JÄGER
Computergesteuerte
Messtechnik GmbH

ADwin-Gold-
USB / -ENET

Handbuch

ADwin-Gold USB / ENET, Handbuch Version 4.3, Januar 2012

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.

Dateinamen und -verzeichnisse sind in spitzen Klammern und im Schrifttyp Courier New angeben.

<C:\ADwin\…>

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

Programmtext

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

```
Var_1
```

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 Betrieb Ihres *ADwin-Gold*-Systems. Es wird ergänzt durch

- das Handbuch „*ADwin*-Installation“, das die Schnittstellen- Installation zu allen *ADwin*-Systemen beschreibt.
  Beginnen Sie hier die Installation Ihres Systems!
- die Beschreibung des Konfigurationsprogramms *ADconfig*, mit dem Sie die Kommunikation von der jeweiligen Schnittstelle (Interface) zur Ihrem *ADwin*-Gerät einrichten.
- das Handbuch *ADbasic*, das die Basisbefehle für den gleichnamigen Compiler enthält sowie das Funktionsprinzip von *ADwin*-Systemen näher erläutert.
- die Installations- und Befehlsbeschreibungen für die Treiber der gängigen Entwicklungsumgebungen.

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 IEC 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

Änderungen vorbehalten.

Hotline-Adresse siehe vordere Umschlagseite, innen.

2 Systembeschreibung

2.1 *ADwin* Systemkonzept

ADwin-Systeme garantieren den schnellen und zeitlich präzisen Ablauf von Messdatenerfassungs- und Automatisierungsaufgaben mit sehr schnellen Echtzeitanforderungen. Das bietet eine ideale Basis für Anwendungen wie:

– sehr schnelle digitale Regler

– sehr schnelle Steuerungen

– Datenerfassung mit sehr schneller Online-Analyse der Messdaten

– Überwachung komplexer Triggerbedingungen und vieles mehr

ADwin-Systeme sind optimiert für Abläufe mit **kurzen Prozesszykluszeiten** von einer Millisekunde bis zu wenigen Mikrosekunden.

Systemmerkmale

Das *ADwin*-System besitzt analoge und digitale Ein- und Ausgänge, einen schnellen Prozessor (32-Bit-Floating-Point Signalprozessor) und lokalen Speicher. Der Prozessor übernimmt die gesamte Echtzeitverarbeitung im System. Die Anwendungen **laufen eigenständig** und unabhängig vom PC und dessen Auslastung.

Prozessor

Der Prozessor des *ADwin*-Systems **verarbeitet** *jeden* **Messwert** *sofort*.

In einem Zyklus können die Zustände von Eingängen erfasst, diese mit beliebigen mathematischen Funktionen verarbeitet und auf dieses Ergebnis reagiert werden, und das sogar bei sehr kurzen Prozesszykluszeiten von wenigen Mikrosekunden. Es ergibt sich eine perfekte und logische Arbeitsteilung: auf dem PC läuft ein Programm zur Visualisierung von Daten, zur Eingabe und Bedienung der Abläufe mit Netzwerk- und Datenbankzugriffen, während gleichzeitig auf dem Prozessor des *ADwin*-Systems alle Aufgaben, die Echtzeit erfordern, abgearbeitet werden.

Echtzeitkern

Das Betriebssystem für den DSP des *ADwin*-Systems wurde auf das Erreichen kürzester Reaktionszeiten optimiert. Dieser Echtzeitkern verwaltet parallele Prozesse, die im **Multitasking-Verfahren** gleichzeitig ablaufen können. Prozesse mit niedriger Priorität werden in einem Zeitscheibenverfahren verwaltet. Prozesse mit hoher Priorität unterbrechen bei ihrer Anforderung alle niedrigpriorisierten Prozesse und werden sofort vollständig ausgeführt (präemptives Multitasking). Hochpriorisierte Prozesse werden zeitgesteuert oder von externen Events (Trigger) ausgelöst.

Zeitsteuerung

Für den präzisen Aufruf hochpriorisierter Prozesse sorgt der im System integrierte **Timer**. Er hat eine Auflösung von 25 Nanosekunden (3,3ns ab Prozessor T11). Zu beachten ist die extrem kurze Reaktionszeit von nur 300 Nanosekunden beim Wechsel von einem niedrig- zu einem hochpriorisierten Prozess. Ein ständig laufender Kommunikationsprozess ermöglicht einen kontinuierlichen Datenaustausch zwischen dem *ADwin*-System und dem PC auch während laufenden Anwendungen. Dabei hat die Kommunikation keinen Einfluss auf die Echtzeitfähigkeit des *ADwin*-Systems, trotzdem können jederzeit Daten ausgetauscht werden.

ADbasic

Das Echtzeit-Entwicklungstool *ADbasic* ermöglicht die einfache und schnelle Erstellung von zeitkritischen Programmen für *ADwin*-Systeme. *ADbasic* ist eine **integrierte Entwicklungsumgebung** unter Windows mit Möglichkeiten zum Online-Debugging. Die gewohnte, leicht erlernbare BASIC-Befehlssyntax wurde um Funktionen für den direkten Zugriff auf Ein- und Ausgänge sowie zur Prozesssteuerung und zur Kommunikation mit dem PC erweitert.

Die Kommunikation zwischen *ADwin*-System und PC

Schnittstellen

Das *ADwin*-System ist mit dem PC über eine **USB- oder Ethernet-Schnittstelle** verbunden. Über diese Schnittstelle kann das *ADwin*-System nach dem Einschalten vom PC gebootet werden. Nach dem Booten erwartet das *ADwin*-Betriebssystem Kommandos vom PC, die es abarbeitet.

Befehlsverarbeitung

Es gibt zwei Arten von Kommandos: Zum einen Kommandos, die nur Daten vom PC an das *ADwin*-System schicken, wie z.B. „Prozess laden“, „Prozess starten“ oder „Parameter setzen“, zum anderen Kommandos, die von dem *ADwin*-System eine Antwort erwarten, wie z.B. „Variablen lesen“ oder „Datensätze lesen“. Beide Arten von Kommandos werden vom *ADwin*-System sofort bearbeitet beziehungsweise sofort und vollständig beantwortet. Das *ADwin*-System schickt nie unaufgefordert Daten an den PC. Die Datenübertragung an den PC ist immer nur die Antwort auf ein Kommando vom PC. Dadurch wird die Einbindung des *ADwin*-Systems in die unterschiedlichsten Programmiersprachen und messtechnischen Standardsoftwarepakete sehr erleichtert, denn diese müssen nur in der Lage sein, eine Funktion aufzurufen und den Rückgabewert zu verarbeiten.

Software-Schnittstellen

Unter den aktuellen Windows-Versionen stehen eine **DLL-** und eine **ActiveX-Schnittstelle** zur Verfügung. Darauf basierend gibt es Treiber für die folgenden **Entwicklungsumgebungen**:
.NET, Visual Basic, Visual-C, C/C++, C#, Delphi, VBA (Excel, Access, Word), TestPoint, LabVIEW / LabWINDOWS, Agilent VEE (HP-VEE), InTouch, DIA*dem*, DASYLab, SciLab, MATLAB.

Treiber für Linux, Mac OS und Java stehen ebenfalls zur Verfügung.

Die einfache, kommandoorientierte Kommunikation mit dem *ADwin*-System ermöglicht es, dass mehrere Windows Programme in Abstimmung miteinander gleichzeitig auf das gleiche *ADwin*-System zugreifen. Dies ist vor allem bei der Programmentwicklung und bei der Inbetriebnahme ein großer Vorteil.

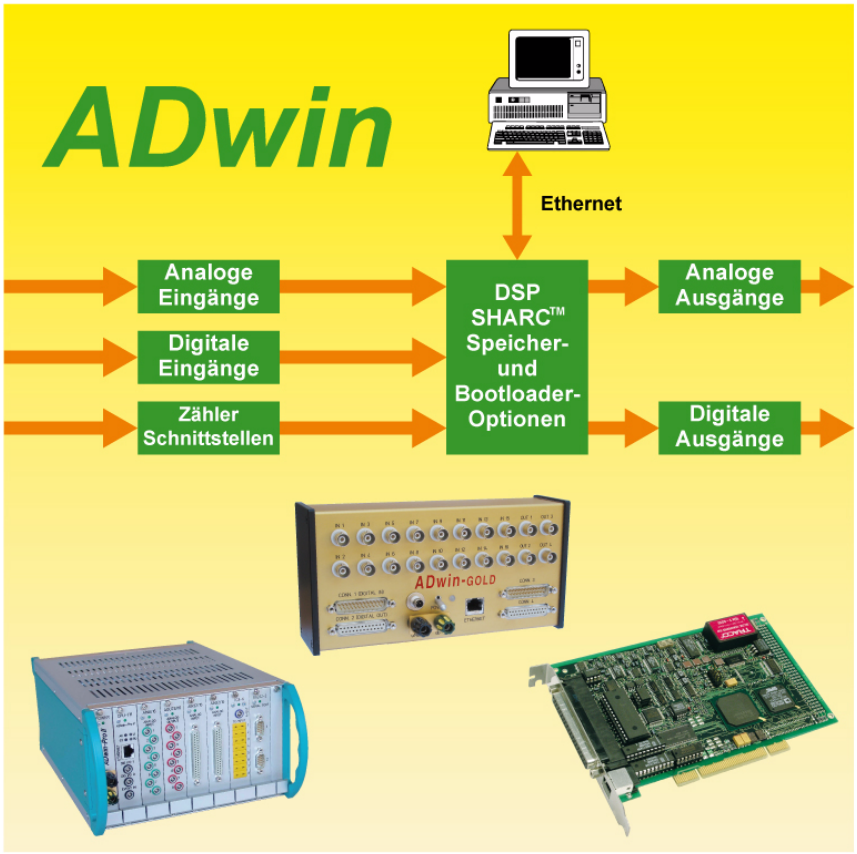


Abb. 1 – Konzept der ***ADwin***-Systeme

2.2 Das *ADwin-Gold*-System

Prozessor und Speicher

Das *ADwin-Gold*-System besitzt den digitalen 32 Bit-Signalprozessor T9 (SHARC ADSP 21062) von Analog Devices mit Floating-Point- und Integer-Verarbeitung. Er übernimmt die gesamte Messwerterfassung, Online-Verarbeitung und Signalausgabe und kann in Verbindung mit A/D-Wandlern jeden Messwert mit Abtastraten bis zu mehreren 100 kHz sofort verarbeiten.

Der interne Speicher mit 256 KiB hat eine sehr kurze Zugriffszeit von 25 ns und nimmt das komplette *ADwin*-Betriebssystem, die *ADbasic*-Prozesse und alle Variablen auf.

Für maximale Zugriffsgeschwindigkeiten liegen alle Ein- und Ausgänge direkt im Adressbereich des DSP. Zum Zwischenspeichern größerer Datenmengen benutzt der DSP einen externen Speicher von 16 MiB (DRAM; optional 64 MiB).

Analoge Eingänge

Das System hat 16 analoge Eingänge mit BNC- oder mit Sub-D-Buchsen, die in zwei Gruppen jeweils mit einem Multiplexer verbunden sind. Die Multiplexer-Ausgänge werden wahlweise mit dem 14 Bit oder dem 16 Bit Analog-Digital-Wandler (ADC) konvertiert (siehe Abb. 2 „Funktionsschema des ADwin-Gold“). Mit dem 14 Bit-ADC kann sehr schnell, mit dem 16 Bit-ADC sehr genau gemessen werden.

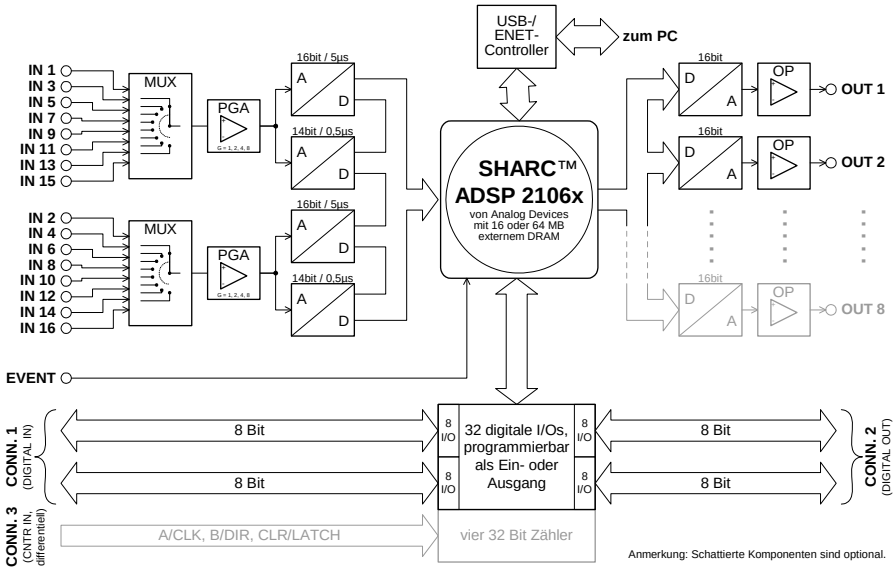


Abb. 2 – Funktionsschema des *ADwin-Gold*

Analoge Ausgänge

In der Standardversion verfügt das ADwin-Gold über 2 analoge Ausgänge (optional 8) mit 16 Bit Auflösung und einem Ausgangsspannungsbereich von -10 V…+10 V. Per Software können Sie die Ausgabe der Spannung aller DAC synchronisieren.

Alle analogen Daten-Eingänge und -Ausgänge des Geräts sind differentiell.

Digitale Ein- und Ausgänge

Auf zwei 25-poligen Sub-D-Anschlüssen stehen 32 digitale Ein- oder Ausgänge zur Verfügung. Sie sind in Gruppen zu jeweils 8 als Ein- oder Ausgang frei programmierbar. Die Ein- bzw. Ausgänge sind TTL-kompatibel.

Trigger-Eingang (EVENT)

Das *ADwin-Gold*-System besitzt einen Trigger-Eingang (EVENT, siehe auch Kapitel 5.3 „Digitale Ein- und Ausgänge und Event-Eingang“). Hiermit können Prozesse durch ein Signal (Trigger) ausgelöst und sofort vollständig abgearbeitet werden (siehe *ADbasic*-Handbuch, Kapitel „Struktur des ADbasic-Programms“).

Standard-Lieferumfang

Der Standard-Lieferumfang des *ADwin-Gold*-Systems umfasst

- das *ADwin-Gold*-Gerät mit USB- oder Ethernet-Schnittstelle, mit BNC- oder Sub-D-Buchsen,
- ein USB-Kabel oder ein „Cross-over“ Ethernet-Kabel vom PC zum Gold-Gerät (Länge ca. 1,8 m),
- Power-Adapter: Ein dreipoliges, verpolungssicheres Stromversorgungs-Kabel an einem Slotblech mit Steckbuchse,
- Stromversorgungskabel vom Power-Adapter zum System,
- *ADwin*-CD,
- Handbuch „ADwin-Installation“,
- das vorliegende Hardware-Handbuch.

2.2.1 Gerätevarianten und Bestelloptionen

Basisvarianten

ADwin-Gold gibt es in vier Basisvarianten:

	Analogkanäle mit BNC-Buchsen	Analogkanäle mit Sub-D-Buchsen
mit Ethernet-Schnittstelle (10/100 MBit/s)	*Gold-ENET*	*Gold-D-ENET*
mit USB-Schnittstelle	*Gold-USB*	*Gold-D-USB*

Zusatzoptionen

Folgende Zusatzoptionen können mitbestellt (aber nicht nachgerüstet) werden:

- *Gold-DA*: 6 zusätzliche analoge Ausgänge (differentiell), jeweils mit einem 16 Bit DAC.
- *Gold-CO1*: 4 Stück 32 Bit-Zähler, wahlweise zur Periodendauermessung, als Impulszähler, als Vorwärts-/Rückwärtszähler mit Takt-/Richtung oder als Vier-Flanken-Auswertung für Inkremental-Enkoder.
- *Gold-CAN*: 4 Decoder zum Anschluss von Inkremental-Encodern mit SSI-Schnittstelle, 2 CAN-Schnittstellen (je nach Bestellung nur „High-speed“ oder nur „Low-speed“) sowie 2 RSxxx-Schnittstellen (RS232, RS485).

  Diese Option ist nur in Kombination mit der Variante *Gold-D* erhältlich.
- *GOLD-MEM-64*: Externer Speicher mit 64 MiB anstatt 16 MiB sowie interner CPU-Speicher mit 512 KiB anstatt 256 KiB.
- *Gold-Boot*: Flash-EPROM-Bootloader zum eigenständigen Betrieb ohne PC (**nur** in Verbindung mit *Gold-ENET* oder *Gold-D-ENET*).

Sofern nicht explizit ausgeschlossen, sind alle Zusatzoptionen miteinander kombiniert lieferbar.

2.2.2 Zubehör

- *ADbasic*, Echtzeit-Entwicklungsumgebung für alle *ADwin*-Systeme.
- *ADwin-Gold*-pow: externes Netzteil (u.a. erforderlich für Notebook-Betrieb).
- *Gold-Mount*: Gehäuseumbau zur Hutschienen-Montage in einem Schaltschrank mit isolierten Clipsen.
- Einzelner Stromversorgungs-Stecker für ein selbst-konfektioniertes Stromversorgungs-Kabel.

3 Betriebliche Umgebung

Die *ADwin-Gold*-Elektronik ist in einem geschlossenen Aluminiumgehäuse untergebracht, und das System darf nur in diesem Zustand betrieben werden. Mit entsprechendem Zubehör ist die Unterbringung in Schaltschränken oder der mobile Betrieb (z.B. im Kfz) möglich (siehe Kapitel 2.2.2 „Zubehör“).

Erdung

Das *ADwin-Gold*-Gerät **muss geerdet werden**, um

– einen Massebezugspunkt für die Elektronik herzustellen und

– Störungsenergie auf die Erde ableiten zu können.

Verbinden Sie dazu die GND-Buchse, die intern mit der Masse und dem Gehäuse verbunden ist, über ein kurzes impedanzarmes Masseband mit dem zentralen Erdungspunkt Ihrer Anlage.

Galvanische Kopplung

Das Stromversorgungskabel vom Power-Adapter stellt eine galvanische Verbindung zwischen dem PC und dem ADwin-Gold-Gerät her.

Die Liefervariante mit USB-Schnittstelle hat über diese eine galvanische Verbindung zum PC sowie ggf. über die Stromversorgung.

Bei der Liefervariante mit Ethernet-Schnittstelle sind die Datenleitungen galvanisch entkoppelt, die Massepotenziale sind jedoch gekoppelt, weil die Schirmung des Ethernet-Steckers (RJ-45) mit GND verbunden ist.

Ausgleichströme ausschließen

Ausgleichsströme, die über das Gehäuse oder die Schirmung abfließen, beeinflussen das Messsignal.
Wenn Sie Ausgleichströme vermindern wollen, müssen Sie darauf achten, dass die Wirkung des Schirmes erhalten bleibt, indem Sie geeignete Maßnahmen zur Ableitung von Störungen treffen, wie z. B. das Auflegen des Schirms kurz vor dem Eintritt in den Schaltschrank. Je häufiger Sie die Schirmung auf dem Weg zur Maschine erden, desto besser ist die Schirmwirkung.

Verwenden Sie für die **Signalleitungen** Kabel mit beidseitig aufgelegtem Schirm. Auch hier sollte das Ableiten von Störungen über das Gehäuse mit der Verwendung von Schirmklemmen reduziert werden.

BNC-Kabel

Die Abschirmung von BNC-Kabeln wird üblicherweise als differentielle Masse verwendet und verliert dadurch an Schirmwirkung. Daher sind BNC-Kabel bei differentiellen Messungen Störeinflüssen ausgesetzt. Für die Signal- und Datenübertragung außerhalb des Schaltschranks ist eine Umsetzung auf Datenübertragungskabel erforderlich, die paarig verdrillt (twistet pair) und kanalweise geschirmt sind.

Schutzkleinspannung

Das *ADwin-Gold*-System wird extern mit einer Schutz-Kleinspannung von 10 V bis 35 V versorgt; intern wird es mit einer Spannung von +5 V und ±15 V gegen GND betrieben. Es stellt von dieser Seite keine Gefahr für Leib und Leben dar. Für den Betrieb mit einem externem Netzteil gelten die Angaben des Herstellers.

Umgebungsklima

ADwin-Gold ist für den Betrieb in trockenen Räumen konzipiert. Am Einbauort sollen eine Umgebungstemperatur von +5 °C … +50 °C und eine relative Luftfeuchte von 0 … 80 % (nicht kondensierend, siehe Anhang) vorhanden sein.

Gehäusetemperatur

Die Gehäusetemperatur (Oberflächentemperatur) darf auch unter extremen betrieblichen Bedingungen, z.B. im Schaltschrank oder bei direkter Sonneneinstrahlung, +60 °C nicht überschreiten. Es besteht sonst die Gefahr, dass Schäden am Gerät entstehen oder nicht definierte Daten (Werte) ausgegeben werden, die unter ungünstigen Umständen zu Schäden in ihrer Anlage führen können.

4 Inbetriebnahme der Hardware

Schließen Sie bei der Inbetriebnahme keine Kabel an das *ADwin-Gold*-Gerät an, bevor Sie nicht folgende Schritte durchgeführt haben:

1. Führen Sie die Installation der Treiber und Stromversorgung am PC oder Notebook vollständig durch (siehe Handbuch „*ADwin*-Installation“).

   Weitere Hinweise zur Stromversorgung finden Sie in Kapitel 5.1.
2. Folgen Sie den Hinweisen im Kapitel 3 „Betriebliche Umgebung“.
3. Lesen Sie das Kapitel 5 „Ein- und Ausgänge“ in diesem Handbuch.
4. Stellen Sie sicher, dass
   - die Datenverbindung zum PC/Notebook über das USB- oder Ethernet-Kabel aufgebaut ist
   - die Stromversorgung zum *ADwin-Gold* aufgebaut ist
     Beachten Sie die unten stehenden Hinweise zu Spannungs- und Stromversorgung.
   - Ihr Rechner eingeschaltet ist
5. Schalten Sie das *ADwin-Gold*-Gerät ein.

   Einschalten

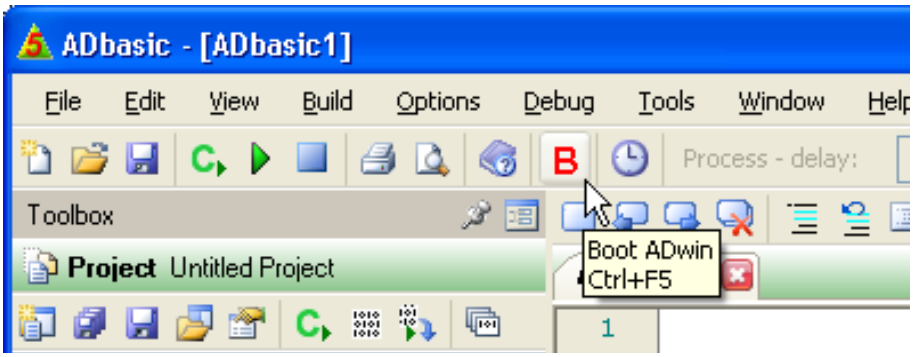
   Um versehentliches Ausschalten zu verhindern, besitzt der Ein-/Ausschalt-Hebel eine Umschaltsperre.
   Ziehen Sie zum Einschalten den Hebel leicht heraus (ca. 1,5mm) und legen ihn in Richtung „`Power`“ um. Damit ist das Gerät eingeschaltet und die LED leuchtet rot auf.
6. Beginnen Sie erst jetzt mit dem Anschluss von Ein- und Ausgängen.

Halten Sie die Reihenfolge ab Punkt 4 jedes Mal ein, wenn Sie das Gerät einschalten.

Nun können Sie beginnen, das *ADwin-Gold*-System zu programmieren. Nehmen Sie auch die Einstellungen im *ADbasic*-Menü „`Options\Compiler`“ vor.

Booten

Starten Sie *ADbasic* und booten das *ADwin*-System durch Anklicken der Boot-Schaltfläche `B`.



Das Blinken der LED (jetzt grün) und die Anzeige in der Statuszeile: „`ADwin is booted`“ zeigen an, dass das Betriebssystem richtig geladen ist und *ADbasic* eine Verbindung zum *ADwin*-System herstellen kann (wenn nicht, überprüfen Sie zuerst die Anschlüsse).

Programme mit *ADbasic*

Die Programmierung von *ADwin*-Systemen ist im *ADbasic*-Handbuch ausführlich beschrieben. *ADbasic*-Befehle für den I/O-Zugriff sind in Kapitel 12 auf Seite 46 beschrieben.

Beginnen Sie mit Programmbeispielen aus dem *ADbasic*-Tutorial.

Anschlüsse

5 Ein- und Ausgänge

Alle Ein- und Ausgänge dürfen nur im Bereich der angegebenen Spezifikation betrieben werden (siehe Anhang A.1 Technische Daten). Im Zweifel wenden Sie sich bitte an den Hersteller des Gerätes, das Sie an das *ADwin-Gold*-System anschließen wollen.

Offene Eingänge können zu Fehlern führen – vor allem in einer nicht störungsfreien Umgebung. Zu Ihrer Sicherheit legen Sie nicht benutzte Eingänge möglichst nah an Stecker oder Buchse des *ADwin-Gold* auf einen definierten Pegel (z.B. GND). Schließen Sie keine Kabel mit offenem Ende an die Eingänge an; dies kann Störimpulse an den Eingängen verursachen.

Die Ein- und Ausgänge der Basisversion *ADwin-Gold* sind auf den folgenden Seiten beschrieben:

– Strom-Eingangsstecker (Seite 10)
– 16 analoge Eingänge über 2 Multiplexer (Seite 10)
– 2 analoge Ausgänge (Seite 12)
– 32 digitale Ein- oder Ausgänge (Seite 14)

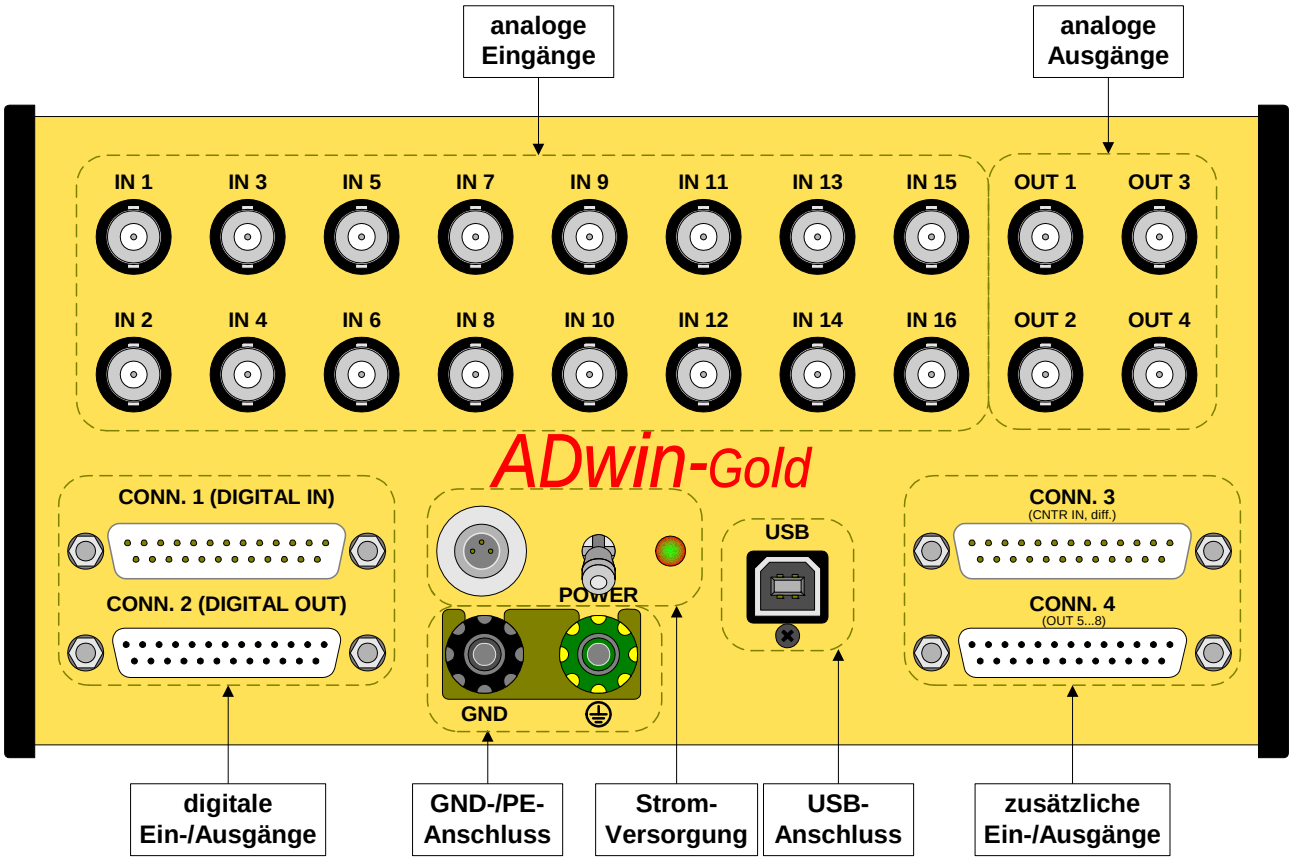


Abb. 3 – Übersichtsbild *ADwin-Gold-USB*

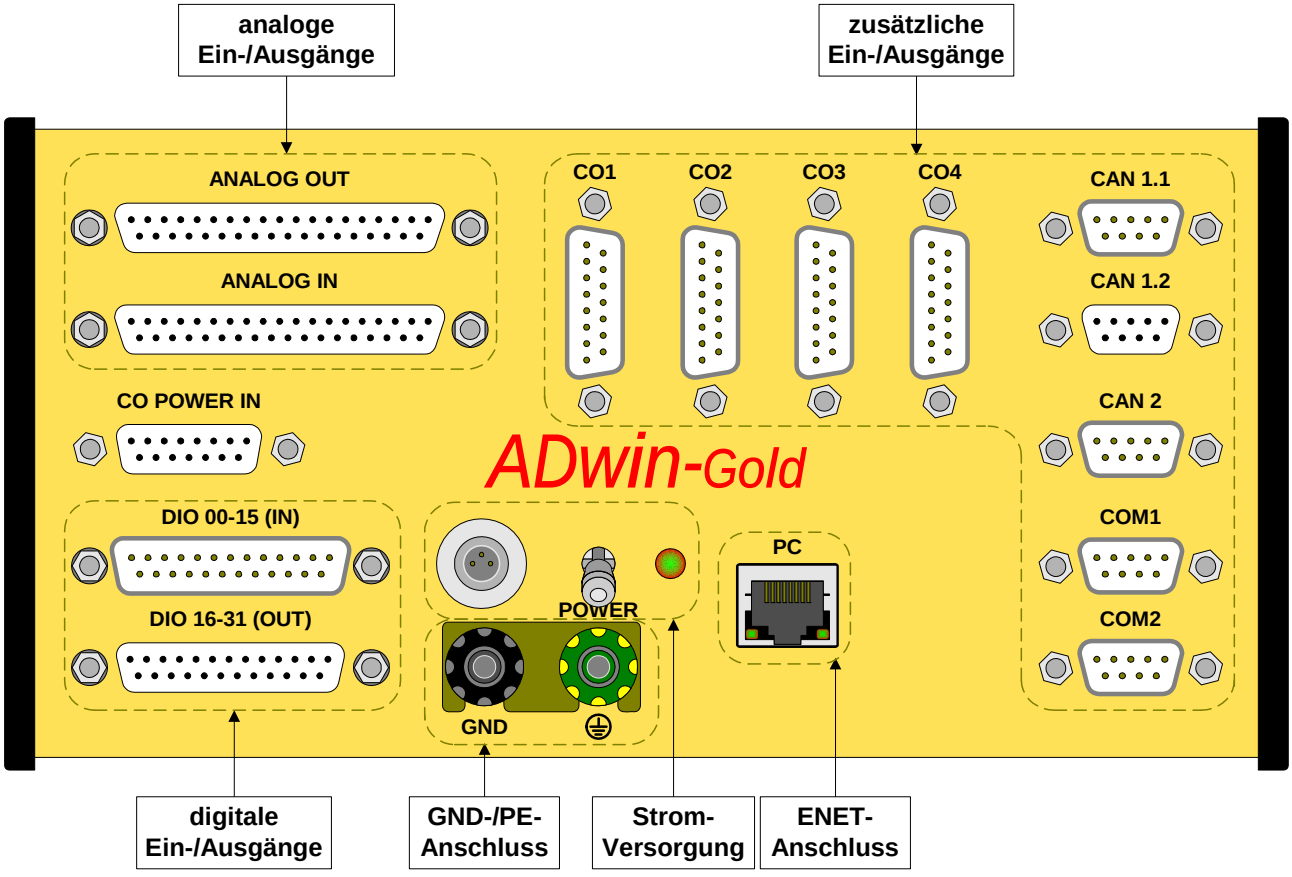


Abb. 4 – Übersichtsbild *ADwin-Gold-D-ENET*

5.1 Stromversorgung

Stromversorgung

Die Stromversorgung des ADwin-Gold (siehe Anhang, Technische Daten) erfolgt über den Einbaustecker links neben dem Power-Schalter (Pinbelegung siehe Abb. 5). Schließen Sie dort einen 3-poligen Subminiatur-Rundsteckverbinder an; die Bezugsadresse finden Sie im Anhang, Abschnitt A.4.

