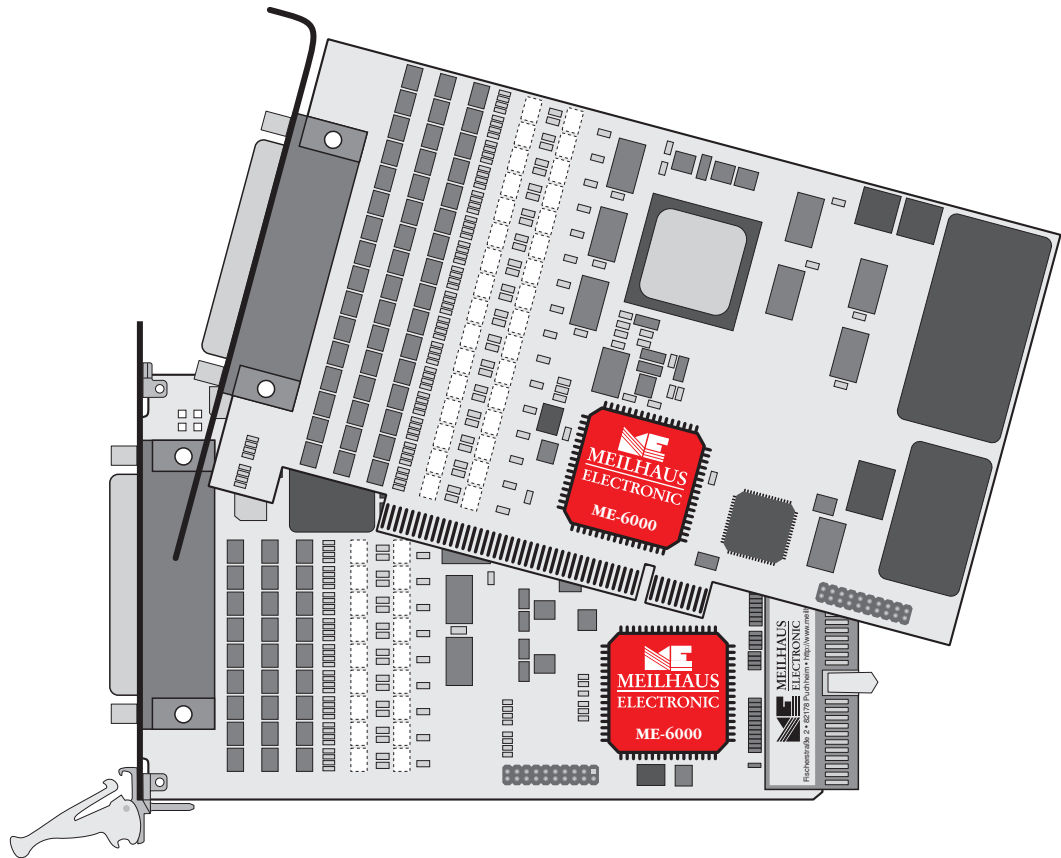


Meilhaus Electronic Handbuch

ME-6000 Serie 2.1D (ME-6000/6100/6200/6300)



**16 Bit D/A-Wandlerkarte mit bis zu 16 Kanälen
und galvanischer Trennung; optional: Insel-Kanäle**

Impressum

Handbuch ME-6000/6100/6200/6300

Revision 2.1D

Ausgabedatum: 30. August 2005

Meilhaus Electronic GmbH
Fischerstraße 2
D-82178 Puchheim bei München
Germany
<http://www.meilhaus.de>

© Copyright 2005 Meilhaus Electronic GmbH

Alle Rechte vorbehalten. Kein Teil dieses Handbuches darf in irgendeiner Form (Fotokopie, Druck, Mikrofilm oder in einem anderen Verfahren) ohne ausdrückliche schriftliche Genehmigung der Meilhaus Electronic GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Wichtiger Hinweis:

Alle in diesem Handbuch enthaltenen Informationen wurden mit größter Sorgfalt und nach bestem Wissen zusammengestellt. Dennoch sind Fehler nicht ganz auszuschließen.

Aus diesem Grund sieht sich die Firma Meilhaus Electronic GmbH dazu veranlaßt, darauf hinzuweisen, daß sie weder eine Garantie (abgesehen von den im Garantieschein vereinbarten Garantieansprüchen) noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen kann.

Für die Mitteilung eventueller Fehler sind wir jederzeit dankbar.

Delphi/Pascal ist ein Warenzeichen von Borland International, INC.

Visual C++ und VisualBASIC sind Warenzeichen von Microsoft.

VEE Pro und VEE OneLab sind Warenzeichen von Agilent Technologies.

ME-VEC und ME-FoXX sind Warenzeichen von Meilhaus Electronic.

Weitere der im Text erwähnten Firmen- und Produktnamen sind eingetragene Warenzeichen der jeweiligen Firmen.



Inhalt

1	Einführung	5
1.1	Lieferumfang	5
1.2	Leistungsmerkmale.....	6
1.3	Systemanforderungen.....	8
1.4	Softwareunterstützung.....	8
2	Installation	11
2.1	Testprogramm.....	11
3	Hardware	13
3.1	Blockschaltbild.....	13
3.2	Generelle Hinweise	15
3.3	D/A-Teil	16
3.3.1	Galvanische Trennung	16
3.3.2	Insel-Kanäle	17
3.3.3	Option „High Current“	18
3.3.4	Externer Trigger D/A-Teil	18
3.4	Digital-I/O-Teil	20
4	Programmierung	21
4.1	D/A-Teil	21
4.1.1	Kennlinien für Windows-Treiber.....	22
4.1.2	Kennlinien für Linux-Treiber	23
4.1.3	Betriebsart „AOSingle“	24
4.1.4	Betriebsart „AOSimultaneous“	25
4.1.5	Timergesteuerte „AO-Betriebsarten“	26
4.1.5.1	Konfiguration des D/A-Teils	27
4.1.5.2	Vorbereitung der Software	29
4.1.5.2.1	Betriebsart „AOContinuous“	29
4.1.5.2.2	Betriebsart „AOWraparound“	33
4.1.5.3	Starten der Ausgabe.....	36
4.1.5.4	Stoppen der Ausgabe	36
4.2	Digital-I/O-Teil	37

4.3	Treiberkonzept	38
4.3.1	Visual C++	39
4.3.2	Visual Basic	39
4.3.3	Delphi	40
4.3.4	Agilent VEE	40
4.3.5	LabVIEW	41
4.3.6	Python	42
5	Funktionsreferenz.....	45
5.1	Allgemeine Hinweise	45
5.2	Nomenklatur.....	46
5.3	Beschreibung der API-Funktionen.....	47
5.3.1	Fehler-Behandlung.....	49
5.3.2	Allgemeine Funktionen.....	53
5.3.3	Analoge Ausgabe	61
5.3.4	Digitale Ein-/Ausgabe	84
Anhang	91
A	Spezifikationen.....	91
B	Anschlußbelegungen.....	94
B1	78pol. Sub-D-Buchse (ST1)	95
B2	Zusatzstecker (ST2)	97
C	Zubehör.....	98
D	Technische Fragen	100
D1	Fax-Hotline	100
D2	Serviceadresse	100
D3	Treiber-Update	100
E	Konstanten-Definitionen	101
F	Index	103

1 Einführung

Sehr geehrte Kundin, sehr geehrter Kunde,

Mit dem Kauf einer PC-Einsteckkarte von Meilhaus Electronic haben Sie sich für ein technologisch hochwertiges Produkt entschieden, das unser Haus in einwandfreiem Zustand verlassen hat.

Überprüfen Sie trotzdem die Vollständigkeit und den Zustand Ihrer Lieferung. Sollten irgendwelche Mängel auftreten, bitten wir Sie, uns sofort in Kenntnis zu setzen.

Bevor Sie die Karte in Ihren Rechner einbauen, lesen Sie bitte aufmerksam diese Bedienungsanleitung, insbesondere Kapitel 2 zur Installation durch.

Die Beschreibungen in diesem Handbuch gelten gleichermaßen für PCI- und CompactPCI-Varianten der ME-6000-Serie, sofern nicht ausdrücklich unterschieden wird.

1.1 Lieferumfang

Wir sind selbstverständlich bemüht, Ihnen ein vollständiges Produktpaket auszuliefern. Um aber in jedem Fall sicherzustellen, daß Ihre Lieferung komplett ist, können Sie anhand nachfolgender Liste die Vollständigkeit Ihres Paketes überprüfen.

Ihr Paket sollte folgende Teile enthalten:

- D/A-Wandlerkarte der ME-6000 Serie für PCI- bzw. CompactPCI-Bus
- Handbuch im PDF-Format auf CD-ROM (optional in gedruckter Form)
- Treiber-Software auf CD-ROM
- Spezial-Anschlußkabel von 78poligem Sub-D-Stecker auf 16 einzeln geschirmte Leitungen mit offenem Ende
- Zusatz-Slotblech ME-AK-D25F/S (cPCI) für PCI- bzw. CompactPCI-Slot
- 25poliger Sub-D-Gegenstecker

1.2 Leistungsmerkmale

ME-6000/6100 PCI/CompactPCI

Modell	16 Bit D/A-Kanäle		Digital-I/Os
	insgesamt	...mit FIFO	
ME-6000/4	4	–	16
ME-6000/8	8	–	16
ME-6000/16	16	–	16
ME-6100/4	4	4	16
ME-6100/8	8	4	16
ME-6100/16	16	4	16
Optional für alle Modelle:	„p“-Option: Alle D/A-Kanäle voneinander entkoppelt (sog. „Insel“-Kanäle)		

Tabelle 1: Modell-Übersicht ME-6000/6100

ME-6200/6300 CompactPCI

Modell	16 Bit D/A-Kanäle		Digital-I/Os
	insgesamt	...mit FIFO	
ME-6200/5	4 + 1 „U-Plus“	–	16
ME-6200/9	8 + 1 „U-Plus“	–	16
ME-6300/5	4 + 1 „U-Plus“	4	16
ME-6300/9	8 + 1 „U-Plus“	4	16
Optional für alle Modelle:	„p“-Option: Alle D/A-Kanäle voneinander entkoppelt (sog. „Insel“-Kanäle)		

Tabelle 2: Modell-Übersicht ME-6200/6300

Je nach Modell verfügen die Karten der ME-6000-Serie über 4, 8 oder 16 D/A-Kanäle zur bipolaren Spannungsausgabe im Bereich ± 10 V sowie über 16 Digital-I/Os. Jeder D/A-Kanal hat einen separaten, hochgenauen High Speed 16 Bit D/A-Wandler.

Der D/A-Teil ist als Ganzes vom Rest der Karte **galvanisch getrennt**. Als Option können zusätzlich die einzelnen Kanäle voneinander entkoppelt werden („p“-Option mit sog. **„Insel“-Kanälen**). Damit können Genauigkeiten besser $\pm 1\%$ erreicht werden.

Unabhängig von der Gesamtzahl der Kanäle können bei den Modellen **ME-6100/6300** die ersten 4 Kanäle (0...3) auch zur zeitgesteuerten Ausgabe von Kurvenformen verwendet werden. Jeder dieser 4 Kanäle verfügt über ein eigenes 8 kByte großes Werte-FIFO. Damit sind Sample-Raten von bis zu **500 kHz pro Kanal** möglich ohne den Host-Rechner zu belasten. Sie können wählen zwischen der Betriebsart „Continuous“ um kontinuierlich Werte auszugeben und der Betriebsart „Wraparound“ für periodische Signale.

Alle ME-6000/6100 ab Hardware-Version 2.6 sowie alle Compact-PCI-Karten verfügen über **16 Digital-I/Os**. Diese sind als 2 bidirektionale 8 Bit breite TTL-Ports organisiert. Der Anschluß erfolgt über die 25polige Sub-D-Buchse des mitgelieferten Zusatz-Slotbleches.

Alle Modelle vom Typ **ME-6200/6300** verfügen über einen speziellen D/A-Kanal (Uout_8), der Ausgangsspannungen im Bereich 0...+50 V bei max. 20 mA liefert. Dieser Kanal wird auch mit „U-Plus“-Kanal bezeichnet. Die Versorgung der galvanisch getrennten Ausgangsstufe dieses Kanals erfolgt standardmäßig über einen DC/DC-Wandler auf der Karte. Die Einschwingzeit dieses Kanals beträgt 25 μs (bei Vollausschlag)

Auf Wunsch sind OEM-Versionen mit anderen Ausgangsspannungsbereichen bis ± 28 V möglich. Auch eine externe Versorgung der Ausgangsstufe im Bereich $\pm 12... \pm 28$ V ist möglich. Bei Interesse an OEM-Versionen setzen Sie sich bitte mit unserem Vertrieb in Verbindung, Tel.: 089/8901660.

1.3 Systemanforderungen

Die ME-6000 Serie setzt einen PC mit Intel® Pentium® Prozessor oder kompatiblen Rechner voraus, der über einen freien Standard-PCI- bzw. CompactPCI-Steckplatz verfügt. Die ME-6000/6100 wird von aktuellen Windows-Versionen sowie Linux und VxWorks unterstützt. Die ME-6200/6300 wird z. Zt. unter Linux unterstützt.

1.4 Softwareunterstützung

Den aktuellen Stand des Software-Lieferumfangs entnehmen Sie bitte den entsprechenden README-Dateien.

Systemtreiber Für alle gängigen Betriebssysteme (siehe README-Dateien)

ME-Software-Developer-Kit (ME-SDK):

Beispiele für alle gängigen Programmiersprachen, sowie Tools und Testprogramme

Graphische Programmierumgebungen:

Meilhaus VEE-Treibersystem für
HP VEE, HP VEE Lab,
Agilent VEE Pro und
Agilent VEE OneLab

LabVIEW™ Treiber

Im Lieferumfang enthalten ist das sog. „Messlabor“. Dies ist ein virtuelles Instrumenten-Panel, das u. a. einen „**Virtuellen Funktionsgenerator**“ für die ME-6100 beinhaltet. Voraussetzung ist die Agilent VEE Runtime Software, die bei Bedarf automatisch mitinstalliert wird.

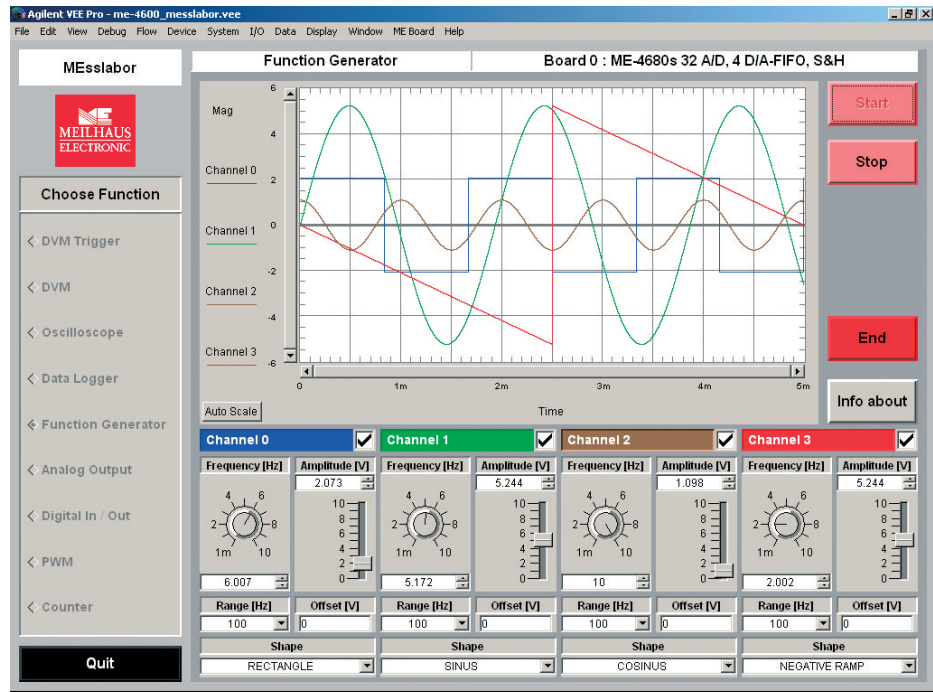


Abb. 1: Virtueller Funktionsgenerator

2 Installation

Bitte lesen Sie **vor Einbau der Karte** das Handbuch Ihres Rechners bzgl. der Installation von zusätzlichen Hardwarekomponenten und das Kapitel „Hardware-Installation“ in diesem Handbuch (sofern zutreffend, z. B. für ISA-Karten).

- **Installation unter Windows (Plug&Play)**

Sie finden eine Anleitung in HTML-Form auf CD-ROM. Bitte **vor Installation lesen** und bei Bedarf ausdrucken!

Grundsätzlich gilt folgende Vorgehensweise:

Falls Sie die Treiber-Software in gepackter Form erhalten haben, entpacken Sie bitte **vor Einbau der Karte** die Software in ein Verzeichnis auf Ihrem Rechner (z. B. C:\Meilhaus).

Bauen Sie die Karte in Ihren Rechner ein und installieren Sie anschließend die Treiber-Software. Diese Reihenfolge ist wichtig, um die Plug&Play-Funktionalität unter Windows 95*/98/Me/2000/XP zu gewährleisten. Für Windows 95* und NT 4.0 gilt dies analog, beachten Sie jedoch die etwas andere Vorgehensweise bei der Treiberinstallation.

**Sofern Windows-Version von der betreffenden Karte unterstützt wird (siehe Readme-Dateien)*

- **Installation unter Linux**

Beachten Sie die Installationshinweise, die in der Archiv-Datei des jeweiligen Treibers enthalten sind.

2.1 Testprogramm

Zum Test der Einsteckkarte verwenden Sie bitte das entsprechende Testprogramm im ME Software Developer Kit (ME-SDK). Nach Entpacken des ME-SDK finden Sie das Testprogramm in entsprechenden Unterverzeichnissen von C:\MEILHAUS (Default). Voraussetzung: Systemtreiber korrekt installiert.

