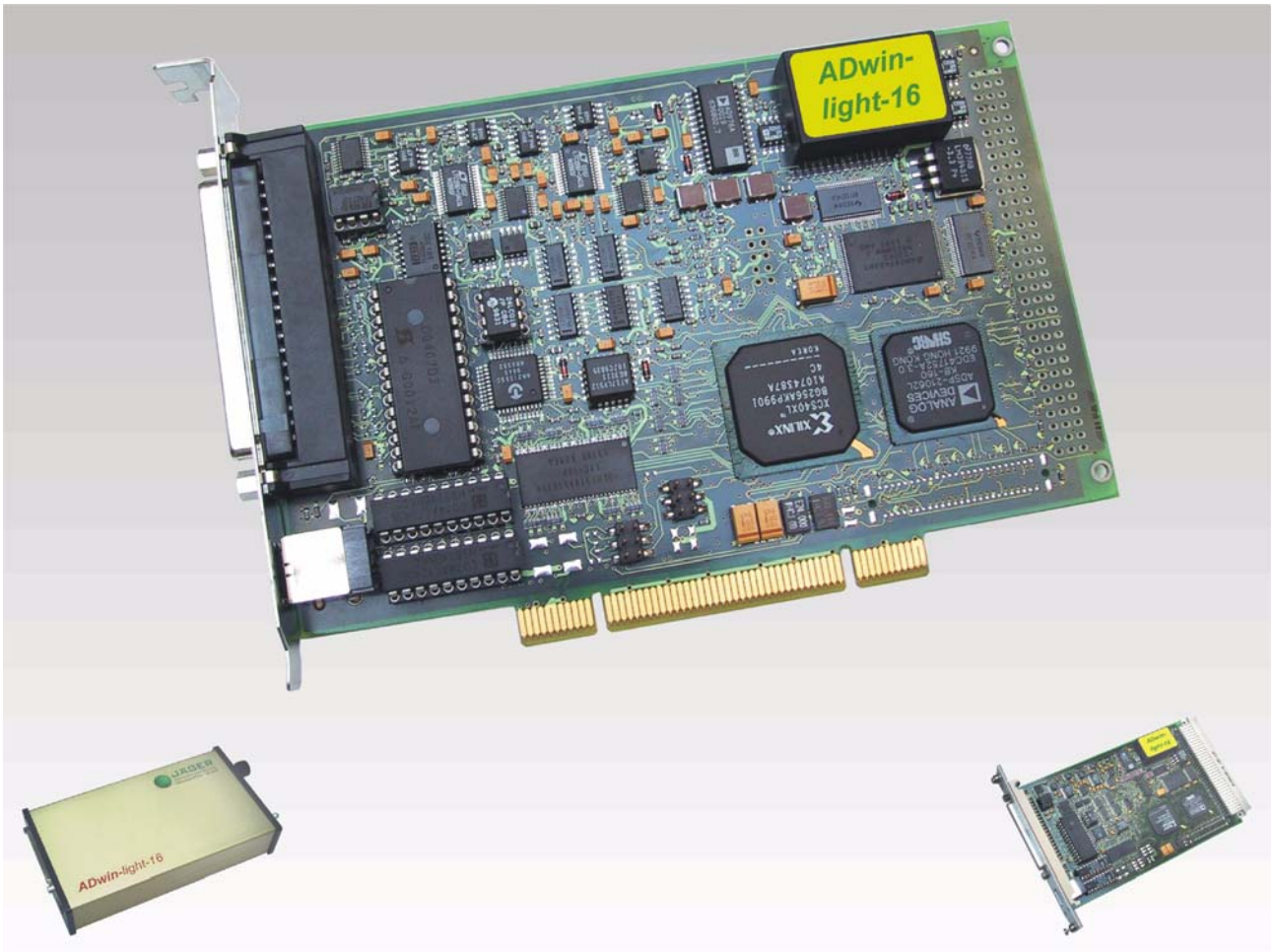
ADwin-light-16

Handbuch



ADwin-light-16 , Handbuch Version 2.7, Oktober 2006

Hier finden Sie immer einen Ansprechpartner für Ihre Fragen:

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

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

Inhaltsverzeichnis

Typografische Konventionen

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

Einen „Hinweis“ finden Sie bei

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

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

<C:\ADwin\ ...>

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

Programmtext

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

Var_1

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

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

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

1 Zu diesem Handbuch

Dieses Handbuch enthält umfassende Informationen für den Betrieb Ihres ***ADwin-light-16***-Systems. Es wird ergänzt durch

– das Handbuch „***ADwin*** Installation“, das die Installation von Software und Hardware beschreibt.
Beginnen Sie hier die Installation Ihres Systems!

– die Beschreibung des Konfigurationsprogramms *ADconfig*, mit dem Sie die Kommunikation von der jeweiligen Schnittstelle zu Ihrem ***ADwin-light-16***-Gerät einrichten.

– das Handbuch ***ADbasic***, das alle Befehle für den gleichnamigen Compiler enthält sowie das Funktionsprinzip von ***ADwin***-Systemen erläutert.

  Die Online-Hilfe von ***ADbasic*** enthält die gleichen Informationen.

– die Installations- und Befehlsbeschreibungen für die Treiber der gängigen Entwicklungsumgebungen.

– das Handbuch „***ADwin*** HSM-24V“, ein Modul am LS-Bus.

Bitte beachten Sie folgende Hinweise

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

Einschränkung der Anwendergruppe

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

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

Verfügbarkeit der Unterlagen

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

Rechtliche Grundlagen

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

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

Änderungen vorbehalten.

Hotline-Adresse siehe vordere Umschlagseite, innen.

2 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 (präemtives 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. 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 Windows 95/98/NT/ME/2000/XP 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.

Ein Linux-Treiber steht 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.

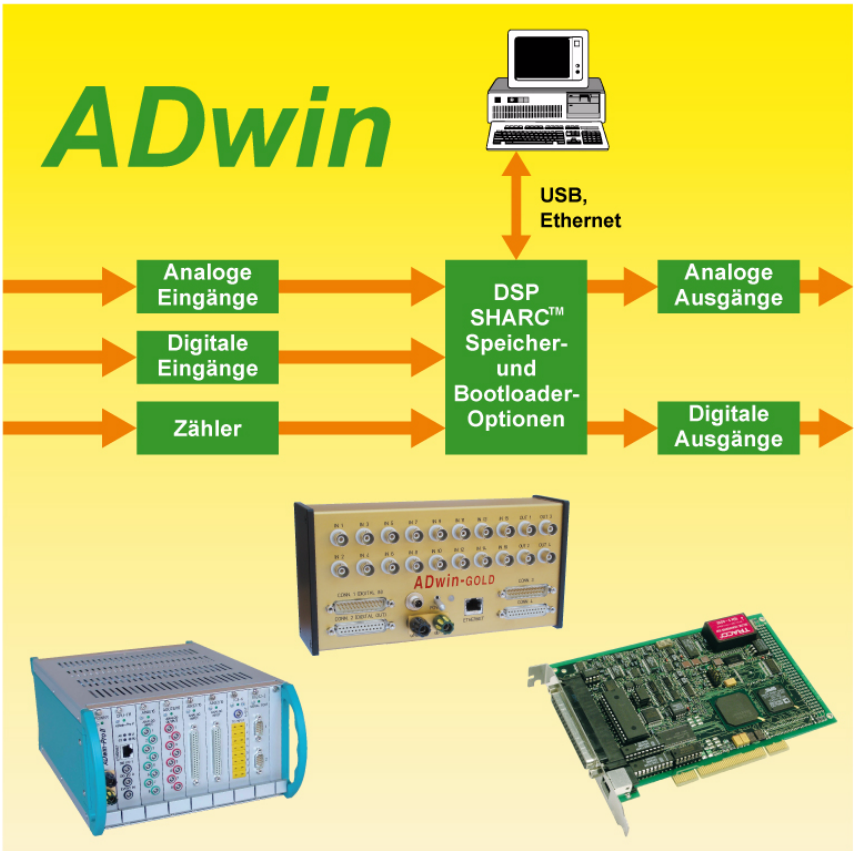


Abb. 1 – Konzept der ***ADwin***-Systeme

2.2 ADwin-light-16

Prozessor und Speicher

Das System ***ADwin-light-16*** besitzt den digitalen **32 Bit-Signalprozessor** ADSP 21062 (SHARC) von Analog Devices mit Floating-Point- und Integer-Verarbeitung. Er übernimmt die gesamte Messwerterfassung, Online-Verarbeitung und Signalausgabe und kann in Verbindung mit dem A/D-Wandler jeden Messwert mit Abtastraten bis zu 100 kHz sofort verarbeiten; ab Rev. B sind optional Abtastraten bis 500 kHz verfügbar.

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

Für maximale Zugriffsgeschwindigkeit liegen alle Ein- und Ausgänge direkt im Adressbereich des DSP. Zum Zwischenspeichern größerer Datenmengen nutzt der DSP einen **externen Speicher** (SDRAM) von **8 MB** (Rev. B: **16 MB**).

Analoge Eingänge

In einer 37-poligen Sub-D-Buchse sind **8 analoge Eingänge** bereit gestellt, die mit einem gemeinsamen Multiplexer verbunden sind. Das Eingangssignal wird mit einem 16 Bit Analog-Digitalwandler (ADC) konvertiert (siehe Abbildung unten). Ab Rev. B ist die automatische Wandlung mehrerer Eingänge mit einer Ablaufsteuerung verfügbar.

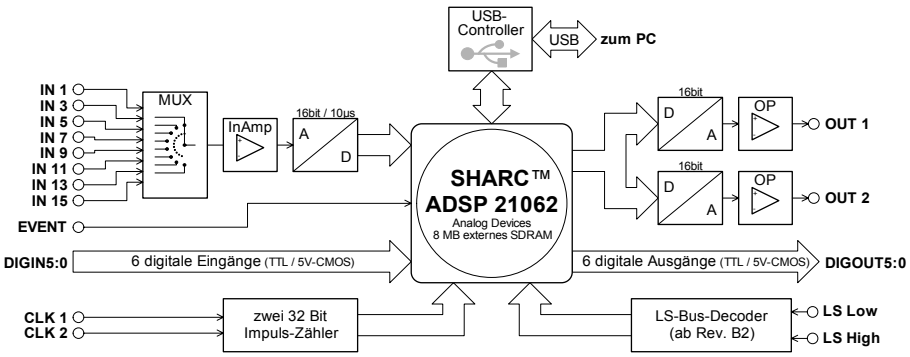


Abb. 2 – Funktionsschema (mit USB-Schnittstelle)

Analoge Ausgänge

ADwin-light-16 verfügt über **2 Analogausgänge** mit 16 Bit Auflösung und einem Ausgangsspannungsbereich von -10 V...+10 V. Per Software können Sie die Ausgabe der Spannung aller DAC synchronisieren sowie kalibrieren. Das Ausgangssignal durchläuft zur Glättung einen Tiefpassfilter mit einer Eckfrequenz von $f_g = 700$ kHz.

Digitale Ein- und Ausgänge

Auf der 37-poligen Sub-D-Buchse stehen **6 digitale Eingänge und 6 digitale Ausgänge** zur Verfügung. Die Ein- bzw. Ausgänge sind TTL-kompatibel. Daneben sind Eingänge für 2 Impuls-/Ereigniszähler mit 32 Bit vorhanden.

Trigger-Eingang

ADwin-light-16 besitzt einen Trigger-Eingang (EVENT, siehe auch Kapitel 5.2 “Digitale Ein- und Ausgänge“). Hiermit können Prozesse durch ein Signal (Trigger) ausgelöst und sofort vollständig abgearbeitet werden (siehe ***ADbasic***-Handbuch oder Online-Hilfe: Kapitel „Prozesse im Betriebssystem“).

Eine serielle Schnittstelle (LS-Bus ab Rev. B2, siehe Seite 17) erlaubt den Anschluss von bis zu 15 zusätzlichen Modulen.

Standardlieferumfang

Der Standardlieferumfang eines ***ADwin-light-16***-Systems beinhaltet

- ***ADwin-light-16***-Gerät
- USB- oder Ethernet-Anschlusskabel, Länge 1,8 Meter
- ***ADwin***-CD
- Handbuch „Treiber-Installation“
- das vorliegende Hardware-Handbuch.

Die Variante mit externem Gehäuse (*L16-EXT*) beinhaltet zusätzlich:

– Power-Adapter: Ein dreipoliges, verpolungssicheres Stromversorgungskabel (PC-intern) an einem Slotblech mit Steckbuchse.

– Stromversorgungskabel (zwischen Slot-Blech und *L16-EXT*)

Bauformen mit USB

Es gibt ***ADwin-light-16*** als Basisversion mit USB-Anschluss in verschiedenen Bauformen:

PC-Einsteckkarte (*L16-PCI*)

19“-Einschub (*L16-EURO*)

externes Gehäuse (*L16-EXT*)

Abb. 3 – Bauformen

Liefervarianten mit Ethernet

Die Bauformen *EURO* und *EXT* sind alternativ mit USB- oder mit 10/100 MBit Ethernet-Schnittstelle lieferbar. Die Liefervarianten der Basisversion sind in der folgenden Tabelle aufgeführt.

Bauform	**Schnittstelle**	
	USB	**Ethernet**
PC-Einsteckkarte	*L16-PCI*	–
19“-Einschub (Euro)	*L16-EURO*	*L16-EURO-ENET*
Externes Gehäuse	*L16-EXT*	*L16-EXT-ENET*

Abb. 4 – Liefervarianten der Basisversion ***ADwin-light-16***

Beachten Sie bitte, dass die Spannungsversorgung für die Bauformen unterschiedlich ist:

– +5 Volt für *L16-PCI* und *L16-EURO*

– +10 ... +18 Volt für *L16-EXT*, ab Rev B +10 ... +35 Volt

2.2.1 Bestelloptionen (nicht nachrüstbar)

Das ***ADwin-light-16***-System kann mit folgenden Optionen ausgerüstet sein:

- ***L16-CO1***: **Zähleroption** mit einem 32 Bit Vor-/Rückwärtszähler mit Vier-Flanken-Auswertung für Inkrementalgeber (Beschreibung siehe Seite 24).
- ***L16-DIO1***: Das Erweiterungsmodul (Beschreibung s. Seite 26) enthält
  - **32 digitale Ein-/Ausgänge** (programmierbar in Gruppen zu 8)
  - Einen **SSI-Decoder** (erst ab Rev. B).
  - **Eine CAN-Schnittstelle** (High-Speed, alternativ Low-Speed)
  - **Zwei 32 Bit Vor-/Rückwärtszähler** zur Impuls-, Periodendauer- und Tastverhältnis-Messung sowie einer Vier-Flanken-Auswertung zum Anschluss von Inkrementalgebern.
- ***L16-DIO2***: Das Erweiterungsmodul (Beschreibung s. Seite 43) enthält
  - **32 digitale Ein-/Ausgänge** (programmierbar in Gruppen zu 8).
  - Einen **SSI-Decoder**.
  - **Zwei 32 Bit Vor-/Rückwärtszähler** zur Impuls-, Periodendauer- und Tastverhältnis-Messung sowie einer Vier-Flanken-Auswertung zum Anschluss von Inkrementalgebern.
- ***L16-DIO3***: Das Erweiterungsmodul (Beschreibung s. Seite 43) enthält **32 digitale Ein-/Ausgänge** (programmierbar in Gruppen zu 8).
- ***L16-Boot***: **Flash-EPROM-Bootloader** zum eigenständigen **Betrieb ohne PC** (Beschreibung siehe Seite 54). Nur lieferbar in Verbindung mit Ethernet-Schnittstelle.
- ***L16-Mount***: Gehäuseumbau für ***L16-EXT*** zur **Hutschienen-Montage** in einem Schaltschrank mit isolierten Befestigungshaltern.

Beachten Sie, dass die Zähler der Erweiterungskarten nicht zusätzlich zur Verfügung stehen, sondern jeweils die Zähler der Basisversion ersetzen. Die Zähler unterschiedlicher Erweiterungen können deswegen nicht gemeinsam genutzt werden.

2.2.2 Zubehör

Für das ***ADwin-light-16***-System ist das folgende Zubehör (auch nachträglich) erhältlich; weitere Beschreibung siehe Seite 55:

ADbasic

- ***ADbasic***, Echtzeit-Entwicklungstool für alle ***ADwin***-Systeme
- Kabel-Stecker für eine externe Spannungsversorgung (nur für *L16-EXT*)
- externes Netzteil ***ADwin-light-16***-pow (u.a. erforderlich für Notebook-Betrieb)

3 Betriebliche Umgebung

Die Platine des ***ADwin-light-16***-Systems darf nur in einem geschlossenen Gehäuse betrieben werden (bei Bauform *L16-EXT* bereits erfüllt).

Je nach Variante und Ausrüstung kann das Gerät in 19“-Systemgehäusen, in Schaltschränken oder im mobilen Betrieb (z.B. im Kfz) eingesetzt werden (siehe Kapitel 2.2.1f: Bestelloptionen und Zubehör).

Das ***ADwin-light-16*-System 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 impedanz-armes Masseband mit dem zentralen Erdungspunkt Ihrer Anlage.

Galvanische Kopplung

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.
Wollen Sie Ausgleichströme vermindern, 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.

Schutzkleinspannung

Das ***ADwin-light-16***-System wird intern mit einer Spannung von +5 V und ±15 V gegen GND betrieben und 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-light-16 ist für den Betrieb in trockenen Räumen konzipiert. Am Einbauort (im PC oder 19“-Gehäuse) sollen eine Umgebungstemperatur von +5 °C ... +50 °C und eine relative Luftfeuchte von 0 ... 80 % (nicht kondensierend, s.a. Anhang) vorhanden sein.

Gehäusetemperatur

Die Gehäusetemperatur (Oberflächentemperatur) der Liefervariante *L16-EXT* darf auch unter extremen betrieblichen Bedingungen, z.B. im Schaltschrank oder bei direkter Sonneneinstrahlung, +55 °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-light-16***-System an, bevor Sie nicht folgende Schritte durchgeführt haben:

1. Software-Installation / Einbau im PC oder 19“-Gehäuse:

   Folgen Sie der Anleitung im Handbuch: „***ADwin***-Installation“.
2. Richten Sie die betriebliche Umgebung ein wie in Kapitel 3 beschrieben.
3. Lesen Sie das Kapitel 5 “Ein- und Ausgänge“ in diesem Handbuch.
4. Schließen Sie erst jetzt Signalleitungen an die Ein- und Ausgänge an.

Hinweise

Sicherstellen der Spannungsversorgung

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

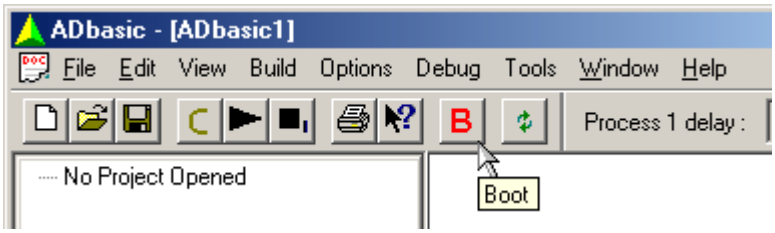
Achten Sie bei der Verwendung strombegrenzender Netzteile darauf, dass beim Einschalten der Strombedarf ein Mehrfaches des Betriebsstroms betragen kann. Genaue Angaben finden Sie bei den Technischen Daten (Anhang).

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

Vermeiden Sie die direkte Berührung unisolierter Teile zum Schutz vor elektrostatischer Aufladung.

Datenverbindung testen

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



Die Anzeige in der Statuszeile: „`ADwin is booted`“ zeigt an, dass das Betriebssystem richtig geladen ist und ***ADbasic*** eine Verbindung zum ***ADwin***-System herstellen kann.

Programme mit *ADbasic*

Die Programmierung von ***ADwin***-Systemen ist im ***ADbasic***-Handbuch (oder der Online-Hilfe) ausführlich beschrieben.
Beginnen Sie mit Programmbeispielen im ***ADbasic***-Tutorial.