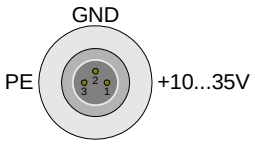


Abb. 5 – Stromversorgungsstecker (männlich)

Beim Betrieb mit einem Notebook muss die Stromversorgung durch ein separates Netzteil erfolgen (siehe Kapitel 2.2.2 auf Seite 5). Bitte beachten Sie, dass das Netzteil ausreichend dimensioniert ist.

Achten Sie bei der Verwendung strombegrenzender Netzteile darauf, dass beim Einschalten der Strombedarf ein Mehrfaches des Betriebsstroms betragen kann. Genaue Angaben finden Sie bei den Technischen Daten (Anhang).

Bei **Ausfall der Betriebsspannung** gehen alle Daten im *ADwin-Gold* verloren. Nicht definierte Daten (Werte) können unter ungünstigen Umständen zu Schäden in Ihrer Anlage führen.

Sicherstellen der Spannungsversorgung

Achten Sie auf eine zuverlässige Spannungsversorgung.
Im Standardlieferumfang betrifft das den PC, ansonsten auch das externe Netzteil, bei Betrieb in einem Fahrzeug die Batteriespannung.

5.2 Analoge Ein- und Ausgänge

Für störungsfreien Betrieb sind bei den Hardware-Varianten mit BNC-Buchsen isolierte BNC-Stecker erforderlich. Es besteht ansonsten die Gefahr von Schäden durch elektrostatische Entladungen und Kurzschlüssen an den Eingängen. Das gilt vor allem bei Verwendung von nicht isolierten BNC-T-Stücken.

Das *ADwin-Gold*-Gerät muss geerdet werden, um Messungen störungsfrei durchführen zu können. Verbinden Sie dazu die GND-Buchse über ein impedanzarmes Masseband mit dem zentralen Erdungspunkt Ihrer Anlage.

Die Spannungsversorgung vom Power-Adapter am PC verbindet auch die Erdung des *ADwin-Gold*-Systems mit der Erdung des PC. Wenn Sie den PC und das System nicht am selben Ort betreiben, können unterschiedliche Massepotenziale am *ADwin-Gold* und am Messobjekt bzw. den Messleitungen Störungen verursachen. Vermeiden Sie solche Einflüsse, indem Sie ein externes Netzteil benutzen.

Neben der Beschreibung der Ein- und Ausgänge finden Sie nachfolgend Hinweise zur Umrechnung zwischen Digits und analogen Spannungswerten und zur Eingangsbeschaltung der analogen Eingänge.

Standardbefehle

Für eine schnelle und einfache Programmierung gibt es im Compiler *ADbasic* Standardbefehle, die **einfaches Messen bzw. Ausgeben von Daten** ermöglichen (siehe ADC oder DAC, Seite 48ff). Verwenden Sie andere Befehle (z.B. für direkten Registerzugriff) erst, wenn extrem zeitkritische oder besondere Aufgaben es erfordern (siehe auch Handbuch *ADbasic*).

5.2.1 Analoge Eingänge

Multiplexer

ADwin-Gold hat 16 analoge Eingänge IN1 … IN16. Die Eingänge mit ungeraden Zahlen (1, 3, … 15) sind dem Multiplexer 1, diejenigen mit geraden Zahlen (2, 4, … 16) sind dem Multiplexer 2 zugeordnet. Der Ausgang jedes Multiple-

xers ist mit je einem 14 Bit-ADC und einem 16 Bit-ADC verbunden (siehe auch Funktionsschema des ADwin-Gold, Seite 4).

16 Bit- und 14 Bit-Messung

Sie können die Signale an den Multiplexer-Ausgängen wahlweise mit einem 14 Bit oder mit einem 16 Bit Analog-Digital-Wandler (ADC) konvertieren (siehe Abb. 2 „Funktionsschema des ADwin-Gold“). Sie messen mit

– dem 14 Bit-ADC sehr schnell (max. 0,5 µs, Auflösung 1,221 mV)

– dem 16 Bit-ADC sehr genau (max. 5 µs, Auflösung 305 µV).

Differentiell

Die analogen Eingänge sind differentiell. Für jeden Messkanal sind je ein Plus- und ein Minuseingang vorhanden, zwischen denen die Spannungsdifferenz gemessen wird (jedoch nicht potenzialfrei). Für jeden Kanal müssen Plus- und Minuseingang angeschlossen werden.

– BNC-Buchsen (Gold)

  Die Eingänge sind mit männlichen BNC-Buchsen bestückt, die in zwei Reihen angeordnet sind. Der Innenleiter ist der Plus-Eingang, der Außenleiter der Minus-Eingang.

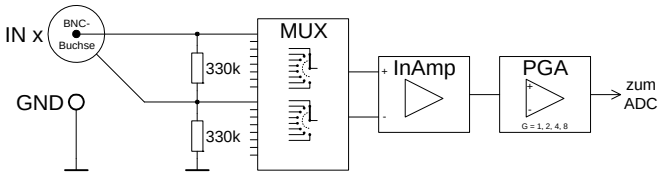
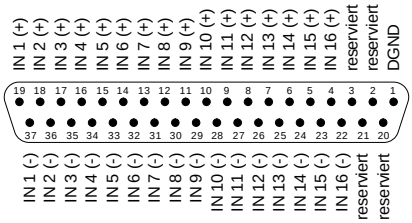


Abb. 6 – Eingangsbeschaltung eines analogen Eingangs

– Sub-D-Buchse (Gold-D)

  Die Eingänge sind auf die Sub-D-Buchse ANALOG IN gelegt. Die Eingangsbeschaltung ist die gleiche wie bei der Variante mit BNC-Buchsen.



ANALOG IN

Abb. 7 – Pin-Belegung der Analogeingänge für Variante Gold-D

Beachten Sie, dass zusätzlich zu Plus- und Minuseingang des Kanals immer eine Masseverbindung zwischen dem System (GND-Buchse) und der Signalquelle bestehen muss.

Achten Sie auf einen möglichst geringen Ausgangswiderstand der Spannungsquelle für die Eingangssignale, denn dieser kann die Messgenauigkeit beeinflussen. Falls dies nicht möglich ist:

– Abhängig vom Ausgangswiderstand der Spannungsquelle wird ein linearer Messfehler erzeugt.
  Sie können dies ausgleichen, indem Sie den Messwert mit einem entsprechenden Faktor multiplizieren und dadurch sozusagen „nachkalibrieren“.

– Ab etwa 3 kΩ Ausgangswiderstand aufwärts verlängert sich zusätzlich die Einschwingzeit des Multiplexers.
  Die in den Standard-Befehlen **ADC** und **ADC12** definierte Einschwingzeit

ist dann zu kurz, so dass zu früh ungenaue Werte abgerufen werden. Verwenden Sie für diesen Fall die in Kapitel 5.4.1 beschriebenen Befehle.

ADC-Befehl

Die Befehle **ADC**( ) für den 16 Bit-ADC und **ADC12**( ) für den 14 Bit-ADC führen mit einem der ADC eine komplette Messung auf einem analogen Eingang durch (siehe Seite 49). So berücksichtigen diese Befehle z.B. die Einschwingzeit des Multiplexers und stellen einwandfreie Messungen sicher.

Programmieren

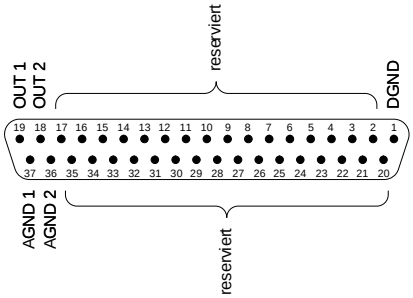
Befehle zur Programmierung der analogen Eingänge sind ab Seite 47 beschrieben. Die Befehle werden auch in der Online-Hilfe erläutert.

Bereich	Befehle
vollständige Messung durchführen	**ADC**, **ADC12**
Messung in Teilschritten durchführen (siehe Kapitel 5.4)	**Set_Mux** **Start_Conv**, **Wait_EOC** **ReadADC**, **ReadADC12**

5.2.2 Analoge Ausgänge

Das System hat 2 analoge Ausgänge (OUT1, OUT2). Den Ausgängen ist je ein eigener Digital-Analog-Wandler (DAC) zugeordnet.

Bei der Variante Gold liegen die Ausgänge auf BNC-Buchsen, bei der Variante Gold-D auf der Sub-D-Buchse ANALOG OUT (siehe Abb. 7).



ANALOG OUT

Abb. 8 – Pin-Belegung der Analogausgänge für Variante Gold-D

Zusätzliche Ausgänge siehe Kapitel 7 „DA-Erweiterung“.

Programmieren

Der Standardbefehl **DAC**(Nummer,Wert) (siehe Seite 48) prüft jeden Wert auf die Über- und Unterschreitung des 16 Bit-Wertebereiches (0…65535). Liegt der Wert innerhalb dieses Bereiches, wird der angegebene Wert auf dem Ausgang Nummer ausgegeben. Liegt er außerhalb, wird der Maximal- bzw. Minimalwert ausgegeben.

5.2.3 Berechnungsgrundlagen

Spannungsbereich

Das *ADwin-Gold*-System arbeitet bei den analogen Ein- und Ausgängen mit einem Spannungsbereich von –10 V bis +10 V (= bipolar 10 V).

Zuordnung von Digits zu Spannung

Die 65536 (2^{16}) Digits sind den jeweiligen Spannungsbereichen der ADC und DAC so zugeordnet, dass

– 0 (Null) Digit der maximalen negativen Spannung und

– 65535 Digit der maximalen positiven Spannung

entspricht.

Der Wert für 65536 Digit, genau 10 Volt, liegt gerade *außerhalb* des Messbereichs, womit sich für die 16 Bit-Wandlung ein maximaler Spannungswert von 9,999695 Volt und für die 14 Bit-Wandlung von 9,998779 Volt ergibt.

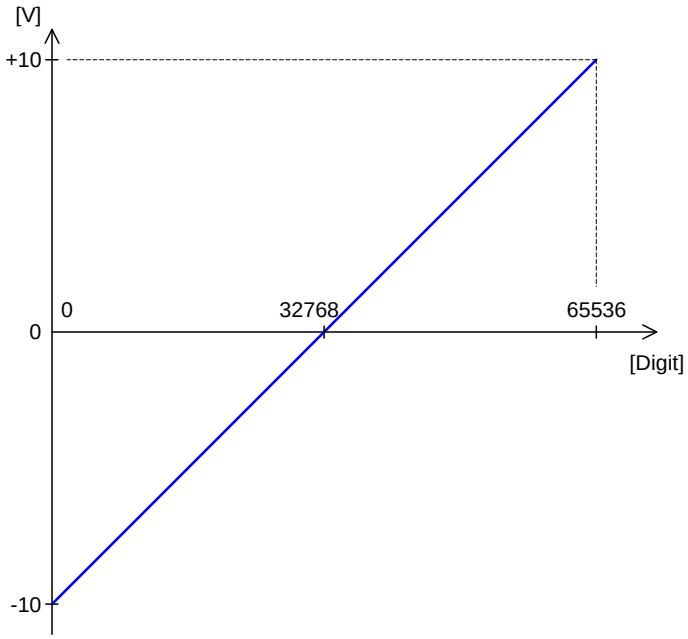


Abb. 9 – Nullpunktverschiebung bei Standardeinstellung bipolar 10 Volt

Nullpunktverschiebung U_{OFF}

Die bipolare Einstellung führt zu einer Nullpunktverschiebung, die im folgenden auch als Offset U_{OFF} bezeichnet wird.
Beim Spannungsbereich –10 V … +10 V gilt:

$U_{OFF} = -10\,V$

Verstärkungsfaktor k_V

Das *ADwin-Gold*-System besitzt einen programmierbaren Verstärker (PGA), mit dem Sie die Eingangsspannung um die Faktoren 1, 2, 4, und 8 verstärken können. Gleichzeitig verkleinert sich damit der Messbereich um den jeweiligen Verstärkungsfaktor k_V (siehe Anhang „Technische Daten“).

Beachten Sie bei Anwendungen mit $k_V > 1$, dass auch die Störsignale entsprechend mit verstärkt werden.

Quantisierungsstufe U_{LSB}

Die Quantisierungsstufe U_{LSB} ist die kleinste digital darstellbare Spannungsdifferenz und ist gleich der Spannung des niederwertigsten Bit (Least Significant Bit, LSB). Sie ist für die beiden ADC unterschiedlich:

– 16 Bit-ADC: $U_{LSB} = 20\,V / 2^{16} = 305{,}175\,\mu V$

– 14 Bit-ADC: $U_{LSB} = 20\,V / 2^{14} = 1220{,}7\,\mu V$

Der gemessene 16 Bit-Wert des ADC wird im unteren Wort der Speicherzelle zurückgeliefert. Dort muss sich auch ein auszugebender DAC-Wert befinden.

Bit-Nr.	31…16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
32 Bit-Speicher	0	16 Bit-Wert des 16 Bit-ADC / DAC im unteren Wort															
	0	14 Bit-Wert des 14 Bit-ADC im unteren Wort														0	0

Abb. 10 – Ablage der ADC/DAC-Bits im Speicher

Um Messwerte des 14 Bit ADC und des 16 Bit ADC einfach vergleichen zu können, wird der gewandelte Wert beim 14 Bit ADC linksbündig in das untere Wort der Speicherzelle geschrieben. Die untersten 2 Bits sind daher stets 0 (Null).

Die 16384 Digits des 14 Bit ADC werden auf die 65536 Digits des 16 Bit ADC abgebildet. Damit entsprechen je 4 Digits des 16 Bit ADC einem Digit des 14 Bit ADC.

Die folgenden Gleichungen gelten somit für beide ADC-Typen:

Umrechnung Digit in Spannung

DAC

Für einen DAC gilt:

$$U_{OUT} = Digits \cdot U_{LSB} + U_{OFF}$$

$$Digits = \frac{U_{OUT} - U_{OFF}}{U_{LSB}}$$

ADC

Für einen ADC (14 Bit und 16 Bit) gilt:

$$Digits = \frac{k_V \cdot U_{IN} - U_{OFF}}{U_{LSB}}$$

$$U_{IN} = \frac{Digits \cdot U_{LSB} + U_{OFF}}{k_V}$$

Toleranzbereiche

Geringe Abweichungen zu den rechnerischen Werten können innerhalb der Toleranzbereiche einzelner Bauteile liegen. Es gibt zwei charakteristische Abweichungsarten, die in diesem Handbuch angegeben sind (in LSB):

INL

– Die integrale Nicht-Linearität (INL) beschreibt die maximale Abweichung von der Geraden über den gesamten Eingangsspannungsbereich.

DNL

– Die differentielle Nicht-Linearität (DNL) beschreibt die maximale Abweichung von der Breite einer Quantisierungsstufe.

5.3 Digitale Ein- und Ausgänge und Event-Eingang

Digitale Ein- / Ausgänge

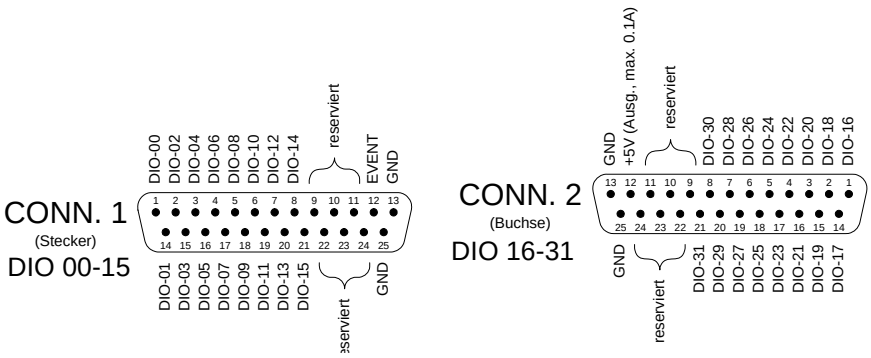
Auf den zwei 25-poligen Sub-D-Buchsen (`CONN. 1` und `CONN. 2`) stehen 32 digitale Kanäle `DIO 00`...`DIO 31` zur Verfügung.
Die Kanäle sind in Gruppen zu jeweils 8 als Ein- oder Ausgang programmierbar. Nach dem Einschalten des Gerätes sind alle 4 Anschlussgruppen als Eingang konfiguriert.

Trigger-Eingang (EVENT)

ADwin-Gold besitzt einen externen Trigger-Eingang (EVENT) auf der Sub-D-Buchse `CONN. 1`.

Ein externes Signal (Trigger) mit steigender Flanke an diesem Eingang kann Prozesse aufrufen, die sofort und vollständig abgearbeitet werden (siehe auch *ADbasic*-Handbuch, Kapitel: „Programmaufbau“).

Der Event-Eingang besitzt einen internen Pull-down-Widerstand (10 kΩ).



DIO-00…DIO-15 DIO-16…DIO-31

Abb. 11 – Pin-Belegung der Digitalkanäle

Die digitalen Eingänge sind TTL-kompatibel und gegen Überspannung nicht geschützt.
Beschalten Sie keine freien Anschlüsse, die als „reserviert“ gekennzeichnet sind. Diese sind Änderungen oder Erweiterungen vorbehalten; Nichtbeachten kann das System beschädigen.

Programmieren

Befehle zur Programmierung der digitalen Ein-/Ausgänge sind ab Seite 59 beschrieben:

Funktion	Befehle
Konfigurieren	**Conf_DIO**
Eingangswerte lesen	**Digin**, **Digin_Word**
Ausgangswerte setzen	**Digout_Word**, **Set_Digout** **Clear_Digout**

Beachten Sie die Hinweise zum Programmieren im folgenden Abschnitt.

5.3.1 Programmzugriff auf digitale Ein- und Ausgänge

Conf_DIO(12)

Der Befehl **Conf_DIO(12)** konfiguriert DIO 15:00 als digitale Eingänge und DIO 31:16 als digitale Ausgänge.

Nur in der Konfiguration **Conf_DIO(12)** können Sie mit den Standard-Befehlen in vollem Umfang auf die Ein- und Ausgänge zugreifen. Über die Programmierung bei anderen Konfigurationen informiert Sie das folgende Kapitel 5.3.1 „Programmzugriff auf digitale Ein- und Ausgänge“ (siehe auch Tutorial).

Die Befehle zum Lesen und Setzen der Kanäle funktionieren

Digout_Word, **Clear_Digout**, **Set_Digout**, **Digin_Word**, **Digin**

Nach dem Einschalten des Gerätes sind alle 4 Anschlussgruppen als Eingang konfiguriert; dies entspricht dem Befehl **Conf_DIO(0)**. Die folgende Tabelle zeigt, wie Ein- und Ausgänge (IN, OUT) konfiguriert werden, wenn Sie den Wert der ersten Spalte als Befehlsargument verwenden.

Conf_DIO()	DIO31:24	DIO23:16	DIO15:08	DIO07:00
0	IN	IN	IN	IN
1	IN	IN	IN	OUT
2	IN	IN	OUT	IN
3	IN	IN	OUT	OUT
4	IN	OUT	IN	IN
5	IN	OUT	IN	OUT
6	IN	OUT	OUT	IN
7	IN	OUT	OUT	OUT

Conf_DIO()	DIO31:24	DIO23:16	DIO15:08	DIO07:00
8	OUT	IN	IN	IN
9	OUT	IN	IN	OUT
19	OUT	IN	OUT	IN
11	OUT	IN	OUT	OUT
12	OUT	OUT	IN	IN
13	OUT	OUT	IN	OUT
14	OUT	OUT	OUT	IN
15	OUT	OUT	OUT	OUT
Anwendbare Befehle:	Digout_Word, Clear_Digout, Set_Digout		Digin_Word, Digin	
Befehl ist anwendbar für den Anschluss DIOnn, bei	Einstellung „OUT“		Einstellung „IN“ Bei Einstellung „OUT“ wird der Register-Inhalt dieses Bytes zurückgelesen	

Abb. 12 – Übersicht der Konfigurationen mit **Conf_DIO**

Beachten Sie folgende Einschränkung:

Nur bei Konfiguration der Ein-/Ausgänge mit dem Befehl **Conf_DIO(12)** (entsprechende Pinbelegung siehe Seite 15) können Sie mit den folgenden Befehlen im vollem Umfang auf die Kanäle zugreifen
Digin, **Digin_Word**, **Digout_Word**, **Set_Digout**, **Clear_Digout**.

Für jede andere Konfiguration müssen Sie das entsprechende Hardware-Register auslesen oder beschreiben (siehe Befehle **Peek** und **Poke** im Handbuch *ADbasic*). Die Hardware-Adressen für den Registerzugriff sind im Anhang dargestellt.

5.4 Zeitkritische Aufgaben

Für extrem zeitkritische Aufgaben können Sie Befehle einsetzen, mit denen Sie direkt auf die *Steuer- und Datenregister der ADC und DAC* zugreifen (siehe Befehle **Peek** und **Poke** im *ADbasic*-Handbuch). Diese Register liegen im Speicheradress-Bereich des ADSP (memory mapped). Die Befehle ermöglichen auch eine Optimierung der Programmstruktur (s.u.).

Im Gegensatz zu den Standardbefehlen **ADC**( ), **ADC12**( ) und **DAC**( ) besitzen die Befehle für den Direktzugriff *keine Prüfroutinen*. Vor der Benutzung sollten Sie sich deshalb genaue Kenntnisse über Zeitabläufe, Programmstrukturen und Funktionsabläufe in einem ADC aneignen.

5.4.1 Analoge Ein- und Ausgänge

ADC( ) und ADC12( )

Die Standardbefehle **ADC**( ) und **ADC12**( ) bestehen aus einer Sequenz von mehreren Befehlen (im folgenden dargestellt, siehe auch Seite 49ff) und benötigen eine werkseitig festgelegte Zeit zur Ausführung. Die Ausführungszeit wird vor allem durch die Einschwingzeit des Multiplexers und die Wandlungszeit bestimmt.

```
Set_Mux( )
...                          'Einschwingzeit abwarten
Start_Conv( )
Wait_EOC( )                  'Auf Wandlungsende warten
Read_ADC( )                  'bzw. READ_ADC12() bei ADC12()
```

Programmstruktur

Sie können die im Standardbefehl enthaltenen Wartezeiten durch Verwendung der Einzelbefehle für andere Zwecke nutzen (oder ggf. auch verlängern). Bei geschicktem Einsatz der Befehle können Sie dadurch schnellere Messvorgänge realisieren.

Es ist wichtig, den **Start_Conv**( ) Befehl in ausreichendem Zeitabstand vom **Set_Mux**( ) Befehl zu setzen, um die Einschwingzeit des Multiplexers zu berücksichtigen.

Nutzen Sie die entstehenden Wartezeiten, z.B. für Rechenoperationen, und sparen Sie somit Rechenzeit ein:

- Einschwingzeit des Multiplexers: Diese beträgt beim maximalen Spannungssprung von 20 Volt für den 16 Bit ADC höchstens 6,5µs und für den 14 Bit ADC höchstens 2,5µs.
- Wandlungszeit der ADC: Sie beträgt beim 14 Bit ADC 0,5µs und beim 16 Bit ADC 5µs.

Direkter Registerzugriff

ADC

Eine Messung kann sehr schnell ausgeführt werden, wenn Sie direkt auf die Steuer- und Datenregister der ADC zugreifen.

DAC

Ist bei den analogen Ausgängen sichergestellt, dass die Werte innerhalb der Bereichsgrenzen liegen, können Sie mit direktem Zugriff auf die Hardware-Register sehr schnell ein oder mehrere DAC-Register beschreiben und synchron die Ausgabe aktivieren (siehe Befehle **Peek** und **Poke** im *ADbasic*-Handbuch).

Die Hardware-Adressen für den direkten Zugriff auf die Steuer- und Datenregister sind im Anhang dargestellt.

6 Kalibrierung

6.1 Allgemeine Hinweise

Die 2 Digital/Analog-Wandler (DAC; optional 8) und die 4 Analog/Digital-Wandler (ADC) Ihres *ADwin-Gold*-Systems sind bei der Auslieferung werkseitig kalibriert. Entsprechend den Vorschriften zur Einhaltung der Messgenauigkeit für Ihr Anwendungsgebiet sind die Geräte in regelmäßigen Abständen zu kalibrieren.

Sie führen die Kalibrierung mit dem Programm <GoldCalib.exe> durch; der Pfad bei Standardinstallation ist <C:\ADwin\Tools\ADwin-Gold>.

Hilfsmittel

Zur Kalibrierung benötigen Sie folgende Hilfsmittel:

- Ein Digital-Multimeter (DMM) mit einer Messgenauigkeit von
  - 30µV bei 16Bit -Wandlern
  - 120µV bei 14Bit -Wandlern
- Eine stabile Referenz-Gleichspannungsquelle mit einer Einstellgenauigkeit von
  - 30µV bei 16Bit -Wandlern
  - 120µV bei 14Bit -Wandlern
- Verbindungskabel von den Ein/Ausgängen zur Referenzspannungsquelle und zum Messgerät (bevorzugt BNC-Kabel).

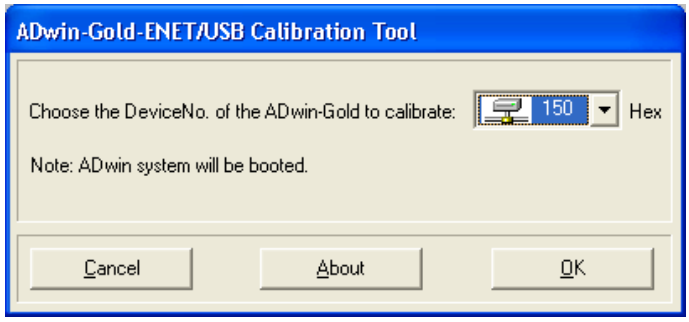
6.2 Kalibrierung durchführen

Verbinden Sie Ihr *ADwin-Gold*-Gerät mit dem PC und konfigurieren Sie es mit dem Programm <ADconfig.exe>.

Die Kalibrierung muss bei Betriebstemperatur des *ADwin-Gold* erfolgen. Bei einer Einschalttemperatur des Gerätes von ca. 20…25 Grad Celsius (Raumtemperatur) ist die Betriebstemperatur etwa 30 Minuten nach dem Einschalten erreicht.

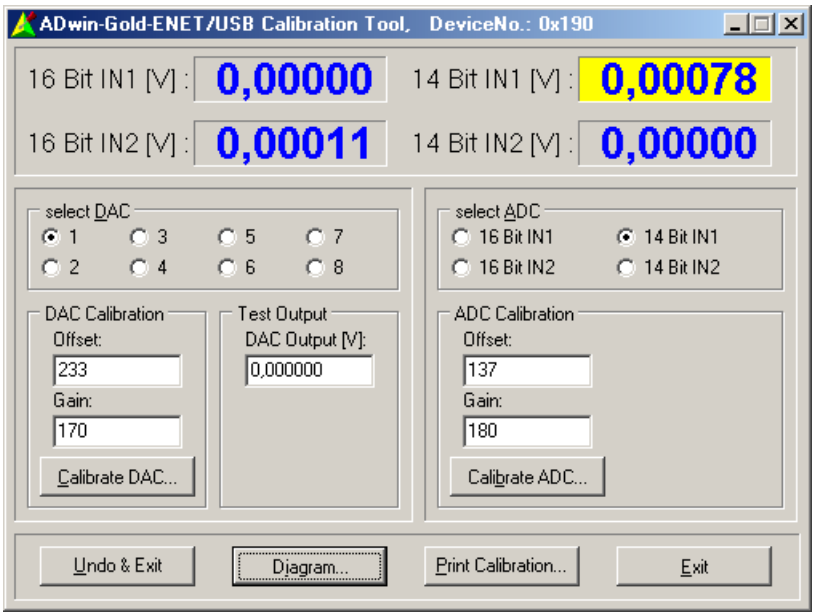
Schritt 1

Starten Sie das **Kalibrierprogramm** <GoldCalib.exe>. Es erscheint das Fenster „ADwin-Gold-ENET/USB Calibration Tool“.



Wählen Sie die Device-Nummer des zu kalibrierenden Geräts und bestätigen Sie durch Drücken von „OK“.
Sie erhalten eine Warnung, wenn Sie kein *ADwin-Gold* gewählt haben oder eines mit einer älteren Firmware-Version. Sie können die Warnung mit „Yes“ übergehen oder mit „No“ zum vorigen Fenster zurückkehren.

Es erscheint das **Übersichtsfenster**. In der Kopfzeile wird die von Ihnen gewählte Device-Nummer angezeigt.



Das obere Feld zeigt die aktuellen Messwerte an den Eingängen IN1 und IN2, jeweils gemessen mit dem 16 Bit- und dem 14 Bit-ADC.

Wählen Sie unteren Feld links den zu kalibrierenden DAC, rechts den zu kalibrierenden ADC. Der Messwert am gewählten ADC wird oben hervorgehoben. In den Eingabefeldern darunter sehen Sie die jeweils gültige Kalibriereinstellung für `Offset` und `Gain` der DAC und ADC; Sie können dort direkt Werte eingeben. Mit den Schaltflächen „`Calibrate DAC`“ oder „`Calibrate ADC`“starten Sie die Kalibrierung des gewählten Wandlers.
Im Eingabefeld „`Test Output`“ können Sie einen Spannungswert eingeben, der automatisch auf den oben gewählten Wandler/Ausgang gelegt wird.

Jede Ihrer Eingaben wird sofort in das *ADwin-Gold*-System übertragen. Wenn Sie das Programm mit „`Exit`“ verlassen, bleiben die neuen Einstellungen erhalten. Mit der Schaltfläche „`Undo&Exit`“ machen Sie alle Eingaben rückgängig und verlassen das Kalibrierprogramm (d.h. die ursprünglichen Einstellungen werden in das *ADwin-Gold*-System übertragen).
„`Diagram`“ zeigt Ihnen in einer Grafik die Genauigkeit der aktuellen Kalibriereinstellung. Ein Protokoll der eingestellten Werte drucken Sie mit der Schaltfläche „`Print Calibration`“.

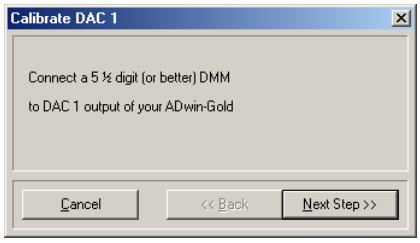
Kalibrieren Sie die Wandler in beliebiger Reihenfolge (nur mit Referenz-Spannungsquelle). Die Kalibrierung eines Wandlers wird jeweils in 3 Stufen ausgeführt; Sie können zwischen den Fenstern der Stufen durch Vorwärts-/ Rückwärts-Schaltflächen hin- und herschalten.

Ohne Referenz-Spannungsquelle ist die Kalibrierung möglich, aber ungenauer. Kalibrieren Sie dabei zuerst die DAC, dann erst die ADC.

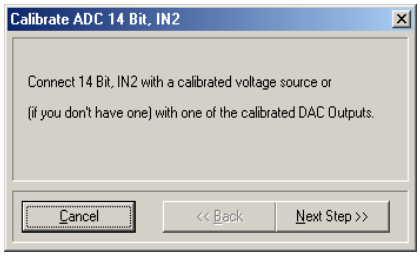
Die 3 Stufen zum **Kalibrieren eines Wandlers** sind nachfolgend beschrieben, jeweils in der linken Spalte für einen DAC, in der rechten für einen ADC.