3 Hardware

3.1 Blockschaltbild

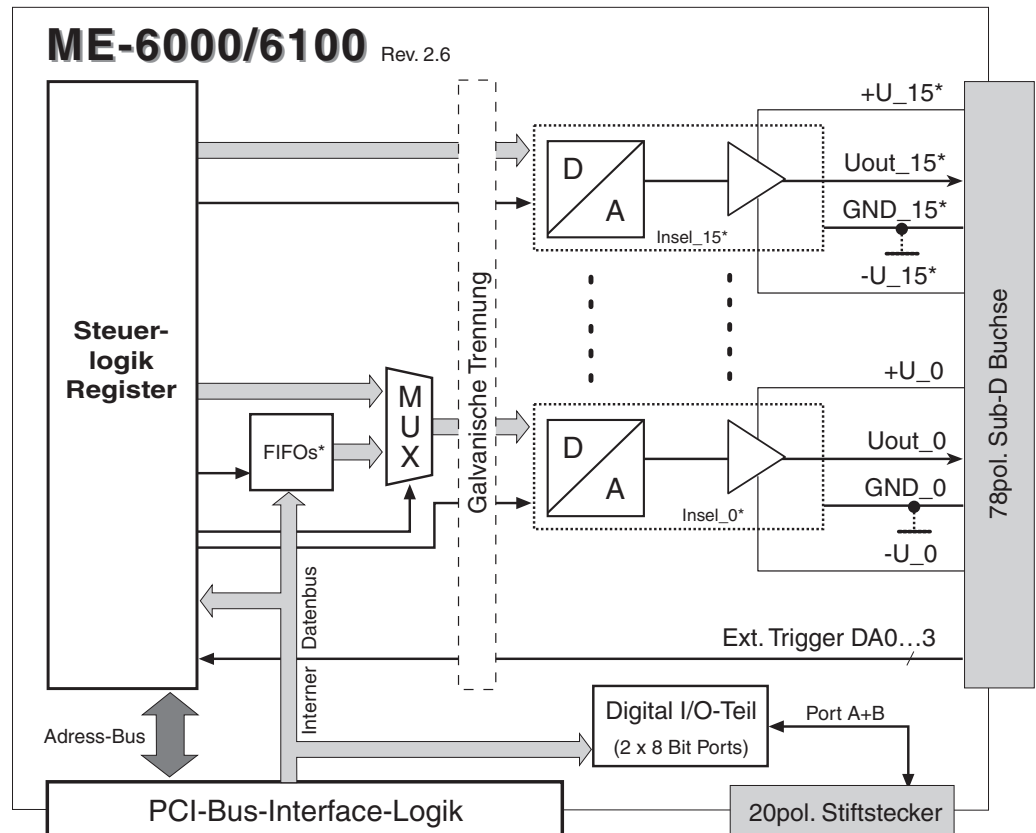


Abb. 2: Blockschaltbild ME-6000/ME-6100

- ME-6000/4:** 4 D/A-Kanäle $\pm 10V$ (Uout_0...3), 16 DIOs
- ME-6000/8:** 8 D/A-Kanäle $\pm 10V$ (Uout_0...7), 16 DIOs
- ME-6000/16:** 16 D/A-Kanäle $\pm 10V$ (Uout_0...15), 16 DIOs
- ME-6100/4:** 4 D/A-Kanäle $\pm 10V$ (Uout_0...3) mit FIFO, 16 DIOs
- ME-6100/8:** 8 D/A-Kanäle $\pm 10V$ (Uout_0...7), davon 4 mit FIFO (Uout_0...3), 16 DIOs
- ME-6100/16:** 16 D/A-Kanäle $\pm 10V$ (Uout_0...15), davon 4 mit FIFO (Uout_0...3), 16 DIOs

* Je nach Modell sind nicht alle Funktionsgruppen verfügbar.

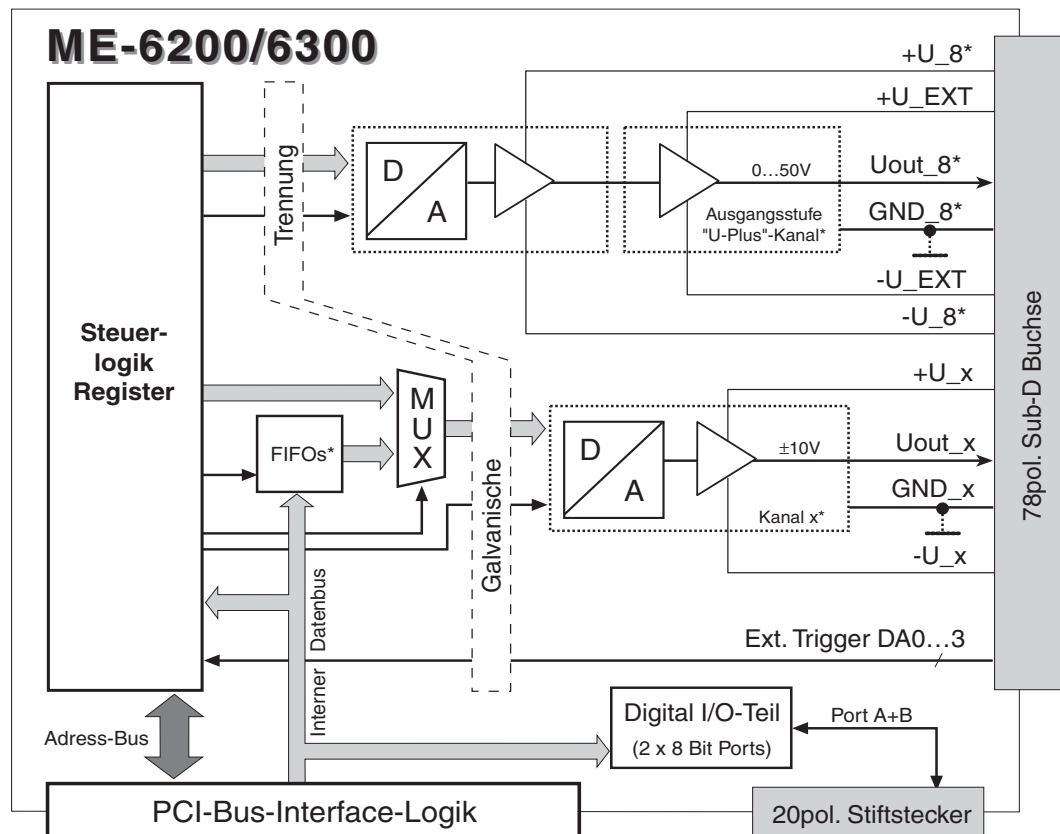


Abb. 3: Blockschaltbild ME-6200/ME-6300

- ME-6200/5:** 4 D/A-Kanäle $\pm 10V$ ($x=0\dots 3$), 1 „U-Plus“-Kanal (Uout_8), 16 DIOs
- ME-6200/9:** 8 D/A-Kanäle $\pm 10V$ ($x=0\dots 7$), 1 „U-Plus“-Kanal (Uout_8), 16 DIOs
- ME-6300/5:** 4 D/A-Kanäle $\pm 10V$ ($x=0\dots 3$) mit FIFO, 1 „U-Plus“-Kanal (Uout_8), 16 DIOs
- ME-6300/9:** 8 D/A-Kanäle $\pm 10V$ ($x=0\dots 7$) davon 4 mit FIFO, 1 „U-Plus“-Kanal (Uout_8), 16 DIOs

* Je nach Modell sind nicht alle Funktionsgruppen verfügbar.

3.2 Generelle Hinweise

Achtung:

Nach Einschalten des Rechners geben die Standard-D/A-Kanäle -10 V aus. Nach dem Starten des Treibers gehen die Ausgänge nach 0 V. Um ein definiertes Einschaltverhalten zu erreichen starten Sie zuerst den Host-Rechner. Schalten Sie Ihre ext. Beschaltung erst nach Start des Treibers ein.

Der „U-Plus“-Kanal der ME-6200/6300 gibt sofort nach Einschalten der Versorgungsspannung 0 V aus.

Wichtiger Hinweis!

Beachten Sie, daß stets ein Bezug von der externen Beschaltung zur Masse der einzelnen Kanäle hergestellt werden muß. Die Pins +U_0...15 und -U_0...15 werden benötigt, falls eine der Optionen „Insel-Kanäle“ oder „High Current“ (HC) verwendet werden und dürfen ansonsten **nicht beschaltet** werden!

Stellen Sie sicher, daß bei Berührung der Karte und beim Stecken des Anschlußkabels keine statische Entladung über die Steckkarte stattfinden kann.

Sämtliche Steckverbindungen der Karte sollten grundsätzlich nur im spannungslosen Zustand hergestellt bzw. gelöst werden. Achten Sie auf sicheren Sitz des Anschlußkabels. Es muß vollständig auf die Sub-D Buchse aufgesteckt und mit den beiden Schrauben fixiert werden. Nur so ist eine einwandfreie Funktion der Karte gewährleistet ist!

Die Belegung der 78poligen Sub-D Buchse finden Sie im Anhang (siehe „Anschlußbelegungen“ auf Seite 94).

3.3 D/A-Teil

3.3.1 Galvanische Trennung

Alle D/A-Kanäle der Karte sind über Optokoppler von der PC-Masse galvanisch getrennt. D. h. GND_0...15 sind miteinander verbunden und beziehen sich auf eine gemeinsame Masse (ISO_GND). Bei der ME-6100 sind auch die externen Triggereingänge optoentkoppelt.

Der Ausgangsstrom I_{\max} pro Kanal hängt von der Anzahl der bestückten bzw. genutzten Kanäle ab (siehe Tabelle).

Kanäle	I_{\max}	Kanäle	I_{\max}
4	$\pm 15\text{mA}$	12	$\pm 10\text{mA}$
8	$\pm 15\text{mA}$	16	$\pm 3\text{mA}$

Tabelle 3: Max. Ausgangsstrom

Beachten Sie, daß $\pm 15\text{ mA}$ pro Kanal nicht überschritten werden dürfen! Die Pins für die ext. $\pm 15\text{V}$ Spannungsversorgung ($+U_x$ und $-U_x$) werden intern getrieben und dürfen nicht extern beschaltet werden!

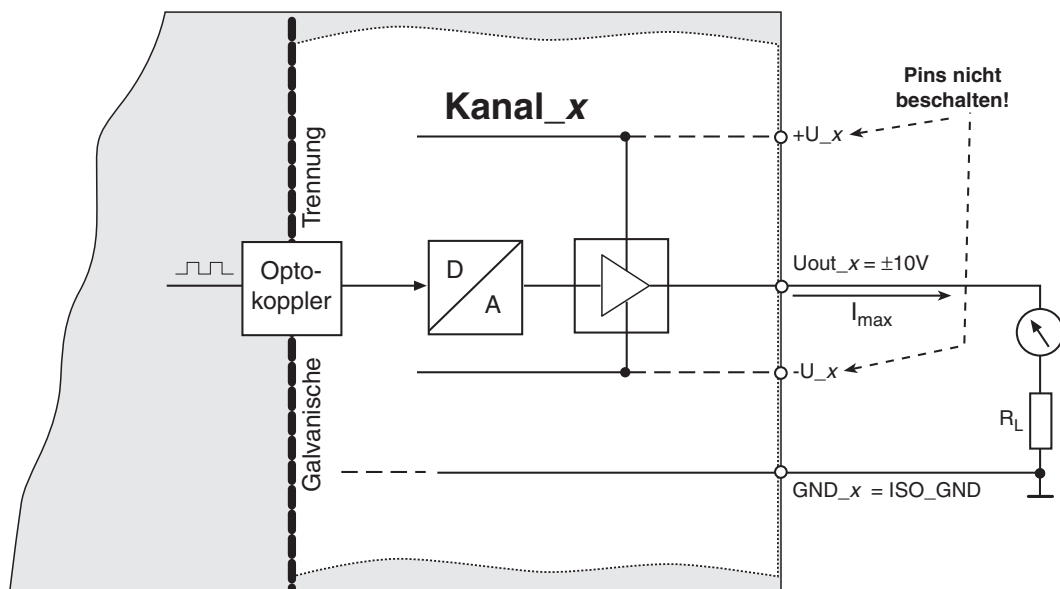


Abb. 4: Galvanische Trennung aller Kanäle

3.3.2 Insel-Kanäle

Mit der „p“-Option („Insel-Kanäle“) haben alle D/A-Kanäle voneinander unabhängige Masse-Potentiale und Versorgungs-Pins. D. h. Sie müssen die Bezugsmasse des jeweiligen Kanals (GND_0...15) einzeln mit der entsprechenden Masse Ihres externen Schaltungsteils verbinden. Außerdem benötigen Sie für jeden Insel-Kanal eine eigenständige, symmetrische Versorgungsspannung von $\pm 15\text{ V}$ ($\pm 22\text{ mA}$ pro Kanal für $I_{\text{max}} = \pm 15\text{ mA}$). Bei Verwendung einer hochwertigen, rauscharmen Spannungsquelle können Sie damit hervorragende **Genauigkeiten** besser $\pm 1\%$ erreichen. Bei der ME-6100/6300 sind auch die ext. Triggereingänge in die „Inseln“ miteinbezogen. Sie können pro D/A-Kanal **max. $\pm 15\text{ mA}$** treiben.

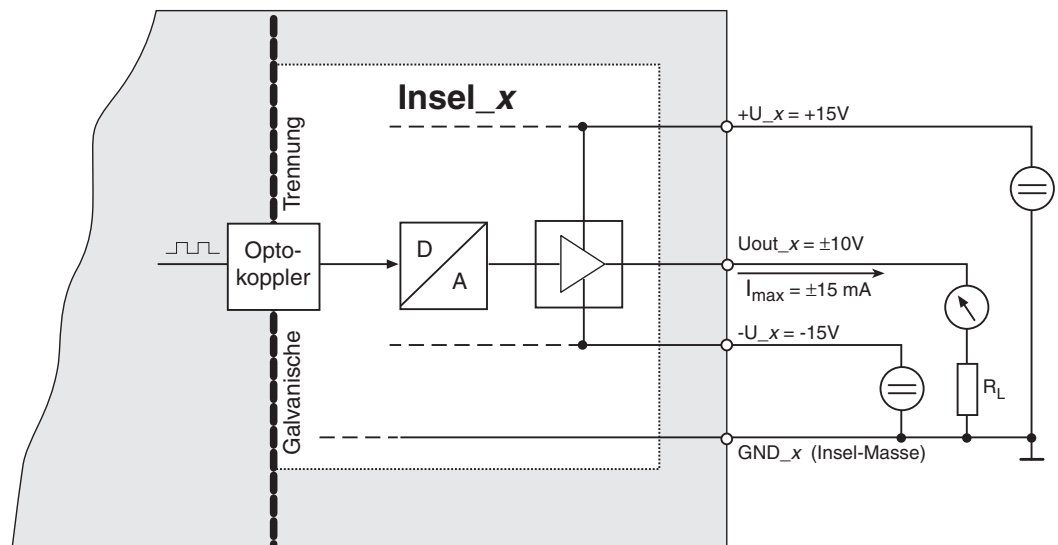


Abb. 5: Insel-Kanäle

3.3.3 Option „High Current“

Die Option „**High Current**“ (HC) können Sie mit den Kartenvarianten ohne „Insel-Kanäle“ kombinieren. Dadurch haben Sie die Möglichkeit den Ausgangsstrom pro Kanal auf $I_{\max} = \pm 15\text{mA}$ zu erhöhen. Dies erfordert eine externe, rauscharme Spannungsversorgung von $\pm 15\text{V}$ ($\pm 22\text{mA}$ pro Kanal für $I_{\max} = \pm 15\text{mA}$) und eine Änderung der Hardware – bitte kontaktieren Sie dazu unsere Service-Abteilung (siehe Seite 100).

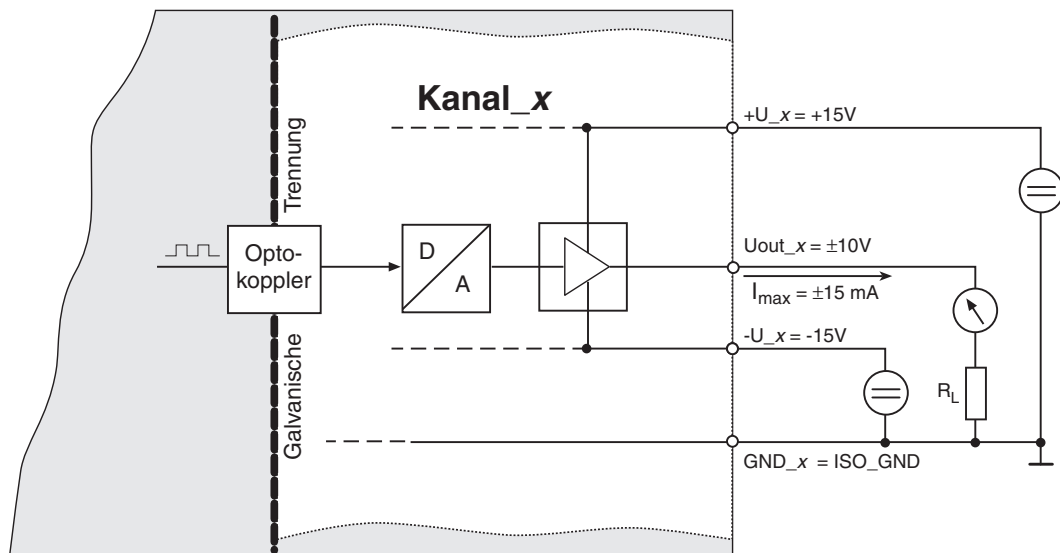


Abb. 6: Schematische Beschaltung der Ausgänge in der Option „High Current“ (mit galvanischer Trennung)

3.3.4 Externer Trigger D/A-Teil

Die D/A-Kanäle 0...3 können durch ein externes Triggersignal (DA_TRIG_x) gestartet werden. Je nach gewählter Flanken-Option („RISING“, „FALLING“ oder „BOTH“) wird die Wandlung durch eine entsprechende Flanke gestartet. Die Option „BOTH“ bedeutet entweder steigende **oder** fallende Flanke.



Abb. 7: Triggerflanken

Achten Sie bei der Beschaltung der ext. Triggereingänge darauf, daß die Spannungspegel eingehalten werden (siehe Spezifikationen auf Seite 92) und ein Masse-Bezug (GND_x) hergestellt werden muß.

Die optoentkoppelten Triggereingänge arbeiten mit einem High-Pegel von +5V. Im Low-Pegel muß ein Strom I_F von min. 7,5 mA gegen Masse (GND_x) fließen.

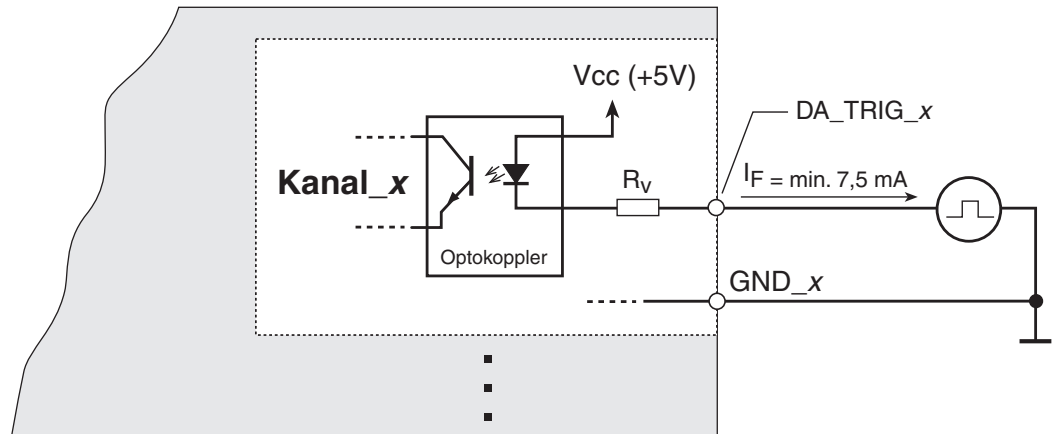


Abb. 8: Beschaltung der Triggereingänge

3.4 Digital-I/O-Teil

Alle ME-6000/6100 ab Hardware-Version 2.6 sowie alle Compact-PCI-Karten verfügen über zwei 8 Bit breite TTL-Ports. Jeder Port kann unabhängig als Ein- oder Ausgang konfiguriert werden. Nach dem Einschalten der Versorgung sind alle Ports auf Eingang geschaltet. Zur Programmierung lesen Sie bitte Kap. 4.2 "Digital-I/O-Teil" auf Seite 37.

Die beiden Ports A und B können über den 20pol. Stiftstecker ST2 auf das mitgelieferte Zusatz-Slotblech (ME-AK-D25F/S) mit einer 25pol. Sub-D-Buchse geführt werden.