5 Ein- und Ausgänge

Anschlüsse

Das System ***ADwin-Light-16*** besitzt folgende Steckverbindungen (Pinbelegung auf der nächsten Seite):

– Anschlussbuchse für USB oder Ethernet

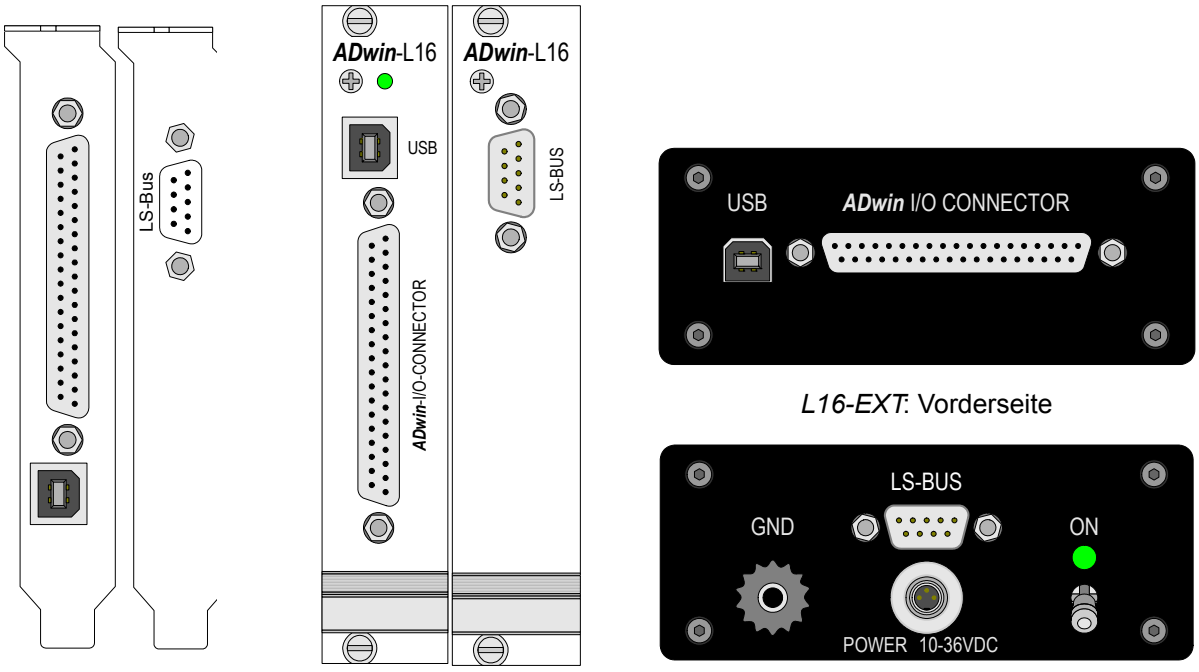
– 37-polige Sub-D-Buchse `ADwin I/O CONNECTOR` für
  • 8 analoge Eingänge, 2 analoge Ausgänge
  • jeweils 6 digitale Ein- und Ausgänge
  • 1 digitaler Trigger-Eingang
  • 2 Impuls-/Ereigniszähler mit 32 Bit
  • Stromversorgungs-Ausgang +5V; bei *L16-PCI* auch ±12V

– 9-polige Sub-D-Buchse `LS-BUS` für die LS-Bus-Schnittstelle (ab Rev. B2).

Die Bauform *L16-EXT* besitzt zusätzlich an der Hinterseite eine GND-Buchse (siehe Seite 7, Erdung), eine Strom-Eingangsbuchse und einen manuellen Ein-/ Ausschalter.

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-light-16***-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 der Sub-D-Buchse auf einen definierten Pegel (z.B. GND). Trennen Sie diese Eingänge auch von offenen Leitungen.
Ausnahme hierzu ist der Event-Eingang, der bereits einen internen Pull-up-Widerstand (10 kΩ) besitzt.



L16-PCI *L16-EURO* *L16-EXT*: Vorderseite *L16-EXT*: Hinterseite

Abb. 5 – Steckverbindungen ***ADwin-light-16***

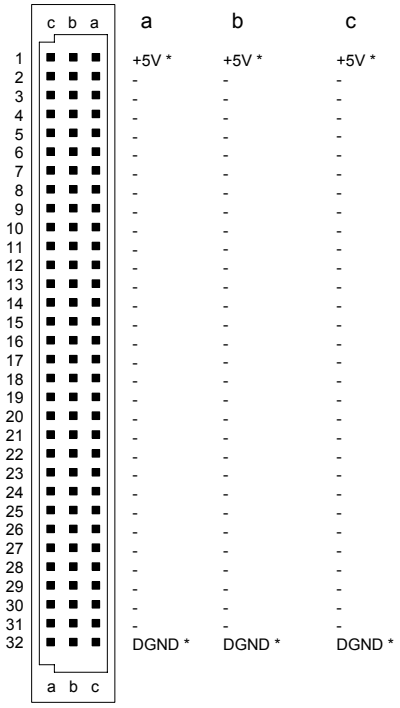


Abb. 6 – *L16-EURO* VGA-Federleiste zur Stromversorgung (Buchse)

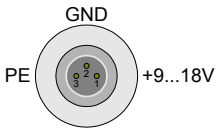


Abb. 7 – *L16-EXT* Strom-Eingangsstecker (Stecker)

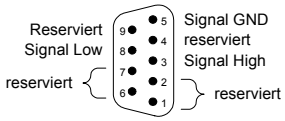
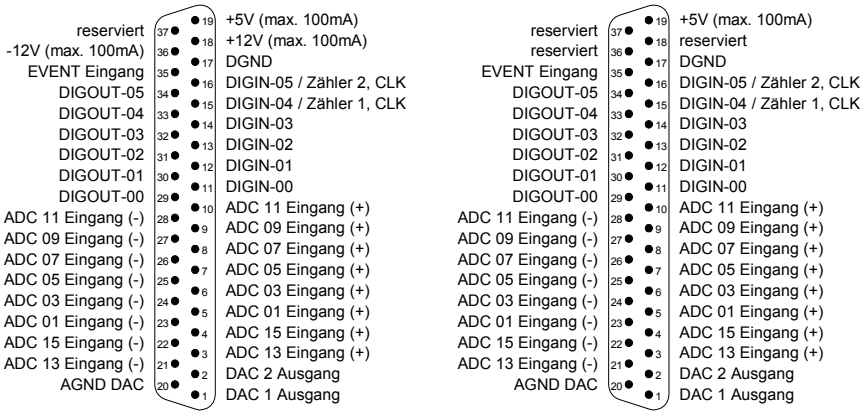


Abb. 8 – Pin-Belegung `LS-BUS` (Buchse)



L16-PCI *L16-EURO* und *L16-EXT*

Abb. 9 – Pin-Belegung Ein-/Ausgänge (Buchse)

Für eine schnelle und einfache Programmierung gibt es im Compiler ***ADbasic*** Standardbefehle, die einfaches Messen bzw. Ausgeben von Daten ermöglichen (siehe ***ADbasic***-Handbuch oder -Online-Hilfe). Verwenden Sie andere Befehle (z.B. direkter Registerzugriff) erst, wenn extrem zeitkritische oder besondere Aufgaben es erfordern.

Standardbefehle

Genauere Angaben zu den analogen sowie den digitalen Ein- und Ausgängen finden Sie in den folgenden Kapiteln.

5.1 Analoge Ein- und Ausgänge

Die Bauform *L16-EXT* muss geerdet werden, um Messungen störungsfrei durchführen zu können. Verbinden Sie dazu die GND-Buchse über ein impedanz-armes Masseband mit dem zentralen Erdungspunkt Ihrer Anlage. Bei der PCI- als auch der EURO-Variante obliegt die korrekte Erdung dem PC bzw. dem 19“-System.

Erdung

Beim *L16-EXT*-System ist das Gehäuse über die GND-Leitung des Stromversorgungskabels als auch über die GND-Leitung des USB-Kabels mit dem Schutzleiter des PC verbunden oder über die Schirmung des Ethernet-Steckers mit GND.

5.1.1 Eingänge

Das System hat 8 analoge Messeingänge, die über einen gemeinsamen Multiplexer zum 16 Bit Analog-Digitalwandler (ADC) geführt werden. Die Einschwingzeit des Multiplexers beträgt 6,5 µs beim maximalen Spannungssprung von 20V.

Multiplexer

Die Eingänge sind ausschliesslich mit ungeraden Zahlen (ADC 01, ADC 03, ... ADC 15) nummeriert, was bei der Programmierung zu berücksichtigen ist.

Die analogen Eingänge sind differentiell. Für jeden Messkanal sind jeweils ein Plus- und ein Minuseingang vorhanden, zwischen denen die Spannungsdifferenz gemessen wird (jedoch nicht potentialfrei).

Differentiell

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

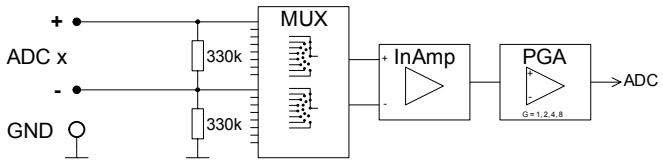


Abb. 10 – Eingangsbeschaltung eines analogen Eingangs

Das Signal am Multiplexer-Ausgang wird mit einem 16 Bit Analog-Digitalwandler (ADC) konvertiert, siehe auch Abb. 2 – Funktionsschema (mit USB-Schnittstelle). Die Wandlungszeit beträgt 10 µs (optional ab Rev. B: 2µs) bei einer Auflösung von 305 µV.

16 Bit-Messung

Der Befehl `ADC()` führt mit einem ADC eine komplette Messung auf einem analogen Eingang durch. So berücksichtigt dieser Befehl z.B. die Einschwingzeit des Multiplexers und stellt einwandfreie Messungen sicher (siehe auch ***ADbasic***-Handbuch oder -Online-Hilfe: Befehlsreferenz).

Komplette Messung

Ab Rev. B können die Signale ausgewählter analoger Eingänge mit einem einzigen Befehl (nacheinander) gewandelt werden. Die Eingänge werden mit `SEQ_INIT` ausgewählt, die Wandlung mit `START_CONV` gestartet und die Ergebnisse mit `SEQ_READ` gelesen.

Rev. B mit Ablaufsteuerung

5.1.2 Ausgänge

DAC-Befehl

Der Standardbefehl `DAC(Nummer,Wert)` prüft jeden Wert auf die Über- und Unterschreitung des 16-Bit Wertebereiches. Liegt der Wert innerhalb des 16-Bit Wertebereiches, wird der angegebene Wert auf dem Ausgang `Nummer` ausgegeben. Liegt er außerhalb, wird der Maximal- bzw. Minimalwert ausgegeben (siehe auch ***ADbasic***-Handbuch oder -Online-Hilfe: Befehlsreferenz).

5.1.3 Berechnungsgrundlagen

Spannungsbereich

Das ***ADwin-light-16***-System arbeitet bei den analogen Ein- und Ausgängen mit einem Spannungsbereich von –10 V bis +10 V (bipolar 10 V).

Zuordnung von Digit und Spannung

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

– 0 (Null) Digit der maximalen negativen Spannung und

– 65535 Digit der maximalen positiven Spannung

entspricht.

Der Wert für 65.536 Digit, genau 10 Volt, liegt gerade außerhalb des Messbereichs, womit sich für die 16 Bit-Wandlung ein maximaler Spannungswert von 9,999695 Volt ergibt.

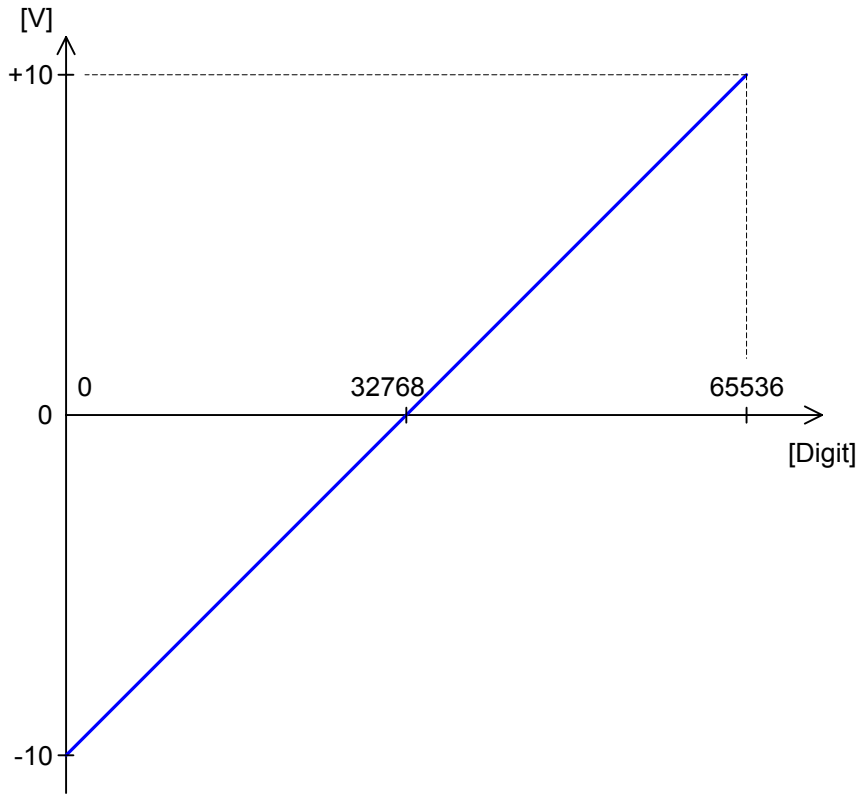


Abb. 11 – Nullpunktverschiebung bei Standardeinstellung bipolar 10 Volt

Die bipolare Einstellung führt zu einer Nullpunktverschiebung, die im folgenden auch als Offset bezeichnet wird.

Beim Spannungsbereich –10 V ... +10 V gilt U_{OFF} = –10 V

Least Significant Bit

Die Quantisierungsstufe U_{LSB} ist die kleinste digital darstellbare Spannungsdifferenz und ist gleich der Spannung des niederwertigsten Bit (Least Significant Bit). Das U_{LSB} entspricht dem 2^{16}-ten Teil von 20 V gleich 305,175 µV.

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	oberes Wort	16 Bit-Wert des ADC / DAC im unteren Wort															

Umrechnung Digit ⇆ Spannung

Für einen DAC gilt: **DAC**

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

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

Für einen ADC gilt: **ADC**

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

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

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):

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

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

5.2 Digitale Ein- und Ausgänge

Digitale Ein- / Ausgänge

Auf der 37-poligen Sub-D-Buchse stehen 6 digitale Eingänge (DIGIN 00 … DIGIN 05) und 6 digitale Ausgänge (DIGOUT 00…DIGOUT 05) zur Verfügung. Die Pin-Belegung der Sub-D-Buchse entnehmen Sie bitte Abbildung 9 auf Seite 10.

Die Eingänge DIGIN 04 und DIGIN 05 sind gleichzeitig als Zählereingang geschaltet und können alternativ für einen der beiden Zwecke (Digital- oder Zählereingang) genutzt werden. Dies gilt nicht für die DIO1-Erweiterung (siehe Seite 26) oder die DIO2-Erweiterung (siehe Seite 43).

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.

Trigger-Eingang (EVENT)

Das ***ADwin-light-16***-System besitzt einen externen Trigger-Eingang (EVENT). Ein externes Signal (Trigger) mit steigender Flanke an diesem Eingang kann Prozesse auslösen, die sofort und vollständig abgearbeitet werden (siehe ***ADbasic***-Handbuch oder -Online-Hilfe, Kapitel „Prozesse im Betriebssystem“).

Programmierung

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

Bereich	Befehle
Einzelnen Digitaleingang lesen.	DIGIN
Alle Digitaleingänge lesen.	DIGIN_WORD
Alle Digitalausgänge setzen.	DIGOUT_WORD
Einen Digitalausgang auf High setzen.	SET_DIGOUT
Einen Digitalausgang auf Low setzen.	CLEAR_DIGOUT

Die Befehle sind in der Include-Datei `<ADWL16.INC>` enthalten und sind im Handbuch ***ADbasic*** und in der Online-Hilfe beschrieben.

5.3 Impuls-/Ereigniszähler

Das ***ADwin-light-16***-System stellt 2 Impuls-/Ereigniszähler mit 32 Bit zur Verfügung, die Sie per Software einzeln oder gemeinsam konfigurieren und auslesen können.

Bei den Lieferoptionen *L16-CO1*, *L16-DIO1* und *L16-DIO2* werden die hier beschriebenen durch andere Zähler ersetzt. Sie finden die entsprechende Beschreibung in Kapitel 7 “CO1-Zählererweiterung“, Kapitel 8 “DIO1-Erweiterung“ oder Kapitel 9 “DIO2- / DIO3-Erweiterung“.

5.3.1 Hardware

Die Grafik zeigt den Aufbau eines einzelnen Zählers.

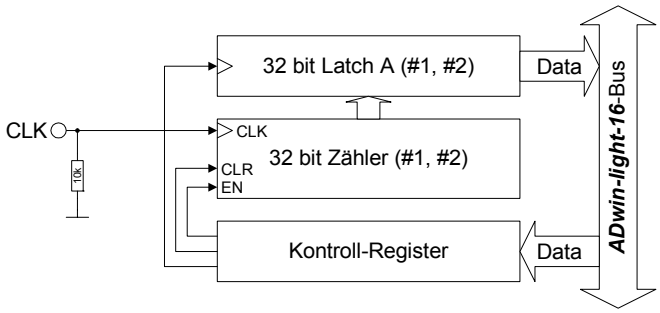


Abb. 12 – Schema des Impuls-/Ereigniszählers

Die Inkrementalzähler werden extern getaktet, d.h. sie erhöhen ihren Zählerstand um eins bei jeder positiven Flanke am Takteingang (CLK). Beide Zähler haben ein Latch A, das den Zählerstand programmgesteuert zum Auslesen übernehmen kann.
Die Zähler werden mit besonderen ***ADbasic***-Befehlen über Kontrollregister gesteuert, siehe Kurzreferenz auf Seite 16.

Eingänge beschalten

Die Takteingänge liegen auf den Pins 15 und 16 (siehe Abb. 9 – Pin-Belegung Ein-/Ausgänge (Buchse), Seite 10); für die korrekte Funktion sind TTL-kompatible Signale erforderlich. Einzelheiten und Grenzwerte finden Sie bei den Technischen Daten im Anhang. Beide Eingänge können alternativ als digitaler Signaleingang eingesetzt werden (siehe auch Kapitel 5.2).

Die Eingänge der Impuls-/Ereigniszähler besitzen einen Pull-down-Widerstand. Dennoch können offene Eingänge vor allem in einer nicht störungsfreien Umgebung zu Fehlern führen. Legen Sie deshalb sicherheitshalber die nicht benutzten Eingänge auf einen definierten Pegel (z.B. GND).

5.3.2 Software

Include-Datei

Die Zähler-Funktionen werden mit ***ADbasic***-Befehlen komfortabel programmiert. Die Befehle sind in einer Include-Datei enthalten; binden Sie diese Datei am Beginn des Programms ein mit der Befehlszeile

```
#INCLUDE ADWL16.INC
```

Die Zähler-Befehle der folgenden Tabelle sind im ***ADbasic***-Handbuch und in der Online-Hilfe vollständig beschrieben:

Zähler-Nr. / Bit	2 / 1	1 / 0	Kommentar
CNT_CLEAR( )	0	0	ohne Einfluss
	1	1	Zähler löschen *
CNT_ENABLE( )	0	0	Zähler sperren
	1	1	Zähler freigeben (laufende Zähler beachten)
CNT_LATCH( )	0	0	ohne Einfluss
	1	1	Zählerstand in Latch A übernehmen *
CNT_READLATCH(#)			Latch A auslesen (# = Zähler-Nr. 1, 2)
CNT_READ(#)			Zählerstand in Latch A übernehmen und auslesen (# = Zähler-Nr. 1, 2)
*Nach der Befehlsausführung werden gesetzte Bits automatisch wieder gelöscht. Bei anderen Befehlen müssen gesetzte Bits durch einen neuen Befehlsaufruf gelöscht werden.			

Abb. 13 – Zähler-Befehle Kurzreferenz

Sie können jeden Zähler einzeln oder beide Zähler gemeinsam konfigurieren.

Initialisierung

Initialisieren Sie die Zähler bitte in dieser Reihenfolge:

1. Gewünschten Zähler sperren (CNT_ENABLE)

   Der Befehl CNT_ENABLE spricht alle Zähler gemeinsam an. Auch wenn Sie nur einen Zähler sperren (Bit = 0) möchten, müssen Sie alle anderen Zähler konfigurieren, deren Zustand unverändert bleiben soll (Bit = 1). Beim Freigeben des Zählers gilt dies entsprechend.

2. Zähler löschen (CNT_CLEAR)
3. Zähler freigeben (CNT_ENABLE)

Für die Verarbeitung der Werte im ***ADbasic***-Programm übertragen Sie die Werte in Latch A und lesen sie dort aus.

5.3.3 Auswerten des Zählerinhalts

Die Binärzähler erzeugen 32 Bit-Werte. Unterscheiden Sie die Auswertung dieser Binärwerte (z.B. Differenzen) klar von der Darstellung der Werte als Dezimalzahl.

unveränderte Bitmuster

Für die Auswertung des Zählerinhalts benötigen Sie dessen 32 Bit-Werte ohne Veränderung, insbesondere bei der Bildung von Differenzen. Dies gewährleisten unter ***ADbasic*** nur Variablen vom Typ LONG.