1. Externes Hilfsgerät (DMM / Spannungsquelle) anschließen:
   Wählen Sie zur Kalibrierung eines Wandlers den gewünschten Wandler und dann die Schaltfläche „`Calibrate ...`“; es erscheint das erste Fenster:

Schritt 2



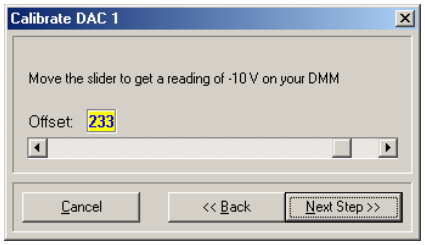
Schließen Sie ein DMM am gewählten Ausgang an.



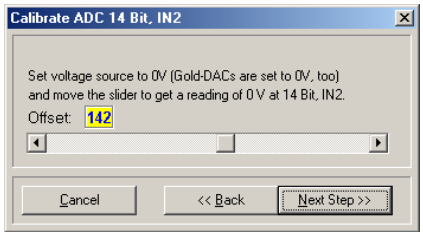
Schließen Sie die Spannungsquelle (oder einen kalibrierten DAC-Ausgang) am gewählten Eingang an.

Beachten Sie bitte Abb. 3 „Übersichtsbild ADwin-Gold-USB“.
Wählen Sie „`Next Step` >>“.

2. Offset einstellen



Verstellen Sie am Rollbalken den Offset-Wert so, dass Ihr Digital-Multimeter -10V anzeigt.

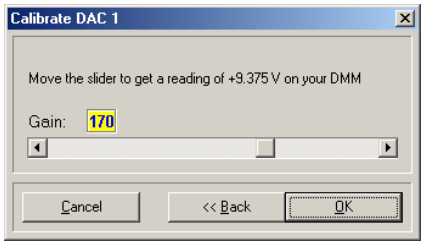


Stellen Sie an der Spannungsquelle den Sollwert 0V ein. Die Einstellung des ADC auf diesen Wert erfolgt automatisch.
Stellen Sie am Rollbalken den Offset-Wert so ein, dass der Sollwert am ADC im Übersichtsfenster angezeigt wird.

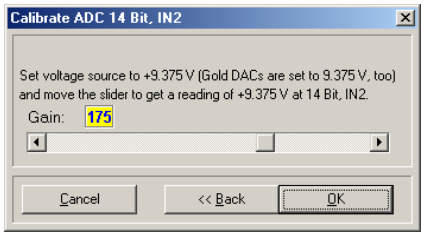
Wählen Sie „`Next Step` >>“.

3. Gain einstellen

:



Verstellen Sie am Rollbalken den Offset-Wert so, dass Ihr Digital-Multimeter 9,375V anzeigt.

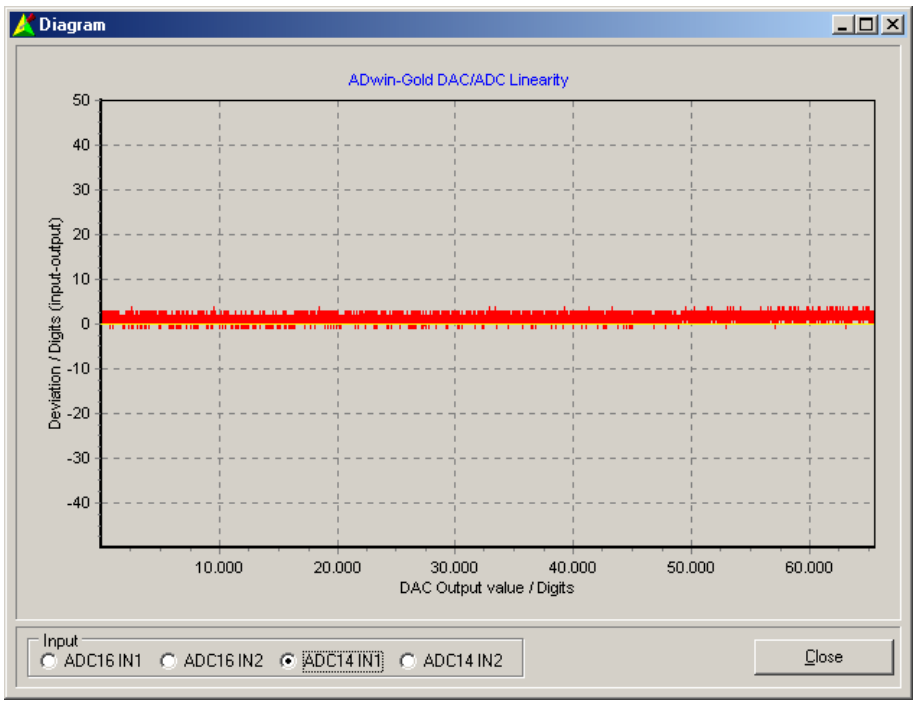


Stellen Sie an der Spannungsquelle den Sollwert 9,375V ein. Die Einstellung des ADC auf diesen Wert erfolgt automatisch.
Stellen Sie am Rollbalken den Offset-Wert so ein, dass der Sollwert am ADC im Übersichtsfenster angezeigt wird.

Die Kalibrierung für diesen Wandler ist beendet. Wählen Sie „OK“. Wiederholen Sie Schritt 2 ggf. für die anderen Wandler.

Schritt 3

Die **Genauigkeit** der eingestellten Kalibrierung können Sie anhand eines Diagramms prüfen (Schaltfläche „`Diagram`“ im Übersichtsfenster). Verbinden Sie zunächst 2 beliebige Ausgänge mit den Eingängen IN1 und IN2. Wählen Sie im Diagramm einen der Eingänge und den zugehörigen Wandler aus.



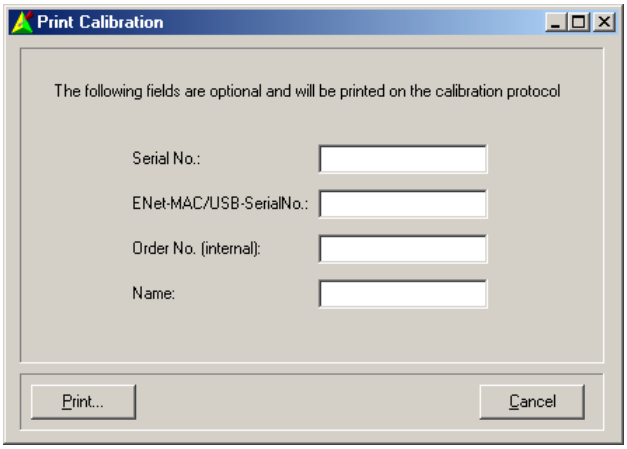
Das Programm gibt die Werte 0…65535 Digits auf alle DAC aus, vergleicht sie mit den Messwerten am gewählten Eingang und stellt die Abweichung in einer Kurve dar.
Die Abweichung sollte kleiner als 5 Digits sein.

Mit `Close` kehren Sie zum Übersichtsfenster zurück.

Schritt 4

Mit der Schaltfläche „`Print Calibration`“ können Sie ein **Protokoll** der eingestellten Kalibrier-Daten ausdrucken.

In dem geöffneten Fenster können Sie verschiedene Informationen eingeben, die in Ihrem Ausdruck erscheinen (für eine spätere Zuordnung des Protokolls). Mit der Schaltfläche „`Print`“ starten Sie den Druckvorgang; das Programm kehrt automatisch zum Übersichtsfenster zurück.
Im Protokoll-Ausdruck sind die Kalibriereinstellungen aller Ein- und Ausgänge für Gain und Offset sowie das Druckdatum enthalten.



Schritt 5

Die Kalibrierung ist beendet.

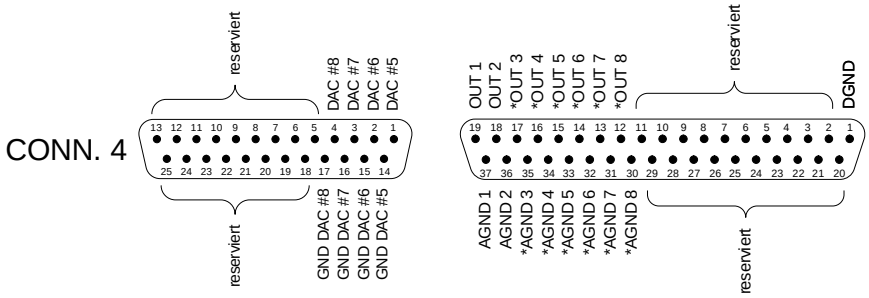
7 DA-Erweiterung

Anschlüsse

Mit der DA-Erweiterung erhalten Sie insgesamt **8 analoge Ausgänge** mit einer Auflösung von 16 Bit und je einem DAC.

In der Variante Gold führen 2 analoge Ausgänge von DAC 3 und DAC 4 auf die BNC-Buchsen OUT 3 und OUT 4. Die weiteren 4 Ausgänge führen von DAC 5…DAC 8 auf die Pins 1…4 und 14…17 der 25-poligen Sub-D-Buchse CONN. 4 (siehe Abbildung).

In der Variante Gold-D liegen alle zusätzlichen Ausgänge auf Pins der Sub-D-Buchse ANALOG OUT.



Variante Gold Variante Gold-D

Abb. 13 – Pin-Belegung der *DA*-Erweiterung

Programmierung und Kalibrierung

Sie programmieren und kalibrieren die zusätzlichen DAC wie bei den DAC 1 und DAC 2 (siehe Kapitel 5.2, Kapitel 6 und Befehlsreferenz in Kapitel 12).

8 CO1-Zählererweiterung

Die Erweiterung *Gold-CO1* (Bestelloption) stellt 4 Stück 32 Bit Vor-/ Rückwärtszähler mit Vierflankenauswertung zur Verfügung.

Die technischen Daten der CO1-Zählererweiterung sind im Anhang A.1 beschrieben.

8.1 Hardware

Die Erweiterung *Gold-CO1* stellt 4 Stück 32 Bit Vor-/ Rückwärtszähler mit Vierflankenauswertung zur Verfügung. Sie können die Zähler per Software sowohl einzeln als auch gemeinsam konfigurieren und auslesen (die Grafik zeigt den Aufbau eines einzelnen Zählers).

Latch A und B

Die Zähler können intern oder extern getaktet werden und werden über zugeordnete Latches ausgelesen. Alle Zähler haben je ein Latch A und ein Latch B. Der Zählerstand kann mit Programmierbefehlen oder (bei entsprechender Einstellung) bei einem externen Signal an CLR/LATCH gelöscht oder in ein Latch übertragen werden.

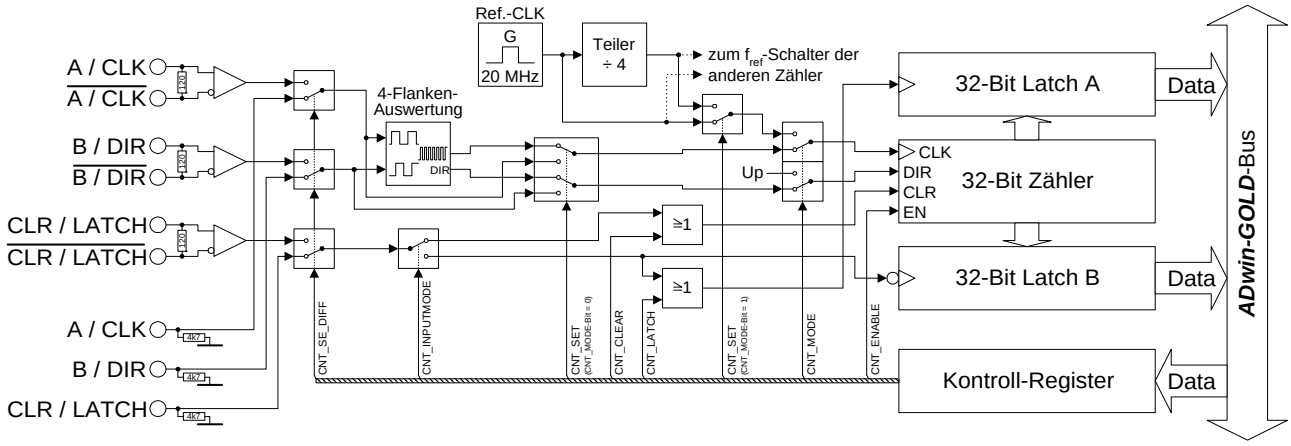


Abb. 14 – Schema der *Gold-CO1*-Zählererweiterung

Es gibt die Betriebsarten Ereigniszählung (externer Takt) und Pulsbreitenmessung (interner Takt); siehe auch Kapitel 8.3 / 8.4:

Externer Takteingang

a) **Ereigniszählung**: Das In-/Dekrementieren des Zählers wird durch externe Rechtecksignale an den Eingängen A/CLK und B/DIR ausgelöst. Eine steigende Flanke an CLR/LATCH bewirkt, dass entweder der Zähler auf Null gesetzt (CLR) oder der Zählerstand ins Latch geschrieben wird (LATCH).

   Es gibt die Modi:

1. **Takt und Richtung**: Eine steigende Flanke an CLK in- oder dekrementiert den Zählerstand um eins. Das Signal an DIR bestimmt die Zählrichtung (0 = Dekrementieren; 1 = Inkrementieren).
2. **Vierflankenauswertung**: Jede Flanke der (um 90 Grad) versetzten Signale an A/CLK und an B/DIR löst ein In-/Dekrementieren des Zählers aus. Die Zählrichtung ergibt sich aus der Reihenfolge der steigenden/fallenden Flanken dieser Signale. Dieser Modus wird besonders für Inkrementalgeber (Winkel-Encoder) eingesetzt.

Interner Takteingang

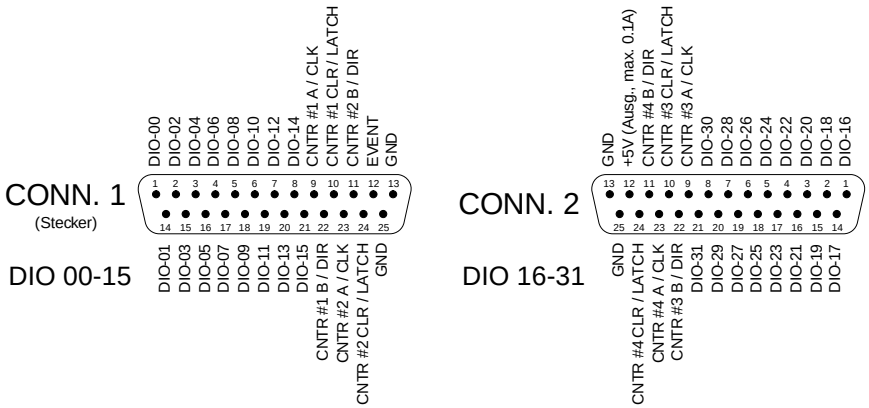
b) **Pulsbreitenmessung**: Das In-/Dekrementieren des Zählers wird von einem internen Referenztaktgeber ausgelöst; es kann eine Signalfrequenz von 5 MHz oder 20 MHz verwendet werden. Ausgewertet wird das an CLR/LATCH anliegende Rechtecksignal: Mit jeder positiven Flanke wird der Zählerstand in Latch A geschrieben, mit jeder negativen in Latch B.

Sie können damit berechnen:

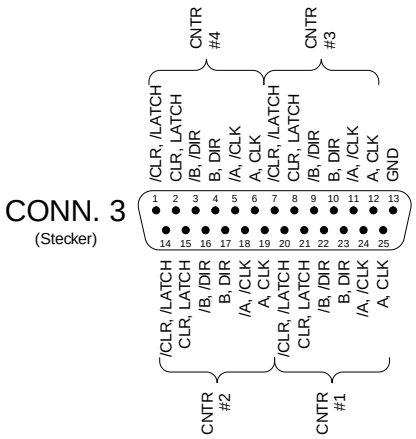
1. die **Periodendauer** des Eingangssignals an CLR/LATCH aus den Werten in Latch A oder Latch B.
2. die **Impulsbreite** und **Pausenzeit** aus den Werten in Latch A und Latch B.

Die Zähler werden mit *ADbasic*-Befehlen über ein Kontrollregister gesteuert (Befehlsübersicht siehe Tabelle in Kapitel 8.2).

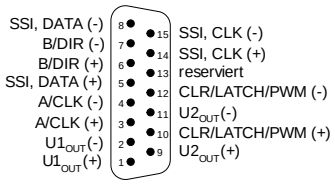
An den differentiellen Eingängen A/CLK, B/DIR und CLR/LATCH sind TTL-ähnliche Signale erforderlich. Einzelheiten und Grenzwerte finden Sie bei den Technischen Daten im Anhang, Tabelle „Digitale Ein- / Ausgänge“.



Zählereingänge bei Betriebsart TTL für Gold / Gold-D
(single-ended; Betriebsart bei Rev. B2 nicht vorhanden)



nur Variante Gold



CO1, ..., CO4 (Stecker)

nur Variante Gold-D

Zählereingänge bei Betriebsart differentiell

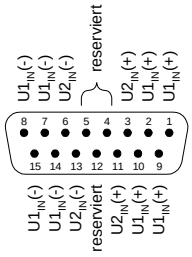
Abb. 15 – Pin-Belegung der CO1-Erweiterung

Es ist in jedem Fall erforderlich, dass Sie die Zählereingänge mit dem Befehl **`Cnt_SE_Diff`** für die gewünschte Betriebsart einstellen. Dies geschieht paarweise, d.h. die Zähler 1 und 2 gemeinsam sowie die Zähler 3 und 4 gemeinsam.
Bei Rev. B2 können Sie nur differentielle Eingänge einstellen, ab Rev. B3 alternativ auch TTL-Eingänge (single ended).

Obwohl alle Eingänge der *CO1*-Erweiterung einen Pull-down-Widerstand besitzen, können offene Eingänge (bei dem zugehörigen Zähler) vor allem in einer nicht störungsfreien Umgebung zu Fehlern führen. Wenn Sie einen Eingang eines Zählers nicht benutzen, legen Sie sicherheitshalber beide Leitun-

gen des (differentiellen) Eingangs auf ein definiertes Potenzial: Legen Sie den Pluseingang auf +5V und den Minuseingang auf GND.

Bei der Liefervariante Gold-D können Sie über den Anschluss `CO Power in` eine Spannung einspeisen, die an den Steckern CO1…CO4 zur Verfügung steht, z.B. für externe Inkrementalgeber.
Beachten Sie bitte: Die Minus-Eingänge U_{1in} (-) sind intern über eine gemeinsame Leitung galvanisch mit GND verbunden; auch die Minus-Eingänge U_{2in} (-) haben eine solche Verbindung.



CO POWER IN
(Buchse)

Abb. 16 – Pin-Belegung Zähler-Spannungsversorgung (Gold-D)

8.2 Software

Die für den Zugriff auf die Zähler benötigten Funktionen befinden sich in der Include-Datei:

`<ADWGCNT.INC>`

Binden Sie diese Include-Datei zum Beginn eines Programms ein, damit Sie die Befehle aus der nachfolgenden Tabelle benutzen können. Die Befehle sind in Kapitel 12 ab Seite 70 beschrieben.

Befehl	Funktion
Cnt_Clear() *	Zähler löschen
Cnt_Enable()	Zähler sperren oder freigeben (achten Sie auf bereits laufende Zähler)
Cnt_GetStatus(#)	Statusregister auslesen (# = Zähler-Nr. 1…4)
Cnt_ResetStatus()	Statusregister zurücksetzen
Cnt_InputMode()	CLR/LATCH-Eingang auf CLR- oder LATCH-Modus stellen
Cnt_Latch() *	Zählerstand in Latch A übernehmen
Cnt_Mode()	Externen Takteingang oder internen Referenztakt verwenden
Cnt_SE_Diff()	Differentielle oder TTL-Eingänge einstellen (paarweise)
Cnt_Set()	In Kombination mit **Cnt_Mode()**: Zähler-Modus oder Höhe des internen Referenztakts einstellen
Cnt_Read(#)	Zählerstand in Latch A übernehmen und auslesen (# = Zähler-Nr. 1…4)
Cnt_ReadLatch(#)	Latch A (getriggert durch pos. Flanke) auslesen (# = Zähler-Nr. 1…4)
Cnt_ReadFLatch(#)	Latch B (getriggert durch neg. Flanke) auslesen (# = Zähler-Nr. 1…4)

* diese Funktionen werden nach der Durchführung wieder zurückgesetzt. Alle anderen Funktionen werden nur durch die entgegengesetzte Funktion aufgehoben.

Abb. 17 – Befehle der *Gold-CO1*-Zählererweiterung

Mit den Befehlen in der Tabellenmatrix beeinflussen Sie immer **alle** Zähler (ausgenommen **Cnt_Read**…). Achten Sie deshalb darauf, immer alle Bits korrekt zu setzen oder zu löschen. Sie können dadurch jeden Zähler einzeln oder beliebig viele Zähler gemeinsam beeinflussen.

Befehlsfolge

Konfigurieren Sie die Zähler bitte in dieser Reihenfolge:

1. Gewünschten Zähler sperren (**Cnt_Enable**)
2. Betriebsart einstellen (**Cnt_Mode**, **Cnt_Set**, **Cnt_InputMode**, **Cnt_SE_Diff**)
3. Zähler löschen (**Cnt_Clear**)
4. Zähler freigeben (**Cnt_Enable**)

Für die Verarbeitung der Werte im *ADbasic*-Programm übertragen Sie die Werte ggf. ins Latch-Register und lesen sie dort aus.

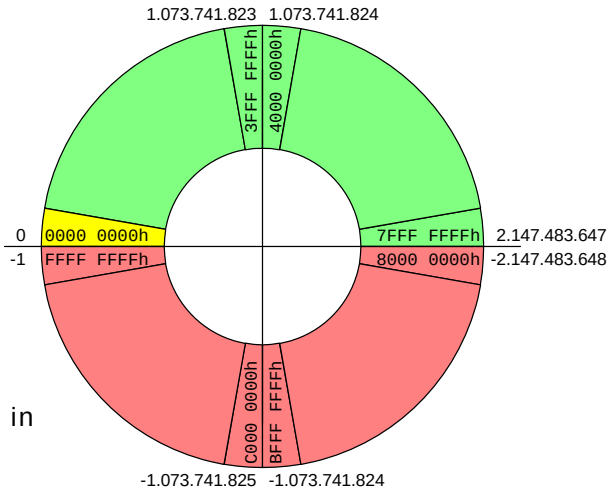
Beachten Sie die Abhängigkeit des Befehls **Cnt_Set** vom Befehl **Cnt_Mode**.

Wenn Sie einen bestimmten Zähler sperren oder freigeben möchten, müssen Sie auch die schon laufenden Zähler freigeben (= Bits setzen). Wenn Sie (unbeabsichtigt) die Bits dieser Zähler nicht setzen, werden diese gesperrt.

8.2.1 Auswerten des Zählerinhalts

Die Binärzähler der *CO1*-Erweiterung erzeugen 32 Bit-Werte, die *ADbasic* nach dem Modell des unten stehenden Zahlenkreises als Zahlen mit Vorzeichen interpretiert: Das höchste Bit (MSB) stellt das Vorzeichen dar; die größte positive Zahl (231-1) schließt an die höchste negative Zahl (-231) an und die kleinste positive (0) an die kleinste negative Zahl (-1).

Zahlenkreis



innen:Wert des Binärzählers

außen:Zahlenwert in *ADbasic*

Abb. 18 – Zahlenkreis als Interpretation von Zählerwerten

Beachten Sie deswegen bei der Programmierung die nachstehenden Regeln:

a) Verarbeiten Sie den gelesenen 32 Bit-Werts nur mit Variablen vom Typ `INTEGER` oder `LONG`. *ADbasic* behält dann intern das gelesene Bitmuster unverändert bei und berücksichtigt automatisch den Übergang zwischen positivem und negativem Zahlenbereich. Damit gilt:

Zählrichtung

b) Die Zählrichtung (vor- oder rückwärts) ergibt sich zuverlässig nur aus dem **Vorzeichen der Differenz**: [**neuer** Zählerstand] **minus** [**alter** Zählerstand] und nicht aus dem Vergleich der Zählerstände.

„Überlauf“

Berücksichtigen Sie bei der Programmierung, dass ein „Überlauf“ zwischen dem Auslesen von zwei Zählerständen - d.h. der aktuelle Zählerstand „überrundet“ den zuletzt gelesenen - nicht erfasst wird. Ein solcher Überlauf tritt bei einer Eingangsfrequenz von 20 MHz nach etwas mehr als 3½ Minuten ein, bei 5 MHz nach über 14 Minuten.

Beispielprozesse

Sie finden mehrere Beispielprozesse zur *CO1*-Erweiterung im Verzeichnis `<C:\ADwin\ADbasic\samples_ADwin_Gold>` (Standardinstallation).

8.3 Betriebsart Impuls-/Ereigniszähler

Externe Rechtecksignale an den Eingängen A/CLK und B/DIR takten in dieser Betriebsart den jeweiligen Zähler. Mit **`Cnt_Set`** aktivieren Sie den Modus zur Ermittlung von Taktfrequenz und Richtung oder die Vierflankenauswertung.

Der Eingang CLR/LATCH kann benutzt werden, um (jeweils bei einem dort anliegenden High-Signal)

– den Zähler zu löschen (CLR) **Löschen**

– den Zählerstand ins Latch-Register A zu übernehmen (LATCH). **Latchen**

8.3.1 Takt und Richtung

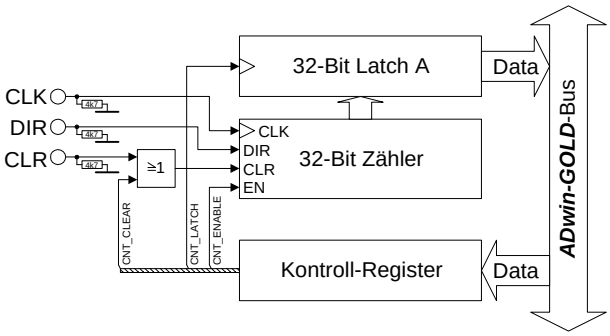


Abb. 19 – Schema *CO1*-Erweiterung im Modus „Takt und Richtung“

Jede positive Flanke eines Rechtecksignals auf dem CLK-Eingang (Clock) wird bis zu einer maximalen Frequenz von 20 MHz gezählt. Die Richtung ergibt sich aus einem High- (vorwärts) bzw. Low-Signal (rückwärts) auf dem DIR-Eingang (Direction); dieses Signal kann sowohl statisch sein, für eine feste Zählrichtung, oder auch dynamisch, für wechselnde Zählrichtungen.

Programmierbeispiel

```
INIT:                       Initialisierung ...
. . .
CNT_ENABLE(0)               Zähler sperren
CNT_CLEAR(1)                Löschen des Zählerinhaltes
CNT_MODE(0)                 Externer Takteingang (CLK)
CNT_SET(1)                  Modus "Takt und Richtung" aktivieren
CNT_INPUTMODE(0)            Eingang CLR/LATCH als CLR-Eingang
CNT_SE_DIFF(0)              Single-Ended-Eingänge auswählen
CNT_ENABLE(1)               Zähler freigeben
. . .
EVENT:                      Event-Schleife ...
CNT_LATCH(1)                Aktuellen Zählerstand in Latch A übernehmen
CNT_READLATCH(1)            Latch A auslesen
. . .                       Zählerstand im Prozess auswerten
```

8.3.2 Vier-Flanken-Auswertung

Dieser Modus ermittelt Takt und Zählrichtung aus zwei Rechteck-Signalen, die an den Eingängen A und B um 90 Grad versetzt anliegen. Die Zählrichtung ergibt sich aus der zeitlichen Abfolge mit der die steigenden und fallenden Flanken der beiden Signale eintreffen.

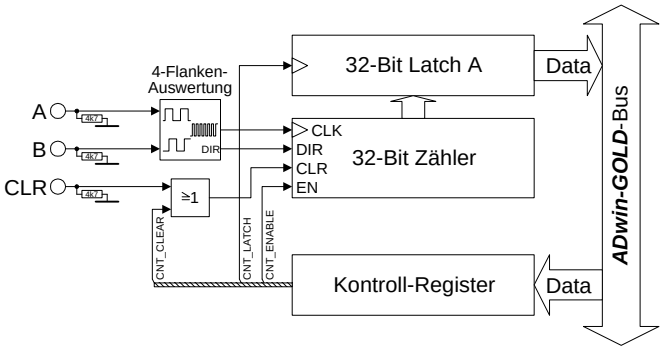


Abb. 20 – Schema *CO1*-Erweiterung im Modus „4-Flanken-Auswertung“

Berücksichtigen Sie bitte:

– Der Zähler registriert bei einem Zyklus des A/B-Signals 4 Flanken.

– Die maximale Zählfrequenz beträgt 20 MHz. Gemeinsam mit den 4 Flanken je Zyklus ergibt sich daraus eine maximale Eingangsfrequenz von 5 MHz.
– Der Abstand zwischen einer Flanke an A und einer Flanke an B darf 50 ns nicht unterschreiten. Impulsbreiten oder Pausenzeiten kürzer als 100 ns werden nicht gezählt.
– Eine Änderung der Phasenverschiebung hat Einfluss auf die maximale Eingangsfrequenz wegen der Mindestabstände der Flanken. Bei einem Abweichen von 90 Grad sinkt die maximale Eingangsfrequenz von 5 MHz beispielsweise bei 45 Grad auf 2,5 MHz.

Programmierbeispiel

```
INIT:                    Initialisierung ...
. . .
CNT_ENABLE(0)            Zähler sperren
CNT_CLEAR(1)             Löschen des Zählerinhaltes
CNT_MODE(0)              Externer Takteingang (CLK)
CNT_SET(0)               Modus "4-Flanken-Auswertung" aktivieren
CNT_INPUTMODE(0)         Eingang CLR/LATCH als CLR-Eingang
CNT_SE_DIFF(0)           Single-Ended-Eingänge auswählen
CNT_ENABLE(1)            Zähler freigeben
. . .

EVENT:                   Event-Schleife ...
CNT_LATCH(1)             Aktuellen Zählerstand in Latch A übernehmen
CNT_READLATCH(1)         Latch A auslesen
. . .                    Zählerstand im Prozess auswerten
```

8.4 Betriebsart Impulsbreiten- und Periodendauer-Messung

Referenztaktgeber

In dieser Betriebsart taktet ein interner Referenztaktgeber den Zähler mit einer Signalfrequenz von 20 MHz oder (nach einem Teiler) 5 MHz. Alle Zähler besitzen einen Umschalter für die Signalfrequenz. Es können die Periodendauer oder die Impuls-/Pausenzeit eines Rechtecksignals am Eingang LATCH gemessen werden.

In diesem Modus müssen Sie bei hohen Frequenzen berücksichtigen, dass Ihr **`Processdelay`** kleiner bleibt als eine Signalperiode, um jeden Zyklus zu erfassen.

8.4.1 Periodendauer-Messung

Alle 4 Zähler der Erweiterung können die Periodendauer messen.

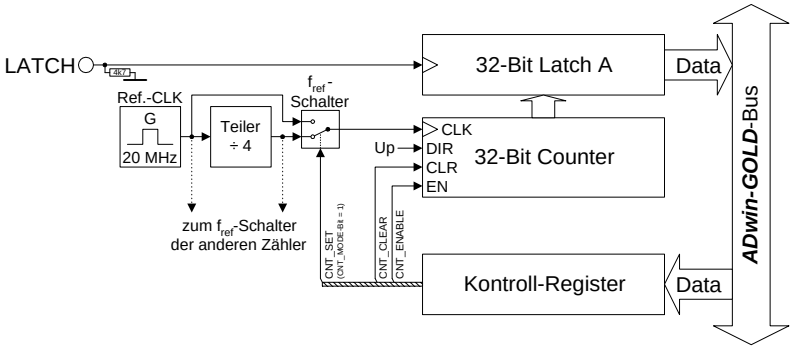
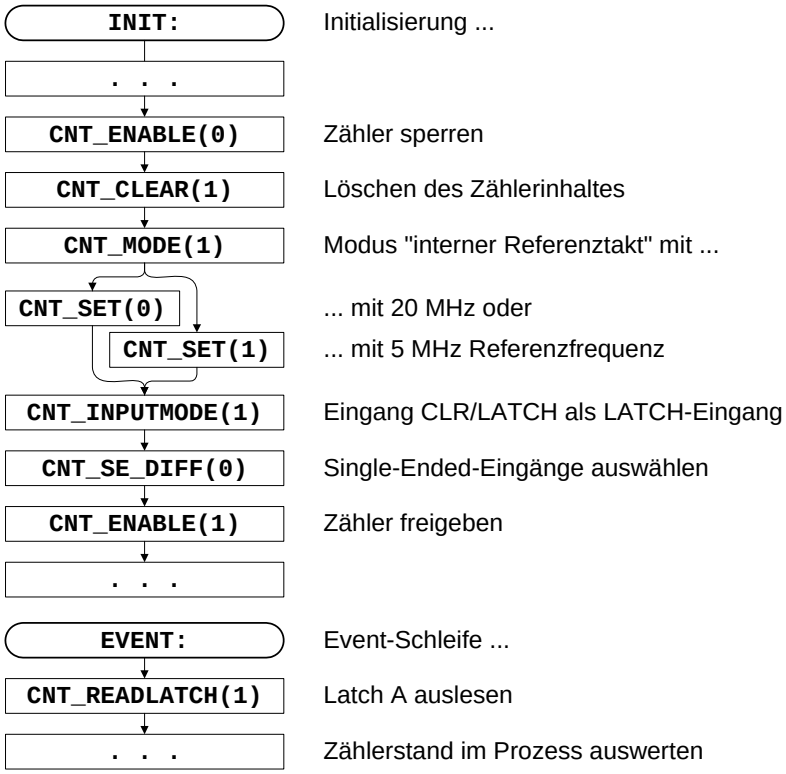


Abb. 21 – Schema *CO1*-Erweiterung im Modus „Periodendauermessung“

In diesem Modus wird bei jeder positiven Flanke am Eingang LATCH der Zählerstand in Latch A geschrieben, wobei der jeweils vorherige Stand überschrieben wird. Die Periodendauer ergibt sich aus dem Produkt von Zählerstandsdifferenz mal Periodendauer des Referenztaktes.