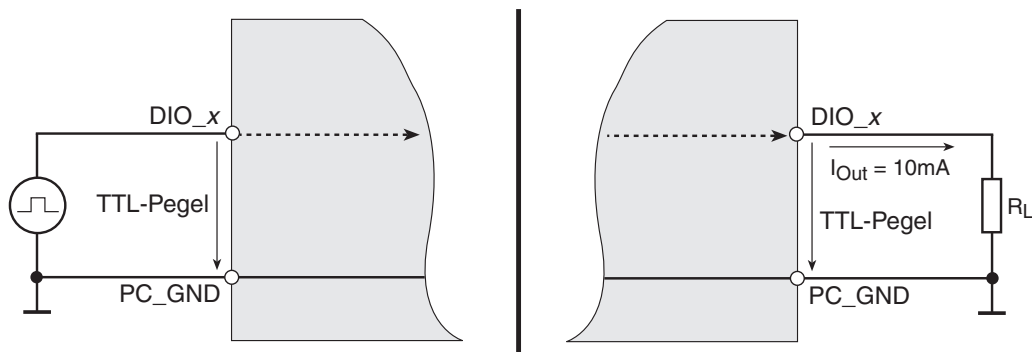


Abb. 9: Beschaltung der digitalen Ein-/Ausgänge

Achten Sie bei der Beschaltung der Ein- und Ausgänge darauf, daß der TTL-Pegel eingehalten werden (siehe Spezifikationen auf Seite 92) und ein Bezug zur PC-Masse (PC_GND) hergestellt werden muß. Der max. Ausgangsstrom beträgt $I_{\text{Out}} = I_{\text{OL}} = I_{\text{OH}} = 10 \text{ mA}$.

4 Programmierung

Hinweis: Für die Programmierung unter Windows steht ein gemeinsamer Systemtreiber für alle Karten der ME-4600 und ME-6000 Serie zur Verfügung. Die Funktionen beginnen mit dem Präfix „me4000“ und sind im Kapitel 5 "Funktionsreferenz" ab Seite 45 beschrieben. Unter Linux kommt für die ME-6000 Serie (incl. ME-6200/6300) eine eigenständige API zum Einsatz, die im ME-6000 Handbuch Rev. 1.1 dokumentiert ist und im Internet unter www.meilhaus.de/download zum Download bereitsteht.

Beachten Sie, daß für die unterschiedlichen Treiber unter Windows und Linux auch unterschiedliche Kennlinien für die Umrechnung der Digital- in Spannungswerte gelten. Unter Windows können Sie hierfür auch die Funktion `...AOVoltToDigit` verwenden (nicht für „U-Plus“-Kanal).

4.1 D/A-Teil

Betriebsart	Anwendung	Trigger	Timing
AOSingle (siehe Seite 24)	1 Wert auf 1 Kanal ausgeben	Software-Start, ext. digital	–
AOSimultaneous (siehe Seite 25)	Mehrere Kanäle simultan ausgeben	Software-Start, ext. digital	–
AOContinuous (siehe Seite 29)	Timergesteuerte Ausgabe kontinuierlich sich ändernder Werte	Software-Start, ext. digital	D/A-Timer (...AOConfig)
AOWraparound (siehe Seite 33)	Timergesteuerte Ausgabe sich wiederholender Werte	Software-Start, ext. digital	D/A-Timer (...AOConfig)

Tabelle 4: D/A-Betriebsarten

4.1.1 Kennlinien für Windows-Treiber

Diese Kennlinie gilt für bipolare Ausgabe bei 16 Bit Auflösung und Verwendung der „ME-4000-API“ unter Windows. Für alle Kanäle mit Ausnahme des „U-Plus“-Kanals kann für „FS“* 10 V eingesetzt werden.

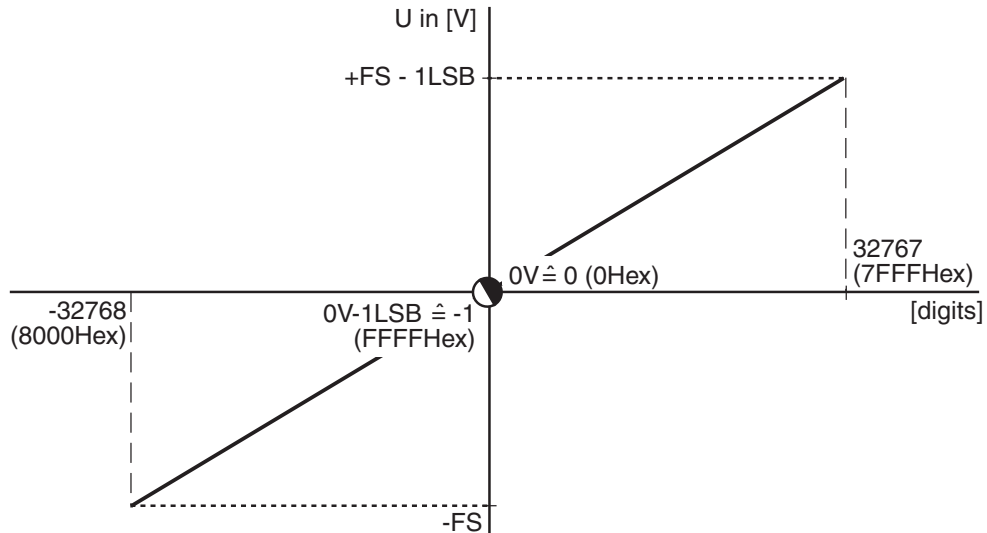


Abb. 10: Bipolare Kennlinie (Windows)

Diese Kennlinie gilt für unipolare Ausgabe bei 16 Bit Auflösung und Verwendung der „ME-4000-API“ unter Windows. Für den „U-Plus“-Kanal der ME-6200/6300 gilt: bei standardmäßiger Bestückung der Ausgangsstufe (0...50 V) kann für „FS“* 50 V eingesetzt werden.

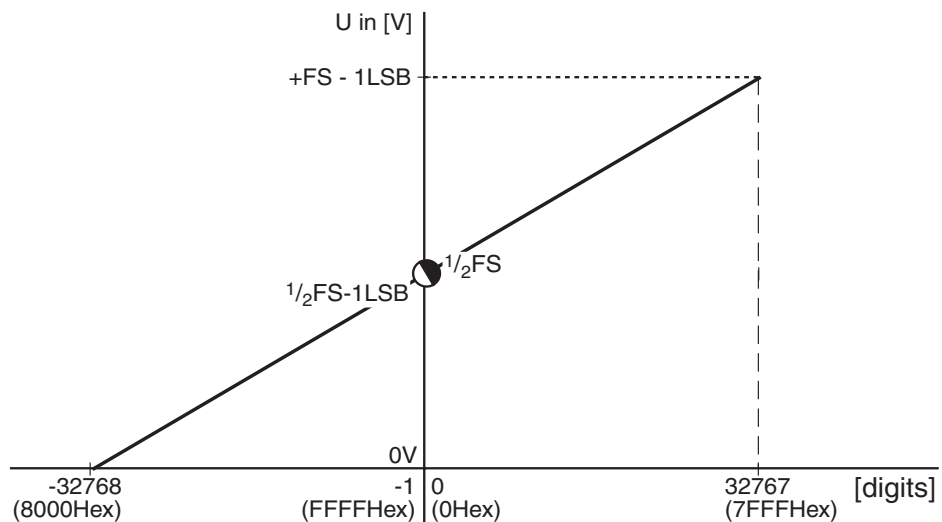


Abb. 11: Unipolare Kennlinie (Windows)

*„FS“ steht für „Full Scale“ (Vollausschlag) im jeweiligen Messbereich; „LSB“ steht für den Spannungswert, der sich für das niederwertigste Bit bei 16 Bit Auflösung ergibt.

4.1.2 Kennlinien für Linux-Treiber

Diese Kennlinie gilt für bipolare Ausgabe bei 16 Bit Auflösung und Verwendung des Linux-Treibers. Für alle Kanäle mit Ausnahme des „U-Plus“-Kanals kann für „FS“* 10 V eingesetzt werden.

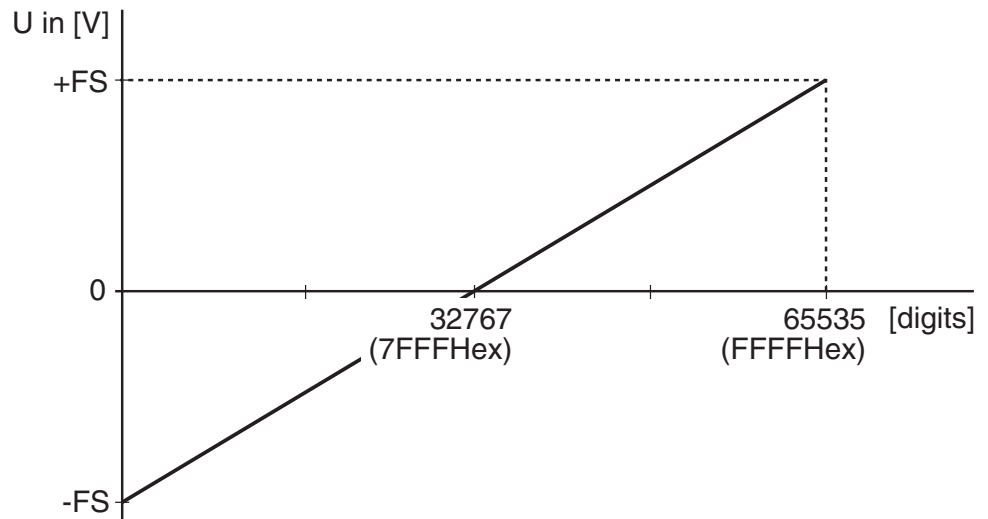


Abb. 12: Bipolare Kennlinie (Linux)

Diese Kennlinie gilt für unipolare Ausgabe bei 16 Bit Auflösung und Verwendung des Linux-Treibers. Für den „U-Plus“-Kanal der ME-6200/6300 gilt: bei standardmäßiger Bestückung der Ausgangsstufe (0...50 V) kann für „FS“* 50 V eingesetzt werden.

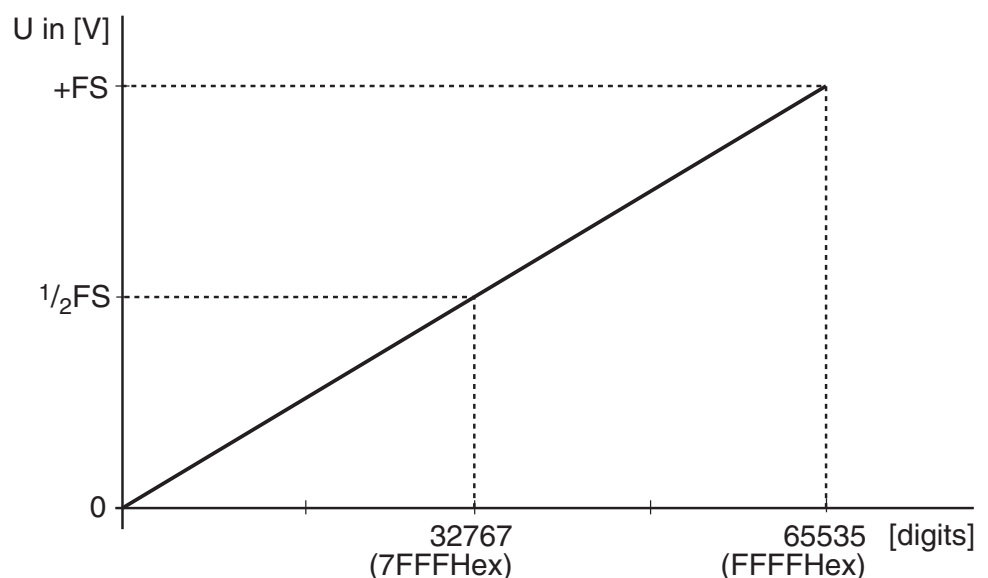


Abb. 13: Unipolare Kennlinie (Linux)

*„FS“ steht für „Full Scale“ (Vollausschlag) im jeweiligen Messbereich; „LSB“ steht für den Spannungswert, der sich für das niederwertigste Bit bei 16 Bit Auflösung ergibt.

4.1.3 Betriebsart „AOSingle“

ME-6000/6200	ME-6100/6300
✓	✓

Der auszugebende Spannungswert wird mit der Funktion ...*AOSingle* in den D/A-Wandler des gewünschten Kanals geladen. Je nach Triggermodus wird der Wert sofort ausgegeben (Software-Start) oder durch eine entsprechende Flanke am zugehörigen Triggereingang (nur für D/A-Kanal 0...3). Es ist keine weitere Konfiguration nötig.

Das folgende Diagramm soll die grundsätzliche Vorgehensweise erläutern (siehe auch Funktionsbeschreibung auf Seite 69 sowie Beispielprogramme im ME-SDK):

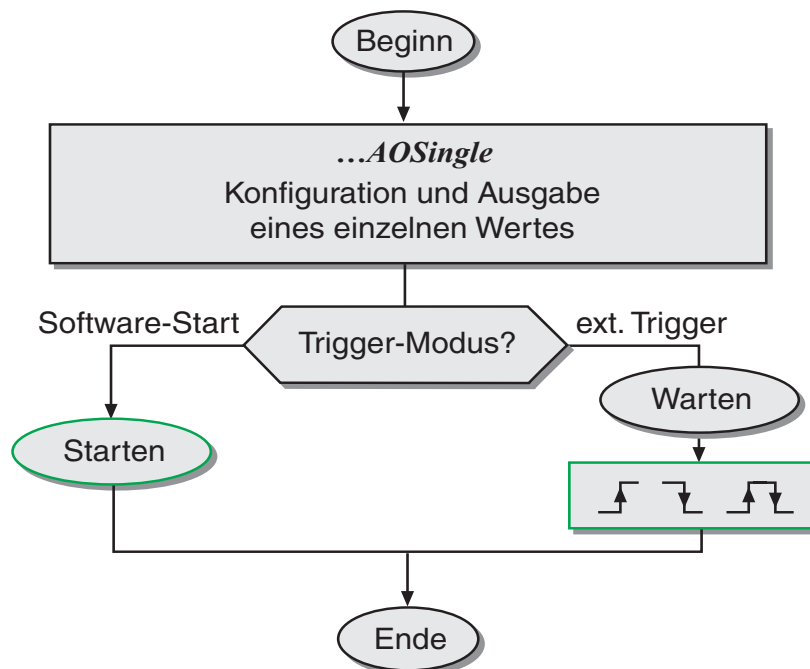


Abb. 14: Programmierung „AOSingle“

4.1.4 Betriebsart „AOSimultaneous“

ME-6000/6200	ME-6100/6300
✓	✓

Durch Aufruf der Funktion `...AOSingleSimultaneous` werden die auszugebenden Spannungswerte zunächst für jeden Kanal, der in die simultane Ausgabe einbezogen werden soll in den entsprechenden D/A-Wandler geladen. Je nach gewähltem Triggermodus werden die Kanäle entweder sofort ausgegeben (Software-Start) oder für die Ausgabe durch einen externen Triggerimpuls „scharfgeschaltet“. Sie können wählen welcher Triggereingang (bzw. Eingänge) der simultanen Kanäle die Ausgabe starten soll. Bei Verwendung des externen Triggers können nur die D/A-Kanäle 0...3 verwendet werden.

Das folgende Diagramm soll die grundsätzliche Vorgehensweise erläutern (siehe auch Funktionsbeschreibung auf Seite 71 sowie Beispielprogramme im ME-SDK):

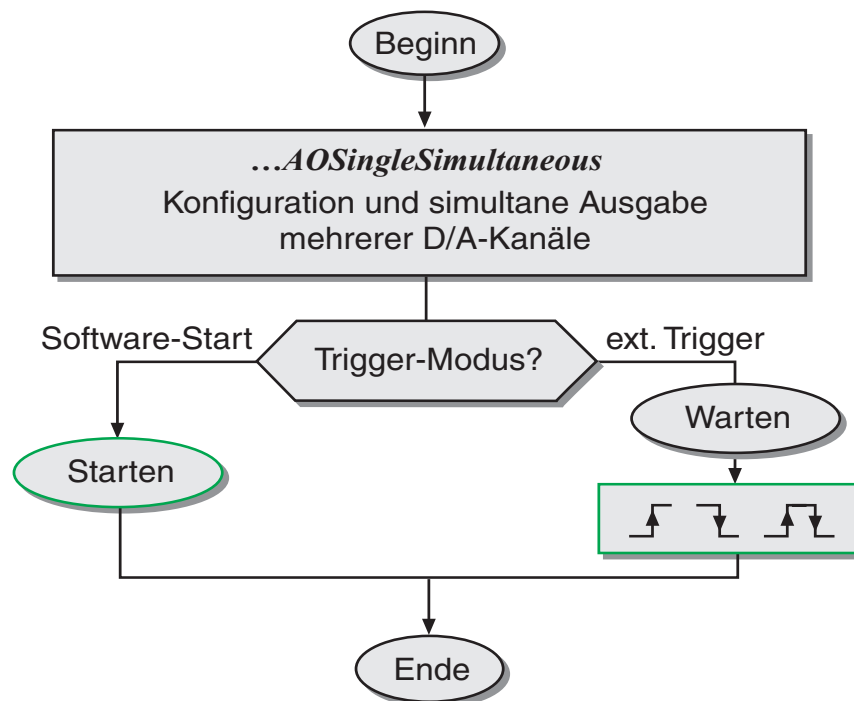


Abb. 15: Programmierung „AOSimultaneous“

4.1.5 Timergesteuerte „AO-Betriebsarten“

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Die Programmierung der timergesteuerten Ausgabe läuft im Wesentlichen in 3 Schritten ab:

1. Konfiguration der Hardware je Kanal (0...3) mit ...*AOConfig* (siehe Kap. 4.1.5.1)
2. Vorbereitung der Software mit ...*AOContinuous* oder ...*AOWraparound* (siehe Kap. 4.1.5.2)
3. Start der Ausgabe mit ...*AOStart* bzw. ...*AOStartSynchronous* (siehe Kap. 4.1.5.3)

Vor Beginn der Ausgabe müssen Sie sich entscheiden, ob Sie während der laufenden Ausgabe neue Werte kontinuierlich nachladen möchten (*AOContinuous*), oder stets die gleichen Werte periodisch ausgegeben (*AOWraparound*) werden sollen. Nach Konfiguration von Hardware und Software kann die Ausgabe per Software oder durch ein externes Triggersignal gestartet werden.

Abbildung 16 soll die grundsätzliche Vorgehensweise in den Betriebsarten „**AOContinuous**“ und „**AOWraparound**“ veranschaulichen. Zur weiteren Vorgehensweise bzgl. Konfiguration, Daten-Handling und Ausführungsmodi beachten Sie bitte die folgenden Kapitel.

4.1.5.1 Konfiguration des D/A-Teils

In einem ersten Schritt wird die Hardware für jeden Kanal mit der Funktion ...*AOConfig* konfiguriert (Funktionsbeschreibung siehe Seite 63).

- Wählen Sie den gewünschten D/A-Kanal.
- Als Zeitgeber dient ein programmierbarer 32 Bit Zähler, der einen 33 MHz Takt als Zeitbasis verwendet. Daraus ergibt sich eine Periodendauer von 30,30ns, die als kleinste Zeiteinheit definiert wird und im Folgenden „1 Tick“ genannt wird. Zur einfachen Umrechnung von Frequenz bzw. Periodendauer in Ticks können Sie die Funktionen ...*FrequencyToTicks* oder ...*TimeToTicks* verwenden. Es sind Sample-Raten im Bereich 500 kS/s bis $\approx 0,5$ Samples/Minute einstellbar.
- Als Triggermodi stehen zur Verfügung:
 - Software-Start: Die Ausgabe wird für den spezifizierten Kanal mit der Funktion ...*AOStart* gestartet bzw. nach Eintreffen des ersten geeigneten Trigger-Ereignisses.
 - Synchroner Software-Start: Synchroner Start mehrerer Kanäle nach entsprechender Konfiguration mit ...*AOConfig*. Die Ausgabe startet unmittelbar nach Aufruf der Funktion ...*AOStartSynchronous*.
 - Start durch den ersten externen Trigger-Impuls (steigende, fallende oder beliebige Flanke) am entsprechenden Eingang DA_TRIG_x.
 - Synchroner Start durch den ersten externen Trigger-Impuls (steigende, fallende oder beliebige Flanke) an einem oder mehreren Triggereingängen (DA_TRIG_x).