Die Anzeige von 32-Bit-Binärwerten in ***ADbasic*** sorgt oft für Verwirrung, weil hierbei der vorzeichenlose Zählerwert als vorzeichenbehafteter Dezimalwert dargestellt wird (siehe Zahlenkreis unten). Als Folge entsteht in der Darstellung ein Übergang zwischen positivem und negativem Zahlenbereich, der jedoch für die Auswertung des Binärwerts ohne Bedeutung ist.

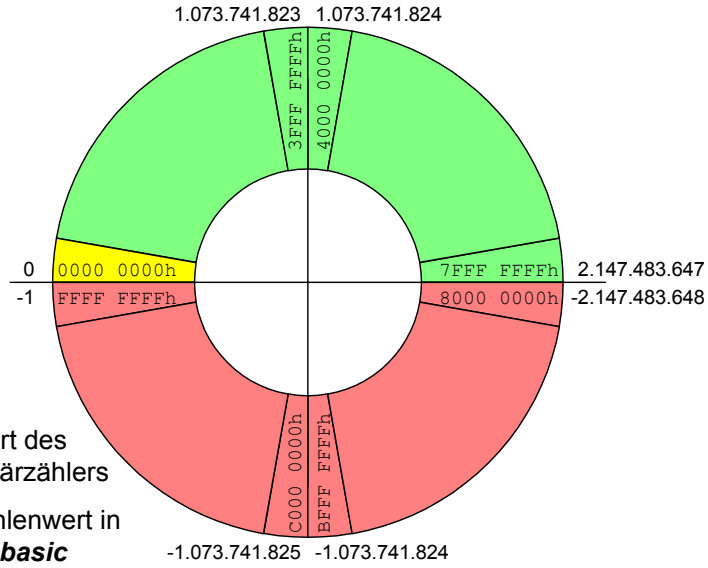


Abb. 14 – Zahlenkreis als Interpretation von Zählerwerten

Zahlenkreis

Zur Vollständigkeit wird nachfolgend die Umwandlung zur Dezimalzahl beschrieben: Das höchste Bit (MSB) des Binärwerts stellt das Vorzeichen dar. Bei positivem Vorzeichen werden die weiteren 31 Bits direkt als Zahl interpretiert, d.h. die Absolutwerte von Dezimalzahl und Binärwert sind gleich. Bei negativem Vorzeichen werden die 31 Bits invertiert, eine Eins addiert und als Zahl interpretiert (2er-Komplement); negative Dezimalzahlen haben deshalb einen anderen Absolutwert als der zugehörige Binärwert.

Bilden Sie Differenzen nur mit Integer-Zahlen (`LONG`).

„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 Frequenz von 20 MHz nach etwas mehr als 3½ Minuten ein, bei 5 MHz nach über 14 Minuten.

5.4 LS-Bus

ADwin-light-16 stellt einen Anschluss für den LS-Bus auf einem 9-poligen Sub-D-Verbinder (Buchse) zur Verfügung; die Pin-Belegung ist auf Seite 10 dargestellt.

Der LS-Bus ist ein bidirektionaler, serieller Bus mit 5 MHz Taktrate.

Der Bus ist als Linienverbindung aufgebaut, d.h. die ***ADwin***-Schnittstelle und bis zu 15 LS-Bus-Module sind jeweils über Zweipunktverbindungen miteinander verbunden. Am letzten LS-Bus-Modul muss der Busabschluss aktiviert sein. Die maximale Buslänge beträgt 5 m.

Die Module am LS-Bus werden mit ***ADbasic***-Befehlen programmiert, die über die LS-Bus-Schnittstelle am ***ADwin***-System geschickt werden. Die Befehle sind meistens modulspezifisch und sind im Handbuch des LS-Bus-Moduls beschrieben (oder in der Online-Hilfe).

5.5 Zeitkritische Aufgaben

Zeitkritische Aufgaben

Für extrem zeitkritische Aufgaben können Sie Befehle einsetzen, mit denen Sie direkt auf die Steuer- und Datenregister der Hardware zugreifen (siehe ***ADbasic***-Handbuch oder -Online-Hilfe). Diese Register liegen im Speicheradressbereich des Prozessors (memory mapped). Die Befehle ermöglichen auch eine Optimierung der Programmstruktur.

Im Gegensatz zu den Standardbefehlen `ADC()` und `DAC()` verzichten die Befehle für den Direktzugriff auf jegliche Prüfroutinen. Vor deren Benutzung sollten Sie deshalb über genaue Kenntnisse der Programmierung und der Zeit- und Funktionsabläufe in einem Analog-Digitalwandler verfügen, da Sie nun hardware-nah programmieren.

Analoge Ein- und Ausgänge

`ADC()`

Führen Sie anstelle des Standardbefehls `ADC()` die folgenden ***ADbasic***-Befehle in dieser Reihenfolge aus:

Programmstruktur

```
SET_MUX()
...                          'Einschwingzeit abwarten
START_CONV()
WAIT_EOC()                   'das Wandlungsende abwarten
READADC()
```

Legen Sie zwischen die Ausführung der Befehle `START_CONV()` und `SET_MUX()` durch weitere Programmierschritte einen ausreichenden Zeitabstand, um die Einschwingzeit des Multiplexers zu berücksichtigen (siehe auch ***ADbasic***-Handbuch oder -Online-Hilfe: Befehlsreferenz).

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

- Einschwingzeit des Multiplexers: Diese beträgt 6,5 µs beim maximalen Spannungssprung von 20V.
- Wandlungszeit des 16 Bit-ADC: 10µs; optional ab Rev. B: 2µs.

Hardware-Adressen der Steuer- und Datenregister

Mit den Befehlen `PEEK` und `POKE` (siehe ***ADbasic***-Handbuch oder -Online-Hilfe) können Sie direkt auf Steuer- und Datenregister zugreifen. Damit können Sie den Prozessablauf beschleunigen, beispielsweise:

ADC

- eine Messung sehr schnell ausführen.

DAC

- sehr schnell ein oder mehrere DAC-Register beschreiben und synchron die Ausgabe aktivieren.

Stellen Sie sicher, dass die Werte der Befehlsparameter innerhalb der zulässigen Bereichsgrenzen liegen.

Die Hardware-Adressen der Register finden Sie nachfolgend, gruppiert nach analogen Eingängen, analogen Ausgängen, digitalen Ein-/Ausgängen und Zählern.

Beachten Sie, dass manche Register mehrere Vorgänge gleichzeitig beeinflussen können.

Adresse [HEX]	Funktion	Bit Nr. 31-16	15-10	9	8	7	6	5	4	3	2	1	0	Kommentar
20 40 00 00	Multiplexer auf Eingangskanal setzen (ADC 01... ADC 15)	-	-	-	-	-	-	0	0	0	n	n	n	„nnn“ binär = 0...7 dezimal; gewählter Kanal = nnn*2 + 1

Adresse [HEX]	Funktion	Bit Nr. 31-16	15-10	9	8	7	6	5	4	3	2	1	0	Kommentar
20 40 00 10	Konvertierung starten: ADC#1	-	-	-	-	-	-	-	1	1	1	1	s	s = 0 : Konvertierung starten s = 1 : kein Einfluss
20 40 00 20	Konvertierungs-Status (EOC): ADC#1	-	-	-	-	-	-	-	-	-	-	-	e	e = 0 : Konvertierung beendet e = 1 : Konvertierung läuft
20 40 00 30	Register auslesen: ADC#1	-	x	x	x	x	x	x	x	x	x	x	x	x : Ergebnis der Konvertierung
20 40 01 00	Register auslesen und Konvertierung starten: ADC#1	-	x	x	x	x	x	x	x	x	x	x	x	

Abb. 15 – ADC Hardware-Adressen der Steuer- und Datenregister

Adresse [HEX]	Funktion	Bit Nr. 31-16	15-10	9	8	7	6	5	4	3	2	1	0	Kommentar
20 40 00 10	Konvertierung starten: alle DAC synchron	-	-	-	-	-	-	-	1	1	s	1	1	s = 0 : Konvertierung starten s = 1 : kein Einfluss
20 40 00 50	Register nur beschreiben: DAC #1	-	x	x	x	x	x	x	x	x	x	x	x	x : zu konvertierender Digitalwert
20 40 00 60	Register nur beschreiben: DAC #2	-	x	x	x	x	x	x	x	x	x	x	x	
20 40 02 00	Register beschreiben und sofort Konvertierung starten: DAC #1	-	x	x	x	x	x	x	x	x	x	x	x	
20 40 02 10	Register beschreiben und sofort Konvertierung starten: DAC #2	-	x	x	x	x	x	x	x	x	x	x	x	

Abb. 16 – DAC Hardware-Adressen der Steuer- und Datenregister

Adresse [HEX]	Funktion	Bit Nr. 31:16	15:6	5	4	3	2	1	0	Kommentar
20 40 00 B0	Eingangs-Register DIGIN-05...DIGIN-00	-	-	x	x	x	x	x	x	x : eingelesener Digitalwert
20 40 00 C0	Ausgangs-Register DIGOUT-05 ... DIGOUT-00	-	-	x	x	x	x	x	x	x : auszugebender Digitalwert
20 40 00 C4	DIGOUT Bit-SET-Register	-	-	0	0	0	0	0	0	kein Einfluss
		-	-	1	1	1	1	1	1	Bit setzen
20 40 00 C8	DIGOUT Bit-CLEAR-Register	-	-	0	0	0	0	0	0	kein Einfluss
		-	-	1	1	1	1	1	1	Bit löschen

Abb. 17 – DIO Hardware-Adressen der Steuer- und Datenregister

Adresse [HEX]	Funktion	Bit Nr. 31:16	15:6	5	4	3	2	1	0	Kommentar
20 40 02 04	Inhalt Latch A, Zähler #1	x	x	x	x	x	x	x	x	x : gelatchter Zählerstand
20 40 02 14	Inhalt Latch A, Zähler #2	x	x	x	x	x	x	x	x	x : gelatchter Zählerstand
20 40 03 00	Zähler freigeben / sperren (= `CNT_ENABLE()`)	-	-	-	-	-	-	0	0	Zähler gesperrt
		-	-	-	-	-	-	1	1	Zähler freigeben
20 40 03 10	Zähler löschen (= `CNT_CLEAR()`)	-	-	-	-	-	-	0	0	kein Einfluss
		-	-	-	-	-	-	1	1	Zähler löschen*
20 40 03 20	Zähler latchen (= `CNT_LATCH()`)	-	-	-	-	-	-	0	0	kein Einfluss
		-	-	-	-	-	-	1	1	Zählerstand in Latch A übertragen*
*Nach der Befehlsausführung werden gesetzte Bits automatisch wieder gelöscht. Bei anderen Befehlen müssen gesetzte Bits durch einen neuen Befehlsaufruf gelöscht werden.										

Abb. 18 – Zähler Hardware-Adressen der Steuer- und Datenregister

6 Kalibrierung

Die beiden Digital/Analog- (DAC) und der Analog/Digital-Wandler (ADC) Ihres ***ADwin***-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 `<L16Calib.exe>` durch; der Pfad bei Standardinstallation ist `<C:\ADwin\Tools\ADwin-L16>`.

Zur Kalibrierung benötigen Sie folgende Hilfsmittel:

– Ein Digital-Multimeter (DMM) mit einer Messgenauigkeit von 30 µV.

– Eine Referenz-Gleichspannungsquelle mit stabiler Einstellgenauigkeit von 30 µV. Ersatzweise verbinden Sie DAC 1 mit ADC 01(+), DAC 2 mit ADC 03 (+) und AGND DAC mit ADC 01(-) und ADC 03(-), z.B. in Form eines Teststeckers.
Diese Verbindungen benötigen Sie auch für das Kalibrier-Diagramm.

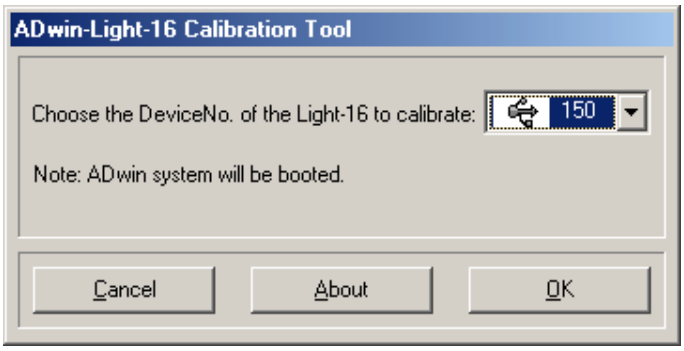
– Verbindungskabel von den Ein/Ausgängen zur Referenzspannungsquelle und zum Messgerät.

Schritt 1

Verbinden Sie Ihr ***ADwin-light-16***-Gerät mit dem PC und konfigurieren Sie es mit dem Programm `<ADconfig.exe>`.

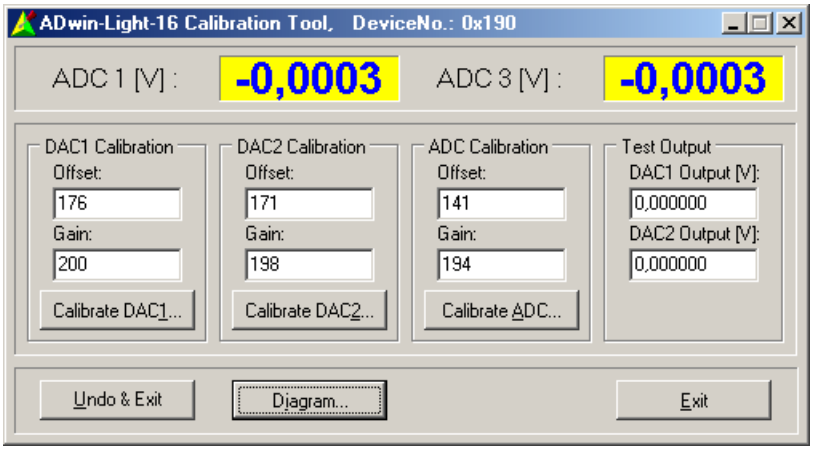
Schritt 2

Starten Sie das **Kalibrierprogramm** `<L16Calib.exe>`. Es erscheint das Fenster „`ADwin-light-16 Calibration Tool`“.



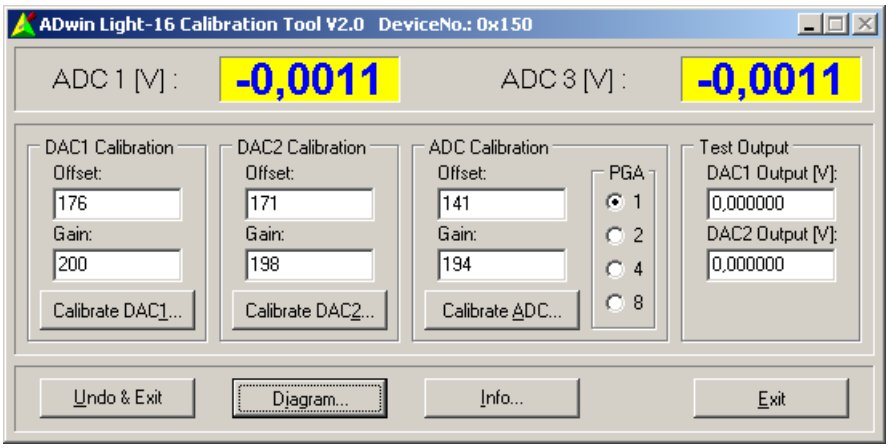
Wählen Sie die Device-Nummer des zu kalibrierenden Geräts und bestätigen Sie durch Drücken von „`OK`“.
Sie erhalten eine Warnung, wenn Sie kein ***ADwin-light-16*** Gerät gewählt haben oder eines mit einer älteren Firmware-Version. Sie können die Warnung mit „`YES`“ übergehen oder mit „`NO`“ zum vorigen 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 ADC 01 und ADC 03. Darunter sehen Sie die gültigen Kalibriereinstellungen für `Offset` und `Gain` der beiden DAC und des ADC; Sie können dort direkt Werte eingeben. Mit der Taste „`Calibrate ...`“ starten Sie die Kalibrierung des jeweiligen Wandlers.
Am rechten Rand können Sie in die Felder DAC1 und DAC2 Spannungswerte eingeben, die automatisch auf die entsprechenden Ausgänge gelegt werden.

Bei Rev. B enthält das Übersichtsfenster zusätzlich das Auswahlfeld `PGA` für den Verstärkungsfaktor und die Schaltfläche `Info...`, über die Versionsinformationen des Geräts angezeigt werden können.



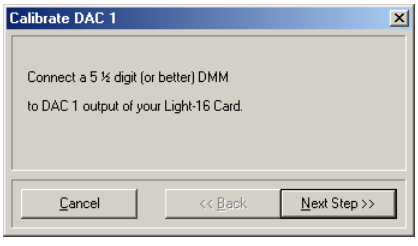
Alle von Ihnen eingegebenen Einstellungen werden automatisch gespeichert. In der unteren Zeile können Sie mit „`Undo&Exit`“ alle Eingaben rückgängig machen und das Kalibrierprogramm verlassen. „`Diagram`“ zeigt Ihnen in einer Grafik die Genauigkeit der aktuellen Kalibriereinstellung. Wenn Sie das Programm mit „`Exit`“ verlassen, bleiben die neuen Einstellungen erhalten.

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.

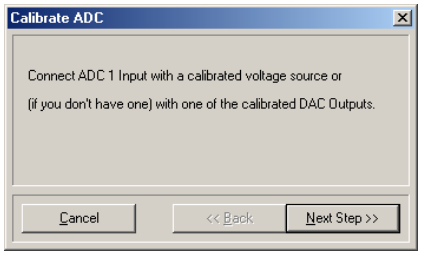
Ohne Referenz-Spannungsquelle ist die Kalibrierung möglich, aber ungenauer. Kalibrieren Sie dabei zuerst DAC1 und DAC2, dann erst den ADC.

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

1. Externes Hilfsgerät (DMM / Spannungsquelle) anschließen:
   Wählen Sie zur Kalibrierung eines Wandlers die entsprechende Taste „`Calibrate ...`“, es erscheint das erste Fenster:



Schließen Sie ein DMM an den Pins AGND DAC und DAC1 oder DAC2 an.

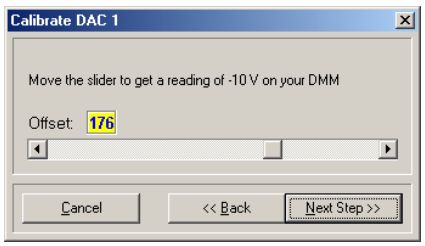


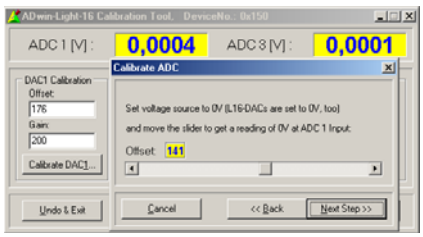
Schließen Sie die Spannungsquelle (oder einen DAC-Ausgang) an den Eingang ADC 01 oder ADC 03 an.

Schritt 3

Beachten Sie bitte Abb. 9 – Pin-Belegung Ein-/Ausgänge (Buchse).
Wählen Sie „Next Step >>“.

2. Offset einstellen:





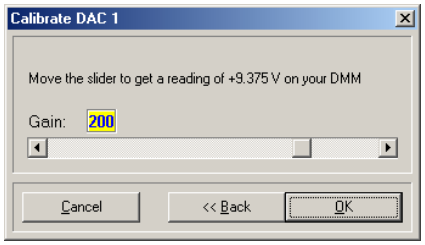
Verstellen Sie am Rollbalken den Offset-Wert so, dass Ihr Digital-Multimeter -10 V anzeigt.

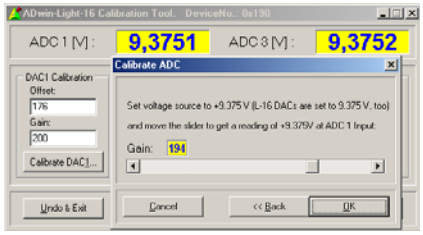
Stellen Sie an der Spannungsquelle den Sollwert 0 V ein. Die Einstellung des ADC auf diesen Wert erfolgt automatisch.
Stellen Sie am Rollbalken den Offset-Wert so ein, dass der Sollwert am ADC 01 im Übersichtsfenster angezeigt wird.

Bei Rev. B erfolgen die Einstellungen dieses Schritts automatisch.

Wählen Sie „Next Step >>“.