Programmierbeispiel



8.4.2 Messung von Impuls- und Pausenzeit

Alle 4 Zähler können Impulsbreite und Pausenzeit messen.

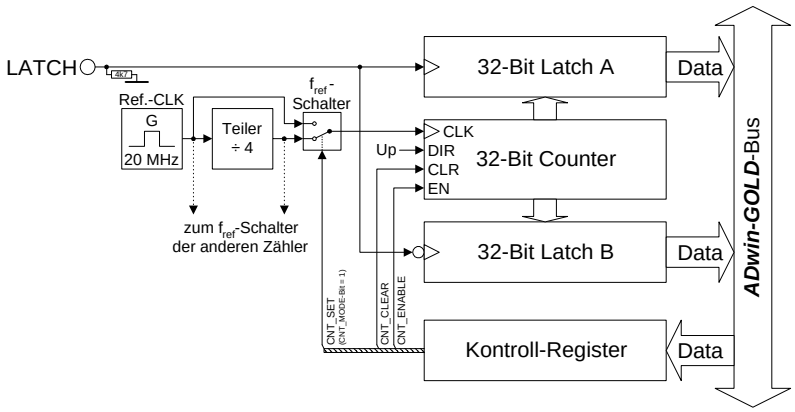


Abb. 22 – Schema *CO1*-Erweiterung im Modus „Impuls-/Pausenzeitmessung“

Die Zähler besitzen je ein Latch A für positive Flanken und ein Latch B für negative Flanken. Aus der Zählerstandsdifferenz der Latches können Impulsbreite und Pausenzeit unabhängig voneinander ermittelt werden.

Programmierbeispiel

```
INIT:                                  Initialisierung ...
. . .
CNT_ENABLE(0)                          Zähler sperren
CNT_CLEAR(1)                           Löschen des Zählerinhaltes
CNT_MODE(1)                            Modus "interner Referenztakt" mit ...
CNT_SET(0)                             ... mit 20 MHz oder
CNT_SET(1)                             ... mit 5 MHz Referenzfrequenz
CNT_INPUTMODE(1)                       Eingang CLR/LATCH als LATCH-Eingang
CNT_SE_DIFF(0)                         Single-Ended-Eingänge auswählen
CNT_ENABLE(1)                          Zähler freigeben
. . .

EVENT:                                 Event-Schleife ...
CNT_READLATCH(1)                       Latch A auslesen
CNT_READFLATCH(1)                      Latch B auslesen
. . .                                  Zählerstand im Prozess auswerten
```

8.5 Hardware-Adressen (*CO1*-Erweiterung)

Ein Prozess kann sehr schnell abgearbeitet werden, wenn Sie direkt auf die Steuer- und Datenregister zugreifen (siehe Kapitel 5.4 sowie Befehle **Peek** und **Poke** im *ADbasic*-Handbuch).

Die Hardware-Adressen der *CO1*-Erweiterung sind im Anhang beschrieben (vgl. Befehlstabelle in Kapitel 8.2).

9 CAN-Erweiterung

Die Erweiterung *Gold-CAN* beinhaltet mehrere zusätzliche Schnittstellen, die unabhängig voneinander konfiguriert und betrieben werden können:

– 4 SSI-Decoder (Seite 34)

  Die Decoder eignen sich zum Anschluss von Inkremental-Encodern mit SSI-Schnittstelle. Alle Eingänge sind differentiell und für RS422/485-Pegel (5V) ausgelegt.

  Die Decoder-Eingänge liegen auf den Steckern CO1…CO4, die ggf. auch Eingänge der Erweiterung CO1 enthalten.

– 2 CAN-Schnittstellen (Seite 36)

  Die beiden Schnittstellen eignen sich je nach Bestelloption entweder für „High-speed“ oder für „Low-speed“. Ein Umschalten ist nicht möglich.

  Die Eingänge der Schnittstelle CAN 1 liegen auf den Steckern CAN 1.1 und CAN 1.2, die der Schnittstelle CAN 2 auf dem Stecker CAN 2.

– 2 RSxxx-Schnittstellen (Seite 40)

  Beide Schnittstellen können unabhängig voneinander per Software auf RS232 oder auf RS485 eingestellt und betrieben werden.

  Die Schnittstellen-Eingänge liegen auf den Steckern COM1 und COM2.

Die Erweiterung *Gold-CAN* ist nur gemeinsam mit der Bestelloption Gold-D erhältlich.

9.1 SSI-Decoder

An die Decoder kann jeweils ein Inkremental-Encoder mit SSI-Schnittstelle angeschlossen werden. Die Signale sind differentiell und haben RS422/485-Pegel.

Die Decoder können entweder (auf Anforderung) einen einzelnen Wert auslesen oder aber kontinuierlich den aktuellen Wert bereit stellen.

Die Anschlüsse der 4 Decoder stehen auf den Steckern CO1…CO4 (15-polig, Sub-D) zur Verfügung, und zwar auf den Pins 5, 8, 14 und 15 (siehe Abb. 23). Falls das Gerät mit der Erweiterung CO1 ausgerüstet ist, sind die übrigen Pins als Zähleranschlüsse geschaltet.

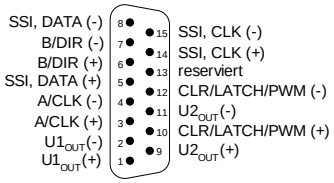
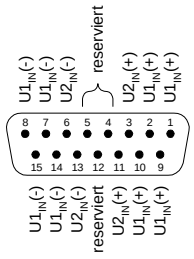


Abb. 23 – Pin-Belegung SSI-Decoder

Über den Anschluss CO Power in kann eine Spannung eingespeist werden, die an den Steckern CO1…CO4 zur Verfügung steht, z. B. für externe Inkrementalgeber.
Beachten Sie bitte: Die Minus-Eingänge U1in (-) sind intern über eine gemeinsame Leitung galvanisch mit GND verbunden; auch die Minus-Eingänge U2in (-) haben eine solche Verbindung.



Eigenschaften einstellen

Folgende Eigenschaften der Decoder sind per Software einstellbar:

– Taktrate: Über einen Vor-Teiler sind Taktraten von ca. 40 kHz bis 1 MHz möglich mit **`SSI_Set_Clock`**.

– Auflösung: Einstellbar bis 32 Bit mit **`SSI_Set_Bits`**.

Beispiel: Umsetzung von Gray-Code

Eine Umsetzung von Gray- in Binär-Code erfolgt durch eine zu programmierende Routine im *ADbasic*-Prozess (siehe unten).

```
REM PAR_1 = zu wandelnder Gray-Wert
REM PAR_2 = Flag für einen neuen Gray-Wert
REM PAR_9 = Ergebnis der Gray-zu-Binär-Wandlung

DIM m, n AS LONG

EVENT:
 IF(PAR_2=1) THEN              'Start der Wandlung
  m=0                          'Variable initialisieren
  PAR_9=0                      ' -"-
  FOR n=1 TO 32                'Alle 32 Bits durchgehen
   m=(SHIFT_RIGHT(PAR_1,(32-n)) AND 1) XOR m
   PAR_9=(SHIFT_LEFT(m,(32-n))) OR PAR_9
  NEXT n
  PAR_2=0                      'Nächste Wandlung ermöglichen
 ENDIF
```

Abb. 24 – Listing: Konvertierung von Gray- in Binär-Code

Programmierung

Die Funktionalität der Decoder wird mit *ADbasic*-Befehlen komfortabel programmiert:

Bereich	Befehle
Initialisierung	SSI_Mode SSI_Set_Bits SSI_Set_Clock
Empfangen von Daten	SSI_Read SSI_Start SSI_Status

Die Befehle sind in der Include-Datei <ADWGCAN.INC> enthalten und werden ab Seite 114 oder in der Online-Hilfe erläutert.

9.2 CAN-Schnittstelle

Die CAN-Schnittstellen 1 und 2 können voneinander unabhängig betrieben werden. Sie eignen sich je nach Bestelloption entweder für „High-speed“ oder für „Low-speed“. Eine Umschaltung ist nicht möglich.

9.2.1 Hardware-Beschreibung

Die Anschlüsse der Schnittstellen 1 und 2 stehen auf 9-poligen Sub-D-Verbindern zur Verfügung:

– Schnittstelle 1: Stecker CAN 1.1 und Buchse CAN 1.2. Die Pins der beiden Steckverbinder sind intern miteinander verbunden.

– Schnittstelle 2: Stecker CAN 2.

Die Pinbelegung für CAN „High speed“ und „Low speed“ ist unterschiedlich.

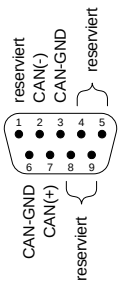
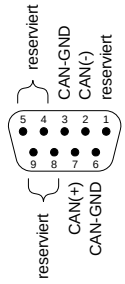
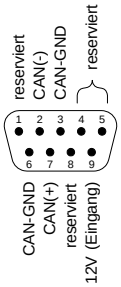
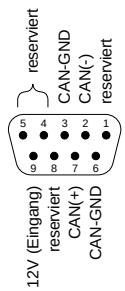
CAN-Bus	Anschlüsse CAN 1.1, CAN 2 (Stecker)	Anschluss CAN 1.2 (Buchse)
High speed		
Low speed		

Abb. 25 – CAN: Pinbelegungen

Beide Schnittstellen haben ein eigenes Potential CAN-GND, die sowohl voneinander galvanisch getrennt sind als auch vom Masse-Potential (GND) des Gehäuses.

Spannungsversorgung (nur Low speed)

Bei der „low speed“-Variante ist eine externe Spannungsversorgung mit 12V Gleichstrom erforderlich, um den CAN-Controller zu betreiben. Die Spannung muss für jede Schnittstelle separat eingespeist werden.

Bus-Terminierung (nur High speed)

Wenn die CAN-Schnittstelle das physikalische Ende eines CAN-Bus vom Typ „High speed“ bildet, muss es mit einem Abschlusswiderstand 120Ω terminiert werden (also nur am ersten oder letzten CAN-Knoten). An CAN-Knoten, die sich nicht an einem physikalischen Ende der Kette befinden, darf nicht terminiert werden.

Wenn für eine der beiden (oder beide) Schnittstellen die Terminierung erforderlich ist, müssen Sie die Pins CAN(+) und CAN(-) durch einen Widerstand von 120Ω verbinden.

9.2.2 Beschreibung der CAN-Schnittstelle

Die CAN-Schnittstelle ist mit dem CAN-Controller AN82527 von Intel® bestückt und arbeitet nach der Spezifikation „CAN 2.0 part A+B“ sowie ISO 11898. Sie programmieren die Schnittstelle mit *ADbasic*-Befehlen, die direkt auf die Register des Controllers zugreifen.

Über den CAN-Bus verschickte Nachrichten sind Datentelegramme mit bis zu 8 Bytes, die durch sogenannte „Identifier“ gekennzeichnet sind. Der CAN-Controller unterstützt Identifier mit 11 Bit und 29 Bit Länge. Die eigentliche Kommunikation, d.h. die Verwaltung der Bus-Nachrichten, erfolgt über 15 „Message-Objekte“.

Zur Konfiguration und Statusanzeige des CAN-Controllers dienen die in ihm enthaltenen Register. Hier werden Busgeschwindigkeit, Interrupt handling usw. eingestellt (siehe separate Dokumentation „82527 - Serial Communications Controller, Architectural Overview“ von Intel®).

Der CAN-Bus (high speed) ist auf Frequenzen bis 1MHz einstellbar und wird standardmäßig mit 1MHz betrieben; bei CAN low speed beträgt die max. Frequenz 125kHz. Der CAN-Bus ist durch Optokoppler vom *ADwin*-System galvanisch getrennt.

Der Eingang einer Nachricht kann einen Interrupt auslösen, der sofort einen Event am Prozessor erzeugt. Dadurch kann eine sofortige Bearbeitung der Nachrichten gewährleistet werden.

Nachrichten verwalten

Identifier

Der CAN-Controller unterscheidet über den Bus verschickte Nachrichten durch „Identifier“, das sind Kennzahlen mit einer definierten Bitlänge. Aus der Bitlänge ergeben sich hier die möglichen Kennzahlen 0...2^{11}-1bzw. 0...2^{29}-1.

Message-Objekte

Jede Nachricht (zu sendende oder zu empfangende) speichert der Controller in einem von 15 „Message-Objekten“. Die Message-Objekte können jeweils entweder zum Senden oder zum Empfangen konfiguriert werden. Als Ausnahme kann das Message-Objekt 15 nur zum Empfangen genutzt werden.
Nach der Initialisierung des CAN-Controllers sind sämtliche Message-Objekte nicht konfiguriert und beteiligen sich nicht am Busverkehr.

Jedes Message-Objekt erhält einen Identifier, der die Zuordnung einer Nachricht zu einem Message-Objekt ermöglicht.

Nachricht übergeben

In *ADbasic* übergeben Sie eine Nachricht an ein Message-Objekt über das Feld **can_msg**, das 8 Datenbytes plus die Anzahl der Datenbytes aufnehmen kann (9 Elemente). Ebenso wird eine Nachricht beim Auslesen aus einem Message Objekt in das Feld **can_msg** übertragen.

Nachricht senden

Das Versenden einer Nachricht läuft in folgenden Schritten ab:

– Sie konfigurieren ein Message-Objekt zum Senden und definieren den Identifier des Objekts (Befehl **En_Transmit**).

– Sie speichern die Nachricht im Feld **can_msg**.

– Sie senden die Nachricht (Befehl **Transmit**). Die Nachricht im Feld **can_msg** wird an das Message-Objekt übergeben. Sobald der Bus frei ist, wird die Nachricht gesendet (mit dem Identifier des Message-Objekts).

Nachricht empfangen

Das Empfangen einer Nachricht läuft in folgenden Schritten ab:

– Sie konfigurieren ein Message-Objekt für Empfang und definieren den Identifier des Objekts (Befehl **En_Receive**).

– Der Controller überwacht den CAN-Bus auf eingehende Nachrichten und speichert Nachrichten mit dem richtigen Identifier in dem Message-Objekt.

– Sie übertragen die Nachricht aus dem Message-Objekt in das Feld **can_msg** (Befehl **Read_Msg**) und lesen den zugehörigen Identifier aus.

Eine eingehende Nachricht überschreibt die alten Daten in dem Message-Objekt, die dadurch unwiderruflich verloren sind. Achten Sie daher beim Programmieren darauf, dass die Daten schneller ausgelesen als empfangen werden. Ein Datenverlust wird durch ein Flag angezeigt.

Bei dem Message Objekt 15 existiert ein zusätzlicher interner Zwischenspeicher, so dass dort 2 Nachrichten gespeichert werden können.

Nachricht zuordnen

Die Zuordnung einer eingehenden Nachricht zu einem Message-Objekt wird automatisch durch einen Vergleich ihrer Identifier gesteuert. Die globale Maske (CAN-Register 6...7 bzw. 6...9) steuert diesen Vergleich:

– Der Identifier der Nachricht wird bitweise mit dem Identifier des Message-Objekts verglichen. Wenn die relevanten Bits gleich sind, wird die Nachricht in das Message-Objekt übernommen. Nicht relevante Bits werden nicht verglichen, d.h. die Nachricht wird (sofern es von diesem Bit abhängt) in das Objekt übernommen.

– Relevante Bits werden in der globalen Maske festgelegt, indem sie dort gesetzt werden.

Globale Maske

Durch die globale Maske kann ein Message-Objekt für den Empfang von Nachrichten mit **verschiedenen Identifiern** (ID) genutzt werden. Das folgende Beispiel zeigt die Zuordnung der Nachrichten-ID 1...4 zu den Message-Objekt-ID 1...4, wenn alle Bits der globalen Maske gesetzt sind bis auf die beiden niederwertigsten (bei einem 11-Bit-Identifier also **11111111100b**).

Nachrichten-ID	ID des Message-Objekts			
	1 …001b	2 …010b	3 …011b	4 …100b
1 (…001b)	x	x	x	0
2 (…010b)	x	x	x	0
3 (…011b)	x	x	x	0
4 (…100b)	0	0	0	x

x: Nachricht wird übernommen
0: Nachricht wird nicht übernommen

In diesem Beispiel entscheidet nur der Vergleich des Bits 2 über die Zuordnung, denn die Bits 3...10 der hier verglichenen Identifier sind identisch (= 0) und die Bits 0 und 1 werden nicht verglichen, weil sie in der globalen Maske auf Null gesetzt sind (= nicht relevant).

Busfrequenz einstellen

Die **CAN-Bus-Frequenz** hängt von der Konfiguration des Controllers ab.

Bei der Initialisierung mit **Init_CAN** wird der Controller automatisch so konfiguriert, dass die CAN-Bus-Frequenz 1MHz beträgt. Soll der CAN-Bus mit einer anderen Frequenz betrieben werden, geschieht dies am einfachsten mit dem Befehl **Set_CAN_Baudrate**.

Bei CAN low speed muss die Busfrequenz auf Werte ≤125 kBit/s eingestellt werden.

Busfrequenz für Sonderfälle

In Sonderfällen kann es vorteilhaft sein, die Einstellungen anders zu wählen, als es mit **Set_CAN_Baudrate** möglich ist. Zu diesem Zweck müssen bestimmte Register mit dem Befehl **Poke** gesetzt werden. Der Registeraufbau ist in der Dokumentation des Controllers beschrieben.

Interrupt freigeben / Event auslösen

Sie können bei einem Message-Objekt freigeben, ob es beim Eingang einer Nachricht einen Interrupt auslöst. Der Interrupt-Ausgang des CAN-Controllers ist intern mit dem Event-Eingang des Prozessors verbunden. Dadurch kann der Prozessor sofort auf eingehende Nachrichten reagieren, ohne den Nachrichteneingang kontrollieren zu müssen (Polling).

Sie können die Interrupts mehrerer Message-Objekte freigeben. Welches Objekt den Interrupt ausgelöst hat, kann aus dem Interrupt-Register (**5Fh**) ersehen werden: Es enthält die Nummer des auslösenden Message-Objekts. Wird das Interrupt-Flag (new message flag) im Message-Objekt zurückgesetzt, wird das Interrupt-Register aktualisiert. Wenn kein Interrupt mehr ansteht, wird das Register auf „0“ gesetzt. Ist während der Bearbeitung des ersten Interrupts ein weiterer aufgetreten, so wird dessen Quelle nun im Interrupt-Register angezeigt. Ein weiterer Hardware-Interrupt erfolgt in diesem Fall nicht.

Programmierung

Die Schnittstelle wird mit *ADbasic*-Befehlen komfortabel programmiert:

Bereich	Befehle
Initialisierung	**Init_CAN** **En_CAN_Interrupt** **Set_CAN_Baudrate**
Empfangen und Senden von Daten	CAN_Msg **En_Receive**, **En_Transmit** **Read_Msg**, **Read_Msg_Con**, **Transmit**
Schreib- / Lesezugriff auf Controller-Register	**Set_CAN_Reg** **Get_CAN_Reg**

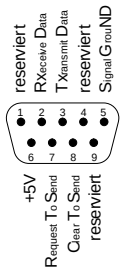
Die Befehle sind in der Include-Datei <ADWGCAN.INC> enthalten und werden ab Seite 89 oder in der Online-Hilfe erläutert.

9.3 RSxxx-Schnittstellen

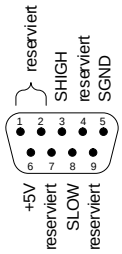
Jede der 2 RSxxx-Schnittstellen ist mit dem Controller „Quad Universal Asynchronous Receiver/Transmitter“ (UART) vom Typ TL16C754 der Firma Texas Intruments® bestückt. Die Funktionalität und Programmierung der Schnittstellen beruhen auf diesem Controller.

Beide Schnittstellen können unabhängig voneinander mit dem Protokoll RS232 oder RS485 betrieben werden. Der physikalische Unterschied der Protokolle liegt in den Pegeln der Signale, die auf dem „Bus“ durch entsprechende Treiber-Bausteine bereit gestellt werden.

Pinbelegung



COM1, COM2
(RS232) *(Stecker)*



COM1, COM2
(RS485) *(Stecker)*

Bus-Terminierung (nur RS485)

Wenn eine RS485-Schnittstelle das physikalische Bus-Ende bildet, muss es mit einem Abschlusswiderstand terminiert werden (also nur der erste oder letzte RS485-Teilnehmer). Bei RS485-Teilnehmern, die sich nicht an einem physikalischen Ende der Kette befinden, darf nicht terminiert werden.

Für die Terminierung steht – falls die gewählte Schaltungsvariante dies erfordert – am Pin 6 eine Spannung von +5 V zur Verfügung. In dieser Leitung ist bereits ein Widerstand von 300 Ω vorhanden.

9.3.1 Einstellung der Schnittstellen-Parameter

Jede Schnittstelle verfügt über einen Eingangs- und einen Ausgangs-FIFO mit einer Länge von jeweils 64 Byte.

Die Schnittstellen-Parameter werden mit Hilfe der Controller-Register eingestellt, und zwar getrennt für jede Schnittstelle. Im folgenden werden die Einstellmöglichkeiten dargestellt.

Handshake

– Handshake: Die Schnittstelle kann in 4 Modi betrieben werden:

1. RS232 ohne Handshake
2. RS232 mit Software-Handshake (Xon/Xoff)
3. RS232 Hardware-Handshake (RTS/CTS). Hierbei müssen die Signale RTS und CTS angeschlossen sein.
4. RS485

Parität

– Parität: Um einen Fehler bei der Übertragung und damit fehlerhafte Daten erkennen zu können, kann ein Paritätsbit mit übertragen werden. Die Parität kann gerade oder ungerade sein, oder es kann auf das Paritätsbit verzichtet werden.

Daten-Bits

– Datenbits: Die Nutzdaten, die übertragen werden sollen, können aus 5…8 Bits bestehen.

Stopp-Bits

– Stopp-Bits: Die Anzahl der Stopp-Bits kann auf 1, 1½ oder 2 eingestellt werden. Dabei ist die Anzahl der Stoppbits von der Anzahl der Datenbits abhängig:
  • 5 Datenbits: 1 oder 1½ Stoppbits.
  • 6…8 Datenbits: 1 oder 2 Stoppbits.

Baudrate

– Baudrate: Die physikalisch erreichbaren Werte liegen zwischen 35 Baud und 2,304 MBaud; bei einer RS-232 Schnittstelle liegt die max. Baudrate laut Spezifikation bei 115,2 kBaud.

  Die einstellbaren Baudraten werden vom moduleigenen Taktgeber abgeleitet; der Grundtakt hat eine Frequenz von 2,304 MHz. Davon ausgehend ist jede Baudrate möglich, die sich durch ganzzahlige Division dieses Grundtakts ergibt. Der Teiler kann Werte im Bereich von 1…0FFFFh annehmen. Die nachfolgende Tabelle zeigt einige gängige Baudraten und die zugehörigen Teiler.

Baudrate	Teiler	
	dez.	hex.
2304000	1	0001h
1152000	2	0002h
460800	5	0005h
230400	10	000Ah
115200	20	0014h
57600	40	0028h
38400	60	003Ch

Baudrate	Teiler	
	dez.	hex.
19200	120	0078h
9600	240	00F0h
4800	480	01E0h
2400	960	03C0h
1200	1920	0780h
600	3840	0F00h
300	7680	1E00h

Abb. 26 – RS-xxx: Gängige Baudraten

Besonderheiten RS485

Über eine RS485-Schnittstelle können – im Gegensatz zu RS232 – mehr als 2 Teilnehmer miteinander kommunizieren. So kann mit Hilfe von RS485-Schnittstellen ein Bus aufgebaut werden.

Daraus ergeben sich folgende Hinweise:

– Es gibt keinen Handshake, da ein Handshake immer nur zwischen 2 Teilnehmern funktioniert.

– Jeder Schnittstelle muss vor dem Betrieb mitgeteilt werden, ob sie auf den Bus schreiben soll oder nur Daten vom Bus übernehmen darf (**RS485_Send**).

9.3.2 Programmierung

Die Funktionalität und Programmierung der Schnittstelle beruhen auf dem eingebauten Schnittstellen-Controller. Der Controller wird mit *ADbasic*-Befehlen komfortabel programmiert:

Bereich	Befehle
Initialisierung	**RS_Init**, **RS_Reset**
Empfangen und Senden von Daten	**Check_Shift_Reg**, **RS485_Send**, **Read_FIFO**, **Write_FIFO**
Schreib- / Lesezugriff auf Controller-Register	**Get_RS**, **Set_RS**

Die Befehle sind in der Include-Datei <ADWGCAN.INC> enthalten und werden ab Seite 104 oder in der Online-Hilfe erläutert.

RS232

Beispielprogramme

Das nachfolgende Programm zeigt die Initialisierung der seriellen RS232-Schnittstelle im Abschnitt **Init:** und das zyklische Lesen und Schreiben von Daten im Abschnitt **Event:**. Der Prozess ist zeitgesteuert.

```
REM Das Programm initialisiert die seriellen
REM Schnittstellen im Abschnitt Init:
REM Im Abschnitt Event: werden Daten zwischen den
REM Schnittstellen 1 & 2 des RS-Moduls ausgetauscht.
REM Mit Hilfe dieses Programms können die Schnittstellen
REM untereinander getestet werden. Dazu müssen Sie die
REM Schnittstellen vor dem Programmstart miteinander
REM verbinden.

#INCLUDE adwgcan.inc
DIM DATA_1[1000] AS LONG'Sendedaten
DIM DATA_2[1000] AS LONG'Empfangsdaten
DIM lauf AS LONG        'Laufvariable

INIT:
 FOR lauf = 1 TO 1000   'Initialisierung der Sendedaten
  DATA_1[lauf] = lauf AND 0FFh
 NEXT lauf
 REM Initialisierung der Schnittstellen:
 REM 9600 Baud, Kein Paritätsbit, 8 Datenbits,
 REM 2 Stoppbits, RS232 ohne Handshake
 RS_Init(1,9600,0,8,1,0)
 RS_Init(2,9600,0,8,1,0)
 PAR_1 = 1
 PAR_4 = 1

EVENT:
 REM Einen Datensatz lesen und schreiben
 IF (PAR_1 <= 1000) THEN'Daten senden
  PAR_2 = Write_FIFO(1,DATA_1[PAR_1])
  IF (PAR_2 = 0) THEN INC PAR_1
 ENDIF

 PAR_3 = Read_FIFO(2)   'Daten lesen
 If (PAR_3 <> -1) THEN
  DATA_2[PAR_4] = PAR_3
  INC PAR_4
 ENDIF
 IF (PAR_4 > 1000) THEN END 'Alle Daten sind übertragen
```

RS485

In diesem Beispiel wird eine RS485-Schnittstelle als passiver Teilnehmer verwendet, der alle Daten liest, die an seinem Eingang anliegen. Wenn ein bestimmter Wert (55) empfangen wird, wird die Schnittstelle aktiv und sendet dann ihrerseits fortlaufend den Wert 44.

```
REM Schnittstelle 2 liest so lange alle Daten vom Bus, bis
REM sie den Wert 55 empfängt. Danach wird die Schnittstelle
REM aktiv und sendet den Wert 44.

#include adwgcan.inc
dim ret_val, val as Long

init:
 RS_Reset()
 REM Initialisierung der Schnittstellen:
 REM 38400 Baud, Kein Paritätsbit, 8 Datenbits,
 REM 1 Stoppbit, RS485 Software-Handshake
 RS_Init(1,38400,0,8,0,3)
 RS_Init(2,38400,0,8,0,3)
 RS485_Send(1,1)        'Schnittstelle 1 senden
 RS485_Send(2,0)        'Schnittstelle 2 empfangen

event:
  val = Read_FIFO(2)    'Daten aus Schnittstelle 2 lesen

  if (val = 55) then
   RS485_Send(2,1)      'Schnittstelle 2 senden
   ret_val = Write_FIFO(2,44) 'Daten schreiben
  endif
```

10 *ADwin-Gold*-Boot

Diese Option ist nur bei *ADwin-Gold-ENET* möglich.

ADwin-Gold-Boot startet eine zuvor programmierte Anwendung automatisch nach dem Einschalten. Damit ist nach dem Einrichten der Anwendung ein Betrieb ohne PC möglich.

Folgende Schritte führt *ADwin-Gold*-Boot nach dem Einschalten aus:

– Laden des Betriebssystems

– Laden der mit dem *ADbasic*-Compiler kompilierten Prozesse (max. 10).

– Automatisches Starten des Prozesses Nr. 10. Hier müssen Sie auch das Starten weiterer Prozesse programmieren.

Bootloader deaktivieren

Wenn Sie nicht mit der Bootloader-Option arbeiten wollen:

– Booten Sie das System nach dem Einschalten, und die gespeicherten Prozesse werden deaktiviert.

– Nach dem Ausschalten und erneuten Einschalten ist die Bootloader-Option wieder aktiv.

Durch Beschreiben des Flash-EEPROM ohne Prozesse und nur mit der Datei <ADwin9.btl> wird das System nach dem erneuten Einschalten nur noch gebootet, aber ein Prozess kann nicht ausgeführt werden.

Wenn Sie *ADwin*-Software für Entwicklungsumgebungen von der beigefügten *ADwin*-CD installieren, wird das Programm für die Bootloader-Option (ADethflash) automatisch kopiert. Die Version der CD sollte 3.002735 oder höher sein.

Benutzen Sie für ein *ADwin-Gold*-System mit einem Ethernet-Interface das Programm <ADethflash.exe>.

Bei Standardinstallation finden Sie das Programm in dem Verzeichnis

<C:\ADwin\Tools\Ethernet Interface\…>.

Hilfe zur Ethernet-Schnittstelle

Hinweise zum Bootloader mit Ethernet-Interface finden Sie im Handbuch „*ADwin*-Installation“.

2000 Werte zur freien Verfügung

In Verbindung mit dem Ethernet-Interface mit Bootloader können Sie bis zu 2000 Long- oder Float-Werte zu 32 Bit mittels *ADbasic*-Prozess im Flash-EEPROM-Speicher ablegen und auslesen. Eine nähere Beschreibung hierzu können Sie im Programm <ADethflash.exe> aufrufen mit der Schaltfläche „Info about eeprom support“.

11 Zubehör

Für das *ADwin-Gold*-System ist folgendes Zubehör lieferbar:

– *ADwin-Gold*-pow: externes 12V-Netzteil

  ADwin-Gold-pow stellt auf der Sekundärseite 12 Volt bei einer maximalen Dauerbelastung von 2 Ampere zur Verfügung. Das Netzteil ist für maximale Erweiterung und Auslastung ausgelegt.

  Achten Sie auf ausreichende Schirmung des USB- und Ethernet-Kabels, um Störungen auf den Datenleitungen zu vermeiden. Störungen müssen vor dem Gehäuse über die Masse abgeleitet werden (siehe auch Kapitel 3 „Betriebliche Umgebung“).

– diverse Längen der Spannungsversorgungs- und USB- bzw. Ethernet-Kabel

– *Gold-Mount*: Gehäuseumbau zur Hutschienen-Montage in einem Schaltschrank mit isolierten Clipsen.