Das folgende Diagramm soll die Vorgehensweise in den Betriebsarten „**AOContinuous**“ und „**AOWraparound**“ veranschaulichen.

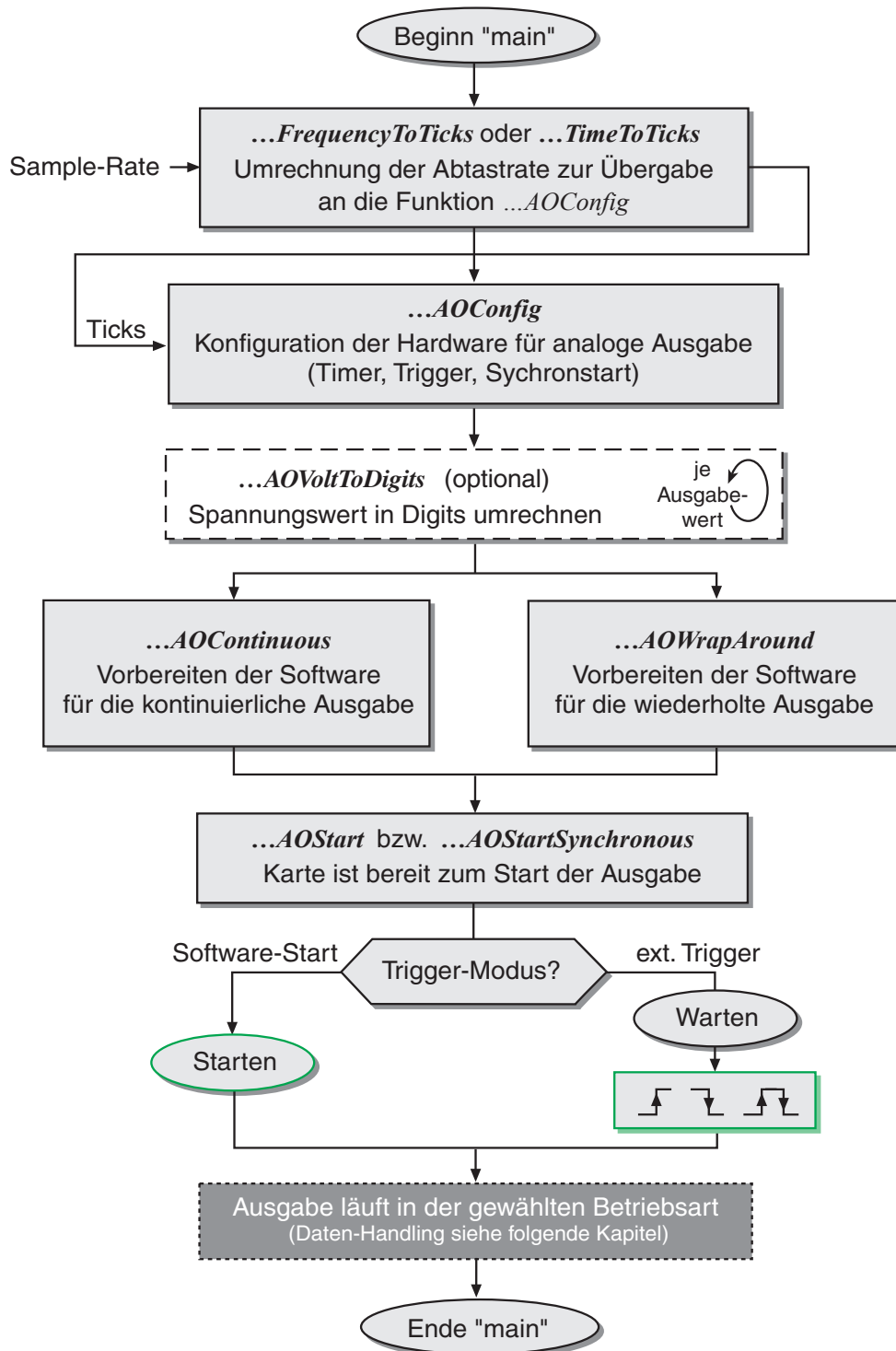


Abb. 16: Vorgehensweise Programmierung AO-Teil

4.1.5.2 Vorbereitung der Software

In einem zweiten Schritt wird die Software für die analoge Ausgabe vorbereitet. Je nach dem ob Sie neue Werte kontinuierlich nachladen (siehe Kap. 4.1.5.2.1) oder die gleichen Werte periodisch ausgegeben möchten (siehe Kap. 4.1.5.2.2), stehen unterschiedliche Funktionalitäten zur Verfügung, die in den folgenden Kapiteln beschrieben sind.

Intern arbeitet der Treiber mit einem Ringpuffer, in den die auszugebenden Werte geschrieben werden müssen.

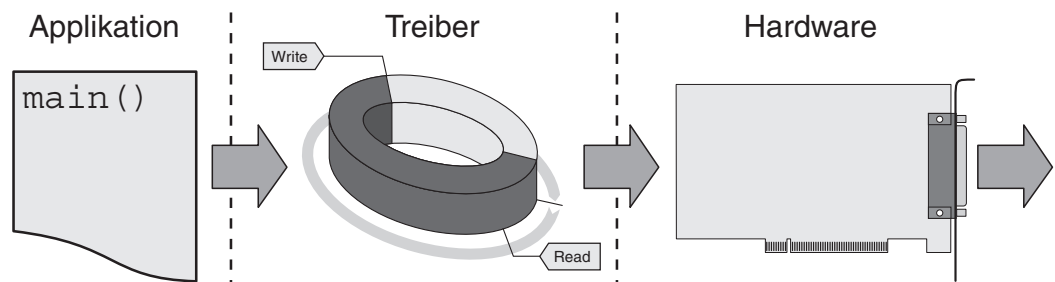


Abb. 17: Ringpuffer AO-Betrieb

4.1.5.2.1 Betriebsart „AOContinuous“

In dieser Betriebsart können Sie auf den D/A-Kanälen 0...3 beliebige, voneinander unabhängige, analoge Signale ausgeben, die sich während der Ausgabe auch ändern bzw. neu berechnet werden können (im Gegensatz zur Betriebsart „AOWrap-around“). Allokieren sie für jeden Kanal, der verwendet werden soll, einen Datenpuffer definierter Größe mit den **ersten** auszugebenden Werten. Vor Beginn der Ausgabe wird das erste Datenpaket in den Ringpuffer geschrieben. Der D/A-Timer gibt ein festes Zeitraster (Sample-Rate) für die Wandlung der einzelnen Werte vor und muß mit der Funktion `...AOConfig` vor Start der Ausgabe konfiguriert werden.

Das Nachladen des Ringpuffers erfolgt mit der Funktion `...AOAppendNewValues`. Im Ausführungsmodus „NON_BLOCKING“ wird nur die Anzahl an Werten „nachgefüllt“, die aktuell im Ringpuffer Platz finden. Der Ausführungsmodus „BLOCKING“ ist hier nicht zu empfehlen, da der interne Thread blockiert bis alle Werte nachgeladen werden konnten. Die Diagramme auf den folgenden Seiten beschreiben den Programm-Fluss unter folgenden Bedingungen:

- a. Die Ausgabe erfolgt im Hintergrund (asynchron) mit der Funktion ...*AOAppendNewValues* im Rahmen einer benutzerdefinierten Callback-Funktion.
- b. Die Ausgabe erfolgt im Hintergrund (asynchron) mit der Funktion ...*AOAppendNewValues*.

Beachten Sie auch die Beispiele im ME-SDK und die Funktionsbeschreibung auf Seite 65.

Zu a: Daten-Handling in der Betriebsart „**AOContinuous**“ mit der Funktion `...AOAppendNewValues` im Rahmen einer benutzerdefinierten Callback-Funktion:

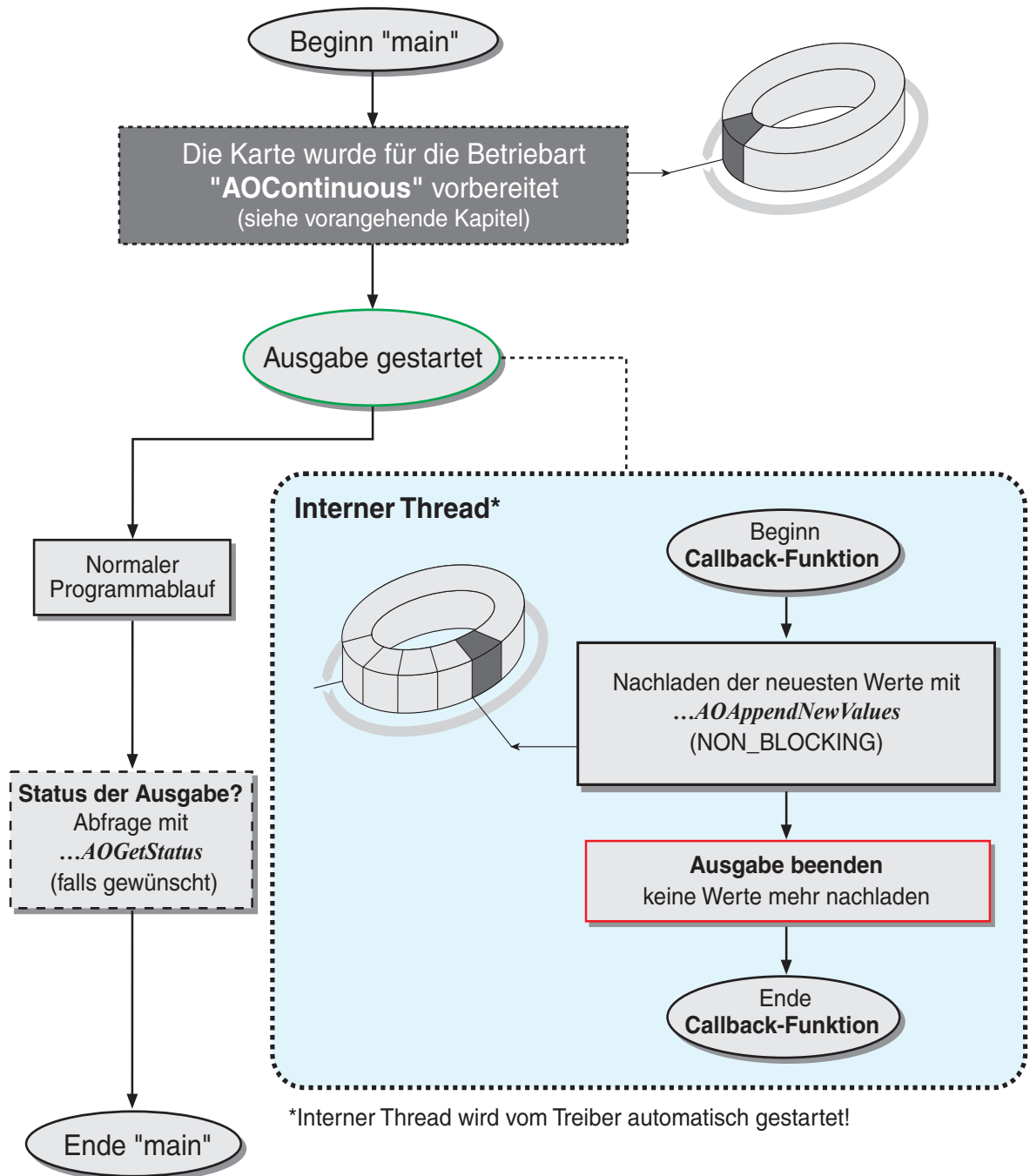


Abb. 18: Programmierung „AOContinuous“ mit Callback-Funktion

Zu b: Daten-Handling in der Betriebsart „**AOContinuous**“ mit der Funktion ...*AOAppendNewValues*:

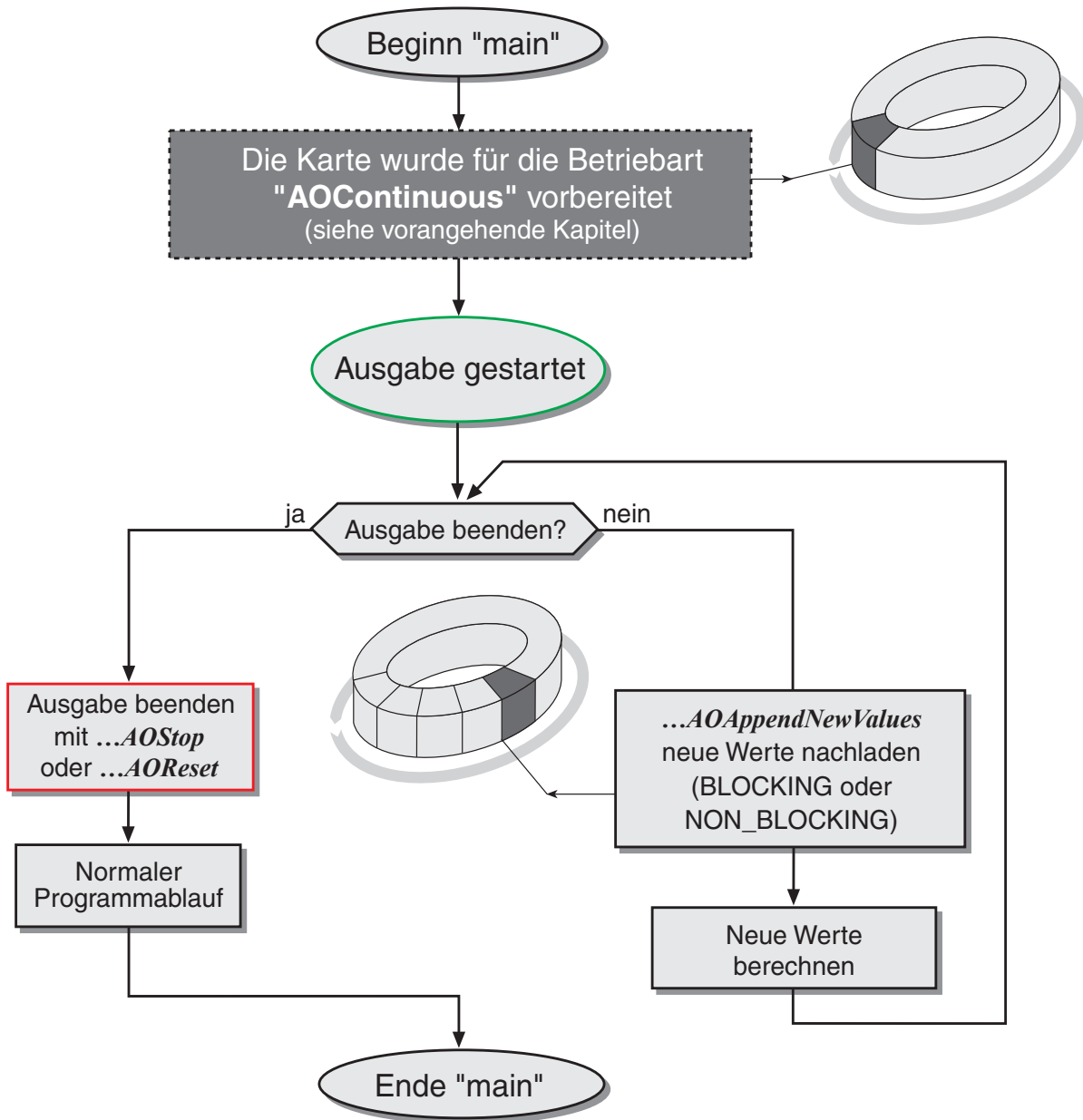


Abb. 19: Programmierung „AOContinuous“ ohne Callback-Funktion

4.1.5.2 Betriebsart „AOWraparound“

In der Betriebsart „AOWraparound“ können Sie auf den D/A-Kanälen 0...3 periodische, voneinander unabhängige, analoge Signale ausgeben. Allokieren Sie für jeden Kanal, der verwendet werden soll, einen Datenpuffer definierter Größe mit den Werten, die periodisch ausgegeben werden sollen. Die Daten werden beim Start **einmalig** in den Datenpuffer geschrieben. Der D/A-Timer gibt ein festes Zeitraster (Sample-Rate) für die Wandlung der einzelnen Werte vor. Die Konfiguration erfolgt mit der Funktion ...*AOConfig*.

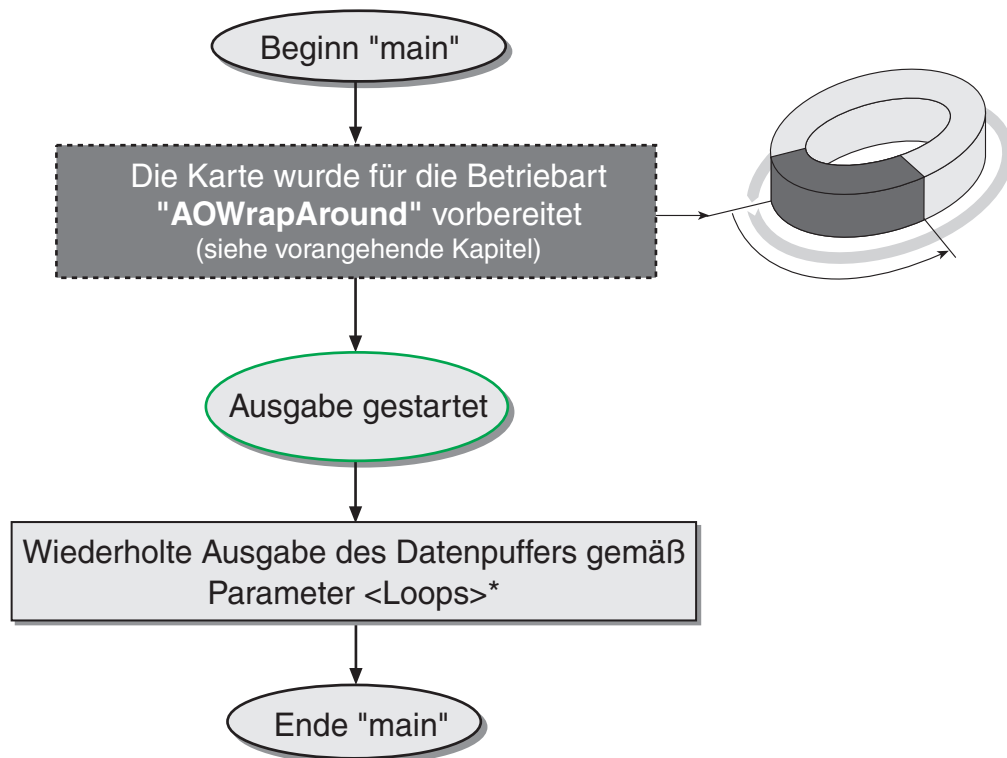
Hinweis: Sofern die Größe des Datenpuffers 8192 Werte nicht übersteigt und die Ausgabe „unendlich“ erfolgt, läuft die Ausgabe auf Firmware-Ebene, d. h. der Host-Rechner wird nicht belastet!