3. Gain einstellen:





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

Stellen Sie an der Spannungsquelle den Sollwert 9,375 V ein. Die Einstellung des ADC auf diesen Wert erfolgt automatisch.
Stellen Sie am Rollbalken den Offset-Wert so ein, dass der Sollwert am ADC 01 im Übersichtsfenster angezeigt wird.

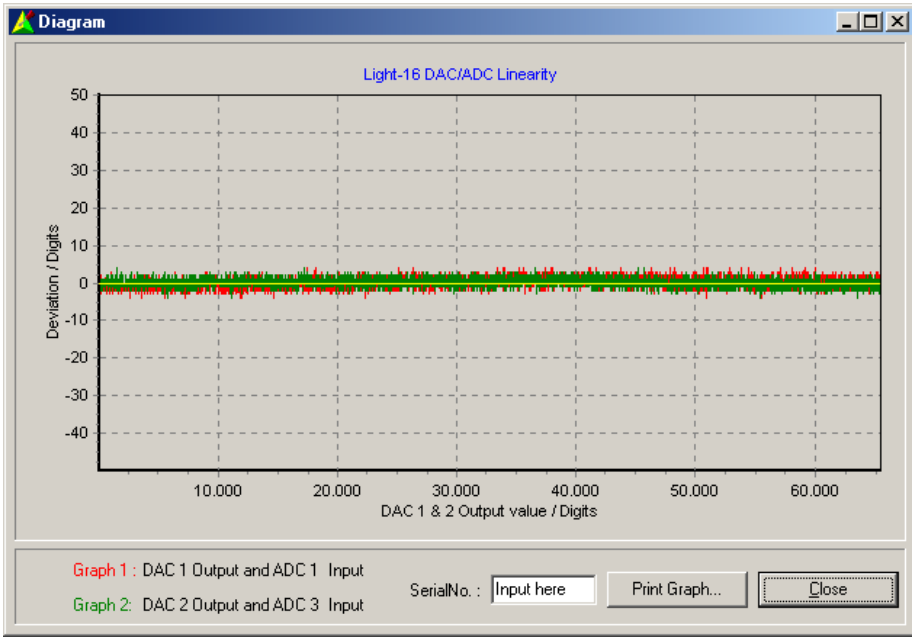
Bei Rev. B erfolgen die Einstellungen dieses Schritts automatisch.

Die Kalibrierung für diesen Wandler ist beendet. Wählen Sie „OK“.

Wiederholen Sie Schritt 3 ggf. für die anderen Wandler.

Schritt 4

Die **Genauigkeit** der eingestellten Kalibrierung können Sie anhand eines Diagramms prüfen (Schaltfläche Diagram im Übersichtsfenster).
Verbinden Sie zunächst den Ausgang DAC 1 mit dem Eingang ADC 01 sowie den Ausgang DAC 2 mit dem Eingang ADC 03.



Das Programm gibt die Werte 0…65.535 Digits auf beide DAC aus, vergleicht sie mit den gemessenen Eingangswerten und stellt die Abweichung in Kurven dar: Kurve 1 (rot) für DAC 1 / ADC 01 und Kurve 2 (grün) für DAC 2 / ADC 03. Die Abweichung sollte kleiner als 5 Digits sein.

Sie können die Grafik mit `Print Graph` ausdrucken (Farbdrucker empfohlen). Geben Sie hierfür die Seriennummer Ihres ***ADwin***-Systems ein, damit Sie den Ausdruck später zuordnen können. Auf dem Ausdruck sind außerdem die Kalibriereinstellungen und das Druckdatum enthalten.

Mit `Close` kehren Sie zum Übersichtsfenster zurück.

Die Kalibrierung ist beendet.

Schritt 5

7 CO1-Zählererweiterung

Die Zählererweiterung CO1 (Bestelloption *L16-CO1*) stellt einen 32 Bit Vor-/Rückwärtszähler mit Vierflankenauswertung zur Verfügung und ersetzt die Inkrementalzähler der Grundversion (Abb. 19 – Schema der L16-CO1 Zählererweiterung zeigt den Zähleraufbau).

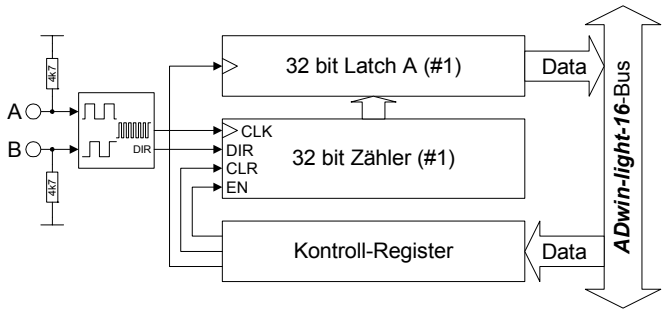


Abb. 19 – Schema der *L16-CO1* Zählererweiterung

7.1 Hardware

Der Zähler wird **extern getaktet** und besitzt eine Vierflanken-Auswertung zum Anschluss eines Encoders. Ausgelesen wird der Zähler über das Latch A, die Steuerung erfolgt mit ***ADbasic***-Befehlen über ein Kontrollregister (siehe "CO1-Befehle, Kurzübersicht" auf Seite 25).

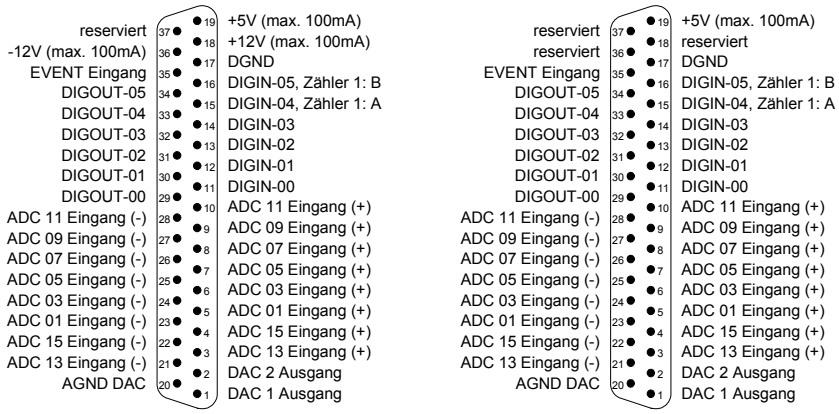
Die Vierflankenauswertung setzt die um 90 Grad versetzten digitalen Eingangssignale A und B in ein Takt- (CLK) und Richtungssignal (DIR) für den Zähler um. Dabei wird aus jeder Flanke des A- und B-Signals ein Taktsignal erzeugt. Die Zählrichtung (DIR) ergibt sich aus der Reihenfolge der steigenden bzw. fallenden Flanken dieser Signale.

Eingänge beschalten

Die Takteingänge A und B liegen auf den Pins 15 und 16 der Sub-D-Buchse `ADwin I/O CONNECTOR` (siehe Pin-Belegung unten); für die korrekte Funktion sind TTL-kompatible Signale erforderlich. Einzelheiten und Grenzwerte finden Sie bei den Technischen Daten im Anhang.

Beide Eingänge können alternativ als digitaler Signaleingang eingesetzt werden (siehe auch Kapitel 5.2).

Obwohl alle Eingänge der CO1-Erweiterung einen Pull-down-Widerstand besitzen, können offene Eingänge vor allem in einer nicht störungsfreien Umgebung zu Fehlern führen. Legen Sie deshalb sicherheitshalber die nicht benutzten Eingänge auf einen definierten Pegel (z.B. GND).



L16-PCI *L16-EURO* und *L16-EXT*

Abb. 20 – Pin-Belegung der *L16-CO1*

7.2 Programmierung

Include-Datei

Die Zähler-Funktionen der CO1-Erweiterung werden mit ***ADbasic***-Befehlen komfortabel programmiert. Die Befehle sind in einer Include-Datei enthalten; binden Sie diese Datei am Beginn des Programms ein mit der Befehlszeile

```
#INCLUDE ADWL16.INC
```

Die Zähler-Befehle der folgenden Tabelle sind im ***ADbasic***-Handbuch oder der Online-Hilfe vollständig beschrieben:

Zähler-Nr. Bit	1 0	Kommentar
CNT_CLEAR( )	0	ohne Einfluss
	1	Zähler löschen *
CNT_ENABLE( )	0	Zähler sperren
	1	Zähler freigeben
CNT_LATCH( )	0	ohne Einfluss
	1	Zählerstand in Latch A übernehmen *
CNT_READLATCH(#)		Latch A auslesen (# = Zähler-Nr. 1)
CNT_READ(#)		Zählerstand in Latch A übernehmen und auslesen (# = Zähler-Nr. 1)
*Nach der Befehlsausführung werden gesetzte Bits automatisch wieder gelöscht. Bei anderen Befehlen müssen gesetzte Bits durch einen neuen Befehlsaufruf gelöscht werden.		

Abb. 21 – CO1-Befehle, Kurzübersicht

Werten Sie bitte den Zählerinhalt nur mit Variablen vom Typ LONG aus, vor allem für die Ermittlung von Differenzen oder der Zählrichtung (siehe auch Seite 16).

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.

Für extrem zeitkritische Aufgaben kann es sinnvoll sein, wenn Sie mit den Befehlen PEEK und POKE direkt auf die Steuer- und Datenregister des Zählers zugreifen. In der Tabelle sind die entsprechenden Hardware-Adressen dargestellt.

Die Hardware-Adressen des CO1-Zählers ersetzen diejenigen der Grundversions-Zähler und sind deswegen identisch.

Adresse [HEX]	Funktion	Bit-Nummer 31:24	23:16	15:08	07:01	00	Kommentar
20 40 02 04	Inhalt Latch A, Zähler #1	x	x	x	x	x	x : gelatchter Zählerstand
20 40 03 00	Zähler freigeben CNT_ENABLE()	-	-	-	-	0	Zähler sperren
		-	-	-	-	1	Zähler freigeben
20 40 03 10	Zähler löschen CNT_CLEAR()	-	-	-	-	0	kein Einfluss
		-	-	-	-	1	Zähler löschen*
20 40 03 20	Zähler latchen CNT_LATCH()	-	-	-	-	0	kein Einfluss
		-	-	-	-	1	Zähler latchen*
*Nach der Befehlsausführung werden gesetzte Bits automatisch wieder gelöscht. Bei anderen Befehlen müssen gesetzte Bits durch einen neuen Befehlsaufruf gelöscht werden.							

Abb. 22 – CO1-Hardware-Adressen der Steuer- und Datenregister

8 DIO1-Erweiterung

Die DIO1-Erweiterung stellt Ihnen zusätzlich zur Verfügung:

– 32 digitale Ein-/Ausgänge (programmierbar in Gruppen zu 8).

– Zwei 32 Bit Vor-/Rückwärtszähler zur Impuls-, Periodendauer- und Tastverhältnismessung sowie einer Vierflankenauswertung zum Anschluss von Encodern.
Eingänge umschaltbar zwischen „single ended“ und „differentiell“.

– CAN-Schnittstelle (High-Speed).

– 1 SSI-Decoder (Seite 42), erst ab Rev. B.

Der SSI-Decoder eignet sich zum Anschluss von Inkremental-Encodern mit SSI-Schnittstelle. Die Eingänge liegen auf dem Stecker COUNTER; die Signale sind differentiell und für RS422/485-Pegel (5V) ausgelegt.

Das Schema zeigt Ihnen gemeinsam die Grundfunktionen eines L16-Geräts mit den Zusatzfunktionen der DIO1-Erweiterung (als USB-Variante).

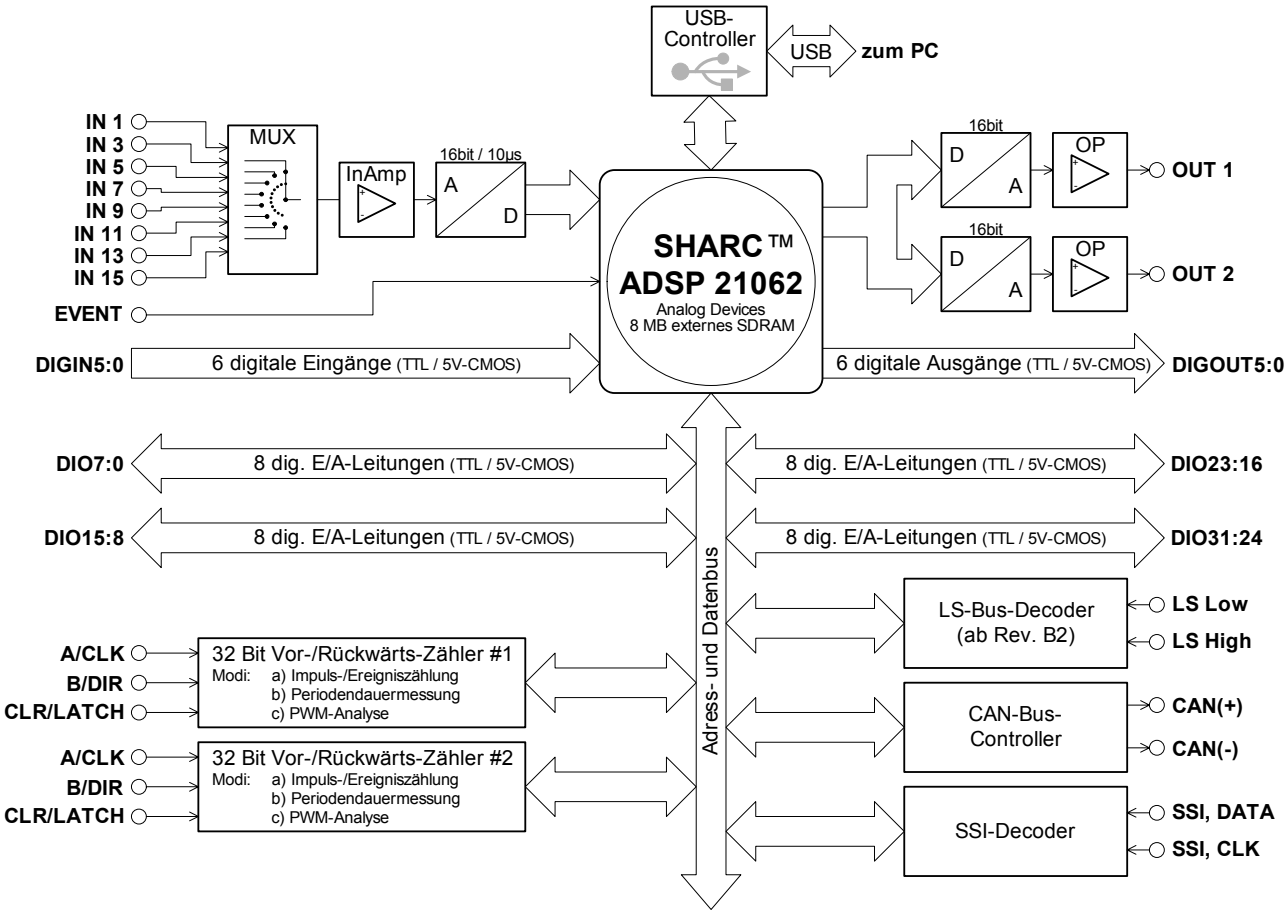


Abb. 23 – Schema *L16-DIO1* (mit USB-Schnittstelle)

Die Zähler der Basisversion werden von den DIO1-Zählern ersetzt. Die digitalen Ein- und Ausgänge stehen weiterhin zur Verfügung.

Die Pin-Belegung am Anschluss „***ADwin*** I/O-CONNECTOR“ ist mit der Basisversion identisch, bis auf einen Unterschied: Die in der Basisversion doppelt beschalteten Pins 15/16 stehen nun ausschließlich als DIGIN-04 und DIGIN-05 zur Verfügung.

Die Pin-Belegung der LS-Bus-Schnittstelle ist auf Seite 10 dargestellt.

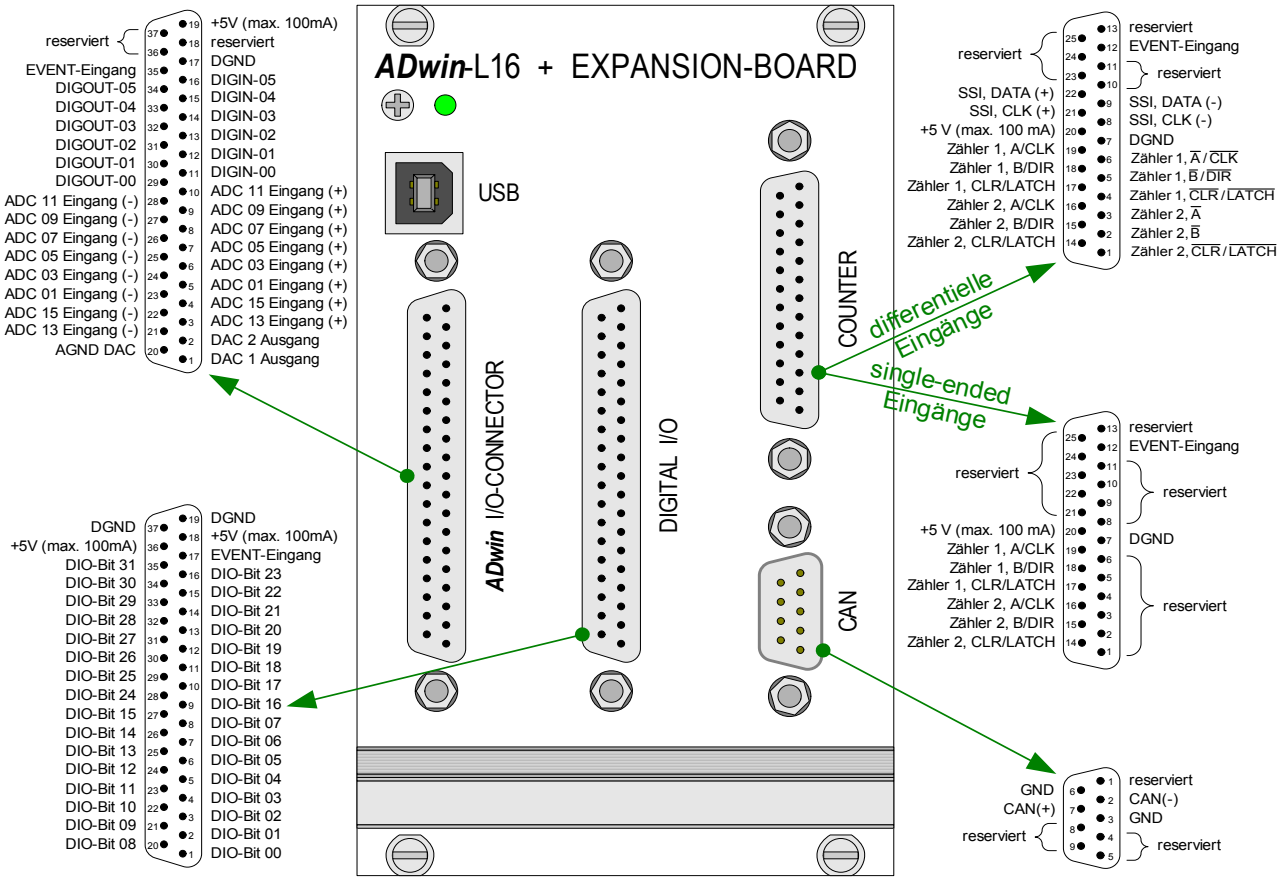


Abb. 24 – Übersichtsbild *L16-EURO-DIO1* mit Pinbelegungen

Auf der DIO1-Platine sind mehrere DIP-Schalter vorhanden, mit denen Sie Einstellungen für die Zähler und die CAN-Schnittstelle verändern können.

Die Einstellung ist nur bei ausgebautem bzw. geöffnetem Gerät möglich. Beachten Sie den Sicherheitshinweis am Anfang dieser Dokumentation.

Einstellungen an der Hardware

Bauform	Vorgehen
L16-EURO, L16-PCI	**Ausbau der Platine** Der Ausbau der Platine erfolgt umgekehrt zum Einbau, der im Handbuch „***ADwin***-Treiber-Installation beschrieben ist.
L16-EXT	**Öffnen des Gehäuses** Entfernen Sie an den schwarzen Seitenplatten die oberen Inbus-Schrauben (Innensechskant) mit einem 2 mm Inbus-Schlüssel und lösen die unteren Schrauben. Stellen Sie die Seitenplatten etwas schräg und ziehen das Oberteil ab. Merken Sie sich die Orientierung des Gehäuses für das spätere Schließen des Geräts. Sie blicken nun direkt auf DIO1-Platine.

Der **Zusammenbau** erfolgt in umgekehrter Reihenfolge zum Ausbau / Öffnen.

