss des Fehlersignals.

Ausgänge

Device No. out	Die am Eingang „Device No.“ angelegte Nummer.
error out	Fehlersignal

Ausgänge

Wenn die Windows-Spracheinstellung übernommen wird, wird für jede Sprache außer deutsch die Einstellung „englisch“ verwendet.

5.6 Hilfsberechnungen

Volt2Value.VI

`Volt2Value.VI` berechnet aus einem Volt-Wert den entsprechenden Digit-Wert für den DAC.

Eingänge

Input voltage	Einzelner Spannungswert in Volt.
Input voltage array	Feld mit Spannungswerten in Volt.
Input voltage array 2dim.	2-dimensionales Feld mit Spannungswerten in Volt.
Converter resolution	Auflösung des ADC.
Voltage range	Spannungsbereich des ADC.

Ausgänge

Output value	Digit-Wert zum Eingang „Input voltage“.
Output value array	Digit-Werte zum Eingang „Input voltage array“.
Output value array 2dim.	Digit-Werte zum Eingang „Input voltage array 2dim.“

Bemerkungen

Das VI berechnet aus einem Spannungswert in Volt den passenden Digit-Wert. Wenn der Digit-Wert an den DAC übergeben wird, erzeugt der DAC den Spannungswert am Analogausgang. Converter resolution und Voltage range stellen Sie je nach verwendetem DAC ein.

Die Eingänge „Input voltage“ können einzeln oder parallel verwendet werden.

Value2Volt.VI

`Value2Volt.VI` berechnet aus einem Digit-Wert eines ADC den entsprechenden Volt-Wert.

Eingänge

Input value	Einzelner Digit-Wert.
Input value array	Feld mit Digit-Werten.
Input value array 2dim.	2-dimensionales Feld mit Digit-Werten.
Converter resolution	Auflösung des ADC.
Voltage range	Spannungsbereich des ADC.

Ausgänge

Output voltage	Volt-Wert zum Eingang „Input value“.
Output voltage array	Volt-Werte zum Eingang „Input value array“.
Output voltage array 2dim.	Volt-Werte zum Eingang „Input value array 2dim.“.

Bemerkungen

Das VI berechnet aus einem vom ADC gelesenen Digit-Wert die Spannung in Volt, die am ADC anliegt. Converter resolution und Voltage range stellen Sie je nach verwendetem ADC ein.

Die Eingänge „Input voltage“ können einzeln oder parallel verwendet werden.

Anhang

A.1 Index der Funktionen

A.2 Fehlermeldungen

Fehler-Nr.	Fehlermeldung
0	Kein Fehler
1	Timeout Fehler beim Schreiben zum ADwin-System.
2	Timeout Fehler beim Lesen vom ADwin-System.
10	Die Device-Nummer ist nicht erlaubt.
11	Die Device-Nummer ist nicht konfiguriert.
15	Funktion für dieses Device nicht erlaubt.
20	Inkompatible Versionen von ADwin-Betriebssystem, Treiberdatei adwin32.dll und / oder ADbasic-Binärdatei.
100	Das Data-Feld ist zu klein.
101	Das Fifo-Feld ist zu klein oder nicht genug Daten.
102	Das Fifo-Feld hat nicht genug Daten.
150	Nicht genug Speicher oder Speicherzugriffsfehler.
200	Datei konnte nicht gefunden werden.
201	Temporäre Datei konnte nicht erstellt werden.
202	Die Datei ist keine ADbasic Binärdatei.
203	Die Datei ist ungültig.[1]
204	Die Datei ist keine BTL.
2000	Netzwerk Fehler (TCP/IP).
2001	Netzwerk timeout.
2002	Falsches Passwort.
3000	USB Gerät nicht gefunden.
3001	Gerät nicht gefunden.

1. Möglicherweise fehlt bei `<ADwin5.btl>` die „memory table“, eine andere Datei wurde zu `<ADwin5.btl>` umbenannt oder diese ist beschädigt