– Einzelner Stromversorgungs-Stecker für ein selbst-konfektioniertes Stromversorgungs-Kabel.

12 Software

Sie programmieren *ADwin-Gold* inklusive aller Erweiterungen mit einfachen *ADbasic*-Befehlen. Die Basis-Befehle sind im Handbuch *ADbasic* beschrieben.

Befehle für den Zugriff auf Ein- und Ausgänge und Schnittstellen befinden sich auf folgenden Seiten:

12.1 Analoge Ein- und Ausgänge

Dieser Abschnitt beschreibt Befehle zum Ansprechen der analogen Eingänge und Ausgänge auf *ADwin-Gold*:

DAC

DAC gibt eine definierte Spannung auf einem bestimmten analogen Ausgang aus.

Syntax

```
DAC (dac_no,value)
```

Parameter

dac_no	Nummer des analogen Ausgangs: 1…8	LONG
value	Wert in Digits, der die auszugebende Spannung definiert: 0…65535	LONG

Beschreibung

Wenn der Digit-Wert value außerhalb des zulässigen Wertebereichs liegt, wird er automatisch auf den systemspezifischen Minimal- oder Maximalwert korrigiert.

Siehe auch

ADC

Gültig für

Gold, Gold-DA

Beispiel

```
Rem Digitaler P-Regler
Dim set_to, gain, diff, out AS LONG'Deklaration
Init:
  Processdelay = 10000

EVENT:
  set_to = PAR_1              'Sollwert
  gain = PAR_2                'Dimensionieren
  diff = set_to - ADC(1)      'Regelabweichung berechnen
  out = diff * gain           'Stellgröße berechnen
  DAC(1, out)                 'Ausgabe der Stellgröße
```

ADC

ADC misst die Spannung an einem analogen Eingang und gibt eine (dem Messergebnis entsprechende) ganze Zahl zurück.

Das Messergebnis wird in Digits zurückgegeben, falls angegeben multipliziert mit einem Verstärkungsfaktor.

Für den 12 Bit-/14 Bit-Wandler verwenden Sie bitte die Anweisung **ADC12**.

Syntax

```
ret_val = ADC(channel{,gain})
```

Parameter

input_ch	Nummer (1…16) des analogen Eingangskanals.	LONG
gain	Optionaler Verstärkungsfaktor (1, 2, 4, 8).	LONG CONST
ret_val	Messergebnis in Digits (0…65535).	LONG

Bemerkungen

ADC ist eine Zusammenstellung aufeinander folgender Funktionen:

- **Set_Mux**: Multiplexer auf den angegebenen Eingangskanal stellen.
- Einschwingen des Multiplexers abwarten.
- **Start_Conv**: Messung starten: Das angelegte Analogsignal in einen Digitalwert konvertieren. Falls angegeben, wird der Verstärkungsfaktor berücksichtigt.
- **Wait_EOC**: Das Ende der Konvertierung abwarten.
- **ReadADC**: Den Digitalwert aus einem Register lesen und zurückgeben.

Multiplexer-Einschwingzeit und Wandlungszeit sind auf Seite 17 angegeben.

Wenn Sie einen nicht vorhandenen Eingangskanal angeben, ist das Messergebnis undefiniert.

Wenn Sie die Zykluszeit des Prozesses (**Processdelay**) auf einen Wert unter 20µs einstellen, beträgt die Ausführungszeit des Befehls nur noch etwa die Hälfte. Dies ist möglich, indem der Compiler für diesen Fall die Wartezeit für das Einschwingen des Multiplexers überspringt. Es wird also angenommen, dass Sie eine Messung ohne Umstellung des Multiplexers beabsichtigen.
Wenn (bei solch kurzen Zykluszeiten) auch die erste Messung korrekt sein soll, müssen Sie eine bestimmte Zeit vor der ersten Benutzung der Anweisung **ADC** mit **Set_Mux** den Multiplexer auf den gewünschten Eingangskanal setzen. Diese Zeit muss mindestens so groß sein wie die Multiplexer-Einschwingzeit.

	Gold Rev. A	Gold Rev. B
Messung mit Multiplexer	14,4µs	14,4µs
Kürzere Ausführungszeit bei Zykluszeit kleiner als	20µs	20µs
Messung ohne Multiplexer	7,7µs	4,7µs
Einschwingzeit des Multiplexers	6,5µs	6,5µs

In folgenden Fällen sollten Sie anstelle der Anweisung **ADC** die Anweisungen **Set_Mux**, **Start_Conv**, **Wait_EOC** und **ReadADC** verwenden:

- Sehr kurze Zykluszeiten: **Processdelay** < 240 (s.o.).
- Hoher Innenwiderstand (>3kΩ) der Spannungsquelle des Messsignals: Dies verlängert die Einschwingzeit des Multiplexers.
- Sie möchten unvermeidliche Wartezeiten für zusätzliche Programmschritte nutzen.

Der Messbereich ist abhängig vom Verstärkungsfaktor:

Verstärkung	Eingangs-Spannungsbereich	Messbereich
1	-10V ... 10V	20V
2	-5V ... 5V	10V
4	-2,5V ... 2,5V	5V
8	-1,25V ... 1,25V	2,5V

Mit der folgenden Formel berechnen Sie aus dem zurückgegebenen Digitalwert die gemessene Spannung:

$$\text{Spannung} = (\text{Digits} - 32768_{\text{bipolar}}) \cdot \frac{\text{Messbereich}}{65536}$$

Für den Fall, dass die Verstärkung gleich 1 gewählt wurde (Messbereich von 20 Volt), gelten die in der Tabelle angegebenen Werte:

Messbereich	Rückgabewert von **ADC**			
	0	32768	65535	1 Digit
20V	-10V	0V	+9,999695 V	305,175µV

Siehe auch

ADC12, ReadADC, Set_Mux, Start_Conv, Wait_EOC, DAC

Gültig für

Gold

Beispiel

```
Dim iw AS LONG                  'Deklaration

EVENT:
  Rem Analogen Eingang 1 mit Verstärkung 4 messen
  iw = ADC(1,4)
  Rem Messwert in globale Variable schreiben, damit er
  Rem vom PC gelesen werden kann
  PAR_1 = iw
```

ADC12

ADC12 misst die Spannung eines analogen Eingangs über den 12 Bit- (Rev. A) oder 14 Bit-Wandler (Rev. B).

Das Messergebnis wird in Digits zurückgegeben, falls angegeben multipliziert mit einem Verstärkungsfaktor.

Für den 16 Bit-Wandler verwenden Sie bitte die Anweisung **ADC**.

Syntax

```
ret_val = ADC12(input_ch{,gain})
```

Parameter

input_ch	Nummer des analogen Eingangskanals (1…16)	LONG
gain	Optional: Verstärkungsfaktor (1, 2, 4, 8)	LONG CONST
ret_val	Messergebnis in Digits: 12 Bit: 0, 16, 32, …, 65520 14 Bit: 0, 4, 8, …, 65532	LONG

Beschreibung

ADC12 ist eine Zusammenstellung aufeinander folgender Funktionen:

- **Set_Mux**: Multiplexer auf den angegebenen Eingangskanal stellen.
- Einschwingen des Multiplexers abwarten.
- **Start_Conv**: Messung starten: Das angelegte Analogsignal in einen Digitalwert konvertieren. Falls angegeben, wird der Verstärkungsfaktor berücksichtigt.
- **Wait_EOC**: Das Ende der Konvertierung abwarten.
- **ReadADC12**: Den Digitalwert aus einem Register lesen und zurückgeben.

Multiplexer-Einschwingzeit und Wandlungszeit sind auf Seite 17 angegeben.

Wenn Sie einen nicht vorhandenen Eingangskanal angeben, ist das Messergebnis undefiniert.

Die Schrittweiten 16 und 4 der zurückgegebenen Messwerte ergeben sich daraus, dass das 12 Bit- und 14 Bit-Wandlungsergebnis jeweils als 16 Bit-Wert übergeben wird: Bei 12 Bit-Wandlern sind die Bits 0 bis 3 immer 0 (Null), bei 14 Bit-Wandlern die Bits 0 und 1.

In folgenden Fällen sollten Sie anstelle der Anweisung **ADC** die Anweisungen **Set_Mux**, **Start_Conv**, **Wait_EOC** und **ReadADC12** verwenden:

- Sehr kurze Zykluszeiten: **Processdelay** < 200: Die Anweisung **ADC12** kann nicht mehr innerhalb der Zykluszeit durchgeführt werden.
- Hoher Innenwiderstand (>3kΩ) der Spannungsquelle des Messsignals: Dies verlängert die Einschwingzeit des Multiplexers.
- Sie möchten unvermeidliche Wartezeiten für zusätzliche Programmschritte nutzen.

Der Messbereich ist abhängig vom Verstärkungsfaktor:

Verstärkung	Eingangs-Spannungsbereich	Messbereich
1	-10V … 10V	20V
2	-5V … 5V	10V
4	-2,5V … 2,5V	5V

Verstärkung	Eingangs-Spannungsbereich	Messbereich
8	-1,25V ... 1,25V	2,5V

Für den Fall, dass die Verstärkung gleich 1 gewählt wurde (Messbereich von 20 Volt), gelten die in der Tabelle angegebenen Werte:

Messbereich	Rückgabewert von **ADC12**			
	0	32768	65520	16 Digits
20V	-10V	0V	+9,99512V	4,88mV

Siehe auch

ADC, ReadADC12, Set_Mux, Start_Conv, Wait_EOC

Gültig für

Gold

Beispiel

```
Dim iw As Long                  'Deklaration

Event:
  Rem Analogen Eingang 1 mit Verstärkung 4 messen
  iw = ADC12(1,4)
  Rem Messwert in globale Variable schreiben, damit er
  Rem vom PC gelesen werden kann
  Par_1 = iw
```

ReadADC

ReadADC gibt einen gewandelten Wert von einem A/D-Wandler mit 16 Bit Auflösung zurück.

Syntax

```
ret_val = ReadADC(adc_no)
```

Parameter

adc_no	Nummer des zu lesenden Wandlers (1, 2)	LONG
ret_val	Messwert in Digits (0…65535), der der anliegenden Spannung am Wandler entspricht.	LONG

Bemerkungen

ReadADC12 liest gewandelte Werte des 12 Bit / 14 Bit A/D-Wandlers aus.

Siehe auch

ADC, ReadADC12, Set_Mux, Start_Conv, Wait_EOC

Gültig für

Gold

Beispiel

```
EVENT:
  Rem Multiplexer setzen: ADC1 auf Kanal 3, ADC2
  Rem auf Kanal 4 (ohne Verstärkung)
  Set_Mux(1001b)
  Rem MUX-Einschwingzeit überbrücken
  Rem …
  Start_Conv(1b)                'Wandlung für beide ADC starten
  Wait_EOC(11b)                 'Ende der Wandlungen abwarten
  PAR_1 = ReadADC(1)            'Wert von ADC1 einlesen
  PAR_2 = ReadADC(2)            'Wert von ADC2 einlesen
```

ReadADC12

ReadADC12 gibt einen gewandelten Wert von einem der beiden 12 Bit / 14 Bit A/D-Wandler zurück.

Syntax

```
ret_val = ReadADC12(adc_no)
```

Parameter

adc_no	Nummer (1, 2) des zu lesenden 12 Bit / 14 Bit -Wandlers.	LONG
ret_val	Messwert in Digits, der der anliegenden Spannung am Wandler entspricht: 12 Bit: 0, 16, 32, …, 65520 14 Bit: 0, 4, 8, 16, …, 65532	LONG

Bemerkungen

ReadADC liest gewandelte Werte des 16 Bit A/D-Wandlers aus.

Die A/D-Wandler (ADC) teilen den Messbereich von 20 Volt in gleich große Bereiche (Digits) ein, bei 12 Bit-Wandlern in 4096 Digits, bei 14 Bit-Wandlern sind es 16384 Digits.

Um den Vergleich mit Messwerten der 16 Bit-Wandler zu vereinfachen, gibt **ReadADC12** das Ergebnis „linksbündig“ zurück, also von Bit 15 absteigend; die Bits 3…0 (12 Bit-Wandler) oder 1 und 0 (14 Bit-Wandler) haben stets den Wert 0.
Bei gleicher anliegender Spannung liefern daher die Anweisungen **ReadADC** und **ReadADC12** in den oberen 12 / 14 Bits das gleiche Ergebnis.

Siehe auch

ADC12, Set_Mux, Start_Conv, Wait_EOC

Gültig für

Gold

Beispiel

```
Dim val1, val2 As Long

Event:
  Rem Multiplexer setzen: ADC12-1 auf Kanal 3, ADC12-2
  Rem auf Kanal 4 (ohne Verstärkung)
  Set_Mux(1001b)
  Rem MUX-Einschwingzeit überbrücken
  Rem …
  Start_Conv(11000b)          'Wandlung für beide ADC starten
  Wait_EOC(11000b)            'Ende der Wandlungen abwarten
  val1 = ReadADC12(1)         'Wert von ADC12-1 einlesen
  val2 = ReadADC12(2)         'Wert von ADC12-2 einlesen
```

Set_Mux

Set_Mux setzt einen oder mehrere Multiplexer auf den gewählten Messkanal und stellt die Verstärkung ein.

Syntax

```
Set_Mux(pattern)
```

Parameter

pattern — Bitmuster zur Zuordnung von Messkanälen und Verstärkung [LONG]

Bitnr.	9 8	7 6	5 4 3	2 1 0
	PGA 2	PGA 1	MUX 2	MUX 1

PGA 1 / 2 Jeweils 2 Bits (6…7 / 8…9) bestimmen den Verstärkungsfaktor des Multiplexers:

2 Bits	PGA 1 / PGA 2
00:	Faktor 1
01:	Faktor 2
10:	Faktor 4
11:	Faktor 8

MUX 1 / 2 Jeweils 3 Bits (0…2 / 3…5) bestimmen den Kanal, auf den der Multiplexer eingestellt wird:

3 Bits	MUX 2	MUX 1
000:	Kanal 2	Kanal 1
001:	Kanal 4	Kanal 3
010:	Kanal 6	Kanal 5
011:	Kanal 8	Kanal 7
100:	Kanal 10	Kanal 9
101:	Kanal 12	Kanal 11
110:	Kanal 14	Kanal 13
111:	Kanal 16	Kanal 15

Beschreibung

Die Umstellung des Multiplexers auf einen anderen Kanal benötigt eine definierte Einschwingzeit. Erst danach darf die Wandlung der anliegenden Spannung gestartet werden.

Die Einschwingzeit und die Wandlungszeit sind auf Seite 17 angegeben.

Wir empfehlen für die Angabe des Bitmusters die binäre Schreibweise (Suffix „**b**“). Sie können damit die erforderlichen Bitmuster besser darstellen als mit der (gleichfalls zulässigen) dezimalen oder hexadezimalen Schreibweise.

Siehe auch

ADC, ADC12, ReadADC, ReadADC12, Start_Conv, Wait_EOC

Gültig für

Gold

Beispiel

Sie möchten den Multiplexer des ADC1 auf den Kanal 5 und die Verstärkung 8 einstellen, und gleichzeitig den Multiplexer des ADC2 auf den Kanal 10 und die Verstärkung 2.
Hierzu benötigen Sie das Bitmuster: **0111100010b** (dezimal: 482).

```
Dim val AS LONG

EVENT:
  Set_Mux(0111100010b)        'Multiplexer setzen (s.o.)
  Rem Nutzen Sie hier die Einschwingzeit des Multi-
  Rem plexers durch einige Befehlszeilen.
  Start_Conv(1)               'Start AD-Wandlung ADC1
  Wait_EOC(1)                 'Wandlungsende des ADC1 abwarten
  val = ReadADC(1)            'Wert von ADC1 einlesen
```

Start_Conv

Start_Conv kann die Wandlung an einem oder mehreren A/D-Wandlern sowie an allen D/A-Wandler starten.

Syntax

```
Start_Conv(adc_pattern)
```

Parameter

pattern — Bitmuster, das die zu startenden Wandler festlegt (nur Bits 0…4 verwendbar): CONST, LONG
1: Wandler starten.
0: Wandler nicht starten.

Bit-Nr.	31…5	4	3	2	1	0
ADC1, 16 Bit	–	–	–	–	–	x
ADC2, 16 Bit	–	–	–	–	x	–
alle DAC	–	–	–	x	–	–
ADC1, 12 Bit / ADC1, 14 Bit	–	–	x	–	–	–
ADC2, 12 Bit / ADC2, 14 Bit	–	x	–	–	–	–

Beschreibung

Bei ADC1 und ADC2 handelt es sich um Analog-Digital-Wandler mit entweder 12 Bit/14 Bit oder 16 Bit. Weitere Informationen siehe Seite 10.

Sie können als Parameter nur Konstanten einsetzen, keine Variablen.

Wir empfehlen für die Angabe des Bitmusters die binäre Schreibweise (Suffix „**b**“). Sie können damit die erforderlichen Bitmuster besser darstellen als mit der (gleichfalls zulässigen) dezimalen oder hexadezimalen Schreibweise.

Siehe auch

ADC, ADC12, ReadADC, ReadADC12, Set_Mux, Wait_EOC

Gültig für

Gold

Beispiel

```
Dim val1 AS LONG

EVENT:
  Set_Mux(0)                  'Multiplexer auf Kanal 1 setzen
  Rem Überbrücken Sie hier die Einschwingzeit des
  Rem Multiplexers mit Befehlszeilen
  Start_Conv(1)               'Start ADC1 A/D-Wandlung
  Wait_EOC(1)                 'Ende der Wandlung abwarten
  val1 = ReadADC(1)           'Wert auslesen
```

Die Multiplexer-Einschwingzeit ist auf Seite 17 angegeben.

Wait_EOC

Wait_EOC wartet auf das Ende der Wandlung an einem bestimmten A/D-Wandler.

Syntax

```
Wait_EOC(adc_pattern)
```

Parameter

adc_pattern	Bitmuster zur Auswahl des Wandlers (nur Bits 0…4 verwendbar).	CONST LONG

Bit-Nr.	31…5	4	3	2	1	0
ADC1, 16 Bit	–	–	–	–	–	x
ADC2, 16 Bit	–	–	–	–	x	–
ADC1, 12/14 Bit	–	–	x	–	–	–
ADC2, 12/14 Bit	–	x	–	–	–	–

Beschreibung

Wenn Sie mehr als eines der Bits setzen, wird so lange gewartet, bis die Wandlung an allen entsprechenden ADC beendet ist.

Setzen Sie immer nur die Bits vorhandener ADC. Anderenfalls führt dies in einem hochprioren Prozess zur Unterbrechung der Kommunikation zwischen *ADwin*-System und PC.

Siehe auch

ADC, ADC12, ReadADC, ReadADC12, Set_Mux, Start_Conv

Gültig für

Gold

Beispiel

```
Dim val AS LONG

EVENT:
  Set_Mux(001000b)            'MUX des ADC2 auf Kanal 4 setzen
  Rem Überbrücken Sie hier die Einschwingzeit des
  Rem Multiplexers mit Befehlszeilen
  Start_Conv(2)               'Start A/D-Wandlung ADC2
  Wait_EOC(2)                 'Wandlungsende an ADC2 abwarten
  val = ReadADC(2)            'Wert auslesen
```

Die Multiplexer-Einschwingzeit ist auf Seite 17 angegeben.

12.2 Digitale Ein- und Ausgänge

Dieser Abschnitt beschreibt Befehle zum Ansprechen der digitalen Eingänge und Ausgänge auf *ADwin-Gold*:

Clear_Digout

Clear_Digout setzt einen der digitalen Ausgänge auf 0 (TTL-Pegel low).

Syntax

```
Clear_Digout(bit_no)
```

Parameter

bit_no	Nummer (0…15) des Bits, das den Ausgang festlegt (siehe Tabelle).	CONST LONG

bit_no	0	1	…	14	15
Ausgang	DIO16	DIO17	…	DIO30	DIO31

Beschreibung

Clear_Digout akzeptiert als Parameter nur eine Konstante. Wenn Sie den Ausgang durch eine Variable festlegen wollen, verwenden Sie den Befehl **Digout_Word**.

Sie müssen den zu löschenden Kanal vorher als Ausgang konfiguriert haben, anderenfalls hat **Clear_Digout** keine Funktion.
Sie können mit der Anweisung **Conf_DIO** die digitalen Kanäle in Gruppen zu je 8 als Ein- oder Ausgang konfigurieren. Wir empfehlen die Einstellung mit **CONF_DIO(1100b)**: Kanäle 0…15 als Eingänge, Kanäle 16…31 als Ausgänge.

Clear_Digout löscht ein Bit in dem Ausgangs-Register der Kanäle DIO16…DIO31. Dadurch wird am zugehörigen Kanal – falls dieser als Ausgang geschaltet ist – der TTL-Pegel low angelegt.

Wenn Sie einen der Kanäle 0…15 auf 0 setzen möchten, löschen Sie das entsprechende Bit im Ausgangs-Register der Kanäle DIO00…DIO15; auch hier gilt: konfigurieren Sie zuerst den Kanal als Ausgang. Gehen Sie vor wie folgt (siehe Beispiel unten):

- Lesen Sie das Register mit **Peek** aus.
- Löschen Sie das zum Kanal gehörige Bit (And-Maskierung).
- Schreiben Sie den Wert mit **Poke** in das Register zurück.

Die Registeradresse entnehmen Sie bitte der Tabelle im Anhang, Kapitel A.2.

Siehe auch

Conf_DIO, Digout_Word, Set_Digout, Peek, Poke, And

Gültig für

Gold

Beispiel

```
Dim val As Long                 'Deklaration

Init:
  Conf_DIO(1100b)               'Dig. Ein-/Ausgänge
konfigurieren
  Set_Digout(0)                 'Dig. Ausgang DIO16 auf 1 setzen

Event:
  val = ADC(1)                  'Messwerterfassung
  If (val > 3000) Then
    Clear_Digout(0)             'Dig. Ausgang DIO16 auf 0
zurücksetzen
  EndIf
```

Ein Unterprogramm, das ein einzelnes Bit der DIO-Leitungen 0…15 auf 0 setzt, könnte wie folgt aussehen:

```
Sub Clear_Digout_CONN1(bitno)
  Poke(204001C0h, Peek(204001C0h) And Not(Shift_Left(1,bitno))
    )
EndSub
```

Conf_DIO

Conf_DIO konfiguriert die 32 digitalen Kanäle in Gruppen zu je 8 als Ein- oder Ausgänge.

Syntax

```
Conf_DIO(pattern)
```

Parameter

pattern	Bitmuster, das die digitalen Kanäle als Ein- oder Ausgang konfiguriert: Bit=0: Kanäle als Eingänge. Bit=1: Kanäle als Ausgänge.	LONG

Bitnr. in pattern	15…4	3	2	1	0
Kanäle	–	DIO31 … DIO24	DIO23 … DIO16	DIO15 … DIO08	DIO07 … DIO00

Beschreibung

Conf_DIO akzeptiert als Parameter pattern nur eine Konstante.

Die digitalen Kanäle sind nach dem Einschalten zunächst als Eingänge konfiguriert (und können also auch noch nicht als Ausgänge angesprochen werden). Sie können nur in Gruppen zu je 8 als Ein- oder Ausgänge konfiguriert werden.

Wir empfehlen für die Angabe des Bitmusters die binäre Schreibweise (Suffix „**b**“). Sie können damit die erforderlichen Bitmuster besser darstellen als mit der (gleichfalls zulässigen) dezimalen oder hexadezimalen Schreibweise.

Wir empfehlen Ihnen die Konfiguration **Conf_DIO(1100b)**, d.h. DIO00…DIO15 sind Eingänge und DIO16…DIO31 sind Ausgänge (siehe auch Seite 14).
Auf diese Einstellung sind die Anweisungen **Clear_Digout**, **Set_Digout**, **Digin_Word**, **Digout_Word**, **Digin** abgestimmt; eine andere Konfiguration kann deren Funktion einschränken oder sie ganz aufheben.

Wenn Sie eine andere als die empfohlene Konfiguration verwenden, können Sie die digitalen Kanäle nur verarbeiten und setzen, indem Sie die entsprechenden Hardware-Register mit **Peek** und **Poke** auslesen und beschreiben (siehe Tabelle im Anhang, Kapitel A.2).

Siehe auch

Clear_Digout, Digin, Digin_Word, Digout_Word, Set_Digout, Peek, Poke

Gültig für

Gold

Beispiel

```
Rem Konfiguriere DIO00…DIO15 als Eingänge
Rem und DIO16…DIO31 als Ausgänge
Conf_DIO(1100b)
```

Digin

Digin gibt den Wert eines der digitalen Eingänge DIO00…DIO15 zurück.

Syntax

```
ret_val = Digin(channel_no)
```

Parameter

channel_no	Nummer, die den abzufragenden Eingang festlegt (siehe Tabelle unten).	CONST LONG
ret_val	1: TTL-Pegel high liegt an 0: TTL-Pegel low liegt an	LONG

channel_no	0	1	…	14	15
Eingang Nr.	DIO00	DIO01	…	DIO14	DIO15

Bemerkungen

Digin akzeptiert als Parameter channel_no nur eine Konstante.

Diese Anweisung ist für das Auslesen weniger Bits geeignet. Wenn mehrere Bits (z.B. auch in Schleifen) auszulesen sind, ist die Verwendung der Anweisung **Digin_Word** deutlich schneller. Beachten Sie dies insbesondere bei zeitkritischen Anwendungen.

Digin erfordert, dass Sie den entsprechenden Kanal vorher als Eingang konfiguriert haben. Bei einem Ausgang wird kein sinnvoller Wert zurückgegeben.

Der Befehl **Conf_DIO** konfiguriert die digitalen Kanäle in Gruppen zu je 8 als Ein- oder Ausgang. Wir empfehlen die Einstellung mit **Conf_DIO(1100b)**: Kanäle 0…15 als Eingänge, Kanäle 16…31 als Ausgänge.

Wenn Sie den Wert eines der Kanäle DIO16…DIO31 benötigen, lesen Sie das entsprechende Bit aus dem Eingangs-Register dieser Kanäle (auch hier: Konfigurieren Sie zuerst den Kanal als Eingang). Gehen Sie vor wie folgt (siehe 2. Beispiel Digin_CONN2):

- Lesen Sie das Register mit **Peek** aus. Die Registernummer entnehmen Sie der Tabelle im Anhang, Kapitel A.2.
- Löschen Sie alle Bits außer dem zum Kanal gehörigen Bit (And-Maskierung).

Siehe auch

Conf_DIO, Digin_Word, Digout_Word, Peek, And

Gültig für

Gold

Beispiel

```
Dim Data_1[10000] As Long As Fifo

Event:
  Rem Ist der digitale Eingang 0 gesetzt?
  If (Digin(0) = 1) Then
    Data_1 = ADC(1)            'Messwerterfassung
  EndIf
```

Eine Funktion, die den Wert eines der Kanäle DIO16…DIO31 zurückgibt, könnte wie folgt aussehen:

```
Function Digin_CONN2(bitno) As Long
 Digin_CONN2=Shift_Right(Peek(204001B0h), bitno) And 1
EndFunction
```

Digin_Word

Digin_Word gibt die Werte aller digitalen Eingänge auf einmal zurück.

Syntax

```
ret_val = Digin_Word()
```

Parameter

ret_val	Bitmuster, das den TTL-Pegeln an den digitalen Eingängen entspricht (Zuordnung s.u.). 1: TTL-Pegel high liegt an 0: TTL-Pegel low liegt an	LONG

Bitnummer in ret_val	31 … 16	15	14	…	1	0
Eingang Nr.	–	DIO15	DIO14	…	DIO01	DIO00

Bemerkungen

Digin_Word erfordert, dass Sie die Kanäle DIO00… DIO15 vorher als Eingänge konfiguriert haben. Für Ausgangskanäle wird kein sinnvoller Wert zurückgegeben.Der Befehl **Conf_DIO** konfiguriert die digitalen Kanäle in Gruppen zu je 8 als Ein- oder Ausgang. Wir empfehlen die Einstellung mit **CONF_DIO**(**1100b**): Kanäle 0…15 als Eingänge, Kanäle 16…31 als Ausgänge.

Wenn Sie die Werte der Kanäle DIO16…DIO31 benötigen, lesen Sie das Eingangs-Register dieser Kanäle aus (auch hier: Konfigurieren Sie zuerst die Kanäle als Eingänge); siehe auch 2. Beispiel Digin_Word_CONN2. Die Registernummer entnehmen Sie der Tabelle im Anhang, Kapitel A.2. Die Bits in diesem Rückgabewert sind den Kanälen wie folgt zugeordnet:

Bitnummer	31…16	15	…	1	0
Eingang Nr.	–	DIO31	…	DIO17	DIO16

Siehe auch

Conf_DIO, Digin, Digout_Word, Peek

Gültig für

Gold

Beispiel

```
Dim Data_1[10000] As Long As Fifo

Event:
  Rem Abfrage, ob die Eingänge 0 und 1 gesetzt sind
  If ((Digin_Word() And 11b) = 11b) Then
    Data_1 = ADC(1)             'Messwerterfassung
  EndIf
```

Eine Funktion, die den Wert der Kanäle DIO16…DIO31 zurückgibt, könnte wie folgt aussehen:

```
Function Digin_Word_CONN2() As Long
  Digin_Word_CONN2=Peek(204001B0h)
EndFunction
```

Digout_Word

Digout_Word setzt die digitalen Ausgänge gleichzeitig auf definierte TTL-Pegel.

Syntax

```
Digout_Word(pattern)
```

Parameter

pattern — Bitmuster, das den TTL-Pegeln an den digitalen Ausgängen entspricht (s. Tabelle). LONG
1: Setzen auf TTL-Pegel high
0: Setzen auf TTL-Pegel low

Bit-Nr. in pattern	31...16	15	...	1	0
Ausgang Nr.	–	DIO31	...	DIO17	DIO16

Bemerkungen

Digout_Word erfordert, dass Sie die Kanäle DIO16… DIO31 vorher als Ausgänge konfiguriert haben. Anderenfalls hat sie keine Funktion.Der Befehl **Conf_DIO** konfiguriert die digitalen Kanäle in Gruppen zu je 8 als Ein- oder Ausgang. Wir empfehlen die Einstellung mit **Conf_DIO**(1100b): Kanäle 0…15 als Eingänge, Kanäle 16…31 als Ausgänge.

Wenn Sie die Pegel der Kanäle DIO00…DIO15 setzen wollen, geben Sie das entsprechende Bitmuster auf das Ausgangs-Register dieser Kanäle aus (auch hier: Konfigurieren Sie zuerst die Kanäle als Ausgänge); siehe auch 2. Beispiel Digout_Word_CONN1. Die Registernummer entnehmen Sie der Tabelle im Anhang, Kapitel A.2.

Siehe auch

Clear_Digout, Conf_DIO, Digin_Word, Set_Digout, Poke

Gültig für

Gold

Beispiel

```
Dim value As Long

Init:
  Rem Ein- und Ausgänge konfigurieren (nur ADwin-Gold)
  Conf_DIO(1100b)

Event:
  value = ADC(1)                'Messwerterfassung
  If (value > 3000) Then'Grenzwert überschritten?
    Digout_Word(101b)           'Ausgänge 0 und 2 setzen, alle
anderen
                                'Ausgänge werden gelöscht!
  EndIf
```

Ein Programm, das TTL-Pegel der Kanäle DIO00 … DIO15 setzt, könnte wie folgt aussehen:

```
Init:
  Conf_DIO(1111b)              'alle Kanäle als Ausgang
                               'konfigurieren

Event:
  If (ADC(1) > 3000) Then'Grenzwert überschritten?
    Digout_Word_CONN1(0FFFFh)  'Ausgänge 0…15 setzen
  EndIf

Sub Digout_Word_CONN1(value)
  Poke(204001C0h,value)
EndSub
```

Set_Digout

Set_Digout setzt einen der digitalen Ausgänge auf 1 (TTL-Pegel high).

Syntax

```
Set_Digout(bit_no)
```

Parameter

bit_no — Nummer (0…15) des Bits, das den Ausgang festlegt (siehe Tabelle). CONST, LONG

bit_no	0	1	…	14	15
Ausgang	DIO16	DIO17	…	DIO30	DIO31

Beschreibung

Set_Digout akzeptiert als Parameter bit_no nur eine Konstante.

Set_Digout ist für das Setzen weniger Bits geeignet. Wenn mehrere Bits (z.B. auch in Schleifen) zu setzen sind, ist die Verwendung der Anweisung **Digout_Word** deutlich schneller. Beachten Sie dies insbesondere bei zeitkritischen Anwendungen.