Lage der DIP-Schalter

Entnehmen Sie bitte die Lage der DIP-Schalter der nachfolgenden Abbildung. Die gestrichelten Rahmen zeigen die Zuordnung der DIP-Schalter zu den Zählern 1 und 2 (obere Hälfte, halbrechts) und zur CAN-Schnittstelle (unten links).

Die Einstellung der DIP-Schalter ist in Kapitel 8.2 “Zähler“ und Kapitel 8.3 “CAN-Bus“ beschrieben.

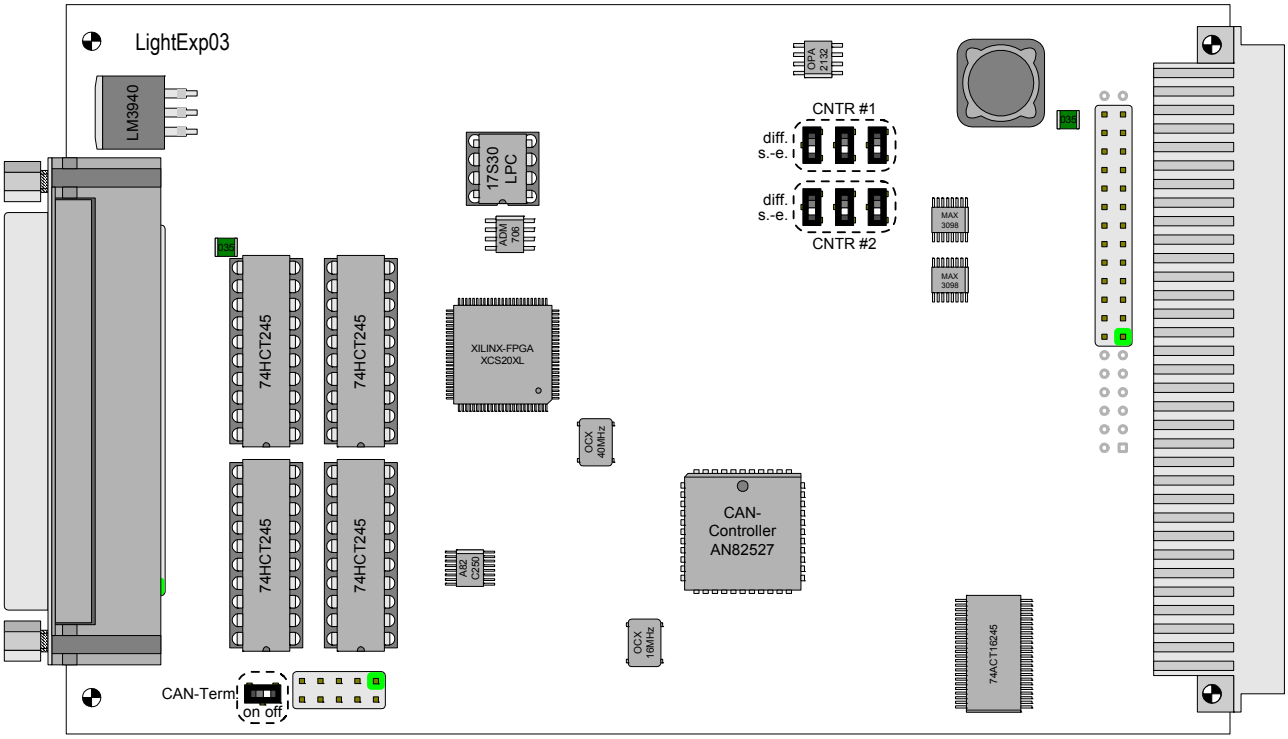


Abb. 25 – Lage der DIP-Schalter auf der DIO1-Platine

Die technischen Daten der DIO1-Erweiterung sind im Anhang enthalten.

8.1 Digitale Ein- und Ausgänge

Zusätzlich zu den digitalen Ein-/Ausgängen der Basisversion (DIGIN, DIGOUT, EVENT) stehen Ihnen auf der 37-poligen Sub-D-Buchse `Digital I/O` 32 digitale Ein- oder Ausgänge (Abkürzung: DIO) zur Verfügung. Sie sind in Gruppen zu jeweils 8 als Ein- oder Ausgang programmierbar.

Trigger-Eingang

Es je einen externen Trigger-Eingang (EVENT) an folgenden Sub-D-Buchsen: `ADwin I/O Connector`, `Counter` und `Digital I/O`. Die Eingänge `Counter` und `Digital I/O` sind galvanisch verbunden und haben einen gemeinsamen Pull-up-Widerstand von 4,7 kΩ, so dass Sie nach Bedarf einen der Eingänge auswählen können. Verwenden Sie nur einen der drei Event-Eingänge.

Mit einem externen Signal (Trigger) am Event-Eingang kann ein Prozesse ausgelöst werden, der sofort und vollständig abgearbeitet wird (siehe ***ADbasic***-Handbuch oder -Online-Hilfe, Kapitel: Prozesse im Betriebssystem“).

Die digitalen Eingänge sind TTL-kompatibel und gegen Überspannung nicht geschützt.

Einschaltkonfiguration

Nach dem Einschalten des Gerätes sind alle Anschlüsse als Eingang konfiguriert; dies entspricht dem Befehl `CONF_DIO_E(0)`. Mit dem Befehl

`CONF_DIO_E(n)`

programmieren Sie die 32 DIO-Leitungen in 4 Gruppen zu jeweils 8 Leitungen als Ein- oder Ausgang (siehe Kapitel 12).

Die folgende Tabelle zeigt Ihnen die 16 möglichen Konfigurationen dieses Befehls. In der untersten Zeile stehen die Befehle, die für bestimmte DIO-Gruppen anwendbar sind.

`CONF_DIO_E(n)`	DIO 31 : DIO 24	DIO 23 : DIO 16	DIO 15 : DIO 08	DIO 07 : DIO 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
8	OUT	IN	IN	IN
9	OUT	IN	IN	OUT
10	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 ***ADbasic***-Befehle:	`DIGIN_WORD2_E`
	`DIGOUT_WORD2_E`
	`DIGOUT_SET2_E`
	`DIGOUT_RESET2_E`
			`DIGIN_WORD1_E`
			`DIGOUT_WORD1_E`
			`DIGOUT_SET1_E`
			`DIGOUT_RESET1_E`	

Abb. 26 – Konfigurationen mit `CONF_DIO_E`

Näheres zur Programmierung zeitkritischer Aufgaben finden Sie im Kapitel auf Seite 17.

8.2 Zähler

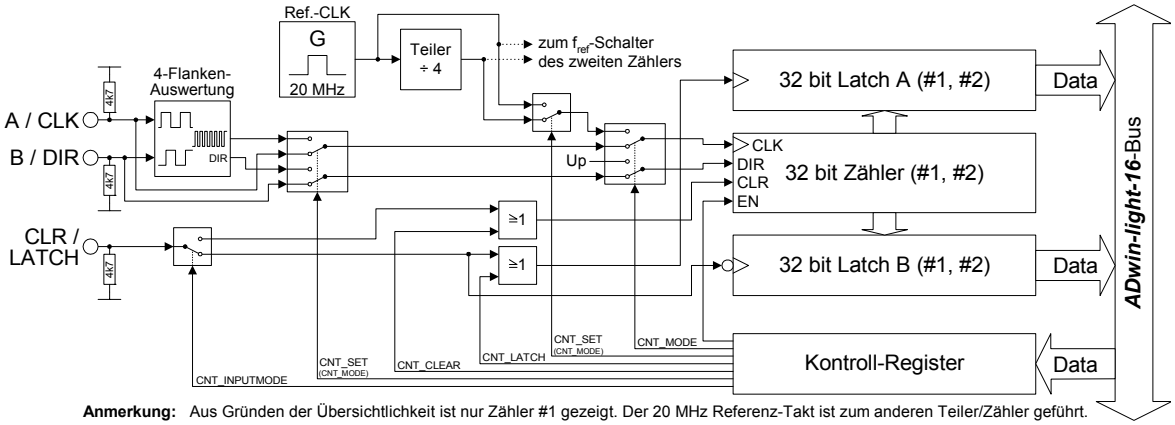
Zähler

Die Erweiterung DIO1 stellt **zwei 32 Bit Zähler** zur Verfügung, die Sie per Software sowohl einzeln als auch gemeinsam konfigurieren und auslesen können. Sie können an den Eingängen differentielle oder „single ended“-Signale anlegen.
Die Zähler ersetzen die Inkrementalzähler der Grundversion.

Latch

Die Zähler können **intern oder extern getaktet** werden und werden über zugeordnete Latches ausgelesen. Alle Zähler haben je ein Latch A sowie ein Latch B (die Grafik zeigt den Aufbau eines einzelnen Zählers).

Der Zählerstand kann mit Programmierbefehlen oder (bei entsprechender Einstellung) bei einem externen Signal an CLR/LATCH gelöscht oder in ein Latch übertragen werden.



Anmerkung: Aus Gründen der Übersichtlichkeit ist nur Zähler #1 gezeigt. Der 20 MHz Referenz-Takt ist zum anderen Teiler/Zähler geführt.

Abb. 27 – Schema DIO1-Zähler

Es gibt die Betriebsarten Ereigniszählung (externer Takt) und Pulsbreitenmessung (interner Takt); siehe auch Kapitel 8.2.2 / 8.2.3:

Externer Takteingang

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

   Es gibt die Modi:
   - **Takt und Richtung**: Jede positive Flanke an CLK in- oder dekrementiert den Zählerstand um eins. Das Signal an B/DIR bestimmt die Zählrichtung (0 = Dekrement; 1 = Inkrement).
   - **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

2. **Pulsbreitenmessung**: Die Signale zum In-/Dekrementieren erhält der Zähler von einem internen Referenztaktgeber mit einer Signalfrequenz von 20 MHz (alternativ 5 MHz nach einem Teiler). 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 berechnen:
   - die Periodendauer des Eingangssignals an CLR/LATCH aus den Werten in Latch A.
   - die Pulsbreite und Pausenzeit aus den Werten in Latch A und Latch B.

Betriebsart der Zählereingänge umstellen

Die Eingänge der Zähler können single ended oder differentiell betrieben werden. Die Einstellung bei Auslieferung ist nicht festgelegt. Stellen Sie daher die von Ihnen gewünschte Betriebsart an den DIP-Schaltern der DIO1-Platine ein (siehe Abb. 25 – Lage der DIP-Schalter auf der DIO1-Platine).

Für differentiellen Betrieb legen Sie alle 3 DIP-Schalter des entsprechenden Zählers nach oben um (siehe ebenfalls Abb. 25); für single ended-Betrieb legen Sie die Schalter nach unten um.

8.2.1 Programmierung

Die Zähler-Funktionen der DIO1-Erweiterung werden mit ***ADbasic***-Befehlen komfortabel programmiert. Die Befehle sind in einer Include-Datei enthalten; binden Sie diese Datei am Beginn des Programms ein mit der Befehlszeile

```
#INCLUDE ADWL16.INC
```

Die Zähler-Befehle der folgenden Tabelle sind im ***ADbasic***-Handbuch oder der Online-Hilfe vollständig beschrieben:

Zähler-Nr.	2	1	Kommentar
Bit	1	0	
CNT_CLEAR( )	0	0	ohne Einfluss
	1	1	Zähler löschen *
CNT_ENABLE( )	0	0	Zähler sperren
	1	1	Zähler freigeben (auf laufende Zähler achten)
CNT_INPUTMODE( )	0	0	CLR/LATCH-Eingang auf CLR-Modus stellen
	1	1	CLR/LATCH-Eingang auf LATCH-Modus stellen
CNT_LATCH( )	0	0	ohne Einfluss
	1	1	Zählerstand in Latch-Register A übernehmen*
CNT_MODE( )	0	0	externer Takteingang
	1	1	interner Referenztakt (20MHz / 5MHz)
CNT_SET( )	0	0	CNT_MODE-Bit = 0 : 4-Flankenauswertung
			CNT_MODE-Bit = 1 : interner Referenztakt von 20MHz
	1	1	CNT_MODE-Bit = 0 : Takt- und Richtungseingang (CLK & DIR)
			CNT_MODE-Bit = 1 : interner Referenztakt von 5MHz
CNT_CLEARENABLE()	0	0	CLR-Eingang sperren
	1	1	CLR-Eingang freigeben
CNT_GETSTATUS(#) **			Status-Register auslesen (Bedeutung der Bits siehe ***ADbasic***-Handbuch / Online-Hilfe)
CNT_READ(#) **			Zählerstand in Latch A übernehmen und auslesen
CNT_READLATCH(#) **			Latch A (getriggert durch positive Flanke) auslesen
CNT_READFLATCH(#) **			Latch B (getriggert durch negative Flanke) auslesen

* Nach der Befehlsausführung werden gesetzte Bits automatisch wieder gelöscht. Bei anderen Befehlen müssen gesetzte Bits durch einen neuen Befehlsaufruf gelöscht werden.
** # = Zähler-Nummer 1 oder 2

Abb. 28 – DIO1-Zähler Befehle Kurzreferenz

Sie können mit den Befehlen der Tabelle jeden Zähler einzeln oder beide Zähler gemeinsam konfigurieren.

Initialisierung

Initialisieren Sie die Zähler bitte in dieser Reihenfolge:

1. Gewünschten Zähler sperren (CNT_ENABLE)

   Der Befehl CNT_ENABLE spricht alle Zähler gemeinsam an. Auch wenn Sie nur einen Zähler sperren (Bit = 0) möchten, müssen Sie alle anderen Zähler konfigurieren, deren Zustand unverändert bleiben soll (Bit = 1). Beim Freigeben des Zählers gilt dies entsprechend.

2. Betriebsart / Modus einstellen
   (CNT_MODE, CNT_SET, CNT_INPUTMODE)

   Beachten Sie die Abhängigkeit des Befehls CNT_SET vom Befehl CNT_MODE.

3. Zähler löschen (CNT_CLEAR)

4. Zähler freigeben (CNT_ENABLE)

Werten Sie bitte den Zählerinhalt nur mit Variablen vom Typ LONG aus. ***ADbasic*** behält dann intern das gelesene Bitmuster unverändert bei und berücksichtigt automatisch den Übergang zwischen positivem und negativem Zahlenbereich (siehe auch Seite 16). Damit gilt:

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.

Ein Prozess kann sehr schnell abgearbeitet werden, wenn Sie mit den Befehlen PEEK und POKE direkt auf die Steuer- und Datenregister zugreifen (siehe auch Kapitel oder ***ADbasic***-Handbuch / Online-Hilfe).
Die Hardware-Adressen der DIO1-Erweiterung können Sie folgender Tabelle entnehmen (vgl. Abb. 28 – DIO1-Zähler Befehle Kurzreferenz).

Adresse [hex]	Funktion	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Kommentar
20 40 02 04	Inhalt Latch A, Zähler-#1	x	x	x	x	x	x	x	x	x	x	x	x	x : Inhalt des Latch
20 40 02 08	Inhalt Latch B, Zähler-#1	x	x	x	x	x	x	x	x	x	x	x	x	x : Inhalt des Latch
20 40 02 14	Inhalt Latch A, Zähler-#2	x	x	x	x	x	x	x	x	x	x	x	x	x : Inhalt des Latch
20 40 02 18	Inhalt Latch B, Zähler-#2	x	x	x	x	x	x	x	x	x	x	x	x	x : Inhalt des Latch
20 40 03 00	Zähler sperren / freigeben CNT_ENABLE()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : Zähler sperren x = 1 : Zähler freigeben
20 40 03 10	Zähler löschen CNT_CLEAR()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : kein Einfluss x = 1 : Zähler löschen
20 40 03 20	Zähler latchen CNT_LATCH()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : kein Einfluss x = 1 : Zähler latchen
20 40 03 30	Eingang: CLR oder LATCH	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : CLR-Eingang x = 1 : LATCH-Eingang
20 40 03 40	Impuls-/Ereigniszähler- oder Puls-breiten-/Periodendauermessung	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : externer Takteingang x =1: interner Referenztakt (20MHz/5MHz)
20 40 03 50	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: 20MHz; x=1: 5MHz
20 40 04 54	DIO_Erweiterung Bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x: setzt/löscht Ausgang <Bit-Nr.>
20 40 04 64	DIO_Erweiterung Bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x: setzt/löscht Ausgang <Bit-Nr.+16>
20 40 04 74	DIO_Erweiterung Set_Bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang <Bit-Nr.> setzen *
20 40 04 84	DIO_Erweiterung Set_Bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang <Bit-Nr.+16> setzen*
* Funktion bei Eingängen ohne Einfluss														

Abb. 29 – DIO1 Hardware-Adressen der Steuer- und Datenregister

Adresse [hex]	Funktion	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Kommentar
20 40 04 94	DIO_Erweiterung Reset_Bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang <Bit-Nr.> löschen *
20 40 04 A4	DIO_Erweiterung Reset_Bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang <Bit-Nr.+16> löschen *
20 40 04 6C	Ein-/Ausgänge konfigurieren CONF_DIO() Bit 0: DIO 00...07; Bit 1: DIO 08...15 Bit 2: DIO 16...23; Bit 3: DIO 24...31	-	-	-	-	-	-	-	-	x	x	x	x	x = 0: Kanäle als Eingang definieren x = 1: Kanäle als Ausgang definieren
* Funktion bei Eingängen ohne Einfluss														

Abb. 29 – DIO1 Hardware-Adressen der Steuer- und Datenregister

8.2.2 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 entweder 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)

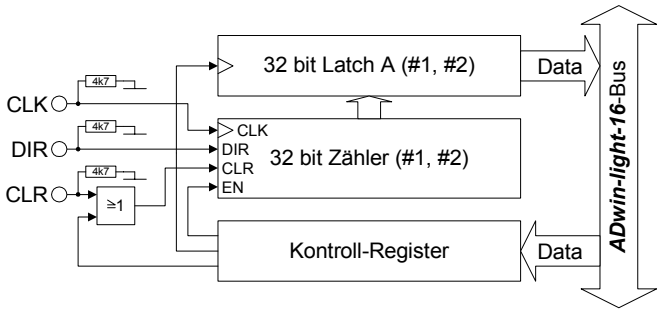
Löschen

– den Zähler zu löschen (CLR)

Latchen

– den Zählerstand in Latch A zu übernehmen (LATCH).

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 als konstante Spannung oder als Rechtecksignal (z.B. vorgegeben durch eine externe Logikschaltung) angelegt sein.

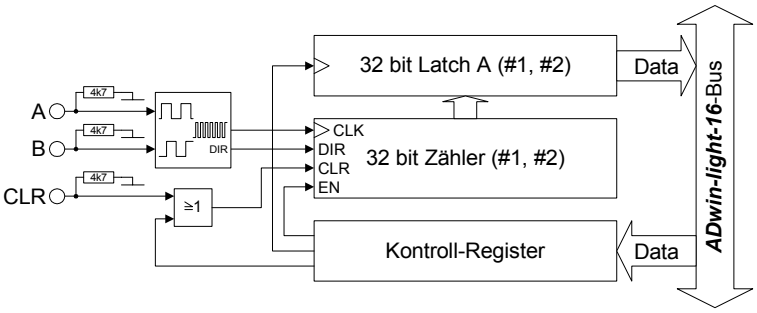
Programmierbeispiel

```
INIT:                          Initialisieren
. . .
CNT_ENABLE(0)                  Alle Zähler sperren
CNT_CLEAR(1)                   Inhalt des Zählers 1 löschen
CNT_MODE(0)                    externer Takteingang am CLK-Eingang
CNT_SET(1)                     Modus Takt und Richtung aktivieren
CNT_INPUTMODE(0)               Eingang CLR/LATCH auf CLR-Modus stellen
CNT_ENABLE(1)                  Zähler 1 freigeben
. . .

EVENT:
CNT_LATCH(1)                   Aktuellen Zählerstand in Latch A übernehmen
CNT_READLATCH(1)               Latch A auslesen
. . .                          Auswerten im Programm
```

Vier-Flanken-Auswertung

Dieser Modus ermittelt Takt und Zählrichtung aus zwei Rechteck-Signalen, die an den Eingängen A und B um 90 Grad (ideal) versetzt anliegen. Die Zählrichtung ergibt sich logisch aus der zeitlichen Reihenfolge der steigenden / fallenden Flanken zueinander.



Berücksichtigen Sie bitte:

– Der Zähler registriert bei einem Zyklus 4 Flanken.