Die Diagramme auf den folgenden Seiten beschreiben den Programm-Fluss unter folgenden Bedingungen:

- a. Der Programmfluß wird blockiert bis die Ausgabe beendet ist (Ausführungsmodus BLOCKING). Der Parameter <Loops> gibt an, wie oft der Datenpuffer ausgegeben werden soll. Der Wert „unendlich“ ist hier nicht möglich.
- b. Die Ausgabe erfolgt im Hintergrund (Ausführungsmodus ASYNCHRONOUS). Der Parameter <Loops> gibt an, wie oft der Datenpuffer ausgegeben werden soll. Im Gegensatz zum BLOCKING-Mode kann hier der Datenpuffer auch „unendlich“ oft ausgegeben werden. Der aufrufende Thread wird dadurch nicht blockiert.

Beachten Sie auch die Beispiele im ME-SDK und die Funktionsbeschreibung auf Seite 81.

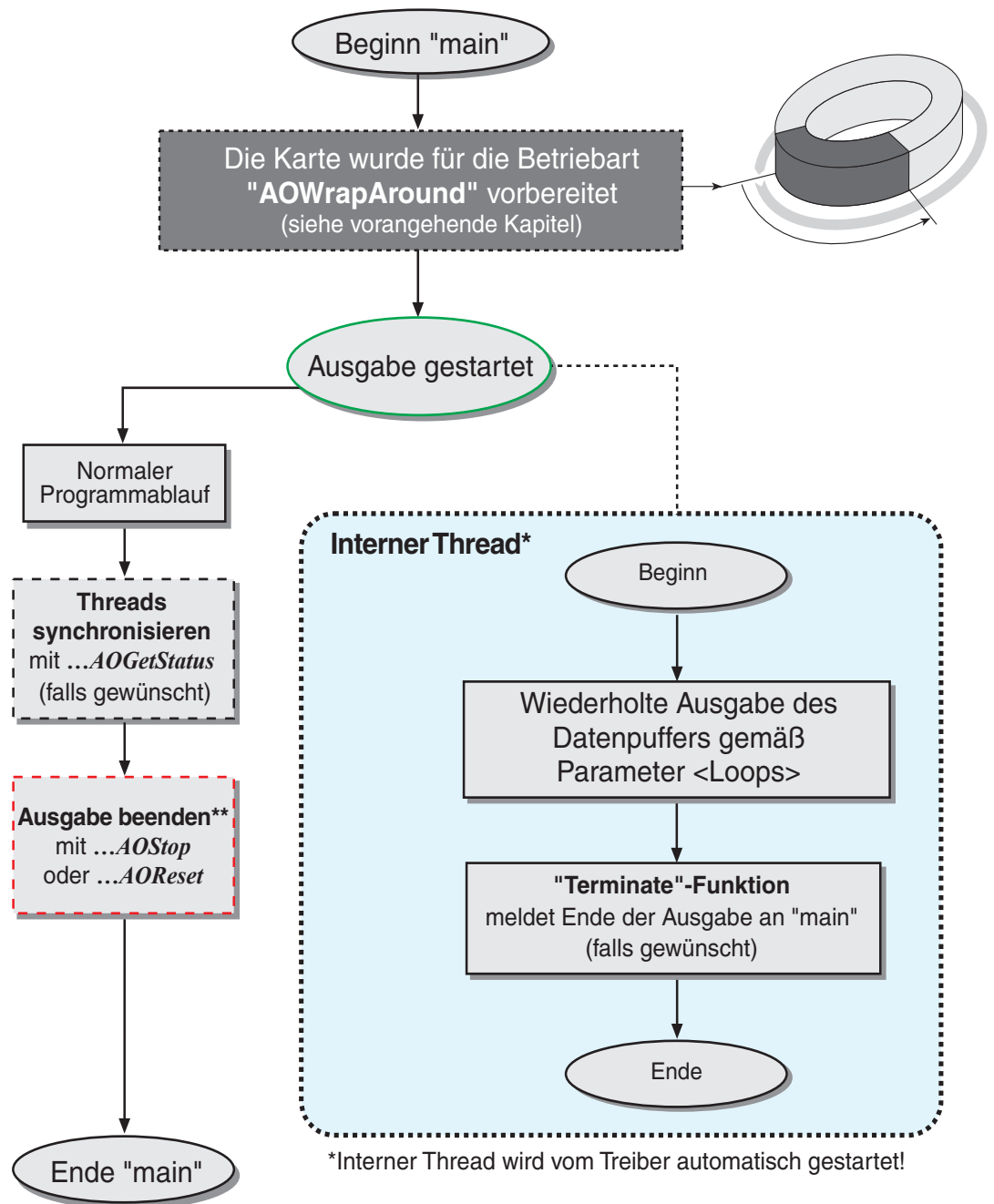
Zu a: Daten-Handling in der Betriebsart „AOWraparound“ im Ausführungsmodus „BLOCKING“:



* Der Parameter <Loops> muß einen endlichen Wert haben (Das Programm blockiert bis die Ausgabe beendet ist).

Abb. 20: Programmierung „AOWraparound“ im „BLOCKING“-Mode

Zu b: Programmierung in der Betriebsart „**AOWraparound**“ im Ausführungsmodus „**ASYNCHRONOUS**“:



**Aufruf nur notwendig falls Ausgabe vorzeitig beendet werden soll oder <Loops> mit „unendlich“ übergeben wurde.

Abb. 21: Programmierung „AOWraparound“ im „ASYNCHRONOUS“-Mode

4.1.5.3 **Starten der Ausgabe**

Zum „Scharfschalten“ der Ausgabe muß entweder die Funktion ...*AOStart* (Start eines einzelnen Kanals) oder die Funktion ...*AOStartSynchronous* (Synchron-Start mehrerer Kanäle) aufgerufen werden. Je nach „Trigger-Modus“ wird die Ausgabe unmittelbar nach Aufruf der Funktion gestartet (Software-Start) oder die Karte wartet auf das entsprechende externe Trigger-Ereignis. Falls Sie mit einem externen Triggersignal arbeiten und dieses ausbleibt können Sie mit einem geeigneten „Time-Out“-Wert die Ausgabe abbrechen.

4.1.5.4 **Stoppen der Ausgabe**

In der Betriebsart „AOContinuous“ wird die Ausgabe in der Regel dadurch beendet, daß keine Werte mehr nachgeladen werden. Mit der Funktion ...*AOStop* können Sie die Ausgabe aber auch sofort beenden - am entsprechenden D/A-Kanal wird danach 0V ausgegeben.

In der Betriebsart „AOWraparound“ können Sie die Ausgabe mit der Funktion ...*AOStop* wahlweise sofort beenden (danach wird 0V ausgegeben) oder definiert „anhalten“, d. h. die Ausgabe wird mit dem letzten Wert im Puffer und somit einem bekannten Spannungswert beendet.

Sofern zwischenzeitlich die Betriebsart für diesen Kanal nicht gewechselt wurde, kann die Ausgabe mit den Funktionen ...*AOStart* bzw. ...*AOStartSynchronous* jederzeit von vorne gestartet werden.

4.2 Digital-I/O-Teil

ME-6000/6200	ME-6100/6300
✓	✓

Alle ME-6000/6100 PCI ab Hardware-Version 2.6 sowie alle CompactPCI-Karten verfügen über zwei 8 Bit breite Digital-I/O-Ports (A, B). Jeder Port kann unabhängig als Ein- oder Ausgang konfiguriert werden. Nach dem Einschalten der Versorgung sind alle Ports auf Eingang geschaltet. Verwenden Sie die Funktion *...DIOConfig* um die Port-Richtung zu definieren.

Hinweis: Ein als Ausgang konfigurierter Port kann auch rückgelesen werden!

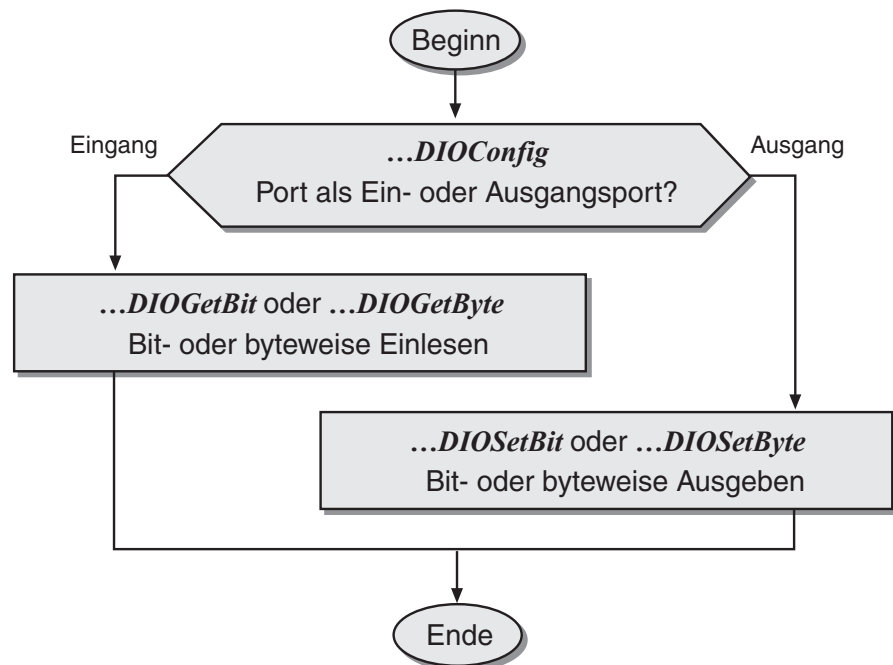


Abb. 22: Programmierung „Digital-I/O“

Zur Beschaltung des Digital-I/O-Teils lesen Sie bitte Kap. 3.4 "Digital-I/O-Teil".

4.3 Treiberkonzept

Wichtiger Hinweis: Die Karten der ME-4600 Serie (ME-4610/4650/4660/4670/4680) sowie die Karten der ME-6000 Serie verwenden unter Windows einen gemeinsamen Systemtreiber. Es wird einheitlich das Präfix „me4000“ in Datei- und Funktionsnamen verwendet.

Der 32 Bit-Treiber wurde für die Windows Treiberarchitektur „Windows Driver Model“ (WDM) entwickelt. Die WDM-Architektur wurde bisher in Windows 98/Me/2000 und XP implementiert. Zur Unterstützung der Karte unter Windows NT4.0 steht zusätzlich ein herkömmlicher Kernel-Treiber zur Verfügung. Der Systemtreiber besteht aus folgenden Komponenten:

- WDM-Treiber (me4000.sys) für Windows 98/Me/2000/XP.
- Kernel-Treiber (me4000.sys) für Windows NT.
- API-DLL (me4000.dll) für Visual C++ und Delphi. Diese API-Funktionen beginnen mit dem Präfix *me4000*...
- API-DLL (me4000Ex.dll) für Agilent VEE, LabVIEW™ und Visual Basic.

Um Ihnen die Hochsprachenprogrammierung zu erleichtern werden einfache Beispiele und kleine Projekte im Source-Code mitgeliefert. Die Programmierbeispiele finden Sie im ME Software Developer Kit (ME-SDK), das standardmäßig ins Verzeichnis C:\Meilhaus\me-sdk installiert wird. Bitte beachten Sie auch die Hinweise in den entsprechenden README-Dateien.

4.3.1 Visual C++

API-DLL	me4000.dll	Systemtreiber
Funktionsprototypen	me4000dll.h	ME-SDK
Konstantendefinition	me4000defs.h	ME-SDK
Funktions-Präfix	me4000...	

Tabelle 5: Visual C++

Die Visual C++ Unterstützung für Ihre Karte finden Sie im ME-SDK auf der „ME-Power-CD“ oder unter www.meilhaus.de/download.

4.3.2 Visual Basic

API-DLL	me4000Ex.dll	Systemtreiber
Funktionsprototypen	me4000.bas	ME-SDK
Konstantendefinition	me4000.bas	ME-SDK
Funktions-Präfix	me4000VB_...	

Tabelle 6: Visual Basic

Die Visual Basic-Unterstützung für Ihre Karte finden Sie im ME-SDK auf der „ME-Power-CD“ oder unter www.meilhaus.de/download.

Wichtige Hinweise: Die Funktionsprototypen für Visual Basic unterscheiden sich zum Teil in der Anzahl der Parameter und dem Datentyp einzelner Parameter. Beachten Sie dazu die Datei `me4000.bas`, die im ME-SDK enthalten ist. Anstatt der Standard-API `me4000.dll` müssen Sie die spezifische API der `me4000Ex.dll` verwenden. „Fehlende“ Parameter werden in der Funktionsreferenz mit dem Hinweis „**VB**“ kenntlich gemacht.

Da in Visual Basic 6.0 das „Threading Model“ geändert wurde, ist die Verwendung von Callback-Funktionen nicht möglich. In Visual Basic 5.0 ist dies jedoch möglich.

4.3.3 Delphi

API-DLL	me4000.dll	Systemtreiber
Funktionsprototypen	me4000dll.pas	ME-SDK
Konstantendefinition	me4000defs.pas	ME-SDK
Funktions-Präfix	me4000...	

Tabelle 7: Delphi

Die Delphi-Unterstützung für Ihre Karte finden Sie im ME-SDK auf der „ME-Power-CD“ oder unter www.meilhaus.de/download.

4.3.4 Agilent VEE

API-DLL	me4000Ex.dll	Systemtreiber
Funktionsprototypen	me4000VEE.h	VEE-Treibersystem
Konstantendefinition	me4000Defines.vee	VEE-Treibersystem
Funktions-Präfix	me4000VEE_...	

Tabelle 8: Agilent VEE

Das Meilhaus VEE-Treibersystem finden Sie auf der „ME-Power-CD“ oder unter www.meilhaus.de/download.

Zur Installation der VEE-Komponenten und für weitere Infos beachten Sie bitte die Dokumentation des VEE-Treibersystems. Zu den Grundlagen der VEE-Programmierung benutzen Sie bitte Ihre VEE Dokumentation und die VEE Online-Hilfe.

Wichtige Hinweise: Die Funktionsprototypen für VEE unterscheiden sich zum Teil in der Anzahl der Parameter und dem Datentyp einzelner Parameter. Beachten Sie dazu die VEE-Header-Datei `me4000VEE.h` die mit dem VEE-Treiber ins Wurzelverzeichnis Ihrer VEE-Installation kopiert wird. Anstatt der Standard-API `me4000.dll` müssen Sie die spezifische API der `me4000Ex.dll` verwenden.

Da VEE keine Callback-Funktionalität unterstützt, fehlen die entsprechenden Parameter in der VEE-spezifischen API (z. B. `<CallbackProc>`). „Fehlende“ Parameter werden in der Funktionsreferenz mit dem Hinweis „**VEE**“ kenntlich gemacht.

4.3.5 LabVIEW

API-DLL	me4000Ex.dll	Systemtreiber
Funktionsprototypen	me4000LV.h*	LabVIEW-Treiber
Konstantendefinition	keine zentrale Definitionsdatei	
Funktions-Präfix	me4000LV_...	siehe Hinweise im Text

Tabelle 9: LabVIEW

*Die Datei me4000LV.h dient nur zur Dokumentationszwecken.

Den LabVIEW™-Treiber für Ihre Karte finden Sie auf der „ME-Power-CD“ oder unter www.meilhaus.de/download.

Zur Installation der LabVIEW™-Komponenten und für weitere Infos beachten Sie bitte die Dokumentation, die Sie mit dem jeweiligen LabVIEW-Treiber erhalten. Zu den Grundlagen der LabVIEW™-Programmierung benutzen Sie bitte Ihre LabVIEW™ Dokumentation und die LabVIEW™ Online-Hilfe.

Wichtige Hinweise: Die Funktionsprototypen für LabVIEW unterscheiden sich zum Teil in der Anzahl der Parameter und dem Datentyp einzelner Parameter. Beachten Sie dazu die Header-Datei me4000LV.h die im LabVIEW-Treiber enthalten ist. Sie dient nur zur Dokumentationszwecken. Anstatt der Standard-API me4000.dll müssen Sie die spezifische API der me4000Ex.dll verwenden.

Da LabVIEW keine Callback-Funktionalität unterstützt, fehlen die entsprechenden Parameter in der LabVIEW-spezifischen API (z. B. <CallbackProc>. „Fehlende“ Parameter werden in der Funktionsreferenz mit dem Hinweis „**LV**“ kenntlich gemacht.

4.3.6 Python

Python ist eine textbasierte, interpretierte (kein Compiler nötig) und interaktive (Eingabe über Kommandozeile möglich) Programmiersprache, die im Quellcode frei verfügbar ist. Sie ermöglicht sowohl das prozedurale als auch das objektorientierte Programmieren.

Python erlaubt einfach und schnell plattformunabhängiges Programmieren unter Windows und Linux. So lässt sich ein Programm unter Windows schreiben, auf ein Linux-System kopieren und mit dem dortigen Interpreter sofort ausführen, ohne daß es kompiliert oder der Quellcode verändert werden muß.

Für den Messtechniker hält Python eine Reihe von Erweiterungsmodulen bereit. Dazu gehören Module zur Programmierung von graphischen Benutzeroberflächen, zur Visualisierung von Messdaten und für numerische Rechenoperationen.

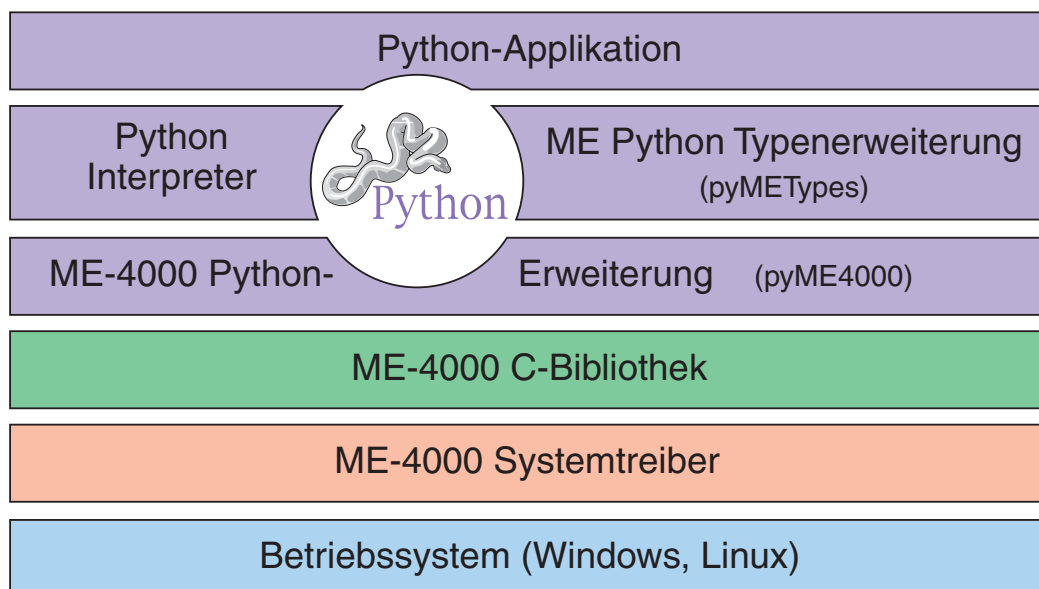


Abb. 23: Python Programmierung

Die Abbildung zeigt den prinzipiellen Aufbau des Softwaresystems. Die Python-Applikation auf der obersten Ebene sieht als Programmierschnittstelle nur den Python-Interpreter und die Erweiterungsmodule. Der plattformabhängige Teil der Softwarearchitektur wird vom Python-Interpreter und den Erweiterungsmodulen vollkommen verdeckt.