Wenn Sie den zu setzenden Ausgang durch den Wert einer Variablen bestimmen wollen, verwenden Sie den Befehl **Digout_Word**.

Set_Digout erfordert, dass Sie den entsprechenden Kanal vorher als Ausgang konfiguriert haben. Anderenfalls hat der Befehl keine Funktion.

Der Befehl **Conf_DIO** konfiguriert die digitalen Kanäle in Gruppen zu je 8 als Ein- oder Ausgang. Wir empfehlen die Einstellung mit **Conf_DIO(1100b)**: Kanäle 0…15 als Eingänge, Kanäle 16…31 als Ausgänge.

Set_Digout setzt ein Bit in dem Ausgangs-Register der Kanäle DIO16 … DIO31. Dadurch wird am zugehörigen Kanal – falls dieser als Ausgang geschaltet ist – der TTL-Pegel high angelegt.

Wenn Sie einen der Kanäle 0…15 auf 1 setzen möchten, setzen Sie das entsprechende Bit im Ausgangs-Register der Kanäle DIO0 … DIO15 (auch hier: Konfigurieren Sie zuerst den Kanal als Ausgang). Gehen Sie vor wie folgt (siehe 2. Beispiel Set_Digout_CONN1):

- Lesen Sie das Register mit **Peek** aus.
- Setzen Sie das zum Kanal gehörige Bit (Or-Maskierung).
- Schreiben Sie den Wert mit **Poke** in das Register zurück.

Die Registernummer entnehmen Sie der Tabelle im Anhang, Kapitel A.2.

Siehe auch

Clear_Digout, Conf_DIO, Digout_Word, Peek, Poke, And

Gültig für

Gold

Beispiel

```
Dim val As Long

Init:
  Rem Dig. Ein-/Ausgänge konfigurieren
  Conf_DIO(1100b)

Event:
  val = ADC(1)                  'Messwerterfassung
  If (val > 3000) Then
    Set_Digout(0)               'Dig. Ausgang DIO16 auf 1 setzen
  EndIf
```

Ein Unterprogramm, das ein einzelnes Bit der DIO-Leitungen 0…15 auf 1 setzt, könnte wie folgt aussehen:

```
Sub Set_Digout_CONN1(bitno)
  Poke(204001C0h, Peek(204001C0h) Or Shift_Left(1,bitno) )
EndSub
```

12.3 Zähler

Dieser Abschnitt beschreibt Befehle zum Ansprechen Zähler auf *ADwin-Gold-CO1*:

Cnt_Clear

Cnt_Clear setzt einen oder mehrere Zähler auf Null, gemäß dem Bitmuster in pattern.

Syntax

```
#Include ADWGCNT.Inc
Cnt_Clear(pattern)
```

Parameter

pattern	Bitmuster Bit = 0: Kein Einfluss Bit = 1: Zähler auf Null setzen	LONG

Bit-Nr.	31…4	3	2	1	0
Zähler-Nr.	–	4	3	2	1

Bemerkungen

Nach Ausführung von **Cnt_Clear** wird das Bitmuster automatisch auf 0 (Null) zurückgesetzt, d.h. die zurückgesetzten Zähler beginnen zu zählen.

Siehe auch

Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.INC

Dim old_1, new_1 AS LONG'Variablen…
Dim old_2, new_2 AS LONG'Dimensionieren

INIT:
  old_1 = 0                    'Variablen…
  old_2 = 0                    'initialisieren
  Cnt_SE_Diff(11b)             'Alle Zähleringänge
differentiell
  Cnt_Mode(0)                  'Alle Zähler auf externen
Takteingang
  Cnt_Set(11b)                 'Zähler 1+2 mit Takt(CLK)- und
                               'Richtungs(DIR)-Eingang
  Cnt_InputMode(0)             'Funktionalität CLR/LATCH
festlegen:
                               'Alle als CLR-Eingang
  Cnt_Clear(11b)               'Zähler 1+2 auf 0 zurücksetzen
  Cnt_Enable(11b)              'Zähler 1+2 starten

EVENT:
  Cnt_Latch(11b)               'Zähler 1+2 gleichzeitig latchen
  new_1 = Cnt_ReadLatch(1)'Latch A Zähler 1 und…
  new_2 = Cnt_ReadLatch(2)'Latch A Zähler 2 auslesen.
  PAR_1 = new_1 - old_1'Differenz bilden (f = Impulse / Zeit)
  PAR_2 = new_2 - old_2'-"-
  old_1 = new_1                'Neuen Zählerstand als alten
speichern
  old_2 = new_2                '-"-
```

Cnt_Enable

Cnt_Enable hält die mittels pattern gewählten Zähler an oder gibt sie frei, um eingehende Impulse zu zählen.

Syntax

```
#Include ADWGCNT.Inc
Cnt_Enable(pattern)
```

Parameter

pattern — Bitmuster — LONG

Bit = 0: Zähler anhalten
Bit = 1: Zähler freigeben

Bit-Nr.	31…4	3	2	1	0
Zähler-Nr.	–	4	3	2	1

Siehe auch

Cnt_Clear, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.INC

Dim old_1, new_1 AS LONG'Variablen…
Dim old_2, new_2 AS LONG'Dimensionieren

INIT:
  old_1 = 0                    'Variablen…
  old_2 = 0                    'initialisieren
  Cnt_SE_Diff(11b)             'Alle Zähleringänge
differentiell
  Cnt_Mode(0)                  'Alle Zähler auf externen
Takteingang
  Cnt_Set(11b)                 'Zähler 1+2 mit Takt(CLK)- und
                               'Richtungs(DIR)-Eingang
  Cnt_InputMode(0)             'Funktionalität CLR/LATCH
festlegen:
                               'Alle als CLR-Eingang
  Cnt_Clear(11b)               'Zähler 1+2 auf 0 zurücksetzen
  Cnt_Enable(11b)              'Zähler 1+2 starten

EVENT:
  Cnt_Latch(11b)               'Zähler 1+2 gleichzeitig latchen
  new_1 = Cnt_ReadLatch(1)'Latch A Zähler 1 und…
  new_2 = Cnt_ReadLatch(2)'Latch A Zähler 2 auslesen.
  PAR_1 = new_1 - old_1'Differenz bilden (f = Impulse / Zeit)
  PAR_2 = new_2 - old_2'-"-
  old_1 = new_1                'Neuen Zählerstand als alten
speichern
  old_2 = new_2                '-"-
```

Cnt_GetStatus

Cnt_GetStatus gibt den Inhalt des Zähler-Statusregisters zurück.

Syntax

```
#Include ADWGCNT.Inc
ret_val = Cnt_GetStatus()
```

Parameter

ret_val	Inhalt des Statusregisters: Hinweise auf mögliche Fehlerquellen; Bedeutung der Bits siehe Tabelle.	LONG

Bit Nr.	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Signal	-	-	-	-	-	-	-	-	N4	N3	N2	N1	-	-	-	-
Bit Nr.	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Signal	L4	C4	L3	C3	L2	C2	L1	C1	B4	A4	B3	A3	B2	A2	B1	A1

- :don't care (Signalzustände undefiniert, mit FF FF 00 F0h ausmaskieren)
Ax:Signal A (statisch)
Bx: Signal B (statisch)
Cx:Korrelationsfehler (Signal A und B sind identisch, d.h. nicht um ca. 90° phasenverschoben)
Lx: Leitungsfehler (Kabel abgezogen oder Leitung unterbrochen)
Nx:CLR-/LATCH-Eingang (statisch)
x:Zählernummer (1…4)

Bemerkungen

Ein Leitungsfehler (L) kann nur bei differentiellen Eingängen detektiert werden! Bei TTL-Eingängen sind diese Bits stets 0.

Das Statusregister wird durch das Auslesen nicht zurückgesetzt; dies wird mit dem Befehl **Cnt_ResetStatus** erreicht.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

- / -

Cnt_InputMode

Cnt_InputMode stellt die Funktion des CLR/LATCH-Eingangs eines oder mehrerer Zähler ein.

Syntax

```
#Include ADWGCNT.Inc
Cnt_InputMode(pattern)
```

Parameter

pattern	Bitmuster Bit = 0: CLR-Modus einstellen Bit = 1: LATCH-Modus einstellen	LONG

Bit-Nr.	31…4	3	2	1	0
Zähler-Nr.	–	4	3	2	1

Bemerkungen

Verwenden Sie diesen Befehl möglichst nur bei gesperrtem Zähler.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.INC

Dim old_1, new_1 AS LONG'Variablen…
Dim old_2, new_2 AS LONG'Dimensionieren

INIT:
  old_1 = 0                    'Variablen…
  old_2 = 0                    'initialisieren
  Cnt_SE_Diff(11b)             'Alle Zähleringänge
differentiell
  Cnt_Mode(0)                  'Alle Zähler auf externen
Takteingang
  Cnt_Set(11b)                 'Zähler 1+2 mit Takt(CLK)- und
                               'Richtungs(DIR)-Eingang
  Cnt_InputMode(0)             'Funktionalität CLR/LATCH
festlegen:
                               'Alle als CLR-Eingang
  Cnt_Clear(11b)               'Zähler 1+2 auf 0 zurücksetzen
  Cnt_Enable(11b)              'Zähler 1+2 starten

EVENT:
  Cnt_Latch(11b)               'Zähler 1+2 gleichzeitig latchen
  new_1 = Cnt_ReadLatch(1)'Latch A Zähler 1 und…
  new_2 = Cnt_ReadLatch(2)'Latch A Zähler 2 auslesen.
  PAR_1 = new_1 - old_1'Differenz bilden (f = Impulse / Zeit)
  PAR_2 = new_2 - old_2'-"-
  old_1 = new_1                'Neuen Zählerstand als alten
speichern
  old_2 = new_2                '-"-
```

Cnt_Latch

Cnt_Latch überträgt den aktuellen Zählerstand eines oder mehrerer Zähler in das zugehörige Latch A, je nach Bitmuster in pattern.

Syntax

```
#Include ADWGCNT.Inc
Cnt_Latch(pattern)
```

Parameter

pattern — Bitmuster — LONG

Bit = 0: keine Funktion
Bit = 1: Zählerstand in Latch A übertragen

Bit-Nr.	31…4	3	2	1	0
Zähler-Nr.	–	4	3	2	1

Bemerkungen

Nach Ausführung des Befehls wird das Bitmuster automatisch auf 0 (Null) zurückgesetzt.

Das Latch A wird mit **Cnt_ReadLatch** in eine Variable ausgelesen.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.INC

Dim old_1, new_1 AS LONG'Variablen…
Dim old_2, new_2 AS LONG'Dimensionieren

INIT:
  old_1 = 0                    'Variablen…
  old_2 = 0                    'initialisieren
  Cnt_SE_Diff(11b)             'Alle Zähleringänge
differentiell
  Cnt_Mode(0)                  'Alle Zähler auf externen
Takteingang
  Cnt_Set(11b)                 'Zähler 1+2 mit Takt(CLK)- und
                               'Richtungs(DIR)-Eingang
  Cnt_InputMode(0)             'Funktionalität CLR/LATCH
festlegen:
                               'Alle als CLR-Eingang
  Cnt_Clear(11b)               'Zähler 1+2 auf 0 zurücksetzen
  Cnt_Enable(11b)              'Zähler 1+2 starten

EVENT:
  Cnt_Latch(11b)               'Zähler 1+2 gleichzeitig latchen
  new_1 = Cnt_ReadLatch(1)'Latch A Zähler 1 und…
  new_2 = Cnt_ReadLatch(2)'Latch A Zähler 2 auslesen.
  PAR_1 = new_1 - old_1'Differenz bilden (f = Impulse / Zeit)
  PAR_2 = new_2 - old_2'-"-
  old_1 = new_1                'Neuen Zählerstand als alten
speichern
  old_2 = new_2                '-"-
```

Cnt_Mode

Cnt_Mode definiert die Betriebsart aller Zähler, d.h. welchen Takteingang diese nutzen, gemäß dem Bitmuster in pattern.

Syntax

```
#Include ADWGCNT.Inc
Cnt_Mode(pattern)
```

Parameter

pattern	Bitmuster Bit = 0: externer Takteingang für Ereigniszählung (CLK/DIR oder A/B) Bit = 1: interner Takteingang für Pulsbreitenmessung (5 MHz oder 20 MHz)	LONG

Bit-Nr.	31…4	3	2	1	0
Zähler-Nr.	–	4	3	2	1

Bemerkungen

Cnt_Set legt den Modus des gewählten Takteingangs fest.

Verwenden Sie **Cnt_Mode** möglichst nur bei gesperrtem Zähler.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.INC

Dim old_1, new_1 AS LONG'Variablen…
Dim old_2, new_2 AS LONG'Dimensionieren

INIT:
  old_1 = 0                    'Variablen…
  old_2 = 0                    'initialisieren
  Cnt_SE_Diff(11b)             'Alle Zähleringänge
differentiell
  Cnt_Mode(0)                  'Alle Zähler auf externen
Takteingang
  Cnt_Set(11b)                 'Zähler 1+2 mit Takt(CLK)- und
                               'Richtungs(DIR)-Eingang
  Cnt_InputMode(0)             'Funktionalität CLR/LATCH
festlegen:
                               'Alle als CLR-Eingang
  Cnt_Clear(11b)               'Zähler 1+2 auf 0 zurücksetzen
  Cnt_Enable(11b)              'Zähler 1+2 starten

EVENT:
  Cnt_Latch(11b)               'Zähler 1+2 gleichzeitig latchen
  new_1 = Cnt_ReadLatch(1)'Latch A Zähler 1 und…
  new_2 = Cnt_ReadLatch(2)'Latch A Zähler 2 auslesen.
  PAR_1 = new_1 - old_1'Differenz bilden (f = Impulse / Zeit)
  PAR_2 = new_2 - old_2'-"-
  old_1 = new_1                'Neuen Zählerstand als alten
speichern
  old_2 = new_2                '-"-
```

Cnt_Read

Cnt_Read überträgt einen aktuellen Zählerstand in das zugehörige Latch A und gibt ihn als Rückgabewert zurück.

Syntax

```
#Include ADWGCNT.Inc
ret_val = Cnt_Read(counter_no)
```

Parameter

channel	Zählernummer: 1…4.	LONG
ret_val	Zählerstand	LONG

Bemerkungen

Verwenden Sie den Rückgabewert in Berechnungen (z.B. Differenzen oder Zählrichtung) nur mit Variablen vom Typ Long.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.INC

Dim old_1, new_1 AS LONG'Variablen…
Dim old_2, new_2 AS LONG'Dimensionieren

INIT:
  old_1 = 0                     'Variablen…
  old_2 = 0                     'initialisieren
  Cnt_SE_Diff(11b)              'Alle Zähleringänge
differentiell
  Cnt_Mode(0)                   'Alle Zähler auf externen
Takteingang
  Cnt_Set(11b)                  'Zähler 1+2 mit Takt(CLK)- und
                                'Richtungs(DIR)-Eingang
  Cnt_InputMode(0)              'Funktionalität CLR/LATCH
festlegen:
                                'Alle als CLR-Eingang
  Cnt_Clear(11b)                'Zähler 1+2 auf 0 zurücksetzen
  Cnt_Enable(11b)               'Zähler 1+2 starten

EVENT:
  Cnt_Latch(11b)                'Zähler 1+2 gleichzeitig latchen
  new_1 = Cnt_ReadLatch(1)'Latch A Zähler 1 und…
  new_2 = Cnt_ReadLatch(2)'Latch A Zähler 2 auslesen.
  PAR_1 = new_1 - old_1'Differenz bilden (f = Impulse / Zeit)
  PAR_2 = new_2 - old_2'-"-
  old_1 = new_1                 'Neuen Zählerstand als alten
speichern
  old_2 = new_2                 '-"-
```

Cnt_ReadLatch

Cnt_ReadLatch gibt den Wert aus dem Latch A eines Zählers als Rückgabewert zurück.

Syntax

```
#Include ADWGCNT.Inc
ret_val = Cnt_ReadLatch(channel)
```

Parameter

channel	Zählernummer: 1…4.	LONG
ret_val	Inhalt des Latch A des Zählers	LONG

Bemerkungen

Verwenden Sie den Rückgabewert in Berechnungen (z.B. Differenzen oder Zählrichtung) nur mit Variablen vom Typ Long.

Der Zeitpunkt, zu dem der Zählerstand ins Latch übertragen wurde, hängt von der Einstellung mit **Cnt_Mode** ab:

- Externer Takteingang (**Cnt_Mode**-Bit = 0): Nur der Befehl **Cnt_Latch** überträgt den Zählerstand (in Latch A).
- Interner Takteingang (**Cnt_Mode**-Bit = 1): Jede Flanke des extern angelegten Messsignals löst einen Latch-Vorgang aus.
  In Latch A wird der Zählerstand bei der positiven Flanke des Eingangssignals zwischengespeichert, während in Latch B (siehe **Cnt_ReadFLatch**) der Zählerstand bei der negativen Flanke des Eingangssignals gespeichert wird.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.Inc

Dim rise, rise_old, fall, fall_old AS LONG
#Define high PAR_1
#Define low PAR_2
#Define T PAR_9
#Define f PAR_10

INIT:
  rise_old = 0                  'Variablen…
  fall_old = 0                  'initialisieren
  Cnt_SE_Diff(11b)              'Alle Zähleringänge
differentiell
  Cnt_Mode(11b)                 'Zähler 1+2 auf internen
Takteingang
  Cnt_Set(0)                    'Alle Zähler mit 20 MHz internem
                                'Referenztakt
  Cnt_InputMode(11b)            'Funktionalität CLR/LATCH
festlegen:
                                'Zähler 1+2 als LATCH-Eingang
  Cnt_Clear(11b)                'Zähler 1+2 auf 0 zurücksetzen
  Cnt_Enable(1)                 'Zähler 1 starten

EVENT:
  rise = Cnt_ReadLatch(1)'Latch A Zähler 1 auslesen
  fall = Cnt_ReadFLatch(1)'Latch B Zähler 1 auslesen
  If (rise <> rise_old) THEN'Steigende Flanke detektiert?
    T = rise - rise_old         'Periodendauer in Nanosekunden
    f = 1E9 / T                 'Frequenz in Hertz
    If (fall <> fall_old) THEN'Fallende Flanke detektiert?
      high = (fall - rise) * 25'Impulsdauer in Nanosekunden
      low = (rise - fall_old) * 25'Pausendauer in Nanosekunden
    ELSE                        'Keine fallende Flanke
detektiert
      high = (fall - rise_old) * 25'Impulsdauer in Nanosekunden
      low = (rise - fall) * 25  'Pausendauer in Nanosekunden
    ENDIF
  ENDIF
  rise_old = rise               'Latch-Inhalt speichern
  fall_old = fall               'Latch-Inhalt speichern
```

Cnt_ReadFLatch

Cnt_ReadFLatch gibt den Wert aus dem Latch B eines Zählers als Rückgabewert zurück (nur im Modus Pulsbreitenmessung einsetzbar).

Syntax

```
#Include ADWGCNT.Inc
ret_val = Cnt_ReadFLatch(CounterNo)
```

Parameter

CounterNo	Zählernummer: 1…4.	LONG
ret_val	Inhalt des Latch B des Zählers	LONG

Bemerkungen

Verwenden Sie den Rückgabewert in Berechnungen (z.B. Differenzen oder Zählrichtung) nur mit Variablen vom Typ Long.

Der Zeitpunkt, zu dem der Zählerstand ins Latch übertragen wurde, hängt von der Einstellung mit **Cnt_Mode** ab:

- Externer Takteingang (**Cnt_Mode**-Bit = 0): Nur der Befehl **Cnt_Latch** überträgt den Zählerstand (in Latch A).
- Interner Takteingang (**Cnt_Mode**-Bit = 1): Jede Flanke des extern angelegten Messsignals löst einen Latch-Vorgang aus.
  In Latch A wird der Zählerstand bei der positiven Flanke des Eingangssignals zwischengespeichert, während in Latch B (siehe **Cnt_ReadFLatch**) der Zählerstand bei der negativen Flanke des Eingangssignals gespeichert wird.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ResetStatus, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.Inc

Dim rise, rise_old, fall, fall_old AS LONG
#Define high PAR_1
#Define low PAR_2
#Define T PAR_9
#Define f PAR_10

INIT:
  rise_old = 0                  'Variablen…
  fall_old = 0                  'initialisieren
  Cnt_SE_Diff(11b)              'Alle Zähleringänge
differentiell
  Cnt_Mode(11b)                 'Zähler 1+2 auf internen
Takteingang
  Cnt_Set(0)                    'Alle Zähler mit 20 MHz internem
                                'Referenztakt
  Cnt_InputMode(11b)            'Funktionalität CLR/LATCH
festlegen:
                                'Zähler 1+2 als LATCH-Eingang
  Cnt_Clear(11b)                'Zähler 1+2 auf 0 zurücksetzen
  Cnt_Enable(1)                 'Zähler 1 starten

EVENT:
  rise = Cnt_ReadLatch(1)'Latch A Zähler 1 auslesen
  fall = Cnt_ReadFLatch(1)'Latch B Zähler 1 auslesen
  If (rise <> rise_old) THEN'Steigende Flanke detektiert?
    T = rise - rise_old         'Periodendauer in Nanosekunden
    f = 1E9 / T                 'Frequenz in Hertz
    If (fall <> fall_old) THEN'Fallende Flanke detektiert?
      high = (fall - rise) * 25'Impulsdauer in Nanosekunden
      low = (rise - fall_old) * 25'Pausendauer in Nanosekunden
    ELSE                        'Keine fallende Flanke
detektiert
      high = (fall - rise_old) * 25'Impulsdauer in Nanosekunden
      low = (rise - fall) * 25  'Pausendauer in Nanosekunden
    ENDIF
  ENDIF
  rise_old = rise               'Latch-Inhalt speichern
  fall_old = fall               'Latch-Inhalt speichern
```

Cnt_ResetStatus

Cnt_ResetStatus löscht das Statusregister aller vier 32 Bit-Zähler.

Syntax

```
#Include ADWGCNT.Inc
Cnt_ResetStatus()
```

Parameter

- / -

Bemerkungen

Das Statusregister wird mit der Anweisung **Cnt_GetStatus** gelesen.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_Set, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.Inc

Dim error As Long

Dim old_1, new_1 As Long 'Variablen...
Dim old_2, new_2 As Long 'dimensionieren

Init:
  Cnt_Enable(0)                 'Alle Zähler stoppen
  Cnt_SE_Diff(11b)              'Alle Zähleringänge
                                'differentiell
  Cnt_Clear(1111b)              'Alle Zähler löschen
  Cnt_SE_Diff(11b)              'Alle Zähler auf diff. Eingänge
  Cnt_Mode(0)                   'Externer Takteingang
  Cnt_Set(0)                    'Vier-Flanken-Auswertung
  Cnt_InputMode(0)              'CLR-Eingang freischalten
  Cnt_Enable(1111b)             'Alle Zähler starten
  old_1 = 0                     'Variablen...
  old_2 = 0                     'initialisieren

  error = 0                     'Fehlerindikator zurücksetzen

Event:
  Par_1 = Cnt_Read(1)           'Zähler 1 auslesen
  Par_2 = Cnt_GetStatus(1) And 0FFFF00F0h 'Statusregister
                                'auslesen und maskieren
  If (Par_2 And 2000000h = 2000000h) Then 'Leitungs- bzw.
                                'Kabelfehler Zähler 1?
    Inc Par_3                   'Anzahl Leitungs- bzw.
                                'Kabelfehler bis
                                'jetzt...
    error = 1                   'Fehlerindikator setzen
  EndIf
  If (Par_2 And 1000000h = 1000000h) Then 'Korrelationsfehler
                                'am Zähler 1?
    Inc Par_4                   'Anzahl Korrelationsfehler bis
                                'jetzt
    error = 1                   'Fehlerindikator setzen
  EndIf
  Cnt_ResetStatus()             'Leitungs- und
                                'Korrelationsfehler-
                                'Bits löschen
  Par_5 = Shift_Right(Par_2 And 10h,4) 'Zustand CLR-Eingg
  Par_6 = Shift_Right(Par_2 And 10000h,16) 'Zustand Eingang A
  Par_7 = Shift_Right(Par_2 And 20000h,17) 'Zustand Eingang B
```

Cnt_Set

Cnt_Set definiert den Betriebsmodus für alle Zähler (in Abhängigkeit von **Cnt_Mode**) gemäß dem Bitmuster in pattern.

Syntax

```
#Include ADWGCNT.Inc
Cnt_Set(pattern)
```

Parameter

pattern — Bitmuster, Bit-Bedeutung siehe Tabelle. LONG

Bit-Wert in pattern	externer Takteingang Bit = 0 in Cnt_Mode	interner Takteingang Bit = 1 in Cnt_Mode
Bit = 0	4-Flankenauswertung	Referenztakt 20 MHz
Bit = 1	Takt- und Richtungseingang	Referenztakt 5 MHz

Bit-Nr.	31…4	3	2	1	0
Zähler-Nr.	–	4	3	2	1

Bemerkungen

Verwenden Sie diesen Befehl möglichst nur bei gesperrtem Zähler.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_SE_Diff

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.Inc

INIT:
  Cnt_SE_Diff(11b)              'Alle Zähleringänge
differentiell
  Cnt_Mode(0)                   'Alle Zähler auf externen
Takteingang
  Cnt_Set(1100b)                'Zähler 3+4:
Takt-Richtungsauswertung
                                'Zähler 1+2:
Vierflankenauswertung
  Cnt_Clear(1100b)              'Zähler 3+4 löschen
  Cnt_Enable(1100b)             'Zähler 3+4 aktivieren,
                                '1+2 deaktivieren
```

Cnt_SE_Diff

Cnt_SE_Diff stellt die Eingänge von jeweils 2 Zählern auf den Betriebsmodus single-ended oder differentiell ein.

Syntax

```
#Include ADWGCNT.Inc
Cnt_SE_Diff(pattern)
```

Parameter

`pattern` Bitmuster zur Auswahl der Zählerpaare (siehe Tabelle) und des Betriebsmodus der Eingänge: LONG
Bit = 0: Betriebsmodus single-ended
Bit = 1: Betriebsmodus differentiell

Bit-Nr. in `pattern`	31 … 2	1	0
Eingang zum Zähler Nr.	–	3 + 4	1 + 2

Bemerkungen

Nach dem Start ist der Betriebsmodus undefiniert; stellen Sie also in jedem Fall an allen Zählereingängen den gewünschten Betriebsmodus ein.

Siehe auch

Cnt_Clear, Cnt_Enable, Cnt_GetStatus, Cnt_InputMode, Cnt_Latch, Cnt_Mode, Cnt_Read, Cnt_ReadLatch, Cnt_ReadFLatch, Cnt_ResetStatus, Cnt_Set

Gültig für

Gold-CO1

Beispiel

```
#Include ADWGCNT.Inc
Dim error AS LONG              'Variablen…
Dim old_1, new_1 AS LONG'dimensionieren
Dim old_2, new_2 AS LONG

INIT:
  CNT_Enable(0)                'Alle Zähler stoppen
  Cnt_SE_Diff(11b)             'Alle Zählereingänge
differentiell
  CNT_Clear(1111b)             'Alle Zähler löschen
  CNT_Mode(0)                  'Externer Takteingang
  CNT_Set(0)                   'Vier-Flanken-Auswertung
  CNT_InputMode(0)             'CLR-Eingang freischalten
  CNT_Enable(1111b)            'Alle Zähler starten
  old_1 = 0                    'Variablen...
  old_2 = 0                    'initialisieren

  error = 0                    'Fehlerindikator zurücksetzen

EVENT:
  PAR_1 = CNT_Read(1)          'Zähler 1 auslesen
  PAR_2 = CNT_GetStatus(1) AND 0FFFF00F0h 'Statusregister
                               'auslesen und maskieren
  If (PAR_2 AND 2000000h = 2000000h) THEN 'Leitungs- bzw.
                               'Kabelfehler Zähler 1?
    Inc PAR_3                  'Anzahl Leitungs- bzw.
Kabelfehler bis
                               'jetzt...
    error = 1                  'Fehlerindikator setzen
  ENDIF
  If (PAR_2 AND 1000000h = 1000000h) THEN 'Korrelationsfehler
                               'am Zähler 1?
    Inc PAR_4                  'Anzahl Korrelationsfehler bis
jetzt
    error = 1                  'Fehlerindikator setzen
  ENDIF
  CNT_ResetStatus()            'Leitungs- und
Korrelationsfehler-
                               'Bits löschen
  PAR_5 = Shift_Right(PAR_2 AND 10h,4) 'Zustand CLR-Eingg
  PAR_6 = Shift_Right(PAR_2 AND 10000h,16) 'Zustand Eingang A
  PAR_7 = Shift_Right(PAR_2 AND 20000h,17) 'Zustand Eingang B
```

12.4 CAN-Schnittstelle

Dieser Abschnitt beschreibt zum Ansprechen der CAN-Schnittstellen auf *ADwin-Gold-CAN*:

CAN_Msg

CAN_Msg ist ein eindimensionales Feld mit 9 Elementen, in dem die Message-Objekte gespeichert sind oder werden.

Syntax

```
#Include ADWGCAN.Inc
CAN_Msg[n] = value
```

oder

```
value = CAN_Msg[n]
```

Parameter

n	Elementnummer im Feld **CAN_Msg** (1…9)	LONG
value	Wert (8 Bit), der in das Message-Objekt geschrieben oder daraus gelesen wird.	LONG

Bemerkungen

Die Elemente des Felds **CAN_Msg**[] haben folgende Funktion:

Elementnr. in **CAN_Msg**	1…8	9
Inhalt	Message-Objekt(e) = Datenbyte(s)	Anzahl (0…8) belegter Datenbytes

Tragen Sie die zu übertragenden Werte in das Feld **CAN_Msg**[] ein, *bevor* Sie diese mit **Transmit** übertragen.

Siehe auch

Init_CAN, Read_Msg, Read_Msg_Con, Transmit

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Sendet eine 32 Bit-FLOAT-Zahl (hier: Pi) als Folge
  von
Rem 4 Bytes in einem Message-Objekt

#Define pi 3.14159265
Dim i AS LONG

INIT:
 Init_CAN(1)                  'CAN-Controller 1
                              'initialisieren

 Rem Message-Objekt 6 der Schnittstelle 1
  initialisieren
 Rem zum Senden von CAN-Nachrichten mit dem Identifier
  40
 En_Transmit(1,6,40,0)

 Rem Bitmuster von Pi mit Datenformat Long erzeugen
 PAR_1 = Cast_FloatToLong(pi)

 Rem Bitmuster (32 Bit) in 4 Bytes aufteilen
 CAN_Msg[4] = PAR_1 AND 0FFh 'LSB zuweisen
 FOR i = 1 TO 3
  CAN_Msg[4-i] = Shift_Right(PAR_1,8*i) AND 0FFh
 NEXT i
 CAN_Msg[9] = 4               'Länge der Nachricht in Bytes

EVENT:
 Transmit(1,6)                'Message-Objekt 6 senden
```

En_CAN_Interrupt

En_CAN_Interrupt konfiguriert ein bestimmtes Message-Objekt einer CAN-Schnittstelle so, dass bei Eintreffen einer Nachricht ein externer Event erzeugt wird.

Syntax

```
#Include ADWGCAN.Inc
En_CAN_Interrupt(can_no, msg_no)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle	LONG
msg_no	Nummer (1…15) des Message-Objektes	LONG

Bemerkungen

- / -

Siehe auch

CAN_Msg, En_Receive, En_Transmit

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
INIT:
  Init_CAN(1)                   'CAN-Controller 1 initialisieren
  En_Receive(1,1,200,0)         'Initialisiere das
Message-Objekt 1
                                'der CAN-Schnittstelle 1 zum
Empfangen
                                'von CAN-Nachrichten mit dem
                                'Identifier 200
  En_CAN_Interrupt(1,1)         'Gibt das Auslösen von
Interrupts
                                '(ext. EVENT) beim Empfang des
                                'Message-Objektes 1 frei
```

En_Receive

En_Receive gibt ein bestimmtes Message-Objekt einer CAN-Schnittstelle zum Nachrichten-Empfang frei.

Syntax

```
#Include ADWGCAN.Inc
En_Receive(channel, msg_no, id, id_extend)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle	LONG
msg_no	Nummer (1…15) des Message-Objekts.	LONG
id	Identifier (0…2^{11} oder 0…2^{29}) der Nachrichten, die in diesem Message-Objekt empfangen werden können.	LONG
id_extend	Länge des Identifiers: 0: 11 Bit 1: 29 Bit	LONG

Bemerkungen

Ein Message-Objekt kann nur Nachrichten vom CAN-Bus empfangen, wenn Sie es zuvor mit **En_Receive** zum Empfang freigegeben haben.