– Die maximale Zählfrequenz beträgt 20 MHz. Gemeinsam mit den 4 Flanken je Zyklus ergibt sich daraus eine maximale Eingangsfrequenz (an A oder B) 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 (d.h. ≠90 Grad) 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:`	Initialisieren
`. . .`	
`CNT_ENABLE(0)`	Alle Zähler sperren
`CNT_CLEAR(1)`	Inhalt des Zählers 1 löschen
`CNT_MODE(0)`	externer Takteingang am CLK-Eingang
`CNT_SET(0)`	Modus Vier-Flankenauswertung aktivieren
`CNT_INPUTMODE(0)`	Eingang CLR/LATCH auf CLR-Modus stellen
`CNT_ENABLE(1)`	Zähler 1 freigeben
`. . .`	
`EVENT:`	
`CNT_LATCH(1)`	Aktuellen Zählerstand in Latch A übernehmen
`CNT_READLATCH(1)`	Latch A auslesen
`. . .`	Auswerten im Programm

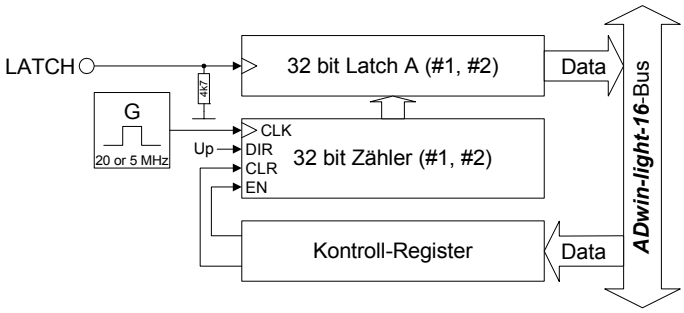
8.2.3 Betriebsart Pulsbreiten- und Periodendauer-Messung

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 Pulsbreite eines Rechtecksignals am Eingang CLR/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.

Periodendauer-Messung

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



Programmierbeispiel

```
INIT:
. . .
CNT_ENABLE(0)
CNT_CLEAR(1)
CNT_MODE(1)
CNT_SET(0)
CNT_SET(1)
CNT_INPUTMODE(1)
CNT_ENABLE(1)
. . .

EVENT:
CNT_READLATCH(1)
. . .
```

Initialisieren

Alle Zähler sperren

Inhalt des Zählers 1 löschen

Modus interner Referenztakt am internen CLK-Eingang des Zählers ...

... mit 20 MHz oder

... mit 5 MHz

Eingang CLR/LATCH auf LATCH-Modus stellen

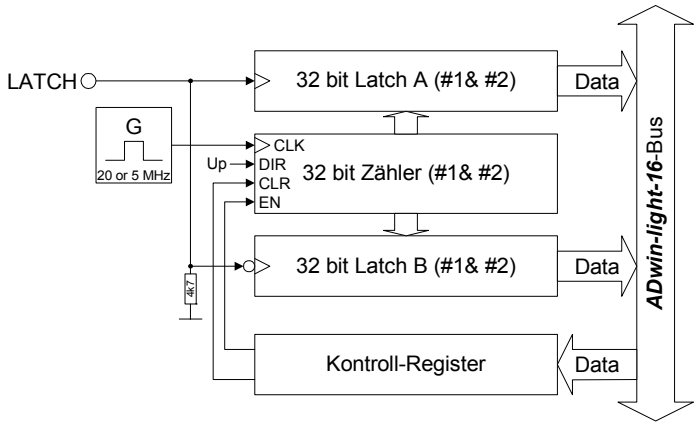
Zähler 1 freigeben

Latch A auslesen

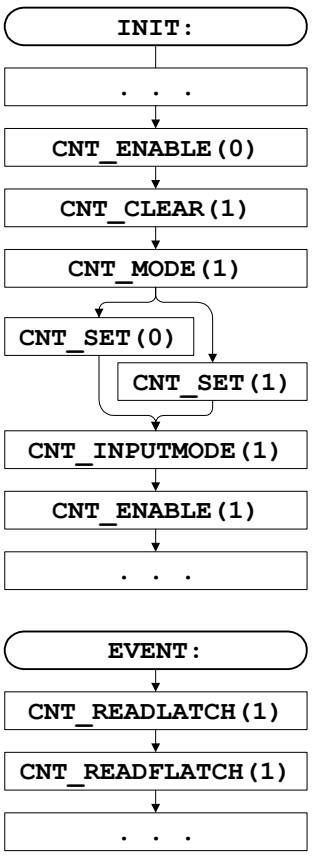
Auswerten im Programm

Messung von Pulsbreite und Pausenzeit

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



Programmierbeispiel



Initialisieren

Alle Zähler sperren

Inhalt des Zählers 1 löschen

Modus interner Referenztakt am internen CLK-Eingang des Zählers ...

... mit 20 MHz oder

... mit 5 MHz

Eingang CLR/LATCH auf LATCH-Modus stellen

Zähler 1 freigeben

Latch A auslesen

Latch B auslesen

Auswerten im Programm

8.3 CAN-Bus

Die Erweiterungskarte DIO1 besitzt eine CAN-Bus Schnittstelle für das „high speed“-CAN-Protokoll. Die Anschlüsse stehen auf einem 9-poligen Sub-D-Verbinder (Stecker / männlich) zur Verfügung; die Pin-Belegung ist auf Seite 27 dargestellt.

Bus-Terminierung

Der CAN-Bus muss an seinem physikalischen Ende (und nur dort) mit einem Abschlusswiderstand terminiert werden, also nur am ersten und am letzten CAN-Knoten. Dies geschieht, indem Sie einen DIP-Schalter auf der DIO1-Platine entsprechend einstellen (siehe Abb. 25 – Lage der DIP-Schalter auf der DIO1-Platine). Wenn die Terminierung erforderlich ist, legen Sie den DIP-Schalter in Richtung zum CAN-Stecker um.

8.3.1 Funktionsbeschreibung des CAN-Controllers

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 255 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 1 MHz einstellbar und wird standardmäßig mit 1 MHz betrieben; bei CAN low speed beträgt die max. Frequenz 125 kHz. 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.

Verwaltung von Nachrichten

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 1 MHz beträgt. Soll der CAN-Bus mit einer anderen Frequenz betrieben werden, müssen die Werte im „Bit Timing Register 0“ (BTR0, Adresse 3Fh) und „Bit Timing Register 1“ (BTR1, Adresse 4Fh) geändert werden. Für eine große Auswahl an Busfrequenzen 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 nachfolgend beschrieben.

	Bit Timing Register 0 (BTR0)		Bit Timing Register 1 (BTR1)		
Bits	7…6	5…0	7	6…4	3…0
Sub-Reg.	SJW	BRP	SPL	TSEG2	TSEG1

Die folgende Tabelle zeigt zulässige Werte und die Bedeutung der Bereiche:

Bereich	zulässige Werte	Bedeutung
SJW	0 … 3	Maximale Pulsdehnung bei der Bus-Synchronisation
BRP	0 … 63	Vor-Teiler
SPL	0 … 1	Sampling Mode
TSEG1	2 … 15	Zeitsegmente vor der Abtastung
TSEG2	1 … 7	Zeitsegmente nach der Abtastung

Die Bereiche SJW und SPL sind standardmäßig 0 und sollten nur bei Bedarf geändert werden. Der Sample-Punkt (bestimmt durch TSEG1 und TSEG2) sollte so gewählt werden, dass er zwischen 50% und 80% der Gesamt-Bitlänge liegt.

Die CAN-Bus-Frequenz berechnet sich folgendermaßen:

$$f_{CAN} = \frac{8\,MHz}{(BRP + 1)(TSEG1 + TSEG2 + 3)}$$

Die folgende Tabelle zeigt die Einstellungen für die gängigsten Baudraten.

Baudrate [kBit/s]	125	250	500	1000
BRP	3	1	0	0
TSEG1	6	6	6	2
TSEG2	7	7	7	3
BTR0	03h	01h	00h	00h
BTR1	76h	76h	76h	32h
Sample-Punkt [%]	54	54	54	60

Abb. 30 – CAN: Gängige Baudraten einstellen

Der Zugriff auf die beiden Timing-Register BTR0 und BTR1 ist nur möglich, wenn der Zugriff zuvor freigegeben wird. Dies geschieht mit dem „CCE-Bit“ im „Control Register“. Das Bit muss anschließend wieder zurückgesetzt werden.

Interrupt / Event

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.

Wenn Sie Message-Objekte für das Auslösen von Interrupts freigeben, dürfen die Event-Eingänge (siehe Seite 29) nicht gleichzeitig beschaltet sein.

Programmierung

Die CAN-Funktionen der DIO1-Erweiterung werden mit ***ADbasic***-Befehlen komfortabel programmiert. Die Befehle sind in einer Include-Datei enthalten; binden Sie diese Datei am Beginn des Programms ein mit der Befehlszeile

```
#INCLUDE ADWL16.INC
```

Die unten stehenden CAN-Befehle sind im ***ADbasic***-Handbuch und der Online-Hilfe vollständig beschrieben:

Bereich	Befehle
Initialisierung des CAN-Controllers	`INIT_CAN`
Setzen und Lesen von Registern	`SET_REG`, `GET_REG`
Initialisieren von Message Objekten	`EN_RECEIVE`, `EN_TRANSMIT`
Senden und Empfangen von Datensätzen	`TRANSMIT`, `READ_MSG`
Freigeben von Interrupts	`EN_INTERRUPT`
Baudrate einstellen	`SET_CAN_BAUDRATE`

Abb. 31 – DIO1 Übersicht CAN-Befehle

8.4 SSI-Decoder

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

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

Die Anschlüsse des Decoders stehen auf dem Stecker COUNTER (25-polig, Sub-D) zur Verfügung, und zwar auf den Pins 8, 9, 21 und 22 (siehe Abb. 24 – Übersichtsbild L16-EURO-DIO1 mit Pinbelegungen). Auf den Pins 7 und 20 stehen DGND und +5V zur Verfügung.

Eigenschaften einstellen

Folgende Eigenschaften des SSI-Decoders 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. 32 – Listing: Konvertierung von Gray- in Binär-Code

Programmierung

Die Funktionalität des SSI-Decoders wird mit ***ADbasic***-Befehlen komfortabel programmiert:

Bereich	Befehle
Decoder-Modus einstellen	**SSI_MODE**
Daten empfangen	**SSI_READ**
Encoder-Auflösung einstellen	**SSI_SET_BITS**
Encoder-Taktrate einstellen	**SSI_SET_CLOCK**
Auslesen des Encoders starten	**SSI_START**
Lesestatus	**SSI_STATUS**

Die Befehle sind in der Include-Datei <ADWL16.INC> enthalten und werden im Handbuch ***ADbasic*** und der Online-Hilfe erläutert.

9 DIO2- / DIO3-Erweiterung

Die DIO2-Erweiterung stellt Ihnen zusätzlich zur Verfügung:

– 32 digitale Ein-/Ausgänge (programmierbar in Gruppen zu 8), Seite 45.

  Die digitalen Ein- und Ausgänge der Basisversion stehen weiterhin zur Verfügung.

– 2 Zähler, Seite 46: 32 Bit Vor-/Rückwärtszähler zur Impuls-, Perioden-dauer- und Tastverhältnismessung sowie einer Vierflankenauswertung zum Anschluss von Encodern.
  Zähler 1 hat TTL-Eingänge (single ended), Zähler 2 hat differentielle Eingänge.

  Die Zähler der Basisversion werden von den DIO2-Zählern ersetzt.

– 1 SSI-Decoder (Seite 53)

  Der SSI-Decoder eignet sich zum Anschluss von Inkremental-Encodern mit SSI-Schnittstelle. Die Eingänge liegen auf dem Stecker COUNTER; die Signale sind differentiell und für RS422/485-Pegel (5V) ausgelegt.

Das Schema zeigt gemeinsam die Grundfunktionen eines L16-Geräts mit den Zusatzfunktionen der DIO2-Erweiterung (als USB-Variante).

Die DIO3-Erweiterung enthält nur die 32 digitalen Ein-/Ausgänge (siehe Seite 45). Eigenschaften und Funktion dieser Ein-/ Ausgänge sind identisch zu den Ein-/Ausgängen der DIO2-Erweiterung.

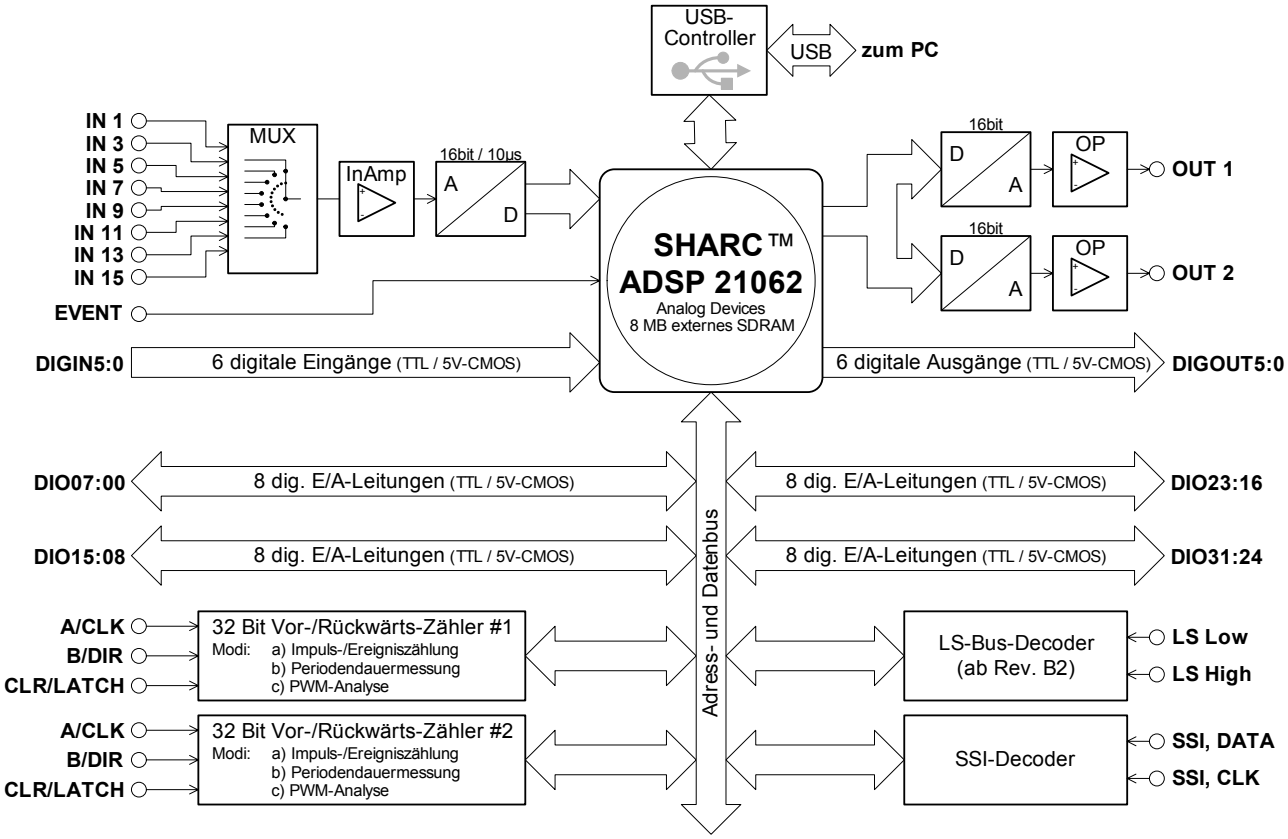


Abb. 33 – Schema *L16-DIO2* (mit USB-Schnittstelle)

Die Zähler der Basisversion werden von den DIO2-Zählern ersetzt. Die digitalen Kanäle der Basisversion bleiben unverändert bestehen.

Die Pin-Belegung am Anschluss „***ADwin*** I/O-CONNECTOR“ ist identisch mit der Basisversion. Abweichend dazu dienen bei der Erweiterung DIO2 die Pins 14…16 nun alternativ auch als Eingänge für den Zähler 1.

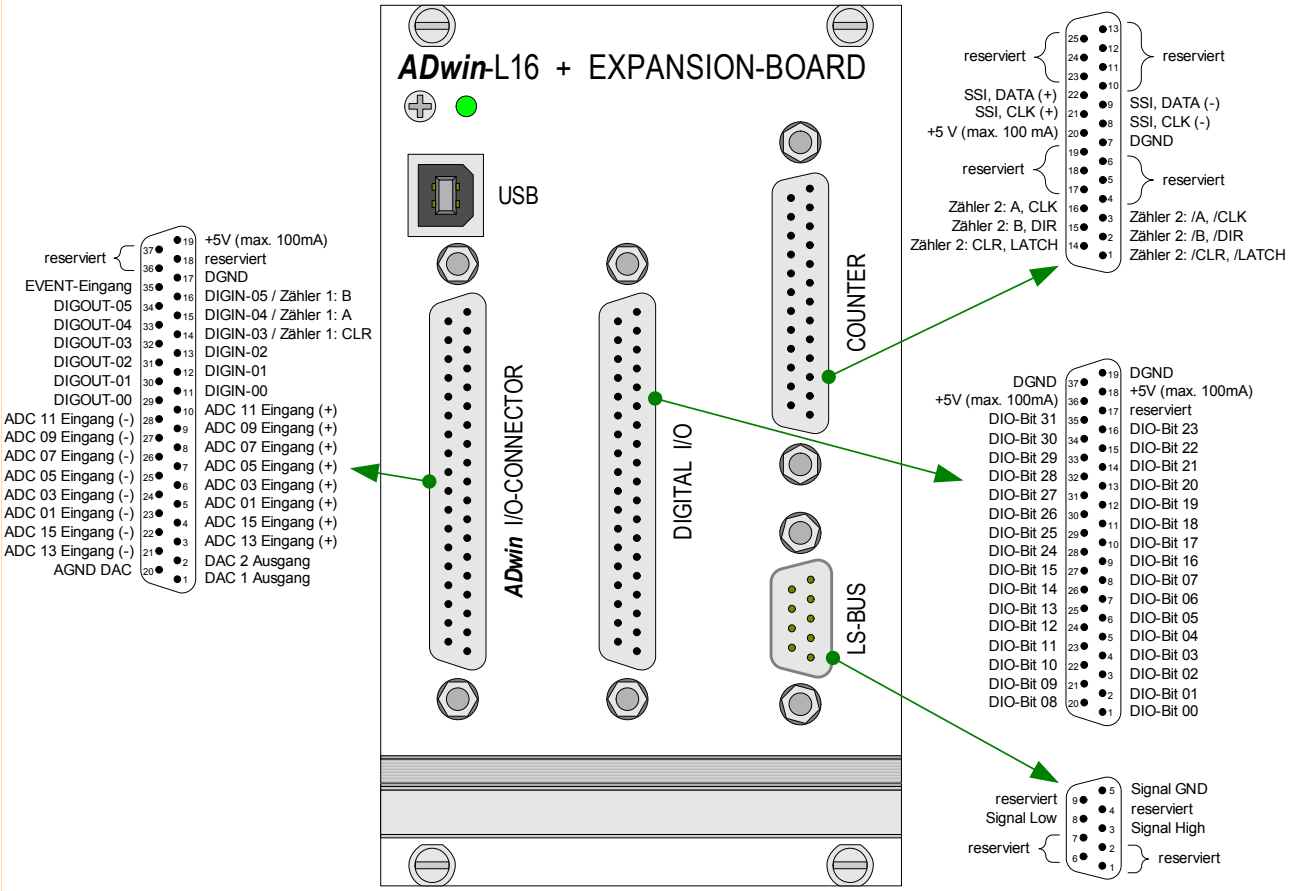


Abb. 34 – Übersichtsbild *L16-EURO-DIO2* mit Pinbelegungen

Die technischen Daten der DIO2-Erweiterung sind im Anhang enthalten.

9.1 Digitale Ein- und Ausgänge

Zusätzlich zu den digitalen Ein-/Ausgängen der Basisversion (DIGIN, DIGOUT, EVENT) stehen Ihnen auf der 37-poligen Sub-D-Buchse „Digital I/O“ 32 digitale Ein- oder Ausgänge (Abkürzung: DIO) zur Verfügung. Sie sind in Gruppen zu jeweils 8 als Ein- oder Ausgang programmierbar.

Die digitalen Eingänge sind TTL-kompatibel und gegen Überspannung nicht geschützt.

Einschaltkonfiguration

Nach dem Einschalten des Gerätes sind alle 32 DIO-Leitungen als Eingang konfiguriert; dies entspricht dem Befehl `CONF_DIO_E(0)`.

Mit dem Befehl `CONF_DIO_E(n)` programmieren Sie die 32 DIO-Leitungen in 4 Gruppen zu jeweils 8 Leitungen als Ein- oder Ausgang (siehe Kapitel 12 oder ***ADbasic*** Online-Hilfe).

Die folgende Tabelle zeigt Ihnen die 16 möglichen Konfigurationen dieses Befehls. In der untersten Zeile stehen die Befehle, die für die DIO-Gruppen anwendbar sind.