Um mit einer Karte vom Typ ME-46x0 oder ME-6x00 (nur unter Windows) unter Python zu Arbeiten benötigen Sie neben dem Systemtreiber und der dazugehörigen Funktionsbibliothek einen Python-Interpreter. Dieser ist unter <http://www.python.org> kostenlos erhältlich (in gängigen Linux-Distributionen bereits enthalten). Zusätzlich benötigen Sie das ME-4000 Erweiterungsmodul, das alle Funktionen und Konstanten für Windows bzw. Linux enthält. Beides wird von Meilhaus Electronic kostenlos zur Verfügung gestellt unter:

<http://www.sourceforge.net/projects/meilhaus>

Dort finden Sie die Pakete „pyME4000“ und „pyMETypes“ als sog. Source-Distribution für Linux und Windows. Neben den Quellen der Erweiterungsmodule sind auch Beispiel- und Testprogramme sowie README-Dateien und Installationshinweise enthalten. Zusätzlich gibt es für Windows Installationsprogramme für die Pakete „pyME4000“ und „pyMETypes“ (Voraussetzung: gültige Python-Installation).

Hinweis: Bei den Funktionsnamen wurde auf das Präfix „me4000“ generell verzichtet, da durch den Import-Befehl für das ME-4000 Erweiterungsmodul automatisch die Zeichen „me4000.“ vorangestellt werden. Beachten Sie auch, daß unter Python (wie unter Linux) für alle Funktionsgruppen die Programmierung mit der Funktion ...*Open* eröffnet und mit der Funktion ...*Close* abgeschlossen wird.

Beispiel für Konsolenprogramm:

```
1 # Python
2 > import me4000
3 > me4000.DIOOpen(0)
4 > value = me4000.DIOGetByte(0, 0)
5 > me4000.DIOClose(0)
6 > print 'Value = 0x%X' % value
7 Value = 0xAA
```


5 Funktionsreferenz

5.1 Allgemeine Hinweise

- **Funktionsprototypen:**

In der folgenden Funktionsbeschreibung werden die generischen Funktionsprototypen für Visual C++ verwendet. Die Definitionen für andere unterstützte Programmiersprachen mit zum Teil unterschiedlichen Datentypen entnehmen Sie bitte den entsprechenden Definitionsdateien im ME-SDK bzw. den Header-Dateien des LabVIEW- bzw. VEE-Treibers (siehe auch Kap. 4.3 ab Seite 38).

- **Parameter „BoardNumber“**

Beim Einsatz einer einzigen Karte einer Kartenfamilie, ist die „BoardNumber“ stets „0“ (Integerwert). In Systemen mit mehreren Karten aus der gleichen Kartenfamilie entscheidet das System unter welcher „BoardNumber“ die jeweilige Karte anzusprechen ist. Ermitteln Sie nach Installation der Karten die Zuordnung der „BoardNumber“.

Wichtiger Hinweis: Softwaretechnisch gehören die Karten der ME-4600 Serie (aktuell: ME-4610/4650/4660/4670/4680) und die Karten der ME-6000-Serie zur gleichen Familie und verwenden einen gemeinsamen Treiber. D, h. die Karten „teilen“ sich den Wertebereich des Parameters <BoardNumber> von 0...31.

Tip: Verifizieren Sie zu Beginn Ihres Programms die Zuordnung von „BoardNumber“ und Seriennummer (siehe Funktion ...*GetSerialNumber*).

- **Ausführungsmodus BLOCKING:**

Beachten Sie, daß es in Verbindung mit geringer Rechnerleistung, langen Sample-Raten oder nicht bzw. nur langsam eintreffenden externen Triggersignalen zu einer längeren Blockierung des Rechners kommen kann.

Beachte: Agilent VEE und LabVIEW arbeiten grundsätzlich im Ausführungsmodus „BLOCKING“.

- **Externer Trigger mit Time-Out:**

Bei Funktionen mit externem Trigger ist es möglich, ein Zeitintervall anzugeben, in dem der **erste** Triggerimpuls eintreffen muß, ansonsten wird die Operation abgebrochen (Parameter <TimeOutSeconds>). Das Ausbleiben weiterer Triggerimpulse z. B. in den analogen Erfassungsmodi „Extern-Einzelwert“ und „Extern-Kanalliste“ wird dadurch nicht abgefangen. Berücksichtigen Sie dies gegebenenfalls bei der Wahl des Ausführungsmodus.

5.2 Nomenklatur

Die API-Funktionen der ME-4000 Funktionsbibliothek gelten für alle Karten vom Typ ME-4610/4650/4660/4670/4680 sowie die Karten der ME-6000 Serie (sofern Funktionalität vom jeweiligen Kartentyp unterstützt wird). Es wird einheitlich das Präfix „*me4000*“ in den Funktionsnamen verwendet. Der Funktionsname besteht aus dem Präfix und mehreren Bestandteilen, die die jeweilige Funktion näher beschreiben und weitgehend „selbstredend“ sind (z. B. „AO“ für analoge Ausgabe).

Für Visual C++ und Delphi gibt es keine sprachspezifische Kennung, z B. *me4000AOConfig*. Für Agilent VEE werden die Zeichen „*VEE_*“ (z. B. *me4000VEE_AOConfig*), für LabVIEW die Zeichen „*LV_*“ (z. B. *me4000LV_AOConfig*) und für Visual Basic die Zeichen „*VB_*“ (z. B. *me4000VB_AOConfig*) eingefügt.

Für die Funktionsbeschreibung gelten folgende Vereinbarungen:

<i>Funktionsnamen</i>	werden im Fließtext kursiv geschrieben z. B. <i>me4000GetBoardVersion</i> .
<Parameter>	werden in spitzen Klammern in der Schriftart Courier geschrieben.
[eckige Klammern]	werden zur Kennzeichnung physikalischer Einheiten verwendet.
main (...)	Programmausschnitte sind in der Schriftart Courier geschrieben.

5.3 Beschreibung der API-Funktionen

Die Funktionsbeschreibung ist nach den folgenden Funktionsgruppen geordnet; innerhalb einer Funktionsgruppe gilt alphabetische Reihenfolge:

„5.3.1 Fehler-Behandlung“ auf Seite 49

„5.3.2 Allgemeine Funktionen“ auf Seite 53

„5.3.3 Analoge Ausgabe“ auf Seite 61

„5.3.4 Digitale Ein-/Ausgabe“ auf Seite 84

Funktion	Kurzbeschreibung	Seite
<i>Fehler-Behandlung</i>		
...ErrorMessage	Fehlernummer einen Fehlerstring zuweisen.	49
...ErrorGetLastMessage	Zuletzt aufgetretenem Fehler einen Fehlerstring zuweisen	50
...ErrorSetDefaultProc	Vordefinierte, globale Fehlerroutine für API installieren	51
...ErrorSetUserProc	Benutzerdefinierte, globale Fehlerroutine für API installieren	52
<i>Allgemeine Funktionen</i>		
...FrequencyToTicks	Frequenz in Ticks umrechnen	53
...GetBoardVersion	Kartenversion ermitteln	55
...GetDLLVersion	DLL-Versionsnummer ermitteln	57
...GetDriverVersion	Treiber-Versionsnummer ermitteln	57
...GetSerialNumber	Seriennummer der Karte ermitteln	58
...TimeToTicks	Periodendauer in Ticks umrechnen	59
<i>Analoge Ausgabe</i>		
...AOAppendNewValues	Ausgabepuffer nachladen	61
...AOConfig	D/A-Teil konfigurieren	63
...AOContinuous	Kontinuierliche Ausgabe	65
...AOGetStatus	Status-Abfrage für „AOContinuous“ und „AOWraparound“	67
...AOReset	Ausgabekanal rücksetzen	68

Tabelle 10: Übersicht der Bibliotheksfunktionen

Funktion	Kurzbeschreibung	Seite
...AOSingle	Einzelwert-Ausgabe	69
...AOSingleSimultaneous	Start der simultanen Ausgabe in der Betriebsart „AOSimultaneous“	71
...AOSTart	Start einer timergesteuerten Ausgabe	73
...AOSTartSynchronous	Synchron-Start in den Betriebsarten „AOContinuous“ & „AOWraparound“	74
...AOSTop	Ausgabe stoppen	77
...AOVoltToDigit	Umrechnung des auszugebenden Spannungswertes in Digitwert	78
...AOWaveGen	Einfacher Funktionsgenerator	79
...AOWraparound	Periodische Ausgabe	81
Digitale Ein/Ausgabe		
...DIOConfig	Digital-Ports für digitale Ein-/Ausgabe konfigurieren	84
...DIOGetBit	Bit einlesen	85
...DIOGetByte	Byte einlesen	86
...DIOResetAll	DIO-Teil rücksetzen	87
...DIOSetBit	Bit ausgeben	88
...DIOSetByte	Byte ausgeben	89

Tabelle 10: Übersicht der Bibliotheksfunktionen

5.3.1 Fehler-Behandlung

me4000ErrorMessage

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Diese Funktion kann dazu verwendet werden um eine Fehlernummer, die von einer API-Funktion zurückgegeben wurde in einen lesbaren Text umzuwandeln.

● Definitionen

VC: me4000ErrorMessage(int iErrorCode, char* pcBuffer, unsigned int uiBufferSize);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<ErrorCode>

Nummer des Fehlers, den die API-Funktion verursacht hat.

<Buffer>

Zeiger auf die Fehlerbeschreibung.

<BufferSize>

Puffergröße in Bytes für Fehlerbeschreibung (max. 256 Zeichen).

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000ErrorGetLastMessage

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Diese Funktion gibt den letzten, von einer „me4000...“ API-Funktion verursachten Fehler zurück. Ein entsprechender Fehlertext kann angezeigt werden.

● Definitionen

VC: me4000ErrorGetLastMessage(char* pcBuffer, unsigned int uiBufferSize);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<Buffer>

Zeiger auf die Fehlerbeschreibung.

<BufferSize>

Puffergröße in Bytes für Fehlerbeschreibung (max. 256 Zeichen).

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000ErrorSetDefaultProc

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Diese Funktion dient dazu eine vordefinierte globale Fehlerroutine für die ganze API zu installieren. Die Fehlerroutine wird automatisch aufgerufen, sobald eine Funktion einen Fehler zurück gibt. Sie erhalten folgende Infos in Form einer Message-Box:

- Name der Funktion, die den Fehler verursacht hat
- Kurze Fehler-Beschreibung
- Fehlercode

Hinweis:

Es kann stets nur eine globale Fehlerroutine installiert sein (...*ErrorSetDefaultProc* oder ...*ErrorSetUserProc*).

● Definitionen

VC: me4000ErrorSetDefaultProc(int iDefaultProcStatus);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<DefaultProcStatus>

- ME4000_ERROR_DEFAULT_PROC_ENABLE
Installieren der vordefinierten Fehlerroutine.
- ME4000_ERROR_DEFAULT_PROC_DISABLE
Deinstallieren der vordefinierten Fehlerroutine.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000ErrorSetUserProc

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Diese Funktion dient dazu eine benutzerdefinierte, globale Fehlerroutine für die API zu installieren. Danach wird diese Routine automatisch aufgerufen, sobald eine Funktion einen Fehler zurück gibt. Verwenden Sie die Funktion `...ErrorMessage` um dem Error-code eine Fehlerbeschreibung zuzuordnen.

Hinweis:

Es kann stets nur eine globale Fehlerroutine installiert sein (`...ErrorSetDefaultProc` oder `...ErrorSetUserProc`).

● Definitionen

Typdefinition für ME4000_P_ERROR_PROC:

```
typedef void (_stdcall * ME4000_P_ERROR_PROC)
(char* pcFunctionName, int iErrorCode)
```

VC: `me4000ErrorSetUserProc(ME4000_P_ERROR_PROC pErrorProc);`

LV: `me4000LV_...` (siehe `me4000LV.h`)

VB: `me4000VB_...` (siehe `me4000.bas`)

VEE: `me4000VEE_...` (siehe `me4000VEE.h`)

→ Parameter

<ErrorProc>

Zeiger auf eine Fehlerroutine. Es wird der Name der fehlerhaften Funktion und der Fehlercode an die hier „installierte“ Funktion übergeben. Durch Übergabe von NULL wird eine bereits installierte Fehlerroutine wieder deinstalliert.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (`ME4000_NO_ERROR`) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

5.3.2 Allgemeine Funktionen

me4000FrequencyToTicks

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Konvertiert die gewünschte Frequenz [Hz] in die Anzahl der „Ticks“ zur Übergabe an die Timer in den entsprechenden „...*Config*“-Funktionen dieser Funktionsbibliothek. Der erlaubte Wertebereich ist abhängig vom jeweiligen Timer. Falls Hardwaregrenzen überschritten werden, führt dies zu einer Fehlermeldung der entsprechenden ...*Config*-Funktion.

Beispiel: Die max. Sample-Rate des A/D-Teils von 500 kS/s entspricht 66 Ticks (ChanTicks).

Hinweis:

Die Anzahl der Ticks errechnet sich folgendermaßen:

Allgemein:
$$\frac{1}{\text{Frequenz[Hz]} \cdot 30,30 \cdot 10^{-9} \text{s}} = \text{Ticks}$$

Die Periodendauer läßt sich in Schritten von 30,30 ns innerhalb des erlaubten Wertebereichs (siehe Parameter) einstellen.

Beispiel:

Anzahl der Ticks zur Übergabe an den Parameter <ChanTicks> für die maximale Abtastrate von 500 kS/s:

$$\frac{1}{500000\text{Hz} \cdot 30,30 \cdot 10^{-9} \text{s}} = 66 \text{ Ticks (42Hex)}$$

Beachten Sie, daß der Parameter <TicksHighPart> nur für eine SCAN-Frequenz <0.0077 Hz benötigt wird (siehe Funktionen „...*AConfig*“ und „...*MultiSigAConfig*“). Sie können damit SCAN-Frequenzen bis ca. 0,00048 Hz einstellen.

Programmierbeispiele finden Sie im ME Software-Developer-Kit (ME-SDK).

● Definitionen

VC: me4000FrequencyToTicks(double dRequiredFreq, unsigned long* pulTicksLowPart, unsigned long* pulTicksHighPart, double* pdAchievedFreq);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<RequiredFreq>

Gewünschte Frequenz in [Hz] zur Umrechnung in Ticks. Bei Übergabe von „0“ wird in <TicksLowPart> und <TicksHighPart> FFFFFFFFHex zurückgegeben. Der betreffende Timer wird damit mit minimaler Frequenz programmiert.

<TicksLowPart>

Pointer auf die errechneten Ticks (niederwertige 32 Bits) zur Übergabe an die entsprechenden Parameter der „...Config“-Funktionen.

<TicksHighPart>

Pointer auf die errechneten Ticks für den höherwertigen Teil (Bits 32...35) des insgesamt 36 Bit breiten Scan-Timers. Zur Übergabe an den Parameter ScanTicksHigh der Funktionen ...AIConfig und ...MultiSigAIConfig. Wird nur für SCAN-Frequenzen <0.0077 Hz benötigt, ansonsten liefert dieser Parameter stets „0“ zurück.

<AchievedFreq>

Zeiger auf einen Double-Wert der nach Rückkehr der Funktion die tatsächlich einstellbare Frequenz [Hz] enthält (es wird stets die nächst höhere Frequenz gewählt). Übergeben Sie ME4000_POINTER_NOT_USED falls Parameter nicht genutzt werden soll.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000GetBoardVersion

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Diese Funktion ermittelt die Kartenversion (Device-ID).

● Definitionen

VC: me4000GetBoardVersion(unsigned int uiBoardNumber, unsigned short* pusVersion);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<Version>

Zeiger auf die Device-ID. Mögliche Werte sind:

6014/6054¹⁾Hex: ME-6000/4 „Opto“, 4 D/A, 16 DIO¹⁾

6018/6058¹⁾Hex: ME-6000/8 „Opto“, 8 D/A, 16 DIO¹⁾

601F/605F¹⁾Hex: ME-6000/16 „Opto“, 16 D/A, 16 DIO¹⁾

6034/6074¹⁾Hex: ME-6000/4 „Insel“, 4 D/A, 16 DIO¹⁾

6038/6078¹⁾Hex: ME-6000/8 „Insel“, 8 D/A, 16 DIO¹⁾

603F/607F¹⁾Hex: ME-6000/16 „Insel“, 16 D/A, 16 DIO¹⁾

6114/6154¹⁾Hex: ME-6100/4 „Opto“, 4 D/A, FIFOs, 16 DIO¹⁾

6118/6158¹⁾Hex: ME-6100/8 „Opto“, 8 D/A, FIFOs, 16 DIO¹⁾

611F/615F¹⁾Hex: ME-6100/16 „Opto“, 16 D/A, FIFOs, 16 DIO¹⁾

6134/6174¹⁾Hex: ME-6100/4 „Insel“, 4 D/A, FIFOs, 16 DIO¹⁾

6138/6178¹⁾Hex: ME-6100/8 „Insel“, 8 D/A, FIFOs, 16 DIO¹⁾

613F/617F¹⁾Hex: ME-6100/16 „Insel“, 16 D/A, FIFOs, 16 DIO¹⁾

¹⁾ 16 Digital-I/Os auf PCI-Karten ab Hardware-Revision 2.6 (mit geänderten Device-IDs).

6255Hex:	ME-6200/5	„Opto“, 5 D/A, 16 DIO, 0-50V ²⁾
6265Hex:	ME-6200/5	„Opto“, 5 D/A, 16 DIO, U_EXT ³⁾
6259Hex:	ME-6200/9	„Opto“, 9 D/A, 16 DIO, 0-50V ²⁾
6269Hex:	ME-6200/9	„Opto“, 9 D/A, 16 DIO, U_EXT ³⁾
6275Hex:	ME-6200/5	„Insel“, 5 D/A, 16 DIO, 0-50V ²⁾
6285Hex:	ME-6200/5	„Insel“, 5 D/A, 16 DIO, U_EXT ³⁾
6279Hex:	ME-6200/9	„Insel“, 9 D/A, 16 DIO, 0-50V ²⁾
6289Hex:	ME-6200/9	„Insel“, 9 D/A, 16 DIO, U_EXT ³⁾
6355Hex:	ME-6300/5	„Opto“, 5 D/A, FIFOs, 16 DIO, 0-50V ²⁾
6365Hex:	ME-6300/5	„Opto“, 5 D/A, FIFOs, 16 DIO, U_EXT ³⁾
6359Hex:	ME-6300/9	„Opto“, 9 D/A, FIFOs, 16 DIO, 0-50V ²⁾
6369Hex:	ME-6300/9	„Opto“, 9 D/A, FIFOs, 16 DIO, U_EXT ³⁾
6375Hex:	ME-6300/5	„Insel“, 5 D/A, FIFOs, 16 DIO, 0-50V ²⁾
6385Hex:	ME-6300/5	„Insel“, 5 D/A, FIFOs, 16 DIO, U_EXT ³⁾
6379Hex:	ME-6300/9	„Insel“, 9 D/A, FIFOs, 16 DIO, 0-50V ²⁾
6389Hex:	ME-6300/9	„Insel“, 9 D/A, FIFOs, 16 DIO, U_EXT ³⁾

²⁾ „U-Plus“-Kanal wird über die Karte versorgt. Diese Modelle sind grundsätzlich mit EPROM bestückt.

³⁾ „U-Plus“-Kanal benötigt externe Spannungsversorgung.

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000GetDLLVersion

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Ermittelt die Versionsnummer der Treiber-DLL.