Das Message-Objekt empfängt nur Nachrichten mit dem von Ihnen angegebenen Identifier.

Siehe auch

CAN_Msg, En_Transmit, Read_Msg, Read_Msg_Con

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

INIT:
  Init_CAN(1)                   'CAN-Controller 1 initialisieren
  En_Receive(1,1,200,0)         'Initialisiere Message-Objekt 1
der
                                'Schnittstelle 1 zum Empfangen
von
                                'Nachrichten mit dem Identifier
200
```

En_Transmit

En_Transmit gibt ein bestimmtes Message-Objekt einer CAN-Schnittstelle für das Senden von Nachrichten frei.

Syntax

```
#Include ADWGCAN.Inc
En_Transmit(channel, msg_no, id, id_extend)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle	LONG
msg_no	Nummer (1…14) des Message-Objektes	LONG
id	Identifier, der mit den Nachrichten dieses Message-Objekts gesendet wird.	LONG
id_extend	Länge des Identifiers: 0: 11 Bit 1: 29 Bit	LONG

Bemerkungen

Erst wenn ein Message-Objekt mit **En_Transmit** zum Senden freigegeben ist, kann das Objekt Nachrichten auf dem CAN-Bus senden.

Siehe auch

CAN_Msg, En_Receive, Transmit

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

INIT:
  Init_CAN(1)                  'CAN-Controller 1 initialisieren
  Rem Initialisere Message-Objekte der Schnittstelle 1:
  Rem Objekt 2 zum Empfangen mit Identifier 200,
  Rem Objekt 6 zum Senden mit Identifier 40
  En_Receive(1,1,200,0)
  En_Transmit(1,6,40,0)
```

Get_CAN_Reg

Get_CAN_Reg gibt den Wert eines bestimmten Registers im Controller einer CAN-Schnittstelle zurück.

Syntax

```
#Include ADWGCAN.Inc
ret_val = Get_CAN_Reg(can_no, regno)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle	LONG
regno	Register-Nummer (0…255) im CAN-Controller	LONG
ret_val	Inhalt des Registers (übergeben in den unteren 8 Bit)	LONG

Bemerkungen

Sie finden die Registernummern des CAN-Controllers AN82527 im Intel®-Datenblatt. Beispiele sind:

- Adresse **00h**: Kontroll-Register
- Adresse **01h**: Status-Register
- Adresse **5fh**: Interrupt-Register

Siehe auch

Init_CAN, Set_CAN_Baudrate, Set_CAN_Reg

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
INIT:
  Init_CAN(1)                  'CAN-Controller 1 initialisieren
  PAR_1 = Get_CAN_Reg(1,0)'Control-Register auslesen
```

Init_CAN

Init_CAN initialisiert den Controller einer CAN-Schnittstelle.

Syntax

```
#Include ADWGCAN.Inc
Init_CAN(channel)
```

Parameter

channel	Nummer (1, 2) der CAN-Schnittstelle	LONG

Bemerkungen

Die Anweisung führt folgende Aktionen aus:

- Reset (Hardware-Reset des CAN-Controllers)
- Alle Filter auf "must match" setzen.
- Clockout-Register auf 0 setzen (= externe Frequenz wird nicht geteilt).
- Register „Bus-Configuration“ auf 0 setzen.
- Übertragungsrate für den CAN-Bus auf 1 MBit/s setzen.
- Alle Message-Objekte sperren.

Sie müssen diese Anweisung ausführen, bevor Sie mit anderen Befehlen auf den CAN-Controller zugreifen. Wir empfehlen die Angabe im Prozessabschnitt **LowInit:** oder **Init:**.

Siehe auch

CAN_Msg, En_CAN_Interrupt, En_Receive, En_Transmit, Get_CAN_Reg, Set_CAN_Baudrate, Set_CAN_Reg

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

INIT:
  Init_CAN(1)                  'Initialisiere den
CAN-Controller 1
```

Read_Msg

Read_Msg gibt zurück, ob eine neue Nachricht in einem Message-Objekt einer CAN-Schnittstelle empfangen wurde.

Falls ja, wird die Nachricht in **CAN_Msg** gespeichert und der Identifier der Nachricht zurückgegeben.

Syntax

```
#Include ADWGCAN.Inc
ret_val = Read_Msg(channel, msg_no)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle	LONG
msg_no	Nummer (1…15) des Message-Objektes	LONG
ret_val	-1: keine neue Nachricht >0:Neue Nachricht; Wert = Identifier der Nachricht	LONG

Bemerkungen

Um eine Nachricht zu empfangen, müssen Sie folgende Reihenfolge einhalten:

- Einmal: Geben Sie das Message-Objekt mit **En_Receive** zum Empfangen frei.
- Sooft erforderlich: Prüfen Sie auf eine neue Nachricht und – falls vorhanden – speichern die Nachricht in **CAN_MSG** mit **Read_Msg**.

Sie können eine empfangene Nachricht nur einmal auslesen.

Siehe auch

CAN_Msg, En_CAN_Interrupt, En_Receive, En_Transmit, Read_Msg_Con

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Wenn eine neue Nachricht mit dem passenden Identifier
Rem empfangen wurde, werden die Daten gelesen. Die
Rem ersten 4 Bytes der Nachricht werden zu einer Fließkomma-
Rem Zahl mit 32 Bit Länge zusammengesetzt.
Dim n AS LONG

INIT:
  PAR_1 = 0
  Init_CAN(1)                   'CAN-Controller 1 initialisieren
  En_Receive(1,1,40,0)          'Message-Objekt 1 initialisieren
                                'zum Empfangen von
CAN-Nachrichten mit
                                'dem Identifier 40


EVENT:
  Rem Wenn das Message-Objekt geändert wurde, werden die
  Rem empfangenen Daten aus Objekt 1 gelesen und der
  Rem Identifier an PAR_9 übergeben.
  Rem Die Daten stehen im Feld CAN_Msg[] bereit.
  PAR_9 = Read_Msg(1,1)

  If (PAR_9 = 40) THEN
    Rem Für das Message-Objekt ist eine neue Nachricht mit dem
    Rem Identifier 40 eingetroffen
    PAR_1 = CAN_Msg[1]          'High-Byte auslesen
    FOR n = 2 TO 4              'Mit restlichen 3 Bytes zu 32
Bit-Zahl
      PAR_1 = Shift_Left(PAR_1,8) + CAN_Msg[n] 'zusammenfügen
    NEXT n
    Rem Das Bitmuster in PAR_1 in den Datentyp FLOAT wandeln und
    Rem der Variablen FPAR_1 zuweisen.
    FPAR_1 = Cast_LongToFloat(PAR_1)
  ENDIF
```

Senden einer Fließkomma-Zahl siehe Bsp. bei Transmit.

Read_Msg_Con

Read_Msg_Con prüft, ob eine vollständige neue Nachricht in einem bestimmten Message-Objekt in einer CAN-Schnittstelle empfangen wurde.

Falls ja, wird die Nachricht in **CAN_Msg** gespeichert und der Identifier der Nachricht zurückgegeben.

Syntax

```
#Include ADWGCAN.Inc
ret_val = Read_Msg_Con(channel,msg_no)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle.	LONG
msg_no	Nummer (1…15) des Message-Objekts.	LONG
ret_val	-1: keine neue Nachricht >0:Neue Nachricht; ret_val = Identifier der Nachricht	LONG

Bemerkungen

Im Unterschied zu **Read_Msg** stellt **Read_Msg_Con** sicher, dass die Nachricht konsistent ist: Wenn während des Auslesens eine neue Nachricht eintrifft, kann es nicht zu einer Mischung der alten und der neuen Nachricht kommen.

Um eine Nachricht zu empfangen, müssen Sie folgende Reihenfolge einhalten:

- Einmal: Geben Sie das Message-Objekt mit **En_Receive** zum Empfangen frei.
- Sooft erforderlich: Prüfen Sie auf eine neue Nachricht und – falls vorhanden – speichern die Nachricht in **CAN_MSG** mit **Read_Msg**.

Sie können eine empfangene Nachricht nur einmal auslesen.

Siehe auch

CAN_Msg, En_CAN_Interrupt, En_Receive, En_Transmit, Read_Msg

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Wenn eine neue Nachricht mit dem passenden Identifier
Rem empfangen wurde, werden die Daten gelesen. Die
Rem ersten 4 Bytes der Nachricht werden zu einer Fließkomma-
Rem Zahl mit 32 Bit Länge zusammengesetzt.
Dim n AS LONG

INIT:
  PAR_1 = 0
  Init_CAN(1)                   'CAN-Controller 1 initialisieren
  En_Receive(1,1,40,0)          'Message-Objekt 1 initialisieren
                                'zum Empfangen von
CAN-Nachrichten mit
                                'dem Identifier 40


EVENT:
  Rem Wenn das Message-Objekt geändert wurde, werden die
  Rem empfangenen Daten aus Objekt 1 gelesen und der
  Rem Identifier an PAR_9 übergeben.
  Rem Die Daten stehen im Feld CAN_Msg[] bereit.
  PAR_9 = Read_Msg_Con(1,1)

  If (PAR_9 = 40) THEN
    Rem Für das Message-Objekt ist eine neue Nachricht mit dem
    Rem Identifier 40 eingetroffen
    PAR_1 = CAN_Msg[1]          'High-Byte auslesen
    FOR n = 2 TO 4              'Mit restlichen 3 Bytes zu 32
Bit-Zahl
      PAR_1 = Shift_Left(PAR_1,8) + CAN_Msg[n] 'zusammenfügen
    NEXT n
    Rem Das Bitmuster in PAR_1 in den Datentyp FLOAT wandeln und
    Rem der Variablen FPAR_1 zuweisen.
    FPAR_1 = Cast_LongToFloat(PAR_1)
  ENDIF
```

Senden einer Fließkomma-Zahl siehe Bsp. bei Transmit.

Set_CAN_Baudrate

Set_CAN_Baudrate stellt die Baudrate des Controllers einer CAN-Schnittstelle ein.

Syntax

```
#Include ADWGCAN.Inc
ret_val = Set_CAN_Baudrate(channel, rate)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle	LONG
rate	Baudrate des CAN-Controllers in Bit/Sekunde.	LONG
ret_val	Status der Befehlsausführung: 0: Baudrate wurde eingestellt 1: Baudrate unzulässig	LONG

Bemerkungen

Die möglichen Baudraten (= Busfrequenzen) entnehmen Sie bitte der Tabelle „Einstellbare Baudraten“ im Anhang. Übernehmen Sie bitte die genaue Schreibweise, d.h. nicht ganzzahlige Baudraten mit 4 Nachkommastellen; Werte mit abweichender Schreibweise werden als nicht zulässig zurückgewiesen.

Die Anweisung führt folgende Aktionen aus:

- Prüfen, ob die übergebene Baudrate zulässig ist. Falls nicht, dann den Rückgabewert auf 1 setzen und die Bearbeitung beenden.
- Die Register des CAN-Controllers für die Baudrate setzen.
- Sampling Mode auf 0 setzen: Ein Sample pro Bit.
- Die Einstellungen so wählen, dass der Sample-Punkt immer zwischen 60% und 72% der Gesamt-Bitlänge liegt.
- Die Sprungweite zur Synchronisation auf 1 setzen.

In Sonderfällen kann es vorteilhaft sein, die Einstellungen anders zu wählen als oben beschrieben. Sie finden hierzu eine Erläuterung im Hardware-Handbuch.

Die Anweisung sollte in den Programm-Abschnitten **LOWINIT:** oder **INIT:** aufgerufen werden, und zwar erst nach der Anweisung Initialisierung, weil sonst die eingestellte Baudrate wieder mit der Standardeinstellung (1MBit/s) überschrieben wird.

Siehe auch

Get_CAN_Reg, Init_CAN, Set_CAN_Reg

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

Dim status As Long

INIT:
  Init_CAN(1)                  'CAN-Controller 1 initialisieren
  Rem Baudrate 125 kBit/s setzen
  status = Set_CAN_Baudrate(1,125000)
```

Set_CAN_Reg

Set_CAN_Reg schreibt einen Wert in ein bestimmtes Register des Controllers einer CAN-Schnittstelle.

Syntax

```
#Include ADWGCAN.Inc
Set_CAN_Reg(channel, regno, value)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle	LONG
regno	Register-Nummer (0…255) im CAN-Controller	LONG
value	Wert (8 Bit), der ins Register geschrieben wird.	LONG

Bemerkungen

Sie finden die Registernummern des CAN-Controllers AN82527 im Intel®-Datenblatt.

Siehe auch

Get_CAN_Reg, Init_CAN, Set_CAN_Baudrate

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

INIT:
  Init_CAN(1)                 'CAN-Controller 1 initialisieren
  Set_CAN_Reg(1,0,1)          'Control-Register auf den Wert 1
                              'setzen
```

Transmit

Transmit sendet die Nachricht in CAN_Msg über ein bestimmtes Message-Objekt einer CAN-Schnittstelle.

Syntax

```
#Include ADWGCAN.Inc
Transmit(channel, msg_no)
```

Parameter

can_no	Nummer (1, 2) der CAN-Schnittstelle	LONG
msg_no	Nummer (1…14) des Message-Objektes	LONG

Bemerkungen

Um eine Nachricht zu senden, müssen Sie folgende Reihenfolge einhalten:

- Einmal: Geben Sie das Message-Objekt mit **En_Transmit** zum Senden frei.
- Sooft erforderlich: Geben Sie die Nachricht in das Feld **CAN_MSG** ein: Die Datenbytes und die Anzahl der Datenbytes.
- Senden Sie die Nachricht mit **Transmit**.

Die CAN-Schnittstelle sendet die Nachricht, sobald das Message-Objekt Zugriffsrecht auf den CAN-Bus hat.

Siehe auch

CAN_Msg, En_Transmit, Init_CAN, Set_CAN_Baudrate

Gültig für

Gold-CAN

Beispiel

```
  #Include ADWGCAN.Inc
Rem Sendet eine 32 Bit-FLOAT-Zahl (hier: Pi) als Folge von
Rem 4 Bytes in einem Message-Objekt
#Define pi 3.14159265
Dim i AS LONG

INIT:
 Init_CAN(2)              'CAN-Controller 2
                          'initialisieren

 Rem Initialisiere das Message-Objekt 6 der Schnittstelle
  2
 Rem zum Senden von CAN-Nachrichten mit dem Identifier 40
 En_Transmit(2, 6,40,0)

 Rem Bitmuster von Pi mit Datenformat Long erzeugen
 PAR_1 = Cast_FloatToLong(pi)

 Rem Bitmuster (32 Bit) in 4 Bytes aufteilen
 CAN_Msg[4] = PAR_1 AND 0FFh 'LSB zuweisen
 FOR i = 1 TO 3
  CAN_Msg[4-i] = Shift_Right(PAR_1,8*i) AND 0FFh
 NEXT i
 CAN_Msg[9] = 4            'Länge der Nachricht in Bytes

EVENT:
 Transmit(2,6)             'Message-Objekt 6 senden
```

Empfangen einer Fließkomma-Zahl siehe Bsp. bei Read_Msg.

12.5 RSxxx-Schnittstelle

Dieser Abschnitt beschreibt Befehle zum Ansprechen der RSxxx-Schnittstellen auf *ADwin-Gold-CAN*:

Check_Shift_Reg

Check_Shift_Reg gibt zurück, ob alle Daten gesendet sind, die in den Sende-FIFO der RSxxx-Schnittstelle geschrieben wurden.

Syntax

```
#Include ADWGCAN.Inc
ret_val = Check_Shift_Reg(interface)
```

Parameter

interface	Nummer (1, 2) der Schnittstelle, deren Sende-Status geprüft wird.	LONG
ret_val	Sende-Status: 0: Daten sind gesendet (= keine Daten im Sende-FIFO vorhanden). 1: Noch nicht alle Daten gesendet (= im Sende-FIFO sind noch Daten vorhanden).	LONG

Bemerkungen

Bei dem Rückgabewert 0 ist sowohl das Sende-FIFO als auch das Ausgangs-Shiftregister leer. Bei dem Rückgabewert 1 ist mindestens ein Bit noch nicht gesendet.

Benutzen Sie diesen Befehl nur, wenn Sie sich bereits eingehend mit dem eingesetzten Controller vertraut gemacht haben (Datenblatt des Herstellers Texas Instruments). Für allgemeine Anwendungen stehen Ihnen komfortablere Befehle aus der Include-Datei zur Verfügung.

Siehe auch

Get_RS, RS_Init, RS_Reset, Write_FIFO

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

EVENT:
  Rem …
  Rem Prüft, ob Schnittstelle 1 noch Daten zu senden hat
  PAR_1 = Check_Shift_Reg(1)
  Rem …
```

Get_RS

Get_RS liest den Inhalt eines bestimmten Controller-Registers aus.

Syntax

```
#Include ADWGCAN.Inc
ret_val = Get_RS(reg_addr)
```

Parameter

reg_addr	Adresse des zu lesenden Controller-Registers.	LONG
ret_val	Inhalt des Controller-Registers.	LONG

Bemerkungen

Benutzen Sie diesen Befehl nur, wenn Sie sich bereits eingehend mit dem eingesetzten Controller vertraut gemacht haben (Datenblatt des Herstellers Texas Instruments). Für allgemeine Anwendungen stehen Ihnen komfortablere Befehle aus der Include-Datei zur Verfügung.

Siehe auch

Check_Shift_Reg, RS_Init, RS_Reset, Set_RS

Gültig für

Gold-CAN

Beispiel

- / -

Read_FIFO

Read_Fifo liest einen Wert aus dem Eingangs-FIFO einer bestimmten Schnittstelle.

Syntax

```
#Include ADWGCAN.Inc
ret_val = Read_Fifo(interface)
```

Parameter

interface	Nummer (1, 2) der auszulesenden Schnittstelle	LONG
ret_val	Inhalt des Eingangs-FIFO: -1: FIFO ist leer ≥0:Übertragener Datenwert	LONG

Bemerkungen

-/-

Siehe auch

RS_Init, RS_Reset, RS485_Send, Write_FIFO

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

INIT:
  RS_Reset()
  Rem Schnittstelle 1 Initialisieren:  9600 Baud, ohne Parität,
  Rem 8 Datenbits, 1 Stoppbit und Hardwarehandshake.
  RS_Init(1,9600,0,8,0,1)

EVENT:
  Rem Einen Wert aus dem FIFO holen. Wenn der FIFO leer ist,
wird -1
  Rem zurückgeliefert.
  PAR_1 = Read_Fifo(1)
```

RS485_Send

RS485_Send legt die Übertragungsrichtung für eine bestimmte Schnittstelle fest.

Syntax

```
#Include ADWGCAN.Inc
RS485_Send(interface,dir)
```

Parameter

interface	Einzustellende Schnittstelle (1, 2)	LONG
dir	Übertragungsrichtung der Schnittstelle: 0: Schnittstelle als Empfänger einstellen. 1: Schnittstelle als Sender einstellen. 2: Schnittstelle als Sender einstellen, der gleichzeitig die gesendeten Daten empfängt. 3: Schnittstelle stumm schalten, d.h. die Schnittstelle arbeitet als Empfänger, nimmt aber keine Daten in den Eingangs-FIFO auf.	LONG

Bemerkungen

Die Einstellung der Übertragungsrichtung bedeutet:

- Empfänger: Der Controller kann Daten auf dem Bus ausschließlich lesen, auch wenn Daten im Ausgangs-FIFO liegen.
- Sender: Der Controller kann Daten auf den Bus legen, die von anderen Teilnehmern gelesen werden können.
- Sender/Empfänger: Der Controller kann Daten auf den Bus legen und gleichzeitig zurücklesen. Dadurch ist eine Überprüfung der ausgegebenen Daten möglich.

Siehe auch

Check_Shift_Reg, Get_RS, RS_Init, RS_Reset, Set_RS

Gültig für

Gold-CAN

Beispiel

- / -

RS_Init

RS_Init initialisiert die angegebene Schnittstelle.

Folgende Kennwerte werden gesetzt:

- Übertragungsgeschwindigkeit in Baud
- Anwendung von Prüf-Bits
- Datenlänge
- Anzahl der Stopp-Bits
- Übertragungs-Protokoll (Handshake)

Syntax

```
#Include ADWGCAN.Inc
RS_Init(interface,baud,parity,bits,stop,handshake)
```

Parameter

interface	Nummer (1, 2) der Schnittstelle, die initialisiert werden soll.	LONG
baud	Übertragungsgeschwindigkeit in Baud: RS232: 35 … 115.200 RS485: 35 … 2.304.000	LONG
parity	Anwendung von Prüf-Bits: 0: ohne Paritäts-Bit 1: gerade Parität (even) 2: ungerade Parität (odd)	LONG
bits	Anzahl der Daten-Bits (5, 6, 7 oder 8).	LONG
stop	Anzahl der Stopp-Bits 0: 1 Stopp-Bit 1: 1½ Stopp-Bits bei 5 Daten-Bits; 2 Stopp-Bits bei 6, 7 oder 8 Daten-Bits	LONG
handshake	Übertragungs-Protokoll: 0: RS232, kein Handshake 1: RS232, Hardware Handshake (RTS/CTS) 2: RS232, Software-Handshake (Xon/Xoff) 3: RS485 (Voreinstellung)	LONG

Bemerkungen

RS_Init ist vor dem ersten Arbeiten mit der gewählten Schnittstelle notwendig, um deren Parameter einzustellen. Sie müssen für eine korrekte Übertragung mit der Gegenstelle identisch sein.

Die Initialisierung ist auch dann erforderlich, wenn Sie mit **RS_Reset** einen Hardware-Reset ausgeführt haben.

Wenn das Übertragungs-Protokoll RS485 eingestellt wird, muss auch die Übertragungsrichtung festgelegt werden (mit **RS485_Send**).

Eine Liste gängiger Baudraten finden Sie auf Seite 41 (Abb. 26).

Siehe auch

Check_Shift_Reg, Get_RS, RS485_Send, RS_Reset, Set_RS

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

INIT:
  RS_Reset()                    'RS-Controller zurücksetzen
  RS_Init(1,9600,0,8,0,1)       'Initialisierung von
Schnittstelle 1
                                'mit 9600 Baud, ohne Parität,
                                '8 Datenbits, 1 Stoppbit und
                                'Hardware-Handshake.
```

RS_Reset

RS_Reset führt einen Hardware-Reset des RSxxx-Controllers durch.

Syntax

```
#Include ADWGCAN.Inc
RS_Reset()
```

Parameter

- / -

Bemerkungen

RS_Reset sendet einen Reset-Impuls auf den entsprechenden Eingang des Controllers TL16C754. Sie können dem Datenblatt des Controllers 16C754 von Texas Instruments entnehmen, auf welche Werte die Register durch den Hardware-Reset gesetzt werden.

Nach einem Hardware-Reset muss eine Initialisierung mit **RS_Init** folgen, um den Controller zu initialisieren und die gewünschten Schnittstellen-Parameter einzustellen.

Siehe auch

Check_Shift_Reg, Get_RS, RS_Init, Set_RS

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc

INIT:
  RS_Reset()                    'RSxxx Controller zurücksetzen
  RS_Init(1,9600,0,8,0,1)       'Initialisierung von
Schnittstelle 1
                                'mit 9600 Baud, ohne Parität,
                                '8 Datenbits, 1 Stoppbit und
                                'Hardware-Handshake.
```

Set_RS

Set_RS schreibt einen Wert in ein bestimmtes Register.

Syntax

```
#Include ADWGCAN.Inc
Set_RS(reg_addr,value)
```

Parameter

reg_addr	Nummer des zu beschreibenen Registers	LONG
value	Wert, der in das Register geschrieben werden soll	LONG

Bemerkungen

Benutzen Sie diesen Befehl nur, wenn Sie sich bereits eingehend mit dem eingesetzten Controller vertraut gemacht haben (Datenblatt des Herstellers: TL16C754 von Texas Instruments). Für allgemeine Anwendungen stehen Ihnen komfortablere Befehle aus der Include-Datei zur Verfügung.

Siehe auch

Get_RS, RS_Init, RS_Reset

Gültig für

Gold-CAN

Beispiel

- / -

Write_FIFO

Write_FIFO schreibt einen Wert in den Sende-FIFO einer bestimmten Schnittstelle.

Syntax

```
#Include ADWGCAN.Inc
ret_val = Write_FIFO(interface,value)
```

Parameter

interface	Nummer (1, 2) der Schnittstelle, deren Sende-FIFO beschrieben wird	LONG
value	Wert der ins Sende-FIFO geschrieben werden soll	LONG
ret_val	Statusmeldung: 0: Daten wurden erfolgreich geschrieben. 1: Daten konnten nicht geschrieben werden, Sende-FIFO ist voll.	LONG

Bemerkungen

Die Anweisung prüft zuerst, ob noch mindestens ein Speicherplatz im Sende-FIFO frei ist. Ist dies der Fall, wird der übergebene Wert ins FIFO geschrieben (Rückgabewert 0); anderenfalls wird eine 1 zurückgeliefert, die angibt, dass das FIFO voll ist und ein Schreiben nicht möglich war.

Siehe auch

Check_Shift_Reg, Read_FIFO, RS_Init, RS_Reset, RS485_Send

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Dim val As Long

INIT:
  RS_Reset()
  Rem Initialisierung von Schnittstelle 1 mit 9600 Baud,
  Rem keine Parität, 8 Datenbits, 1 Stoppbit und
  Rem Hardware-Handshake.
  RS_Init(1,9600,0,8,0,1)

EVENT:
  Rem Ist das FIFO nicht voll, wird val ins FIFO geschrieben.
  Rem Wenn das FIFO-Feld voll ist, wird dies mit dem Wert 1
  Rem in PAR_1 angezeigt.
  PAR_1 = Write_FIFO(1,val)
```

12.6 SSI-Schnittstelle

Dieser Abschnitt beschreibt Befehle zum Ansprechen der SSI-Decoder auf *ADwin-Gold-CAN*:

SSI_Mode

SSI_Mode stellt den Modus aller SSI-Decoder ein, entweder „single shot“ (einzeln lesen) und „continuous“ (kontinuierlich lesen).

Syntax

```
#Include ADWGCAN.Inc
SSI_Mode(pattern)
```

Parameter

pattern	Betriebsmodus der SSI-Decoder, angegeben als Bitmuster. Jedem Decoder ist ein Bit zugeordnet (siehe Tabelle). Bit = 0: Modus „Single shot“, der Encoder wird einmal ausgelesen. Bit = 1: Modus „Continuous“, der Encoder wird kontinuierlich ausgelesen.	LONG

Bitnr.	31:2	3	2	1	0
SSI-Decoder	–	4	3	2	1

Bemerkungen

Wenn Sie den Modus „continuous“ wählen, startet das Auslesen des entsprechenden Encoders sofort. **SSI_Start** ist hierzu nicht erforderlich.

Manche Encoder-Typen liefern im Modus „continuous“ gelegentlich den falschen Messwert 0 (Null) anstelle des korrekten Messwerts zurück. Im Modus „single shot“ tritt dieser Fehler nicht auf.

Siehe auch

SSI_Read, SSI_Set_Bits, SSI_Set_Clock, SSI_Start, SSI_Status

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Decoder 1 läuft mit 1.0 MHz, Decoder 2 mit 0,4 MHz
INIT:
  SSI_Set_Clock(1,10)          'Taktrate einstellen, Decoder 1
  SSI_Set_Clock(2,25)          'Taktrate einstellen, Decoder 2
  SSI_Mode(11b) 'Continuous-Mode setzen, Decoder 1+2
  SSI_Set_Bits(1,10)           '10 Encoder-Bits,  Encoder 1
  SSI_Set_Bits(2,25)           '25 Encoder-Bits, Encoder 2

EVENT:
  PAR_1 = SSI_Read(1)          'Pos.wert auslesen, Encoder 1
  PAR_2 = SSI_Read(2)          'Pos.wert auslesen, Encoder 2
```

SSI_Read

SSI_Read gibt den zuletzt gespeicherten Zählerstand eines bestimmten SSI-Zählers zurück.

Syntax

```
#Include ADWGCAN.Inc
ret_val = SSI_Read(dcdr_no)
```

Parameter

dcdr_no	Nummer (1…4) des SSI-Decoders, dessen Zählerstand auszulesen ist.	LONG
ret_val	Letzter Zählerstand des SSI-Zählers (= Absolutwert-Position des Encoders)	LONG

Bemerkungen

Ein Encoder-Wert wird dann gespeichert, wenn die durch **SSI_Set_Bits** angegebene Anzahl von Bits eingelesen wurde.

Siehe auch

SSI_Mode, SSI_Set_Bits, SSI_Set_Clock, SSI_Start, SSI_Status

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Decoder 1 läuft mit 200 kHz
Dim m, n, y AS LONG

INIT:
  Rem Einstellungen für Decoder 1
  SSI_Set_Clock(1,50)          'Taktrate
  SSI_Mode(1)                  'Continuous-Mode
  SSI_Set_Bits(1,23)           '23 Encoder-Bits

EVENT:
  PAR_1 = SSI_Read(1)          'Pos.wert auslesen

  Rem Wert von Gray-Code in Binärwert wandeln:
  m = 0                        'vorigen Wert löschen
  y = 0                        ' -"-
  FOR n = 1 TO 32              'Alle 32 mögl. Bits durchgehen
    m = (Shift_Right(PAR_1,(32 - n)) AND 1) XOR m
    y = (Shift_Left(m,(32 - n))) OR y
  NEXT n
  Rem Das Ergebnis der Gray-/Binär-Wandlung in PAR_9
  PAR_9 = y
```

SSI_Set_Bits

SSI_Set_Bits stellt für einen bestimmten SSI-Zähler die Anzahl der zu Bits ein, die einen vollständigen Encoder-Wert bilden.

Die Zahl der Bits sollte mit der Auflösung des Encoders identisch sein.

Syntax

```
#Include ADWGCAN.Inc
SSI_Set_Bits(dcdr_no,bit_count)
```

Parameter

dcdr_no	Nummer (1…4) des SSI-Decoders, dessen Status gefragt ist.	LONG
bit_count	Anzahl der Bits (1…32) der zu lesenden Bits für einen Encoder-Wert (entspricht der Encoder-Auflösung).	LONG

Bemerkungen

Die Auflösung (Anzahl der Bits) des SSI-Encoders sollte mit der Anzahl der zu übertragenden Bits übereinstimmen.

Achten Sie darauf, dass die zu lesenden Bits mit der Encoder-Auflösung genau übereinstimmen.

Siehe auch

SSI_Mode, SSI_Read, SSI_Set_Clock, SSI_Start, SSI_Status

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Decoder 1 läuft mit 1.0 MHz, Decoder 2 mit 0,4 MHz
INIT:
  SSI_Set_Clock(1,10)          'Taktrate einstellen, Decoder 1
  SSI_Set_Clock(2,25)          'Taktrate einstellen, Decoder 2
  SSI_Mode(11b) 'Continuous-Mode setzen, Decoder 1+2
  SSI_Set_Bits(1,10)           '10 Encoder-Bits,  Encoder 1
  SSI_Set_Bits(2,25)           '25 Encoder-Bits, Encoder 2

EVENT:
  PAR_1 = SSI_Read(1)          'Pos.wert auslesen, Encoder 1
  PAR_2 = SSI_Read(2)          'Pos.wert auslesen, Encoder 2
```

SSI_Set_Clock

SSI_Set_Clock stellt die Taktrate (ca. 40 kHz bis 1 MHz) ein, mit der der Encoder getaktet wird.

Syntax

```
#Include ADWGCAN.Inc
SSI_Set_Clock(dcdr_no,prescale)
```

Parameter

dcdr_no	Nummer (1…4) des SSI-Decoders, dessen Status gefragt ist.	LONG
prescale	Teilerfaktor (10…255) zur Einstellung der Taktrate nach der Formel: Taktrate = 10 MHz / prescale	LONG

Bemerkungen

Teilerfaktoren kleiner 10 werden automatisch auf den Wert 10 korrigiert; bei Werten über 255 werden die niederwertigsten 8 Bits als Teilerfaktor verwendet.

Die mögliche Taktfrequenz ist abhängig von Kabellänge, Kabeltyp und den jeweils verwendeten Sende- und Empfangsbausteinen des Encoders und des Decoders. Grundsätzlich gilt als Regel: Je höher die Taktfrequenz, desto kürzer die mögliche Kabellänge.