`CONF_DIO_E(n)`	DIO 31 : DIO 24	DIO 23 : DIO 16	DIO 15 : DIO 08	DIO 07 : DIO 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
8	OUT	IN	IN	IN
9	OUT	IN	IN	OUT
10	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 ***ADbasic***-Befehle:	`DIGIN_WORD2_E`
	`DIGOUT_WORD2_E`
	`DIGOUT_SET2_E`
	`DIGOUT_RESET2_E`
			`DIGIN_WORD1_E`
			`DIGOUT_WORD1_E`
			`DIGOUT_SET1_E`
			`DIGOUT_RESET1_E`	

	`DIGIN_LONG_E`, `DIGOUT_LONG_E`			

Abb. 35 – Konfigurationen mit `CONF_DIO_E`

Näheres zur Programmierung zeitkritischer Aufgaben finden Sie im Kapitel auf Seite 17.

9.2 Zähler

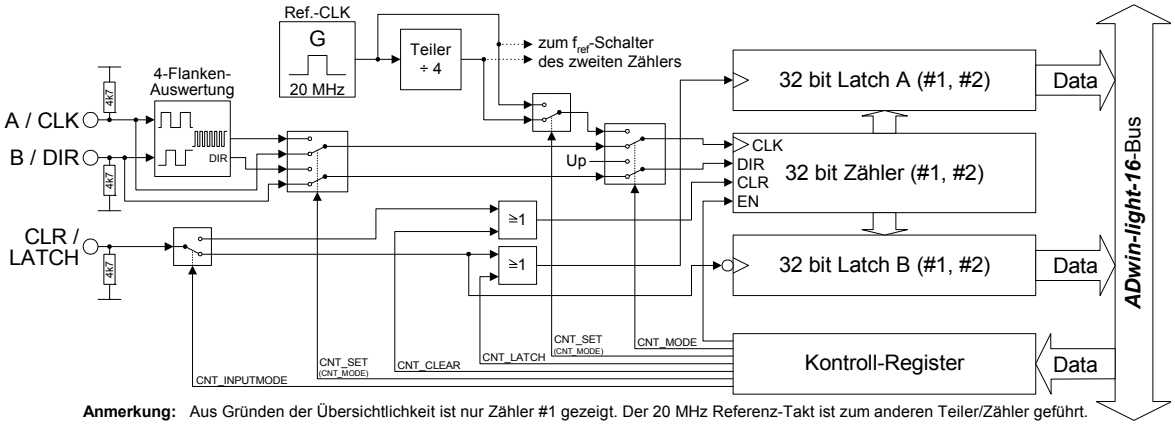
Zähler

Die Erweiterung DIO2 stellt **zwei 32 Bit Zähler** zur Verfügung, die Sie per Software sowohl einzeln als auch gemeinsam konfigurieren und auslesen können. Sie können an Zähler 1 TTL-Signale (single ended) anlegen, an Zähler 2 differentielle Signale.
Die Zähler ersetzen die Inkrementalzähler der Grundversion.

Latch

Die Zähler können **intern oder extern getaktet** werden und werden über zugeordnete Latches ausgelesen. Alle Zähler haben je ein Latch A sowie ein Latch B (die Grafik zeigt den Aufbau eines einzelnen Zählers).

Der Zählerstand kann mit Programmierbefehlen oder (bei entsprechender Einstellung) bei einem externen Signal an CLR/LATCH gelöscht oder in ein Latch übertragen werden.



Anmerkung: Aus Gründen der Übersichtlichkeit ist nur Zähler #1 gezeigt. Der 20 MHz Referenz-Takt ist zum anderen Teiler/Zähler geführt.

Abb. 36 – Schema DIO2-Zähler

Es gibt die Betriebsarten Ereigniszählung (externer Takt) und Pulsbreitenmessung (interner Takt); siehe auch Kapitel 9.2.2 / 9.2.3:

Externer Takteingang

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

   Es gibt die Modi:
   - **Takt und Richtung**: Jede positive Flanke an CLK in- oder dekrementiert den Zählerstand um eins. Das Signal an B/DIR bestimmt die Zählrichtung (0 = Dekrement; 1 = Inkrement).
   - **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

2. **Pulsbreitenmessung**: Die Signale zum In-/Dekrementieren erhält der Zähler von einem internen Referenztaktgeber mit einer Signalfrequenz von 20 MHz (alternativ 5 MHz nach einem Teiler). 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 berechnen:
   - die Periodendauer des Eingangssignals an CLR/LATCH aus den Werten in Latch A.
   - die Pulsbreite und Pausenzeit aus den Werten in Latch A und Latch B.

9.2.1 Programmierung

Die Zähler-Funktionen der DIO2-Erweiterung werden mit ***ADbasic***-Befehlen komfortabel programmiert. Die Befehle sind in einer Include-Datei enthalten; binden Sie diese Datei am Beginn des Programms ein mit der Befehlszeile

```
#INCLUDE ADWL16.INC
```

Die Zähler-Befehle der folgenden Tabelle sind im ***ADbasic***-Handbuch oder der Online-Hilfe vollständig beschrieben:

Zähler-Nr.	2	1	Kommentar
Bit	1	0	
CNT_CLEAR( )	0	0	ohne Einfluss
	1	1	Zähler löschen *
CNT_ENABLE( )	0	0	Zähler sperren
	1	1	Zähler freigeben (auf laufende Zähler achten)
CNT_INPUTMODE( )	0	0	CLR/LATCH-Eingang auf CLR-Modus stellen
	1	1	CLR/LATCH-Eingang auf LATCH-Modus stellen
CNT_LATCH( )	0	0	ohne Einfluss
	1	1	Zählerstand in Latch-Register A übernehmen*
CNT_MODE( )	0	0	externer Takteingang
	1	1	interner Referenztakt (20MHz / 5MHz)
CNT_SET( )	0	0	CNT_MODE-Bit = 0 : 4-Flankenauswertung
			CNT_MODE-Bit = 1 : interner Referenztakt von 20MHz
	1	1	CNT_MODE-Bit = 0 : Takt- und Richtungseingang (CLK & DIR)
			CNT_MODE-Bit = 1 : interner Referenztakt von 5MHz
CNT_CLEARENABLE()	0	0	CLR-Eingang sperren
	1	1	CLR-Eingang freigeben
CNT_GETSTATUS(#) **			Status-Register auslesen (Bedeutung der Bits siehe ***ADbasic***-Handbuch / Online-Hilfe)
CNT_READ(#) **			Zählerstand in Latch A übernehmen und auslesen
CNT_READLATCH(#) **			Latch A (getriggert durch positive Flanke) auslesen
CNT_READFLATCH(#) **			Latch B (getriggert durch negative Flanke) auslesen

* Nach der Befehlsausführung werden gesetzte Bits automatisch wieder gelöscht. Bei anderen Befehlen müssen gesetzte Bits durch einen neuen Befehlsaufruf gelöscht werden.
** # = Zähler-Nummer 1 oder 2

Abb. 37 – DIO2-Zähler Befehle Kurzreferenz

Sie können mit den Befehlen der Tabelle jeden Zähler einzeln oder beide Zähler gemeinsam konfigurieren.

Initialisierung

Initialisieren Sie die Zähler bitte in dieser Reihenfolge:

1. Gewünschten Zähler sperren (CNT_ENABLE)

   Der Befehl CNT_ENABLE spricht alle Zähler gemeinsam an. Auch wenn Sie nur einen Zähler sperren (Bit = 0) möchten, müssen Sie alle anderen Zähler konfigurieren, deren Zustand unverändert bleiben soll (Bit = 1). Beim Freigeben des Zählers gilt dies entsprechend.

2. Betriebsart / Modus einstellen (CNT_MODE, CNT_SET, CNT_INPUTMODE)

   Beachten Sie die Abhängigkeit des Befehls CNT_SET vom Befehl CNT_MODE.

3. Zähler löschen (CNT_CLEAR)

4. Zähler freigeben (CNT_ENABLE)

Werten Sie bitte den Zählerinhalt nur mit Variablen vom Typ LONG aus. ***ADbasic*** behält dann intern das gelesene Bitmuster unverändert bei und berücksichtigt automatisch den Übergang zwischen positivem und negativem Zahlenbereich (siehe auch Seite 16). Damit gilt:

Die Zählrichtung (vor- oder rückwärts) ergibt sich zuverlässig nur aus dem

Vorzeichen der Differenz: [neuer Zählerstand] – [alter Zählerstand]

und nicht aus dem *Vergleich* (< >) der Zählerstände.

Bestimmen Sie die Differenz nur mit Variablen vom Typ LONG.

Ein Prozess kann sehr schnell abgearbeitet werden, wenn Sie mit den Befehlen PEEK und POKE direkt auf die Steuer- und Datenregister zugreifen (siehe auch Kapitel oder ***ADbasic***-Handbuch / Online-Hilfe).
Die Hardware-Adressen der DIO2-Erweiterung können Sie folgender Tabelle entnehmen (vgl. Abb. 37 – DIO2-Zähler Befehle Kurzreferenz).

Adresse [hex]	Funktion	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Kommentar
20 40 02 04	Inhalt Latch A, Zähler-#1	x	x	x	x	x	x	x	x	x	x	x	x	x : Inhalt des Latch
20 40 02 08	Inhalt Latch B, Zähler-#1	x	x	x	x	x	x	x	x	x	x	x	x	x : Inhalt des Latch
20 40 02 14	Inhalt Latch A, Zähler-#2	x	x	x	x	x	x	x	x	x	x	x	x	x : Inhalt des Latch
20 40 02 18	Inhalt Latch B, Zähler-#2	x	x	x	x	x	x	x	x	x	x	x	x	x : Inhalt des Latch
20 40 03 00	Zähler sperren / freigeben CNT_ENABLE()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : Zähler sperren x = 1 : Zähler freigeben
20 40 03 10	Zähler löschen CNT_CLEAR()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : kein Einfluss x = 1 : Zähler löschen
20 40 03 20	Zähler latchen CNT_LATCH()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : kein Einfluss x = 1 : Zähler latchen
20 40 03 30	Eingang: CLR oder LATCH	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : CLR-Eingang x = 1 : LATCH-Eingang
20 40 03 40	Impuls-/Ereigniszähler- oder Pulsbreiten-/Periodendauermessung	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : externer Takteingang x =1: interner Referenztakt (20MHz/5MHz)
20 40 03 50	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: 20MHz; x=1: 5MHz
20 40 04 54	DIO_Erweiterung Bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x: setzt/löscht Ausgang <Bit-Nr.>
20 40 04 64	DIO_Erweiterung Bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x: setzt/löscht Ausgang <Bit-Nr.+16>
20 40 04 74	DIO_Erweiterung Set_Bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang <Bit-Nr.> setzen *
20 40 04 84	DIO_Erweiterung Set_Bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang <Bit-Nr.+16> setzen*
20 40 04 94	DIO_Erweiterung Reset_Bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang <Bit-Nr.> löschen *
20 40 04 A4	DIO_Erweiterung Reset_Bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang <Bit-Nr.+16> löschen *
20 40 04 6C	Ein-/Ausgänge konfigurieren CONF_DIO() Bit 0: DIO 00...07; Bit 1: DIO 08...15 Bit 2: DIO 16...23; Bit 3: DIO 24...31	-	-	-	-	-	-	-	-	x	x	x	x	x = 0: Kanäle als Eingang definieren x = 1: Kanäle als Ausgang definieren
* Funktion bei Eingängen ohne Einfluss														

Abb. 38 – DIO2 Hardware-Adressen der Steuer- und Datenregister

9.2.2 Betriebsart Impuls-/Ereigniszähler

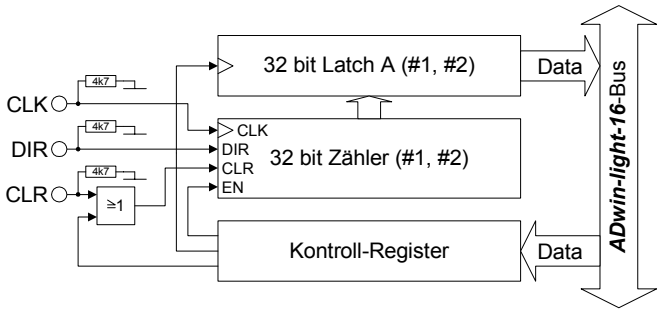
Externe Rechtecksignale an den Eingängen A/CLK und B/DIR takten in dieser Betriebsart den jeweiligen Zähler. Mit `CNT_SET` aktivieren Sie entweder den Modus zur Ermittlung von Taktfrequenz und Richtung oder die Vierflankenauswertung.

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

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

– den Zählerstand in Latch A zu übernehmen (LATCH). **Latchen**

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 als konstante Spannung oder als Rechtecksignal (z.B. vorgegeben durch eine externe Logikschaltung) angelegt sein.

Programmierbeispiel

```
INIT:
. . .
CNT_ENABLE(0)
CNT_CLEAR(1)
CNT_MODE(0)
CNT_SET(1)
CNT_INPUTMODE(0)
CNT_ENABLE(1)
. . .
```

Initialisieren

Alle Zähler sperren

Inhalt des Zählers 1 löschen

externer Takteingang am CLK-Eingang

Modus Takt und Richtung aktivieren

Eingang CLR/LATCH auf CLR-Modus stellen

Zähler 1 freigeben

```
EVENT:
CNT_LATCH(1)
CNT_READLATCH(1)
. . .
```

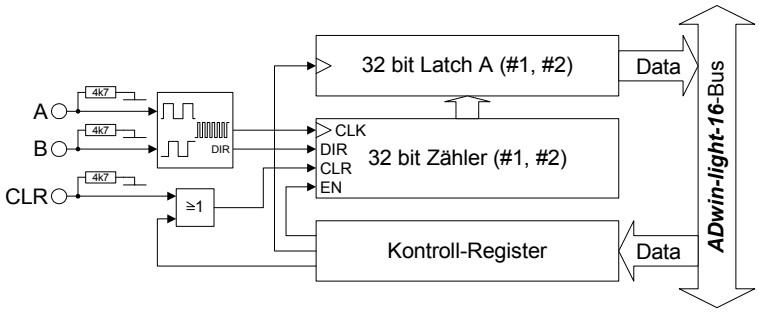
Aktuellen Zählerstand in Latch A übernehmen

Latch A auslesen

Auswerten im Programm

Vier-Flanken-Auswertung

Dieser Modus ermittelt Takt und Zählrichtung aus zwei Rechteck-Signalen, die an den Eingängen A und B um 90 Grad (ideal) versetzt anliegen. Die Zählrichtung ergibt sich logisch aus der zeitlichen Reihenfolge der steigenden / fallenden Flanken zueinander.



Berücksichtigen Sie bitte:

– Der Zähler registriert bei einem Zyklus 4 Flanken.

– Die maximale Zählfrequenz beträgt 20 MHz. Gemeinsam mit den 4 Flanken je Zyklus ergibt sich daraus eine maximale Eingangsfrequenz (an A oder B) 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 (d.h. ≠90 Grad) 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:                                  Initialisieren
. . .
CNT_ENABLE(0)                          Alle Zähler sperren
CNT_CLEAR(1)                           Inhalt des Zählers 1 löschen
CNT_MODE(0)                            externer Takteingang am CLK-Eingang
CNT_SET(0)                             Modus Vier-Flankenauswertung aktivieren
CNT_INPUTMODE(0)                       Eingang CLR/LATCH auf CLR-Modus stellen
CNT_ENABLE(1)                          Zähler 1 freigeben
. . .

EVENT:
CNT_LATCH(1)                           Aktuellen Zählerstand in Latch A übernehmen
CNT_READLATCH(1)                       Latch A auslesen
. . .                                  Auswerten im Programm
```

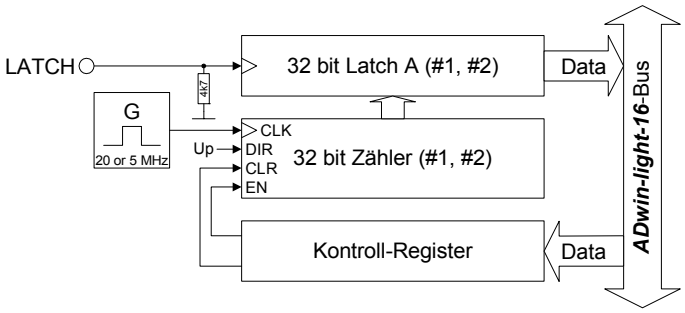
9.2.3 Betriebsart Pulsbreiten- und Periodendauer-Messung

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 Pulsbreite eines Rechtecksignals am Eingang CLR/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.

Periodendauer-Messung

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



Programmierbeispiel

```
INIT:
. . .
CNT_ENABLE(0)
CNT_CLEAR(1)
CNT_MODE(1)
CNT_SET(0)
CNT_SET(1)
CNT_INPUTMODE(1)
CNT_ENABLE(1)
. . .

EVENT:
CNT_READLATCH(1)
. . .
```

Initialisieren

Alle Zähler sperren

Inhalt des Zählers 1 löschen

Modus interner Referenztakt am internen CLK-Eingang des Zählers ...

... mit 20 MHz oder

... mit 5 MHz

Eingang CLR/LATCH auf LATCH-Modus stellen

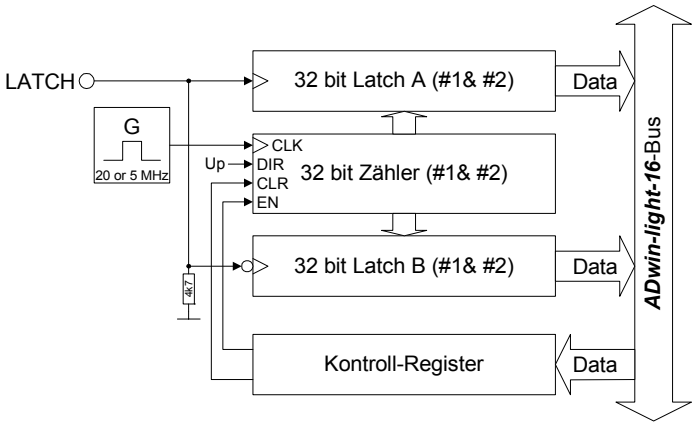
Zähler 1 freigeben

Latch A auslesen

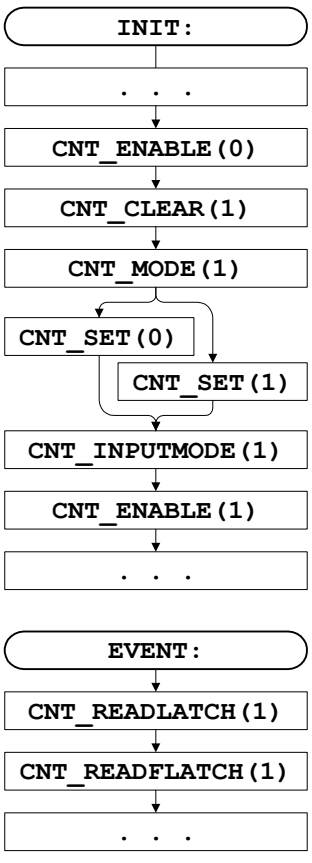
Auswerten im Programm

Messung von Pulsbreite und Pausenzeit

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



Programmierbeispiel



Initialisieren

Alle Zähler sperren

Inhalt des Zählers 1 löschen

Modus interner Referenztakt am internen CLK-Eingang des Zählers ...

... mit 20 MHz oder

... mit 5 MHz

Eingang CLR/LATCH auf LATCH-Modus stellen

Zähler 1 freigeben

Latch A auslesen

Latch B auslesen

Auswerten im Programm

9.3 SSI-Decoder

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

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

Die Anschlüsse des Decoders stehen auf dem Stecker COUNTER (25-polig, Sub-D) zur Verfügung, und zwar auf den Pins 8, 9, 21 und 22 (siehe Abb. 34 – Übersichtsbild L16-EURO-DIO2 mit Pinbelegungen). Auf den Pins 7 und 20 stehen DGND und +5V zur Verfügung.

Eigenschaften einstellen

Folgende Eigenschaften des SSI-Decoders 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. 39 – Listing: Konvertierung von Gray- in Binär-Code

Programmierung

Die Funktionalität des SSI-Decoders wird mit ***ADbasic***-Befehlen komfortabel programmiert:

Bereich	Befehle
Decoder-Modus einstellen	**SSI_MODE**
Daten empfangen	**SSI_READ**
Encoder-Auflösung einstellen	**SSI_SET_BITS**
Encoder-Taktrate einstellen	**SSI_SET_CLOCK**
Auslesen des Encoders starten	**SSI_START**
Lesestatus	**SSI_STATUS**

Die Befehle sind in der Include-Datei <ADWL16.INC> enthalten und werden im Handbuch ***ADbasic*** und der Online-Hilfe erläutert.

10 *ADwin-light-16*-Boot

Diese Option ist nur in Verbindung mit dem Ethernet-Interface möglich, d.h. nur bei *L16-EXT-ENET* oder *L16-EURO-ENET*.

ADwin-light-16-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-light-16***-Boot nach dem Einschalten aus:

– Laden des Betriebssystems

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

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

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 erneutem Einschalten ist die Bootloader-Option wieder aktiv.

Durch Beschreiben des Flash-EEPROMs ohne Prozesse und nur mit der `<ADwin9.btl>`-Datei 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 Bootloaderoption (ADethflash) automatisch kopiert. Die Version der CD sollte 3.00.2735 oder höher sein.

Benutzen Sie für ein ***ADwin-light-16***-System mit einem Ethernet-Interface das Programm `ADethflash`.

Bei Standardinstallation finden Sie das Programm in dem Verzeichniss

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

Hinweise zum Bootloader mit Ethernet-Interface finden Sie in der ***ADwin*** Treiber-Installation.

In Verbindung mit dem Ethernet-Interface mit Bootloader können Sie bis zu 2.000 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 dem Schaltknopf „Info about eeprom support“.

11 Zubehör

Für das ***ADwin-light-16***-System ist folgendes Zubehör lieferbar:

– ***ADbasic***, das Echtzeit-Entwicklungstool für die Programmierung aller ***ADwin***-Systeme.

  ADbasic wird benötigt, um Prozesse für ***ADwin***-Systeme zu entwickeln, die aus einer Entwicklungsumgebung (z. B. C#, Visual Basic oder Matlab) geladen und gesteuert werden.

– ***ADwin-light-16***-pow: externes 12V-Netzteil

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

– Kabel in verschiedenen Längen zur Spannungsversorgung, für USB und für Ethernet.

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

– Kabel-Stecker für eine externe Spannungsversorgung

  Jedem ***ADwin-light-16***-Gerät ist ein Kabelstecker zur Montage an ein externes Spannungsversorgungs-Kabel beigelegt, falls Sie eine alternative Spannungsquelle nutzen möchten.

12 Software

Sie programmieren das ***ADwin-light-16***-System inklusive aller Erweiterungen mit einfachen ***ADbasic***-Befehlen.

Sie finden alle Befehle im Handbuch (Befehlsliste im Anhang) oder in der Online-Hilfe ***ADbasic*** beschrieben.

12.1 Beispiel CAN: Zyklisches Lesen und Senden

Dieses Programm zeigt die Initialisierung des CAN-Controllers im Abschnitt `INIT:` und das zyklische Auslesen und Senden im Abschnitt `EVENT:`.