● Definitionen

VC: me4000GetDLLVersion(unsigned long* pulVersion);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<Version>

Versionsnummer. Der 32-Bit-Wert enthält in den höherwertigen 16 Bit die Hauptversion und in den niederwertigen 16 Bit die Unterversion. Beispiel: 0x00020001 ergibt die Version 2.01

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000GetDriverVersion

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Ermittelt die Versionsnummer des Treibers.

● Definitionen

VC: me4000GetDriverVersion(unsigned long* pulVersion);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<Version>

Zeiger auf die Treiberversion (hexadezimal codiert).

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000GetSerialNumber

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Ermittelt die Seriennummer der ausgewählten ME-4600.

● Definitionen

VC: me4000GetSerialNumber(unsigned int uiBoardNumber, unsigned long* pulSerialNumber);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<SerialNumber>

Zeiger auf die Seriennummer.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000TimeToTicks

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Konvertiert die gewünschte Periodendauer [s] in die Anzahl der „Ticks“ zur Übergabe an die Timer in den entsprechenden „...*Config*“-Funktionen dieser Funktionsbibliothek. Der erlaubte Wertebereich ist abhängig vom jeweiligen Timer. Falls Hardwaregrenzen überschritten werden, führt dies zu einer Fehlermeldung der entsprechenden ...*Config*-Funktion.

Beispiel: Die min. CHAN-Zeit von 2 µs entspricht 66 Ticks (Chan-Ticks).

Hinweis:

Die Anzahl der Ticks errechnet sich folgendermaßen:

$$\text{Allgemein:} \quad \frac{\text{Periodendauer[s]}}{30,30 \cdot 10^{-9} \text{ s}} = \text{Ticks}$$

Die Periodendauer läßt sich in Schritten von 30,30 ns innerhalb des erlaubten Wertebereichs (siehe Parameter) einstellen.

Beispiel:

Anzahl der Ticks zur Übergabe an den Parameter <ChanTicks> für eine minimalen Periodendauer von 2 µs:

$$\frac{(2 \cdot 10^{-6}) \text{ s}}{30,30 \cdot 10^{-9} \text{ s}} = 66 \text{ Ticks (42Hex)}$$

Beachten Sie, daß der Parameter <TicksHighPart> nur für SCAN-Zeiten >130s benötigt wird (siehe Funktionen „...*AICConfig*“ und „...*MultiSigAICConfig*“). Sie können damit SCAN-Zeiten bis ca. 34 Minuten einstellen.

Programmierbeispiele finden Sie im ME Software-Developer-Kit (ME-SDK).

● Definitionen

VC: me4000TimeToTicks(double dRequiredTime, unsigned long* pulTicksLowPart, unsigned long* pulTicksHighPart, double* pdAchievedTime);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<RequiredTime>

Periodendauer [s] zur Umrechnung in Ticks. Bei Übergabe von „0“ wird in <TicksLowPart> und <TicksHighPart> 0Hex zurückgegeben.

<TicksLowPart>

Pointer auf die errechneten Ticks (niederwertige 32 Bits) zur Übergabe an die entsprechenden Parameter der „...Config“-Funktionen.

<TicksHighPart>

Pointer auf die errechneten Ticks für den höherwertigen Teil (Bits 32...35) des insgesamt 36 Bit breiten Scan-Timers. Zur Übergabe an den Parameter ScanTicksHigh der Funktionen ...AIConfig und ...MultiSigAIConfig. Wird nur für SCAN-Zeiten >130s benötigt, ansonsten liefert dieser Parameter stets „0“ zurück.

<AchievedTime>

Zeiger auf einen Double-Wert der nach Rückkehr der Funktion die tatsächlich einstellbare Periodendauer [s] enthält (es wird stets die nächst niedrigere Periodendauer gewählt). Übergeben Sie ME4000_POINTER_NOT_USED falls Parameter nicht genutzt werden soll.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

5.3.3 Analoge Ausgabe

me4000AOAppendNewValues

Beschreibung

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Diese Funktion dient dem kontinuierlichen Nachladen des D/A-FIFOs während einer laufenden Ausgabe. Mit den Funktionen *...AOStop* oder *...AOReset* wird die Ausgabe sofort und vollständig beendet.

Hinweis!

Sie müssen nicht den gleichen, wie in *...AOContinuous* verwendeten Datenpuffer verwenden.

● Definitionen

VC: me4000AOAppendNewValues(unsigned int uiBoardNumber, unsigned int uiChannelNumber, short* psBuffer, unsigned long ulNumberOfValuesToAppend, int iExecutionMode, unsigned long* pulNumberOfValuesAppended);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

D/A-Kanal 0...3

<Buffer>

Zeiger auf Datenpuffer mit den nachzuladenden Werten.

<NumberOfValuesToAppend>

Anzahl der Werte im Datenpuffer. Wenn Sie hier „0“ übergeben, erhalten Sie im Parameter <NumberOfValuesAppended> die Anzahl der Werte zurück, die aktuell im Datenpuffer Platz finden würden.

<ExecutionMode>

Ausführungsmodus für diese Funktion wählen:

- ME4000_AO_APPEND_NEW_VALUES_BLOCKING:
Das Programm ist blockiert bis alle Werte im Ringpuffer Platz gefunden haben.
- ME4000_AO_APPEND_NEW_VALUES_NON_BLOCKING:
Das Programm „füllt“ nur die Anzahl an Werten nach, die aktuell im Ringpuffer Platz finden.

Falls Sie im Parameter <NumberOfValuesToAppend> den Wert „0“ übergeben haben, ist dieser Parameter nicht relevant.

<NumberOfValuesAppended>

Anzahl der tatsächlich in den Ringpuffer geladenen Werte. Siehe auch Parameter <NumberOfValuesToAppend>.

< Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOConfig

Beschreibung

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Diese Funktion konfiguriert die Hardware des D/A-Teils für eine timergesteuerte Ausgabe in den Betriebsarten „AOContinuous“ und „AOWraparound“. Gestartet wird die Ausgabe entweder mit der Funktion `...AOStart` (für einen Kanal) oder mit der Funktion `...AOStartSynchronous` (Synchronstart mehrerer Kanäle). Sie können wählen zwischen Software-Start oder Start durch ein externes Triggersignal.

Als Zeitbasis dient ein 32 Bit Zähler der mit einem 33 MHz Takt gespeist wird. Daraus ergibt sich eine Periodendauer von 30,30ns, die als kleinste Zeiteinheit definiert wird und im Folgenden „1 Tick“ genannt wird. Die Sample-Rate für die analoge Ausgabe muß als Vielfaches eines Ticks im Parameter `<Ticks>` übergeben werden. D. h. die Sample-Rate läßt sich in Schritten von 30,30ns zwischen minimaler und maximaler Sample-Rate einstellen. Die min. Sample-Rate beträgt ca. 0,5 Samples/Minute, die max. Sample-Rate beträgt in Abhängigkeit von Betriebsart und Performance Ihres Rechners bis zu 500 kS/s pro Kanal.

Beachten Sie auch das Kapitel „Programmierung“ ab Seite 21, sowie die Programmbeispiele im ME-SDK.

Hinweis:

Die Funktion `...FrequencyToTicks` bzw. `...TimeToTicks` bietet Ihnen eine bequeme Umrechnungsmöglichkeit von Frequenz bzw. Periodendauer in Ticks zur Übergabe an den D/A-Timer (siehe S. 53ff). Sollte bei Aufruf dieser Funktion ein betroffener D/A-Kanal bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

● Definitionen

VC: `me4000AOConfig(unsigned int uiBoardNumber, unsigned int uiChannelNumber, unsigned long ulTicks, int iTriggerMode, int iExtTriggerEdge);`

LV: `me4000LV_...` (siehe `me4000LV.h`)

VB: `me4000VB_...` (siehe `me4000.bas`)

VEE: `me4000VEE_...` (siehe `me4000VEE.h`)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

D/A-Kanal 0...3

<Ticks>

Anzahl der Ticks für den D/A-Timer (32 Bit), der die Sample-Rate für die timergesteuerte Ausgabe bestimmt. Der Wertebereich liegt zwischen 66 (42Hex) und $2^{32}-1$ (FFFFFFFFHex) Ticks.

<TriggerMode>

Trigger-Ereignis zum Start der analogen Ausgabe (bei Synchron-Start gelten die Einstellungen in *...AOSTartSynchronous*):

- ME4000_AO_TRIGGER_SOFTWARE
Start per Software unmittelbar nach Aufruf der Funktion *...AOSTart*.
- ME4000_AO_TRIGGER_EXT_DIGITAL
Bereit zur Ausgabe nach Aufruf der Funktion *...AOSTart*. Ausgabe wird durch externes Trigger-Signal gestartet.

<ExtTriggerEdge>

Auswahl der Triggerflanke für den entsprechenden Triggereingang DA_TRIG_x (bei Synchron-Start gelten die Einstellungen in *...AOSTartSynchronous*):

- ME4000_AO_TRIGGER_EXT_EDGE_RISING
Start durch steigende Flanke.
- ME4000_AO_TRIGGER_EXT_EDGE_FALLING
Start durch fallende Flanke.
- ME4000_AO_TRIGGER_EXT_EDGE_BOTH
Start durch fallende oder steigende Flanke.
- ME4000_VALUE_NOT_USED
Kein ext. Trigger verwendet. Siehe Parameter <Trigger-Mode>.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOContinuous

Beschreibung

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Diese Funktion dient der Vorbereitung der Betriebsart „AOContinuous“. Sie können damit beliebige Signalverläufe ausgeben, die sich nach Beginn der Ausgabe auch ändern können (im Gegensatz zur Betriebsart „AOWraparound“). Der D/A-Timer gibt ein festes Zeitraster (Sample-Rate) für die Ausgabe vor (siehe ...*AOConfig*). Allokieren sie für jeden Kanal, der verwendet werden soll, einen Datenpuffer definierter Größe, der die ersten auszugebenden Werte enthält. Verwenden Sie die Funktion ...*AOAppendNewValues* zum kontinuierlichen Nachladen der Werte. Dies kann mit oder ohne Callback-Funktion geschehen.

Gestartet wird die Ausgabe mit der Funktion ...*AOStart(Synchronous)* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe ...*AOConfig*). Mit der Funktion ...*AOStop* können Sie die Ausgabe sofort beenden. Sofern zwischenzeitlich die Betriebsart für diesen Kanal nicht gewechselt wurde, kann die Ausgabe mit der Funktion ...*AOStart(Synchronous)* jederzeit von vorne gestartet werden. Mit der Funktion ...*AOReset* wird im Vergleich zu ...*AOStop* auch das D/A-FIFO gelöscht und damit die Ausgabe vollständig beendet.

Siehe auch Kap. 4.1 auf Seite 21 und Programmierbeispiele im ME-Software-Developer-Kit (ME-SDK).

Hinweis:

Sollte bei Aufruf dieser Funktion ein betroffener D/A-Kanal bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

● Definitionen

Typdefinition für ME4000_P_AO_CALLBACK_PROC:

```
typedef void (_stdcall *
ME4000_P_AO_CALLBACK_PROC)
(unsigned long ulBufferAvailable,
void* pCallbackContext);
```

VC: me4000AOContinuous(unsigned int uiBoardNumber, unsigned int uiChannelNumber, short* psBuffer, unsigned long ulDataCount, ME4000_P_AO_CALLBACK_PROC pCallbackProc, void* pCallbackContext, unsigned long ulTimeOutSeconds, unsigned long* pulNumberOfValuesWritten);

LV: me4000LV_... (siehe me4000LV.h)
 VB: me4000VB_... (siehe me4000.bas)
 VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

D/A-Kanal 0...3

<Buffer>

Zeiger auf benutzerallokierten Datenpuffer, der mit den **ersten** auszugebenden Werten gefüllt ist.

<DataCount>

Anzahl der Werte im Datenpuffer „Buffer“

<CallbackProc>

LV, VB, VEE

Callback-Funktion, die regelmäßig aufgerufen wird um den Datenpuffer nachzuladen. Falls diese Funktionalität nicht erwünscht ist, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<CallbackContext>

LV, VB, VEE

Benutzerdefinierter Zeiger, der an die Callback-Funktion übergeben werden kann. Falls keine Callback-Funktion verwendet wird, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<TimeOutSeconds>

Optional können Sie hier ein Zeitintervall in Sekunden angeben innerhalb dessen der erste Triggerimpuls eintreffen muß. Ansonsten wird die Operation abgebrochen. Falls Sie ohne ext. Trigger arbeiten oder kein Time-Out nutzen möchten, übergeben Sie hier die Konstante ME4000_VALUE_NOT_USED.

Bei Synchron-Start gilt der Time-Out-Wert in der Funktion *...AOSTartSynchronous*.

<NumberOfValuesWritten>

Anzahl der Werte, die tatsächlich in den Datenpuffer geschrieben werden konnten.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOGetStatus

Beschreibung

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Funktion dient der Abfrage ob eine analoge Ausgabe in den Betriebsarten „AOContinuous“ und „AOWraparound“ noch läuft oder das FIFO bereits „leer gelaufen“ ist. Dies ist dann der Fall wenn Sie entweder das FIFO bewußt nicht mehr nachgeladen haben um die Ausgabe zu beenden oder das FIFO aufgrund zu geringer Rechnerleistung nicht rechtzeitig nachgeladen werden konnte.

Über den Parameter <WaitIdle> können Sie steuern, ob die Funktion sofort den aktuellen Status zurückgeben soll oder ob Sie warten möchten bis die Ausgabe beendet ist.

● Definitionen

VC: me4000AOGetStatus(unsigned int uiBoardNumber, unsigned int uiChannelNumber, int iWaitIdle, int* piStatus);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

D/A-Kanal 0...3

<WaitIdle>

„Rückkehr-Verhalten“ dieser Funktion:

- ME4000_AO_WAIT_NONE
Funktion gibt im Parameter <Status> den aktuellen Betriebszustand sofort zurück.
- ME4000_AO_WAIT_IDLE
Funktion kehrt erst nach Ende der Ausgabe zurück (FIFO leer). Der Parameter <Status> gibt immer den Wert ME4000_AO_STATUS_IDLE zurück.

<Status>

Aktueller Betriebszustand:

- ME4000_AO_STATUS_IDLE
Die Ausgabe ist beendet, d. h. das FIFO ist leer.
- ME4000_AO_STATUS_BUSY
Die Ausgabe läuft noch.

< Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOReset** Beschreibung**

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Diese Funktion beendet eine laufende Ausgabe des betreffenden Kanals in den Betriebsarten „AOContinuous“ oder „AOWraparound“. Die Ausgabe wird sofort und vollständig beendet. Danach wird der korrespondierende Ringpuffer gelöscht und der Kanal auf 0V gesetzt.

● Definitionen

VC: me4000AOReset(unsigned int uiBoardNumber, unsigned int uiChannelNumber);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter**<BoardNumber>**

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

- Reset eines gewünschten Kanals. Übergeben Sie die entsprechende Kanalnummer 0...3.

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOSingle

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Diese Funktion dient der „transparenten“ Einzelwert-Ausgabe auf einen bestimmten D/A-Kanal.

Für die Einzelwert-Ausgabe („AOSingle“) ist keine weitere Konfiguration mit anderen Funktionen nötig.

Für die simultane Ausgabe auf mehreren Kanälen (Betriebsart „AO-Simultaneous“) verwenden Sie bitte die Funktion *...AOSingleSimultaneous*.

Hinweis:

Sollte bei Aufruf dieser Funktion ein betroffener D/A-Kanal bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

Rückkehr-Verhalten in Abhängigkeit vom Triggermodus:

- Software-Start: sofort
- ext. Trigger ohne Time-Out: sofort
- ext. Trigger mit Time-Out: nach Ablauf des Time-Out oder nach Eintreffen des ext. Triggersignals.

● Definitionen

VC: me4000AOSingle(unsigned int uiBoardNumber, unsigned int uiChannelNumber, int iTriggerMode, int iExtTriggerEdge, unsigned long ulTimeOutSeconds, short sValue);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

D/A-Kanal 0...15

<TriggerMode>

Trigger-Ereignis zum Start der analogen Ausgabe:

- ME4000_AO_TRIGGER_SOFTWARE
Einzel-Wert unmittelbar nach Aufruf dieser Funktion ausgeben (Software-Start).
- ME4000_AO_TRIGGER_EXT_DIGITAL
Bereit zur Ausgabe nach Aufruf dieser Funktion. Die Ausgabe wird durch externes Trigger-Signal am betreffenden Trigger-eingang DA_TRIG_x gestartet.

<ExtTriggerEdge>

Auswahl der Triggerflanke für den entsprechenden Trigger-Eingang DA_TRIG_x.

- ME4000_AO_TRIGGER_EXT_EDGE_RISING
Start durch steigende Flanke.
- ME4000_AO_TRIGGER_EXT_EDGE_FALLING
Start durch fallende Flanke.
- ME4000_AO_TRIGGER_EXT_EDGE_BOTH
Start durch fallende oder steigende Flanke.
- ME4000_VALUE_NOT_USED
Kein ext. Trigger verwendet. Siehe Parameter <Trigger-Mode>.

<TimeOutSeconds>

Optional können Sie hier ein Zeitintervall in Sekunden angeben innerhalb dessen der erste Triggerimpuls eintreffen muß. Ansonsten wird die Operation abgebrochen. Falls Sie ohne ext. Trigger arbeiten oder kein Time-Out nutzen möchten, übergeben Sie hier die Konstante ME4000_VALUE_NOT_USED.

<Value>

16 Bit Ausgabewert; beachten Sie die Kennlinien auf Seite 22/23.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOSingleSimultaneous

Beschreibung

ME-6000/6200	ME-6100/6300
✓	✓

Funktion zur simultanen Ausgabe auf mehrere D/A-Kanäle (Betriebsart „AOSimultaneous“). Über das Wertefeld <ChannelNumber> können Sie festlegen welche Kanäle in die simultane Ausgabe einbezogen werden sollen. Sie können wählen welcher Triggereingang (bzw. Eingänge) der simultanen Kanäle die Ausgabe starten soll.

Hinweis:

Bei Verwendung des externen Triggers können nur die D/A-Kanäle 0...3 verwendet werden.

Rückkehr-Verhalten in Abhängigkeit vom Triggermodus:

- Software-Start: sofort
- ext. Trigger ohne Time-Out: sofort
- ext. Trigger mit Time-Out: nach Ablauf des Time-Out oder nach Eintreffen des ext. Triggersignals.

Zum Start der synchronen Ausgabe in den timergesteuerten Betriebsarten „AOContinuous“ und „AOWraparound“ verwenden Sie bitte die Funktion ...*AOStartSynchronous*.

● Definitionen

VC: me4000AOSingleSimultaneous(unsigned int uiBoardNumber, unsigned int *puiChannelNumber, unsigned long ulCount, int iTriggerMode, int* piExtTriggerEnable, int* piExtTriggerEdge, unsigned long ulTimeOutSeconds, short* psValue);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

Wertefeld mit den Kanalnummern (0...15) jener Kanäle, die simultan ausgegeben werden sollen. In Verbindung mit externem Trigger sind nur die Kanäle 0...3 möglich.

<Count>

Anzahl der im Wertefeld <ChannelNumber> gelisteten Kanäle. Gilt auch für die Parameter <ExtTriggerEnable>, <ExtTriggerEdge> und <Value>.