Siehe auch

SSI_Mode, SSI_Read, SSI_Set_Bits, SSI_Start, SSI_Status

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Decoder 1 läuft mit 1.0 MHz, Decoder 2 mit 0,4 MHz
INIT:
  SSI_Set_Clock(1,10)          'Taktrate einstellen, Decoder 1
  SSI_Set_Clock(2,25)          'Taktrate einstellen, Decoder 2
  SSI_Mode(11b) 'Continuous-Mode setzen, Decoder 1+2
  SSI_Set_Bits(1,10)           '10 Encoder-Bits,  Encoder 1
  SSI_Set_Bits(2,25)           '25 Encoder-Bits, Encoder 2

EVENT:
  PAR_1 = SSI_Read(1)          'Pos.wert auslesen, Encoder 1
  PAR_2 = SSI_Read(2)          'Pos.wert auslesen, Encoder 2
```

SSI_Start

SSI_Start startet das Auslesen der gewählten SSI-Encoder (nur im Modus „single shot“).

Syntax

```
#Include ADWGCAN.Inc
SSI_Start(pattern)
```

Parameter

pattern — Bitmuster zur Auswahl der SSI-Decoder, die gestartet werden sollen: LONG
Bit = 0: keine Funktion
Bit = 1: Auslesen des SSI-Decoders starten

Bitnr.	31:2	3	2	1	0
SSI-Decoder	–	4	3	2	1

Bemerkungen

Im Modus „continuous“ ist diese Anweisung ohne Funktion, weil dann die Encoder-Werte ohnehin kontinuierlich ausgelesen werden.

Ein Encoder-Wert wird dann gespeichert, wenn die durch **SSI_Set_Bits** angegebene Anzahl von Bits eingelesen wurde.

Es wird immer ein vollständiger Encoder-Wert übertragen, auch wenn währenddessen der Modus umgestellt wird.

Siehe auch

SSI_Mode, SSI_Read, SSI_Set_Bits, SSI_Set_Clock, SSI_Status

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Beide Decoder laufen mit 40 kHz
INIT:
  SSI_Set_Clock(1,250)          'Taktrate für Decoder 1
einstellen
  SSI_Set_Clock(2,250)          'Taktrate für Decoder 2
einstellen
  SSI_Mode(0)                   'Single shot-Mode einstellen
                                'für alle Zähler
  SSI_Set_Bits(1,23)            '23 Encoder-Bits auf Encoder 1
  SSI_Set_Bits(2,23)            '23 Encoder-Bits auf Encoder 2

EVENT:
  SSI_Start(11b)                'Positionswert von Encoder 1&2
lesen
  DO                            'Für Encoder 1:
  UNTIL (SSI_Status(1) = 0)  'Wenn Positionswert komplett
                                'gelesen ist …
  PAR_1 = SSI_Read(1)           'Positionswert auslesen
  DO                            'Für Encoder 2:
  UNTIL (SSI_Status(2) = 0)  'Wenn Positionswert komplett
                                'gelesen ist …
  PAR_1 = SSI_Read(2)           'Positionswert auslesen
```

SSI_Status

SSI_Status liefert für einen bestimmten Decoder den aktuellen Lese-Status zurück.

Syntax

```
#Include ADWGCAN.Inc
ret_val = SSI_Status(dcdr_no)
```

Parameter

dcdr_no	Nummer (1…4) des SSI-Decoders, dessen Status gefragt ist.	LONG
ret_val	Lese-Status des Decoders: 0: Decoder ist bereit, d.h. ein vollständiger Wert wurde gelesen. 1: Decoder liest einen Encoder-Wert ein.	LONG

Bemerkungen

Verwenden Sie die Status-Abfrage nur im SSI-Modus „single shot“. Im Modus „continuous“ ist eine Status-Abfrage nicht sinnvoll.

Siehe auch

SSI_Mode, SSI_Read, SSI_Set_Bits, SSI_Set_Clock, SSI_Start

Gültig für

Gold-CAN

Beispiel

```
#Include ADWGCAN.Inc
Rem Beide Decoder laufen mit 40 kHz
INIT:
  SSI_Set_Clock(1,250)        'Taktrate für Decoder 1
einstellen
  SSI_Set_Clock(2,250)        'Taktrate für Decoder 2
einstellen
  SSI_Mode(0)                 'Single shot-Mode einstellen
                              'für alle Zähler
  SSI_Set_Bits(1,23)          '23 Encoder-Bits auf Encoder 1
  SSI_Set_Bits(2,23)          '23 Encoder-Bits auf Encoder 2

EVENT:
  SSI_Start(11b)              'Positionswert von Encoder 1&2
lesen
  DO                          'Für Encoder 1:
  UNTIL (SSI_Status(1) = 0) 'Wenn Positionswert komplett
                              'gelesen ist …
  PAR_1 = SSI_Read(1)         'Positionswert auslesen
  DO                          'Für Encoder 2:
  UNTIL (SSI_Status(2) = 0) 'Wenn Positionswert komplett
                              'gelesen ist …
  PAR_1 = SSI_Read(2)         'Positionswert auslesen
```

Anhang

A.1 Technische Daten

Sämtliche technischen Daten beziehen sich auf ein eingeschaltetes *ADwin-Gold*.

Allgemeine Daten / Grenzwerte						
	Symbol	Konditionen	min.	typ.	max.	Einheit
Versorgungs-Spannnung						
Spannung	U_b		10	12	35	V
Ruhestrom, USB-Schnittstelle	I_{idle}	U_b=10V		1,1		A
		U_b=12V[a]		0,9		
		U_b=35V		0,3		
		U_b=12V; Gold-DA		1,4		
Einschaltstrombedarf, USB-Schnittstelle	$I_{power-on}$	U_b=12Va	1,7			
		U_b=12V; Gold-DA	2,9			
Ruhestrom, Ethernet-Schnittstelle	I_{idle}	U_b=10V		1,3		A
		U_b=12Va		1,1		
		U_b=35V		0,4		
		U_b=12V; Gold-DA		1,5		
Einschaltstrombedarf, Ethernet-Schnittstelle	$I_{power-on}$	U_b=12Va	2,1			
		U_b=12V; Gold-DA	3,1			
Zulässiger Betriebsbereich						
Temperatur	$T_{Gehäuse}$		+5		+60	°C
rel. Feuchte	F_{rel}	nicht kondensierend	0		80	%
Lagerung						
Temperatur	T		-20		+70	°C
Steckverbinder						
Sub-D-Verbinder	Metrisches ISO-Gewinde; UNC-Gewinde als Bestelloption erhältlich					
Abmessungen						
Breite × Höhe × Tiefe (Höhe inkl. Buchsen)	B × H × T	Gold-USB, Gold-ENET	214 × 75 × 109			mm
		mit CAN-Erweiterung	Höhe: +20			
		mit Clipsen[b]	Höhe: +7; Tiefe: +26			
Nettogewicht						
Gewicht	m_{Netto}	Gold-USB, Gold-ENET	1320			g
		mit CAN-Erweiterung	1760			
		Clipse[b]	32			

[a] gilt auch für Gold-CO1

[b] Zubehör zur Hutschienenmontage: *Gold-Mount*

Digitale Ein- / Ausgänge						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
I/O-Leitungen						
Anzahl	DIO00:DIO31	32 (in Gruppen zu 8 als Ein-oder Ausgang programmierbar)				
	EVENT	ext. Trigger-Eingang (positive TTL-Logik)				
Eingänge						
max. Eingangsspanng.		V_{CC} = 5V	-0,5		+5,5	V
Logik-Eingangsspannung	V_{IH} (High)	V_{CC} = 5V	2,4			
	V_{IL} (Low)	V_{CC} = 5V			0,8	
Logik-Eingangsstrom	I_I	V_{CC} = 5V		±0,01	±2	µA
Ausgänge						
Logik-Ausgangsspannung	V_{OH} (High)	I_{OH}= -6mA	3,84	4,3		V
	V_{OL} (Low)	I_{OL}= +6mA		0,17	0,33	
Logik-Ausgangsstrom	I_O	je DIO-Leitung			±35	mA
	I_{TOTAL}	alle Digin bzw. alle DIGOUT über V_{CC} / GND			±70	
EVENT-Eingang						
Flankenerkennung, pos.	V_{T+} (Low)	V_{CC} = 5V	1,65	1,9	2,15	V
Schalthysterese	V_{T+} - V_{T-}		0,4	0,9		
Eingangsstrom	I_{IH}	V_I = 2,7V			20	µA
	I_{IL}	V_I = 0,4V			-50	

Analoge Ein- / Ausgänge						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
Eingänge						
Anzahl	2 × 8 über Multiplexer, differentiell					
Eingangswiderstand	R_i		323,4	330	336,6	kΩ
Spannungsfestigkeit	U_{in} max.	ON & OFF			±35	V
Multiplexer-Einschwingzeit	t_{MUX}	1 LSB 14 Bit		2,5		µs
		1 LSB 16 Bit		6,5		µs
ADC 14Bit						
Konvertierungszeit	t_{conv}				0,5	µs
Messbereich	U_{in}	F_V=1	-10		+9,999695	V
		F_V=2	-5		+4,999847	
		F_V=4	-2,5		+2,499924	
		F_V=8	-1,25		+1,249962	
Diff. Gleichtaktspanng.					±2,5	
Integrale Nichtlinearität	INL			±1	±3	LSB
Different. Nichtlinearität	DNL			±0,25	±0,5	

Analoge Ein- / Ausgänge						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
Offset	Drift[a]			±2		ppm/K
	Fehler	abgleichbar				
Gain	Drift[a]			±5		ppm/K
	Fehler	abgleichbar				
ADC 16Bit						
Konvertierungszeit	t_{conv}				5	µs
Messbereich	U_{in}	F_V=1	-10		+9,999695	V
		F_V=2	-5		+4,999847	
		F_V=4	-2,5		+2,499924	
		F_V=8	-1,25		+1,249962	
Diff. Gleichtaktspanng.					±2,5	
Integrale Nichtlinearität	INL			±1	±3	LSB
Different. Nichtlinearität	DNL			±0,25	±0,5	
Offset	Drift[a]			±2		ppm/K
	Fehler	abgleichbar				
Gain	Drift[a]			±5		ppm/K
	Fehler	abgleichbar				
Ausgänge: DAC 16 Bit						
Anzahl	2 (mit DA-Erweiterung: 8)					
Ausgangsspannung	U_{out}		-10		+9,999695	V
Einschwingzeit	t_{settle}	2V-Sprung		3		µs
		FSR [b] (20V)		10		
Zulässiger Strom					±25	mA
Integrale Nichtlinearität	INL				±2	LSB
Different. Nichtlinearität	DNL				±1	
Offset	Drift[a]			±1		ppm/K
	Fehler	abgleichbar				
Gain	Drift[a]			±3		ppm/K
	Fehler	abgleichbar				

[a] bezogen auf den gesamten Spannungsbereich (FSR)

[b] Full Scale Range

Prozessor						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
Typ	ADSP21062 (SHARC™)					
Hersteller	Analog Devices					
Taktfrequenz	f_{CLK}			40		MHz
Register-Breite				32		Bit
Interner Speicher	SRAM	für Programm		128	256 [a]	kByte
		für Daten		128	256 a	

Prozessor						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
Externer Speicher	SDRAM			16	64 a	MByte

[a] kombinierte Speichererweiterung G-MEM-64-512

CO1-Erweiterung						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
Zähler						
Anzahl	4 Zähler (CNTR1 … CNTR4)					
Eingänge	Je Zähler 3 differentielle Eingänge (A/CLK, B/DIR, CLR/LATCH); Zähler paarweise programmierbar mit differentiellen oder single-ended Eingängen.					
Zählerbreite				32		Bit
Zählfrequenz	f_{CLK}	Eingang CLK		20		MHz
		Eingang A/B		5		
Latch-Breite	LATCH			32		Bit
Referenz-Quarzoszillator						
Referenzfrequenz	f_{ref}			20		MHz
Vorteiler durch 4	f_{ref} / 4			5		
Genauigkeit und Drift					100	ppm
Zählereingänge differentiell[a]						
Differentielle Eingangs-Schwellenspannung	V_{TH}	-10V ≤ V_{CM} ≤ 13,2V	-200		+200	mV
Schalthysterese	ΔV_{TH}	-10V ≤ V_{CM} ≤ 13,2V		40		mV
Bereich der Gleichtaktspannung	V_{CM}		-10		+13,2	V
Anstieg-/Abfallgeschw. der differentiellen Spg.			0,33			V/µs
Zulässige differentielle Eingangsspannung		für jeden Eingang			±3,9	V
Zählereingänge single ended[b] (mit Schmitt-Trigger)						
Flankenerkennung, pos.	V_{T+} (Low)	V_{CC} = 5V	1,65	1,9	2,15	V
Flankenerkennung, neg.	V_{T-} (Low)		0,75	1,0	1,25	
Schalthysterese	V_{T+} - V_{T-}		0,4	0,9		
Eingangsstrom	I_H	V_I = 2,7V			20	µA
	I_L	V_I = 0,4V			-50	

[a] siehe auch Datenblatt MAX3098 von MAXIM

[b] siehe auch Datenblatt 74LS19 von Texas Instruments

A.2 Hardware-Adressen

Hardware-Adressen der ADC

Adresse [HEX]	Funktion	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Kommentar
20400000	MUX 1 setzen: Kanäle 1, 3, 5, …, 15	-	-	-	-	-	-	-	-	-	n	n	n	„nnn“ binär = 0…7 dezimal, gewählter Kanal = nnn + 1
	MUX 2 setzen: Kanäle 2, 4, 6, …, 16	-	-	-	-	-	-	n	n	n	-	-	-	„nnn“ binär = 0…7 dezimal, gewählter Kanal = 2 (nnn + 1)
	Verstärkung PGA 1	-	-	-	-	g	g	-	-	-	-	-	-	„gg“ binär = 0…3 dezimal, gewählte Verstärkung = 2gg
	Verstärkung PGA 2	-	-	g	g	-	-	-	-	-	-	-	-	
20400010	Konvertierung starten: ADC 1 (16 Bit)	-	-	-	-	-	-	-	-	-	1	-	s	s = 0 : Konvertierung starten s = 1 : kein Einfluss
	Konvertierung starten: ADC 2 (16 Bit)	-	-	-	-	-	-	-	-	-	1	s	-	
	Konvertierung starten: ADC 1 (14 Bit)	-	-	-	-	-	-	-	-	s	1	-	-	
	Konvertierung starten: ADC 2 (14 Bit)	-	-	-	-	-	-	-	s	-	1	-	-	
20400020	EOC-Status: ADC 1 (16 Bit)	-	-	-	-	-	-	-	-	-	-	-	e	e = 0 : Konvertierung beendet e = 1 : Konvertierung läuft
	EOC-Status: ADC 2 (16 Bit)	-	-	-	-	-	-	-	-	-	-	e	-	
	EOC-Status: ADC 1 (14 Bit)	-	-	-	-	-	-	-	-	e	-	-	-	
	EOC-Status: ADC 2 (14 Bit)	-	-	-	-	-	-	-	e	-	-	-	-	
20400030	Register auslesen: ADC 1 (16 Bit)	-	x	x	x	x	x	x	x	x	x	x	x	x : Ergebnis der Konvertierung
20400040	Register auslesen: ADC 2 (16 Bit)	-	x	x	x	x	x	x	x	x	x	x	x	
20400130	Register auslesen: ADC 1 (14 Bit)	-	x	x	x	x	x	x	x	x	x	0	0	
20400140	Register auslesen: ADC 2 (14 Bit)	-	x	x	x	x	x	x	x	x	x	0	0	
20400100	Register auslesen und Konvertierg. starten: ADC 1 (16 Bit)	-	x	x	x	x	x	x	x	x	x	x	x	
20400110	Register auslesen und Konvertierg. starten: ADC 2 (16 Bit)	-	x	x	x	x	x	x	x	x	x	x	x	
20400120	Register auslesen und Konvertierg. starten: ADC 1 (14 Bit)	-	x	x	x	x	x	x	x	x	x	0	0	
204001D0	Register auslesen und Konvertierg. starten: ADC 2 (14 Bit)	-	x	x	x	x	x	x	x	x	x	0	0	

Hardware-Adressen der DAC

Adresse [HEX]	Funktion	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Kommentar
20400010	Konvertierung starten: Alle DAC synchron	-	-	-	-	-	-	-	1	1	s	1	1	s = 0 : Konvertierung starten s = 1 : kein Einfluss
20400050	Register nur beschreiben: DAC 1	-	x	x	x	x	x	x	x	x	x	x	x	x : zu konvertierender Digital-wert
20400060	Register nur beschreiben: DAC 2	-	x	x	x	x	x	x	x	x	x	x	x	
20400070	Register nur beschreiben: DAC 3 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400080	Register nur beschreiben: DAC 4 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400090	Register nur beschreiben: DAC 5 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
204000A0	Register nur beschreiben: DAC 6 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400190	Register nur beschreiben: DAC 7 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
204001A0	Register nur beschreiben: DAC 8 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400200	Register beschreiben und Konvertierung sofort starten: DAC 1	-	x	x	x	x	x	x	x	x	x	x	x	x : zu konvertierender Digital-wert
20400210	Register beschreiben und Konvertierung sofort starten: DAC 2	-	x	x	x	x	x	x	x	x	x	x	x	
20400220	Register beschreiben und Konvertierung sofort starten: DAC 3 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400230	Register beschreiben und Konvertierung sofort starten: DAC 4 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400240	Register beschreiben und Konvertierung sofort starten: DAC 5 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400250	Register beschreiben und Konvertierung sofort starten: DAC 6 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400260	Register beschreiben und Konvertierung sofort starten: DAC 7 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400270	Register beschreiben und Konvertierung sofort starten: DAC 8 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	

Hardware-Adressen der digitalen Ein- und Ausgänge

Adresse [HEX]	Funktion	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Kommentar
204000B0	Eingangs-Register DIO15:00	-	x	x	x	x	x	x	x	x	x	x	x	x : eingelesener Digitalwert
204001B0	Eingangs-Register DIO31:16	-	x	x	x	x	x	x	x	x	x	x	x	
204001C0	Ausgangs-Register DIO15:00	-	x	x	x	x	x	x	x	x	x	x	x	x : auszugebender Digitalwert
204000C0	Ausgangs-Register DIO31:16	-	x	x	x	x	x	x	x	x	x	x	x	

Hardware-Adressen der CO1-Zählererweiterung

Adresse [HEX]	Funktion	Bit 31:04	3	2	1	0	Kommentar
20400204	Latch A auslesen: Zähler 1	x	x	x	x	x	x : Inhalt des Latches
20400208	Latch B auslesen: Zähler 1	x	x	x	x	x	
20400214	Latch A auslesen: Zähler 2	x	x	x	x	x	
20400218	Latch B auslesen: Zähler 2	x	x	x	x	x	
20400224	Latch A auslesen: Zähler 3	x	x	x	x	x	
20400238	Latch B auslesen: Zähler 3	x	x	x	x	x	
20400234	Latch A auslesen: Zähler 4	x	x	x	x	x	
20400238	Latch B auslesen: Zähler 4	x	x	x	x	x	
20400300	Zähler freigeben / sperren: Cnt_Enable()	-	x	x	x	x	x = 0 : Zähler sperren x = 1 : Zähler freigeben
20400304	Umschalten zwischen Zähler-Eingängen für single-ended oder differentiellen Betrieb (nur paarweise)	-	-	-	y	x	x: Eingänge für Zähler 1+2 y: Eingänge für Zähler 3+4 x,y = 0: single-ended x,y = 1: differentiell
20400310	Zähler löschen: Cnt_Clear()[a]	-	x	x	x	x	x = 0 : Kein Einfluss x = 1 : Zähler löschen
20400320	Zähler latchen: Cnt_Latch()[a]	-	x	x	x	x	x = 0 : Kein Einfluss x = 1 : Zähler latchen
20400330	Zählereingang wählen: CLR oder LATCH	-	x	x	x	x	x = 0 : CLR-Eingang x = 1 : LATCH-Eingang
20400340	Impuls-/Ereigniszähler oder Impuls-/Pausenzeitmessung	-	x	x	x	x	x = 0 : Ext. Takteingang x = 1 : Int. Ref.-Takt (20/5 MHz)
20400350	4-Flankenauswertung / CLK+DIR oder 20MHz / 5MHz Referenztakt	-	x	x	x	x	Cnt_Mode = 0: x = 0 : 4-Fl.; x = 1 : CLK+DIR Cnt_Mode = 1: x = 0 : 20 MHz; x = 1 : 5 MHz
20400370	Zähler: Fehlerregister[b]	div. Bits					Fehler-Bits (Cnt_GetStatus)

[a] Dieses Register wird nach der Durchführung automatisch zurückgesetzt.

[b] Dieses Register müssen Sie von Hand zurücksetzen!

A.3 Hardware-Revisionen

Auf der Rückseite des Geräts befindet sich ein Aufkleber mit der Revisionsbezeichnung des Geräts. Die Unterschiede der Revisionsstände sind nachfolgend dargestellt:

Revision	Erst-ausgabe	Änderung zur Vorgänger-Version
A	1998	Erst-Version mit Link-Datenverbindung.
B1	Nov. 2002	Prototyp (firmenintern, keine Auslieferung an Kunden)
B2	Apr. 2003	Datenverbindung zum PC nicht mehr über Link, sondern über Ethernet oder USB. Alle Analogeingänge und Zählereingänge sind nur in der Betriebsart differentiell verfügbar.
B3	Nov. 2003	Zusätzliche TTL-Zählereingänge für die Betriebsart single-ended (alternativ zu den Zählereingängen für die Betriebsart differentiell nutzbar). Neue Option Gold-D mit Sub-D-Buchsen anstelle von BNC-Buchsen.
B4	Dez. 2003	Verschiedene Verbesserungen
B5	März 2004	Verschiedene Verbesserungen Layout-Änderung
B6	Aug. 2004	Verbesserte Ethernet-Schnittstelle (ENET-2) mit höherem Datendurchsatz. Neue Option Gold-CAN mit verschiedenen Kommunikations-Schnittstellen.

A.4 Bezugsadressen

Für die Benutzung eines separaten Netzteils benötigen Sie einen 3-poligen Subminiatur-Rundsteckverbinder. Sie erhalten den Rundsteckverbinder z.B. beim folgenden Hersteller unter der Bestell-Nummer 712 2 99-0406-00-03 (Serie 712):

Franz Binder GmbH + Co. elektrische Bauelemente KG
Rötelstrasse 27
D-74172 Neckarsulm
Tel.: 07132 / 325-0
www.binder-connector.de

A.5 RoHS Konformitätserklärung

Die Richtlinie 2002/95/EG der Europäischen Union zur Beschränkung und Verwendung gefährlicher Stoffe in elektrischen und elektronischen Geräten (RoHS-Richtlinie) ist am 1. Juli 2006 in Kraft getreten.

Dabei handelt es sich um folgende Substanzen:

- Blei (Pb)
- Cadmium (Cd)
- Hexavalentes Chrom (Cr VI)
- Polybromierte Biphenyle (PBB)
- Polybromierte Diphenylether (PBDE)
- Quecksilber (Hg)

Die Produktlinie ***ADwin-Gold*** erfüllt seit Juni 2006 die Voraussetzungen der RoHS-Richtlinie in allen gelieferten Varianten.

A.6 Baudraten für den CAN-Bus

ADwin-Gold-CAN besitzt Schnittstellen für den CAN-Bus. Dort können folgende Baudraten eingestellt werden (low speed maximal bis 125 kHz):

Einstellbare Baudraten [Bit/s]				
1000000.0000	888888.8889	800000.0000	727272.7273	666666.6667
615384.6154	571428.5714	533333.3333	500000.0000	470588.2353
444444.4444	421052.6316	400000.0000	380952.3810	363636.3636
347826.0870	333333.3333	320000.0000	307692.3077	296296.2963
285714.2857	266666.6667	250000.0000	242424.2424	235294.1176
222222.2222	210526.3158	205128.2051	200000.0000	190476.1905
181818.1818	177777.7778	173913.0435	166666.6667	160000.0000
156862.7451	153846.1538	148148.1481	145454.5455	142857.1429
140350.8772	133333.3333	126984.1270	125000.0000	123076.9231
121212.1212	117647.0588	115942.0290	114285.7143	111111.1111
106666.6667	105263.1579	103896.1039	102564.1026	100000.0000
98765.4321	95238.0952	94117.6471	90909.0909	88888.8889
87912.0879	86956.5217	84210.5263	83333.3333	81632.6531
80808.0808	80000.0000	78431.3725	76923.0769	76190.4762
74074.0741	72727.2727	71428.5714	70175.4386	69565.2174
68376.0684	67226.8908	66666.6667	66115.7025	64000.0000
63492.0635	62500.0000	61538.4615	60606.0606	60150.3759
59259.2593	58823.5294	57971.0145	57142.8571	55944.0559
55555.5556	54421.7687	53333.3333	52631.5789	52287.5817
51948.0519	51282.0513	50000.0000	49689.4410	49382.7160
48484.8485	47619.0476	47337.2781	47058.8235	46783.6257
45714.2857	45454.5455	44444.4444	43956.0440	43478.2609
42780.7487	42328.0423	42105.2632	41666.6667	41025.6410
40816.3265	40404.0404	40000.0000	39215.6863	38647.3430
38461.5385	38277.5120	38095.2381	37037.0370	36363.6364
36199.0950	35714.2857	35555.5556	35087.7193	34782.6087
34632.0346	34482.7586	34188.0342	33613.4454	33333.3333
33057.8512	32921.8107	32388.6640	32258.0645	32000.0000
31746.0317	31620.5534	31372.5490	31250.0000	30769.2308
30651.3410	30303.0303	30075.1880	29629.6296	29411.7647
29304.0293	29090.9091	28985.5072	28673.8351	28571.4286
28070.1754	27972.0280	27777.7778	27681.6609	27586.2069
27210.8844	27027.0270	26936.0269	26755.8528	26666.6667
26315.7895	26143.7908	25974.0260	25806.4516	25641.0256
25396.8254	25078.3699	25000.0000	24844.7205	24767.8019
24691.3580	24615.3846	24390.2439	24242.4242	24024.0240
23809.5238	23668.6391	23529.4118	23460.4106	23391.8129
23255.8140	23188.4058	22988.5057	22857.1429	22792.0228
22727.2727	22408.9636	22222.2222	22160.6648	22038.5675
21978.0220	21739.1304	21680.2168	21621.6216	21505.3763
21390.3743	21333.3333	21276.5957	21220.1592	21164.0212
21052.6316	20833.3333	20779.2208	20671.8346	20512.8205
20460.3581	20408.1633	20202.0202	20050.1253	20000.0000
19851.1166	19753.0864	19704.4335	19656.0197	19607.8431
19512.1951	19323.6715	19230.7692	19138.7560	19047.6190
18912.5296	18867.9245	18823.5294	18648.0186	18604.6512

Einstellbare Baudraten [Bit/s]				
18518.5185	18433.1797	18390.8046	18306.6362	18181.8182
18140.5896	18099.5475	18018.0180	17857.1429	17777.7778
17738.3592	17582.4176	17543.8596	17429.1939	17391.3043
17316.0173	17241.3793	17204.3011	17094.0171	17021.2766
16949.1525	16913.3192	16842.1053	16806.7227	16771.4885
16666.6667	16632.0166	16563.1470	16528.9256	16460.9053
16393.4426	16326.5306	16260.1626	16227.1805	16194.3320
16161.6162	16129.0323	16000.0000	15873.0159	15810.2767
15779.0927	15686.2745	15625.0000	15594.5419	15503.8760
15473.8878	15444.0154	15384.6154	15325.6705	15238.0952
15180.2657	15151.5152	15122.8733	15094.3396	15065.9134
15037.5940	15009.3809	14842.3006	14814.8148	14705.8824
14652.0147	14571.9490	14545.4545	14519.0563	14492.7536
14414.4144	14336.9176	14311.2701	14285.7143	14260.2496
14184.3972	14109.3474	14035.0877	13986.0140	13937.2822
13913.0435	13888.8889	13840.8304	13793.1034	13722.1269
13675.2137	13605.4422	13582.3430	13559.3220	13513.5135
13468.0135	13445.3782	13377.9264	13333.3333	13289.0365
13223.1405	13157.8947	13136.2890	13114.7541	13093.2897
13071.8954	13008.1301	12987.0130	12903.2258	12882.4477
12820.5128	12800.0000	12759.1707	12718.6010	12698.4127
12578.6164	12558.8697	12539.1850	12500.0000	12422.3602
12403.1008	12383.9009	12345.6790	12326.6564	12307.6923
12288.7865	12195.1220	12158.0547	12121.2121	12066.3650
12030.0752	12012.0120	11994.0030	11922.5037	11904.7619
11851.8519	11834.3195	11764.7059	11730.2053	11695.9064
11661.8076	11627.9070	11611.0305	11594.2029	11544.0115
11494.2529	11477.7618	11428.5714	11396.0114	11379.8009
11363.6364	11347.5177	11299.4350	11220.1964	11204.4818
11188.8112	11111.1111	11080.3324	11034.4828	11019.2837
10989.0110	10943.9124	10928.9617	10884.3537	10869.5652
10840.1084	10810.8108	10796.2213	10781.6712	10752.6882
10695.1872	10666.6667	10638.2979	10610.0796	10582.0106
10540.1845	10526.3158	10457.5163	10430.2477	10416.6667
10389.6104	10335.9173	10322.5806	10296.0103	10269.5764
10256.4103	10230.1790	10204.0816	10101.0101	10088.2724
10062.8931	10025.0627	10012.5156	10000.0000	9937.8882
9925.5583	9876.5432	9852.2167	9828.0098	9803.9216
9791.9217	9768.0098	9756.0976	9696.9697	9685.2300
9661.8357	9615.3846	9603.8415	9569.3780	9523.8095
9456.2648	9433.9623	9411.7647	9400.7051	9367.6815
9356.7251	9324.0093	9302.3256	9291.5215	9259.2593
9227.2203	9216.5899	9195.4023	9153.3181	9142.8571
9090.9091	9070.2948	9049.7738	9039.5480	9009.0090
8958.5666	8928.5714	8918.6176	8888.8889	8879.0233
8869.1796	8859.3577	8771.9298	8743.1694	8714.5969
8695.6522	8658.0087	8648.6486	8620.6897	8602.1505
8592.9108	8556.1497	8547.0085	8510.6383	8483.5631
8474.5763	8465.6085	8456.6596	8421.0526	8403.3613

Einstellbare Baudraten [Bit/s]				
8385.7442	8333.3333	8281.5735	8264.4628	8255.9340
8230.4527	8205.1282	8196.7213	8163.2653	8130.0813
8113.5903	8105.3698	8097.1660	8088.9788	8080.8081
8064.5161	8000.0000	7976.0718	7944.3893	7936.5079
7905.1383	7843.1373	7812.5000	7804.8780	7797.2710
7774.5384	7751.9380	7736.9439	7729.4686	7714.5612
7692.3077	7662.8352	7655.5024	7619.0476	7590.1328
7575.7576	7561.4367	7547.1698	7532.9567	7518.7970
7469.6545	7441.8605	7421.1503	7407.4074	7400.5550
7386.8883	7352.9412	7326.0073	7285.9745	7272.7273
7259.5281	7246.3768	7187.7808	7168.4588	7142.8571
7136.4853	7130.1248	7111.1111	7098.4916	7092.1986
7054.6737	7017.5439	6993.0070	6956.5217	6944.4444
6926.4069	6902.5022	6896.5517	6861.0635	6820.1194
6808.5106	6802.7211	6791.1715	6779.6610	6734.0067
6688.9632	6683.3751	6666.6667	6611.5702	6578.9474
6568.1445	6562.7564	6557.3770	6535.9477	6530.6122
6493.5065	6456.8200	6451.6129	6441.2238	6410.2564
6400.0000	6379.5853	6349.2063	6324.1107	6289.3082
6274.5098	6269.5925	6250.0000	6245.1210	6211.1801
6172.8395	6163.3282	6153.8462	6144.3932	6102.2121
6060.6061	6046.8632	6037.7358	5997.0015	5961.2519
5952.3810	5925.9259	5895.3574	5865.1026	5847.9532
5818.1818	5797.1014	5772.0058	5747.1264	5714.2857
5702.0670	5681.8182	5649.7175	5614.0351	5610.0982
5555.5556	5521.0490	5517.2414	5464.4809	5434.7826
5423.7288	5376.3441	5333.3333	5291.0053	5245.9016
5208.3333	5161.2903	5079.3651	5000.0000	

A.7 Abbildungsverzeichnis

A.8 Index