```
REM Das Programm initialisiert den CAN-Controller,
REM konfiguriert je ein Message-Objekt als Sender
REM und als Empfänger. Das Programm tauscht alle 10 ms
REM Daten zwischen CAN-Controller und Transputer aus.

#INCLUDE ADWL16.inc

DIM ergebnis AS LONG

INIT:
 REM Initialisierung des CAN-Controllers
 INIT_CAN()
 REM Baudrate auf 125 kBit/s setzen
 SET_CAN_BAUDRATE(125000)
 REM Message Objekt 2 zum Lesen konfigurieren
 REM mit 11 Bit und Identifier 385
 EN_RECEIVE(2,385,0)
 REM Message Objekt 3 zum Lesen konfigurieren
 REM mit 11 Bit und Identifier 1
 EN_TRANSMIT(3,1,0)

EVENT:
 REM 1 Datensatz lesen und 1 Datensatz schreiben
 ergebnis = READ_MSG(2)      'Daten einlesen
 'Wenn es neue Daten gibt, werden sie in das Feld
 'CAN_MSG geschrieben


 can_msg[1]=1                'Sende-Daten
 can_msg[2]=2                'werden in das Feld CAN_MSG
 can_msg[3]=3                'geschrieben. Aus diesem werden
 can_msg[4]=4                'beim späteren Senden die Daten
 can_msg[5]=5                'entnommen.
 can_msg[6]=6
 can_msg[7]=7
 can_msg[8]=8
 can_msg[9]=8                '8 Datenbytes

 TRANSMIT(3)                 'Nachricht in Objekt 3 senden
```

12.2 Beispiel CAN: Interruptgesteuertes Lesen

Das nachfolgende Beispiel zeigt die Initialisierung des CAN-Controllers und das nachfolgende interruptgesteuerte Auslesen von neuen Nachrichten:

```
REM Programm initialisiert den CAN-Controller und
REM konfiguriert ein Message Objekt als Empfänger.
REM Das Programm liest interruptgesteuert Nachrichten
REM ein, sobald eine neue eintrifft.

#INCLUDE ADWL16.inc

DIM ergebnis,status,objekt AS LONG

INIT:
 REM Initialisierung des CAN-Controllers
 INIT_CAN()

 REM Baudrate auf 125 kBit/s setzen
 SET_CAN_BAUDRATE(125000)

 REM Message Objekt 1zum Lesen konfigurieren
 REM mit 11 Bit und Identifier 385
 EN_RECEIVE(2,385,0)

 status = GET_CAN_REG(1)     'Status einlesen
 EN_INTERRUPT(1)             'Bei Eingang einer Nachricht in
                             'Message Objekt 1 wird ein
                             'Interrupt ausgelöst

EVENT:
 objekt = GET_CAN_REG(5Fh)   'Interruptregister lesen

 IF (objekt = 2) THEN        'Daraus die Nummer des Message
  objekt = 15                'Objekts ermitteln, in dem die
 ELSE                        'neue Nachricht steht
  objekt = objekt - 2
 ENDIF

 ergebnis = READ_MSG(objekt) 'Neue Daten auslesen

 REM Die Daten stehen im Feld CAN_MSG bereit
```

Anhang

A.1 Technische Daten

Allgemeine Daten / Grenzwerte						
	Symbol	Konditionen	min.	typ.	max.	Einheit
Versorgungs-Spannnung						
Spannung	U_b	*L16-PCI, -EURO*	4,75	5	5,25	V
		L16-EXT, Rev. A	10	12	16	
		L16-EXT, Rev. B	10	12	35	
Versorgungs-Strom mit USB-Schnittstelle						
Betriebsstrom						
L16, *L16-CO1*	I_{idle} bei $U_{b, typ.}$	*L16-PCI, -EURO*	0,75	0,9	1,5	A
		L16-EXT	0,35	0,5	0,7	
L16-DIO1		*L16-PCI, -EURO*	0,85	1,0	1,6	
		L16-EXT	0,55	0,7	1,0	
Einschaltstrombedarf						
L16, *L16-CO1*	$I_{power-on}$ bei $U_{b, typ.}$	*L16-PCI, -EURO*		6		A
		L16-EXT		3		
L16-DIO1		*L16-PCI, -EURO*		7		
		L16-EXT		3		
Versorgungs-Strom mit Ethernet-Schnittstelle						
Betriebsstrom						
L16, *L16-CO1*	I_{idle} bei $U_{b, typ.}$	*L16-PCI, -EURO*	1,0	1,2	1,8	A
		L16-EXT	0,55	0,7	0,9	
L16-DIO1		*L16-PCI, -EURO*	1,15	1,3	1,9	
		L16-EXT	0,55	0,7	1,0	
Einschaltstrombedarf						
L16, *L16-CO1*	$I_{power-on}$ bei $U_{b, typ.}$	*L16-PCI, -EURO*		6,4		A
		L16-EXT		3,3		
L16-DIO1		*L16-PCI, -EURO*		7,5		
		L16-EXT		3,3		
Betrieb						
Temperatur	$T_{Umgebung}$	*L16-PCI, -EURO*	+5		+50	°C
	$T_{Gehäuse}$	*L16-EXT*	+5		+55	
rel. Feuchte	F_{rel}	nicht kondensierend	0		80	%
Lagerung						
Temperatur	T		-20		+70	°C
Abmessungen						
Breite x Höhe x Tiefe	B x H x T	L16-PCI-USB	21,5 x 121,0 x 172,5			mm
		L16-EURO-USB	5 TE (1") x 3 HE (5") x 187			
		L16-EURO-ENET	10 TE (2") x 3 HE (5") x 187			
		L16-EXT-USB	226 x 109 x 44			
		L16-EXT-ENET	226 x 109 x 74			
		+ CO1-Erweiterung	wie Basisversion			
		+ DIO1 / DIO2-Erweiterung	*L16-EURO*: Breite +10 TE *L16-EXT*: Höhe +30			

Allgemeine Daten / Grenzwerte						
	Symbol	Konditionen	min.	typ.	max.	Einheit
Nettogewicht						
Gewicht	m_{Netto}	*L16-PCI*-USB		135		g
		L16-EURO-USB		165		
		L16-EURO-ENET		275		
		L16-EXT-USB		1.100		
		L16-EXT-ENET		1.400		
		+ CO1-Erweiterung		wie Basisversion		
		+ DIO1 / DIO2-Erweiterung		+140		
Steckverbinder						
Sub-D-Verbinder	Metrisches ISO-Gewinde; UNC-Gewinde als Bestelloption erhältlich					
Montage						
Standard	*L16-PCI*: Einbau im PC *L16-EURO*: Einbau im 19"-Gehäuse *L16-EXT*: Tischgehäuse					
optional	Hutschienen- und Wandmontage für *L16-EXT*					

Digitale Ein- / Ausgänge						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
I/O-Leitungen						
Anzahl	DIGIN05:00 DIGOUT05:00	6 Eingänge und 6 Ausgänge (TTL- / 5V-CMOS-Pegel) 2 Eingänge alternativ als Zähler-Eingang verwendbar				
	EVENT	1 ext. Trigger-Eingang (positive TTL-Logik)				
Eingänge						
max. Eingangsspannung		V_{CC} = 5V	-0,5		+5,5	V
Logik-Eingangsspannung	V_{IH} (High)	V_{CC} = 5V	2			
	V_{IL} (Low)	V_{CC} = 5V			0,8	
Logik-Eingangsstrom	I_I	V_{CC} = 5V		±0,1	±1000	nA
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	
Zähler						
Anzahl	2 Inkrementalzähler zur Ereigniszählung.					
Zähler- und Latch-Breite				32		Bit
EVENT-Eingang						
Flankenerkennung, pos.	V_{T+} (Low)	V_{CC} = 5V	1,65	1,9	2,15	V
Flankenerkennung, neg.	V_{T-} (High)	V_{CC} = 5V	0,75	1,0	1,25	
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	8, differentiell über Multiplexer					
Eingangswiderstand	R_i		323,4	330	336,6	kΩ
Spannungsfestigkeit	$U_{in\ max.}$	ON & OFF			±35	V
Multiplexer-Einschwing-zeit	t_{MUX}	2 LSB 16Bit		6,5 *		µs
* Bei einem Innenwiderstand der Spannungsquelle von <10Ω						
ADC 16Bit						
Konvertierungszeit	t_{conv}	Rev. A			10	µs
		Rev. B, optional			2	
Messbereich	U_{in}		-10		+9,999695	V
Differentielle Gleichtaktspannung					±2,0	
Integrale Nichtlinearität	INL			±1	±3	LSB
Differentielle Nichtlinearität	DNL			±0,25	±0,5	
Offset	Drift			±2		ppm/°C
	Fehler	abgleichbar				
Gain	Drift			±20		ppm/°C
	Fehler	abgleichbar				
DAC 16 Bit						
Anzahl	2					
Ausgangsspannung	U_{out}		-10		+9,999695	V
Einschwingzeit	t_{settle}	2V-Sprung			3	µs
		FSR* (20V)			10	
Zulässiger Strom					±5	mA
Integrale Nichtlinearität	INL				±2	LSB
Differentielle Nichtlinea-rität	DNL				±1	
Offset	Fehler	abgleichbar				
Gain	Fehler	abgleichbar				
* FSR = 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		kByte
		für Daten		128		
Externer Speicher	SDRAM	Rev. A		8		MByte
		Rev. B		16		

CO1-Erweiterung						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
Zähler						
Anzahl und Funktion	1 Vor-/Rückwärtszähler zur Ereigniszählung mit Takt / Richtung oder Vierflanken-Auswertung. Die Inkremental-Zähler der Grundversion werden ersetzt.					
Zählereingänge	2 Eingänge (A, B), alternativ als Digital-Eingänge verwendbar					
Zähler- und Latch-Breite				32		Bit

DIO1-Erweiterung						
Parameter	Symbol	Konditionen	min.	typ.	max.	Einheit
Referenz-Quarzoszillator						
Referenzfrequenz	f_{ref}			20		MHz
Vorteiler durch 4	f_{ref} / 4			5		
Genauigkeit und Drift					100	ppm
Zähler						
Anzahl und Funktion	2 Vor-/Rückwärtszähler zur Messung von Tastverhältnis, Impuls- und Periodendauer sowie zur Ereigniszählung mit Takt / Richtung oder Vierflanken-Auswertung. Die Inkremental-Zähler der Grundversion werden ersetzt.					
Zählereingänge	Je Zähler 3 differentielle Eingänge (A/CLK, B/DIR, CLR/LATCH); Zähler programmierbar mit differentiellen oder single-ended Eingängen.					
Zähler- und Latch-Breite				32		Bit
Zählfrequenz	f_{CLK}	Eingang CLK		20		MHz
		Eingang A/B		5		
Digitale Ein- / Ausgänge						
Anzahl	DIO31:00	32 (in Gruppen zu 8 als Ein- oder Ausgang programmierbar)				
	EVENT	ext. Trigger-Eingang (pos. TTL-Logik)				
als Eingänge *						
max. Eingangsspannung		V_{CC} = 5V	-0,5		+5,5	V
Logik-Eingangsspannung	V_{IH} (High)	V_{CC} = 5V	2			
	V_{IL} (Low)	V_{CC} = 5V			0,8	
Logik-Eingangsstrom	I_I	V_{CC} = 5V		±0,1	±1000	nA
als 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}	je DIO-Gruppe (8) über V_{CC} / GND			±70	
* siehe auch Datenblatt SN74 HCT 245 von Texas Instruments						

A.2 Hardware-Adressen - Gesamtübersicht

Adresse [HEX]	Funktion	Bit Nr.								Kommentar	Register verfügbar im Modul		
		31...16	15...6	5	4	3	2	1	0		L16	L16+ CO1	L16+ DIO1
20 40 00 00	Multiplexer auf Eingangskanal setzen (ADC 01...ADC 15)	-	-	-	-	-	n	n	n	„nnn“ binär = 0...7 dezimal, gewählter Kanal = nnn*2 + 1	x	x	x
20 40 00 10	Konvertierung starten: ADC #1	-	-	-	-	-	-	-	s	s = 0 : Konvertierung starten s = 1 : kein Einfluss	x	x	x
	Konvertierung starten: alle DAC synchron	-	-	-	-	-	s	-	-				
20 40 00 20	Konvertierungs-Status (EOC) ADC #1	-	-	-	-	-	-	-	e	e = 0 : Konvertierung beendet e = 1 : Konvertierung läuft	x	x	x
20 40 00 30	Register auslesen: ADC #1	-	x	x	x	x	x	x	x	x : Ergebnis der Konvertierung	x	x	x
20 40 00 50	Register nur beschreiben: DAC #1	-	x	x	x	x	x	x	x	x : zu wandelnder Digitalwert	x	x	x
20 40 00 60	Register nur beschreiben: DAC #2	-	x	x	x	x	x	x	x				
20 40 00 B0	Eingangs-Register DIGIN-05:00	-	-	x	x	x	x	x	x	x : eingelesener Digitalwert	x	x	x
20 40 00 C0	Ausgangs-Register DIGOUT-05:00	-	-	x	x	x	x	x	x	x : auszugebender Digitalwert	x	x	x
20 40 00 C4	DIGOUT Bits setzen	-	-	x	x	x	x	x	x	x = 0 : kein Einfluss x = 1 : Bit setzen	x	x	x
20 40 00 C8	DIGOUT Bits löschen	-	-	x	x	x	x	x	x	x = 0 : kein Einfluss x = 1 : Bit löschen	x	x	x
20 40 01 00	Register auslesen und Konvertierung starten: ADC #1	-	x	x	x	x	x	x	x	x : zu wandelnder Digitalwert	x	x	x
20 40 02 00	Register beschreiben und sofort Konvertierung starten: DAC #1	-	x	x	x	x	x	x	x	x : zu wandelnder Digitalwert	x	x	x
20 40 02 04	Inhalt Latch A, Zähler #1	x	x	x	x	x	x	x	x	x : Inhalt des Latch-Registers	x	x	x
20 40 02 08	Inhalt Latch B, Zähler #1 (nur DIO1)	x	x	x	x	x	x	x	x	x : Inhalt des Latch-Registers	-	-	x
20 40 02 10	Register beschreiben und sofort Konvertierung starten: DAC #2	-	x	x	x	x	x	x	x	x : zu wandelnder Digitalwert	x	x	x
20 40 02 14	Inhalt Latch A, Zähler #2	x	x	x	x	x	x	x	x	x : Inhalt des Latch-Registers	x	-	x
20 40 02 18	Inhalt Latch B, Zähler #2	x	x	x	x	x	x	x	x	x : Inhalt des Latch-Registers	-	-	x
20 40 03 00	Zähler freigeben / sperren: `CNT_ENABLE()`	-	-	-	-	-	-	x	x	x = 0 : Zähler sperren x = 1 : Zähler freigeben	x	nur Bit 0	x
20 40 03 10	Zähler löschen: `CNT_CLEAR()` *	-	-	-	-	-	-	x	x	x = 0 : kein Einfluss x = 1 : Zähler löschen	x	nur Bit 0	x
20 40 03 20	Zähler latchen: `CNT_LATCH()` *	-	-	-	-	-	-	x	x	x = 0 : keine Einfluss x = 1 : Zähler latchen	x	nur Bit 0	x
20 40 03 30	Zählereingang CLR oder LATCH	-	-	-	-	-	-	x	x	x = 0 : CLR-Eingang x = 1 : LATCH-Eingang	-	-	x
20 40 03 40	Impuls-/Ereigniszähler- oder Pulsbreiten-/Periodendauermessung	-	-	-	-	-	-	x	x	x = 0 :externer Takteingang x = 1 : interner Referenztakt (20MHz / 5MHz)	-	-	x
20 40 03 50	4-Flankenauswertung / CLK+DIR oder 20MHz / 5MHz Referenztakt	-	-	-	-	-	-	x	x	`CNT_MODE`=0: x=0: 4-Flanken-Auswertung; x=1: CLK+DIR `CNT_MODE`=1: x=0: 20MHz; x=1: 5MHz	-	-	x
20 40 04 54	Bits DIO-15:00	-	x	x	x	x	x	x	x	x = 0: Ausgang löschen x = 1: Ausgang setzen	-	-	x
20 40 04 64	Bits DIO-31:16	-	x	x	x	x	x	x	x	x = 0: Ausgang löschen x = 1: Ausgang setzen	-	-	x
20 40 04 74	Bits setzen DIO-15:00 **	-	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang setzen	-	-	x
20 40 04 84	Bits setzen DIO-31:16 **	-	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang setzen	-	-	x
20 40 04 94	Bits löschen DIO-15:00 **	-	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang löschen	-	-	x
20 40 04 A4	Bits löschen DIO-31:16 **	-	x	x	x	x	x	x	x	x = 0: kein Einfluss x = 1: Ausgang löschen	-	-	x
20 40 04 6C	DIOs konfigurieren CONF_DIO() Bit 0: DIO 07:00; Bit 1: DIO 15:08 Bit 2: DIO 23:16; Bit 3: DIO 31:24	-	-	-	-	x	x	x	x	x = 0: Gruppe als Eingang x = 1: Gruppe als Ausgang	-	-	x

* Das Register wird nach der Durchführung automatisch wieder zurückgesetzt.
** Funktion bei Eingängen ohne Einfluss

A.3 Hardware-Revisionen

Auf dem Gerät befindet sich ein Aufkleber mit der Revisionsbezeichnung. Die Unterschiede der Revisionsstände sind nachfolgend dargestellt:

Revision	Erst-ausgabe	Änderung zur Vorgänger-Version
A	1998	Erst-Version.
B1	März 2006	Überarbeitetes Layout, dennoch kompatibel zu Rev. A. Externer Speicher auf 16MB erweitert. Schnellere A/D-Wandlung über den Befehl `L16_MODE` verfügbar. Zusätzliche Ablaufsteuerung zur Wandlung analoger Eingänge. Zusätzlicher SSI-Schnittstelle für DIO1-Erweiterung, neue Erweiterungen DIO2 und DIO3.
B2	Aug. 2006	Neue Schnittstelle für LS-Bus.

A.4 Abbildungsverzeichnis