<TriggerMode>

Trigger-Ereignis für die simultane Ausgabe der im Wertefeld <ChannelNumber> gelisteten Kanäle:

- ME4000_AO_TRIGGER_SOFTWARE
Simultane Ausgabe unmittelbar nach Aufruf dieser Funktion (Software-Start).
- ME4000_AO_TRIGGER_EXT_DIGITAL
Bereit zur Ausgabe nach Aufruf dieser Funktion. Die ausgewählten Kanäle werden durch externes Trigger-Signal simultan ausgegeben (siehe auch folgende Parameter).

<ExtTriggerEnable>

Wertefeld zur Freischaltung eines oder mehrerer Triggereingänge (DA_TRIG_0...3). Die Reihenfolge der Einträge korrespondiert mit der im Wertefeld <ChannelNumber>. Die Ausgabe erfolgt mit der ersten geeigneten Flanke an einem der hier freigeschalteten Triggereingänge.

- ME4000_AO_TRIGGER_EXT_DISABLE
Korrespondierender Triggereingang wird nicht berücksichtigt.
- ME4000_AO_TRIGGER_EXT_ENABLE
Korrespondierender Triggereingang wird ausgewertet.

Falls Sie im Parameter <TriggerMode> die Konstante ME4000_AO_TRIGGER_SOFTWARE gewählt haben übergeben Sie hier ME4000_POINTER_NOT_USED.

<ExtTriggerEdge>

Wertefeld zur Auswahl der Triggerflanke. Die Reihenfolge der Einträge korrespondiert mit der im Wertefeld <ChannelNumber>.

- ME4000_AO_TRIGGER_EXT_EDGE_RISING
Start durch steigende Flanke.
- ME4000_AO_TRIGGER_EXT_EDGE_FALLING
Start durch fallende Flanke.
- ME4000_AO_TRIGGER_EXT_EDGE_BOTH
Start durch fallende oder steigende Flanke.
- ME4000_VALUE_NOT_USED
Konstante, falls der Trigger-Eingang für den entsprechenden Kanal nicht freigeschaltet ist (<ExtTriggerEnable> = ME4000_AO_TRIGGER_EXT_DISABLE).

Falls Sie im Parameter <TriggerMode> die Konstante ME4000_AO_TRIGGER_SOFTWARE gewählt haben übergeben Sie hier ME4000_POINTER_NOT_USED.

<TimeoutSeconds>

Optional können Sie hier ein Zeitintervall in Sekunden angeben innerhalb dessen der erste Triggerimpuls eintreffen muß. Ansonsten wird die Operation abgebrochen. Falls Sie ohne ext. Trigger arbeiten oder kein Time-Out nutzen möchten, übergeben Sie hier die Konstante ME4000_VALUE_NOT_USED.

<Value>

Wertefeld mit den auszugebenden Werten (16 Bit); beachten Sie die Kennlinien auf Seite 22/23. Die Reihenfolge der Einträge korrespondiert mit der im Wertefeld <ChannelNumber>.

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOSTart** Beschreibung**

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Funktion zum Starten der Ausgabe in den Betriebsarten „AOContinuous“ und „AOWraparound“.

Falls Sie im Parameter <TriggerMode> der Funktion ...*AOConfig* eine externe Trigger-Option gewählt haben, wird die Ausgabe durch eine entsprechende Flanke am Triggereingang des Kanals gestartet. Zum Synchron-Start mehrerer Kanäle verwenden Sie bitte die Funktion ...*AOSTartSynchronous* (siehe Seite 74).

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 26, sowie in den Beispielprogrammen, die im ME-SDK enthalten sind.

 Hinweis:

Sollte bei Aufruf dieser Funktion ein betroffener D/A-Kanal bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

Rückkehr-Verhalten in Abhängigkeit vom Triggermodus:

- Software-Start: sofort
- ext. Trigger ohne Time-Out: sofort
- ext. Trigger mit Time-Out: nach Ablauf des Time-Out oder nach Eintreffen des ext. Triggersignals.

● Definitionen

VC: me4000AOSTart(unsigned int uiBoardNumber, unsigned int uiChannelNumber);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

Start der Ausgabe auf gewünschten Kanal. Übergeben Sie die Kanalnummer des entsprechenden D/A-Kanals 0...3.

•< Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOSTartSynchronous

Beschreibung

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Funktion zum synchronen Start mehrerer Kanäle in den Betriebsarten „AOContinuous“ und „AOWraparound“.

Vor Aufruf dieser Funktion muß die Sample-Rate für jeden Kanal, der in die synchrone Ausgabe einbezogen werden soll mit der Funktion ...AOConfig konfiguriert werden.

Für die Verwendung des externen Triggers gelten die Einstellungen in dieser Funktion. Entsprechende Einstellungen in der Funktion ...AOConfig werden ignoriert.

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 26, sowie in den Beispielprogrammen, die im ME-SDK enthalten sind.

Hinweis:

Sollte bei Aufruf dieser Funktion ein betroffener D/A-Kanal bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

Rückkehr-Verhalten in Abhängigkeit vom Triggermodus:

- Software-Start: sofort
- ext. Trigger ohne Time-Out: sofort
- ext. Trigger mit Time-Out: nach Ablauf des Time-Out oder nach Eintreffen des ext. Triggersignals.

Definitionen

VC: me4000AOStartSynchronous(unsigned int uiBoardNumber, unsigned int *puiChannelNumber, unsigned long ulCount, int iTriggerMode, int* piExtTriggerEnable, int* piExtTriggerEdge, unsigned long ulTimeOutSeconds);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter**<BoardNumber>**

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31).

<ChannelNumber>

Wertefeld mit den Kanalnummern (0...3) jener Kanäle, die synchron gestartet werden sollen.

<Count>

Anzahl der im Wertefeld <ChannelNumber> gelisteten Kanäle. Gilt auch für die Parameter <ExtTriggerEnable> und <ExtTriggerEdge>.

<TriggerMode>

Trigger-Ereignis für den synchronen Start der im Wertefeld <ChannelNumber> gelisteten Kanäle:

- ME4000_AO_TRIGGER_SOFTWARE
Synchron-Start unmittelbar nach Aufruf dieser Funktion.
- ME4000_AO_TRIGGER_EXT_DIGITAL
Bereit zum Starten der Ausgabe nach Aufruf dieser Funktion. Die ausgewählten Kanäle werden durch externes Triggersignal synchron gestartet (siehe auch folgende Parameter).

<ExtTriggerEnable>

Wertefeld zur Freischaltung eines oder mehrerer Triggereingänge (DA_TRIG_x). Die Reihenfolge der Einträge korrespondiert mit der im Wertefeld <ChannelNumber>. Die Ausgabe startet mit der ersten geeigneten Flanke an einem der hier freigeschalteten Triggereingänge.

- ME4000_AO_TRIGGER_EXT_DISABLE
Korrespondierender Triggereingang wird nicht berücksichtigt.
- ME4000_AO_TRIGGER_EXT_ENABLE
Korrespondierender Triggereingang wird ausgewertet.

Falls Sie im Parameter <TriggerMode> die Konstante ME4000_AO_TRIGGER_SOFTWARE gewählt haben übergeben Sie hier ME4000_POINTER_NOT_USED.

<ExtTriggerEdge>

Wertefeld zur Auswahl der Triggerflanke. Die Reihenfolge der Einträge korrespondiert mit der im Wertefeld <ChannelNumber>. Falls der Trigger-Eingang für den entsprechenden Kanal nicht freigeschaltet ist (<ExtTriggerEnable> = ME4000_AO_TRIGGER_EXT_DISABLE) tragen Sie ME4000_VALUE_NOT_USED im Wertefeld ein.

- ME4000_AO_TRIGGER_EXT_EDGE_RISING
Start durch steigende Flanke.
- ME4000_AO_TRIGGER_EXT_EDGE_FALLING
Start durch fallende Flanke.
- ME4000_AO_TRIGGER_EXT_EDGE_BOTH
Start durch fallende oder steigende Flanke.
- ME4000_VALUE_NOT_USED
Konstante, falls der Trigger-Eingang für den entsprechenden Kanal nicht freigeschaltet ist (<ExtTriggerEnable> = ME4000_AO_TRIGGER_EXT_DISABLE).

Falls Sie im Parameter <TriggerMode> die Konstante ME4000_AO_TRIGGER_SOFTWARE gewählt haben übergeben Sie hier ME4000_POINTER_NOT_USED.

<TimeOutSeconds>

Optional können Sie hier ein Zeitintervall in Sekunden angeben innerhalb dessen der erste Triggerimpuls eintreffen muß. Ansonsten wird die Operation abgebrochen. Falls Sie ohne ext. Trigger arbeiten oder kein Time-Out nutzen möchten, übergeben Sie hier die Konstante ME4000_VALUE_NOT_USED.

Der Wert für <TimeOutSeconds>, der in den Funktionen ...*AOContinuous* oder ...*AOWraparound* übergeben wurde, wird ignoriert.

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOSTop

Beschreibung

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Funktion zum Beenden der analogen Ausgabe des jeweiligen Kanals in den Betriebsarten „AOContinuous“ und „AOWraparound“. In der Betriebsart „AOContinuous“ wird die Ausgabe sofort und vollständig beendet. Am entsprechenden D/A-Kanal wird 0V ausgegeben und der Ringpuffer wird gelöscht. In der Betriebsart „AOWraparound“ können Sie mit dem Parameter <StopMode> selbst bestimmen, ob die Ausgabe sofort beendet und 0V ausgegeben werden soll oder mit dem letzten (bekannten) Wert im Datenpuffer beendet werden soll. Sofern die Betriebsart für diesen Kanal nicht gewechselt wurde kann die Ausgabe mit der Funktion ...AOSTart(*Synchronous*) jederzeit von vorne gestartet werden.

● Definitionen

VC: me4000AOSTop(unsigned int uiBoardNumber, unsigned int uiChannelNumber, int iStopMode);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

- Stop der Ausgabe des gewünschten Kanals. Übergeben Sie die entsprechende Kanalnummer 0...3.

<StopMode>

- ME4000_AO_STOP_MODE_LAST_VALUE
Ausgabe mit letztem Wert im Ringpuffer definiert stoppen (nur sinnvoll in Verbindung mit ...AOWraparound).
- ME4000_AO_STOP_MODE_IMMEDIATE
Ausgabe sofort beenden und 0V ausgeben.

< Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOVoltToDigit** Beschreibung**

ME-6000/6200	ME-6100/6300
✓ (nicht für „U-Plus“-Kanal)	✓ (nicht für „U-Plus“-Kanal)

Diese Funktion erlaubt Ihnen die einfache Umrechnung der auszugebenden Spannungswerte [V] in Digit-Werte [Digits]. Sie können die Spannung in Schritten von 0,3 mV = 1 Digit ausgeben. Die Verwendung dieser Funktion ist optional.

Für einen Ausgangsspannungsbereich von ±10 V gilt folgende Formel (Übertragungskennlinie des D/A-Wandlers siehe Abb. 10 auf Seite 22):

$$U[\text{Digits}] = \frac{32768}{10V} \cdot U[V]$$

Hinweis: Für den „U-Plus“-Kanal der ME-6200/6300 kann die Funktion nicht verwendet werden (siehe Kennlinien auf Seite 22/23).

● Definitionen

VC: me4000AOVoltToDigit(double dVolt, short* psDigit);
 LV: me4000LV_... (siehe me4000LV.h)
 VB: me4000VB_... (siehe me4000.bas)
 VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter**<Volt>**

Auszugebender Spannungswert in Volt.

<Digit>

Zeiger auf auszugebenden Digit-Wert.

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOWaveGen

Beschreibung

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Diese Funktion bietet einen einfach zu programmierenden, virtuellen Funktionsgenerator (Rechteck, Sinus, Dreieck,...). Die gesamte Konfigurierung des betreffenden Kanals übernimmt diese Funktion. Die Ausgabe wird nach Aufruf dieser Funktion automatisch gestartet und mit der Funktion *me4000AOSTop* beendet. Das Signal wird in Abhängigkeit von der gewählten Frequenz und Signalform (Rechteck max. 250kHz, andere max. 100kHz), stets mit der max. Anzahl an Stützpunkten (minimal 5) ausgegeben. Ausnahme: Ein Rechtecksignal wird stets mit 2 Stützpunkten pro Periode ausgegeben. Siehe auch Kap. „Betriebsart „AOWraparound““ auf Seite 33.

Siehe Programmierbeispiele, die im ME-SDK enthalten sind.

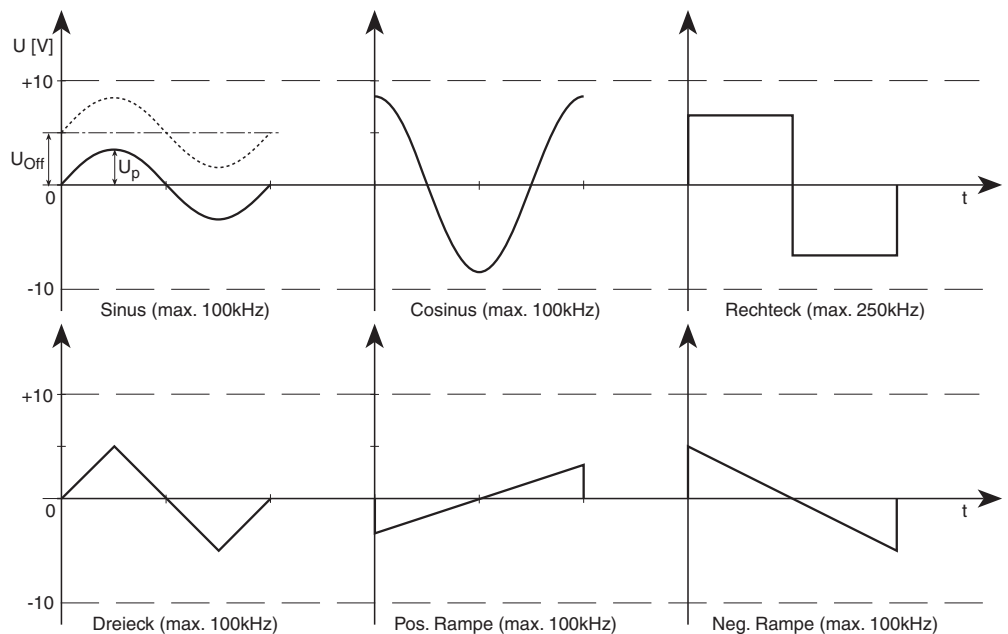


Abb. 24: Offset, Amplitude, Signalformen

☞ Hinweis:**Die Ausgabe mit dieser Funktion läuft stets auf Firmware-Ebene, d. h. der Host-Rechner wird nicht belastet!**

Die Verwendung des externen Triggers ist mit dieser Funktion nicht möglich, verwenden sie dazu die Funktion *...AOContinuous* oder *...AOWraparound*.

Falls in der Summe der Parameter die hardwaretechnischen Grenzen der Karte überschritten werden, gibt der Treiber eine Fehlermeldung aus. Sollten benötigte Hardware-Ressourcen bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

● Definitionen

VC: me4000AOWaveGen(unsigned int uiBoardNumber, unsigned int uiChannelNumber, int iShape, double dAmplitude, double dOffset, double dFrequency);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter**<BoardNumber>**

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

D/A-Kanal 0...3

<Shape>

Signalform; mögliche Werte:

- ME4000_AO_SHAPE_RECTANGLE Rechtecksignal
- ME4000_AO_SHAPE_TRIANGLE Dreiecksignal
- ME4000_AO_SHAPE_SINUS Sinussignal
- ME4000_AO_SHAPE_COSINUS Cosinussignal
- ME4000_AO_SHAPE_POS_RAMP Positive Rampe
- ME4000_AO_SHAPE_NEG_RAMP Negative Rampe

<Amplitude>

Signal-Amplitude U_p [V] als dezimalen Spannungswert; Wertebereich: 0...+10,00V

<Offset>

Offset-Spannung U_{Off} [V] mit der das Signal in positive oder negative Richtung verschoben werden kann; Wertebereich: -10,00V...+10,00V

<Frequency>

Frequenz in [Hz] des periodisch auszugebenden Signals; Wertebereich: 0...100000Hz (Rechteck bis 250000Hz)

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000AOWraparound** Beschreibung**

ME-6000/6200	ME-6100/6300
–	✓ (Kanal 0...3)

Mit dieser Funktion wird der betreffende Kanal für die Betriebsart „AOWraparound“ vorbereitet. Sie können in diesem Modus beliebige periodische Signale auf den Kanälen 0...3 ausgeben. Vor Beginn der Ausgabe müssen die einzelnen Kanäle einmalig beladen werden. Erzeugen sie für jeden Kanal einen Datenpuffer definierter Größe mit den auszugebenden Werten.

Der D/A-Timer gibt ein festes Zeitraster (Sample-Rate) für die Ausgabe vor (siehe ...*AOConfig*).

Gestartet wird die Ausgabe stets mit der Funktion ...*AOStart(Synchronous)* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe ...*AOConfig*).

Mit der Funktion ...*AOStop* können Sie die Ausgabe wahlweise sofort beenden oder definiert „anhalten“, d. h. die Ausgabe wird mit dem letzten Wert im FIFO und somit einem bekannten Spannungswert gestoppt. Sofern zwischenzeitlich die Betriebsart für diesen Kanal nicht gewechselt wurde, kann die Ausgabe mit der Funktion ...*AOStart(Synchronous)* jederzeit von vorne gestartet werden. Mit der Funktion ...*AOReset* wird im Vergleich zu ...*AOStop* auch das D/A-FIFO gelöscht und damit die Ausgabe **vollständig** beendet.

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 21, sowie in den Programmbeispielen, die im ME-Software-Developer-Kit (ME-SDK) enthalten sind.

 Hinweis:

Sofern die Größe des Datenpuffers 8192 Werte nicht übersteigt und die Ausgabe „unendlich“ erfolgt, läuft die Ausgabe auf Firmware-Ebene, d. h. der Host-Rechner wird nicht belastet!

Sollten benötigte Hardware-Ressourcen bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

● Definitionen

Typdefinition für ME4000_P_AO_TERMINATE_PROC:

```
typedef void (_stdcall *
ME4000_P_AO_TERMINATE_PROC)
(void* pTerminateContext);
```

VC: me4000AOWraparound(unsigned int uiBoardNumber, unsigned int uiChannelNumber, short* psBuffer, unsigned long ulDataCount, unsigned long ulLoops, int iExecutionMode, ME4000_P_AO_TERMINATE_PROC pTerminateProc, void* pTerminateContext, unsigned long ulTimeOutSeconds);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<ChannelNumber>

D/A-Kanal 0...3

<Buffer>

Zeiger auf benutzerallokierten Datenpuffer mit den auszugebenden Spannungswerten.

<DataCount>

Anzahl der Werte im Datenpuffer <Buffer>.

<Loops>

Dieser Parameter gibt an, wie oft die Werte im Datenpuffer „Buffer“ ausgegeben werden sollen. Für „unendlich“ übergeben Sie die Konstante: ME4000_AO_WRAPAROUND_INFINITE.

<ExecutionMode>

Ausführungsmodus für diese Funktion wählen:

- ME4000_AO_WRAPAROUND_BLOCKING:
Das Programm ist blockiert bis die Ausgabe beendet ist. In Verbindung mit einer „unendlichen“ Ausgabe (siehe Parameter <Loops>) ist diese Konstante nicht möglich.
- ME4000_AO_WRAPAROUND_ASYNCRONOUS:
Die Ausgabe erfolgt im Hintergrund (asynchron). Der Programmfluß wird nicht unterbrochen.

<TerminateProc>**LV, VB, VEE**

„Terminate“-Funktion, die am Ende der Ausgabe aufgerufen wird. Falls diese Funktionalität nicht erwünscht ist, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<TerminateContext>**LV, VB, VEE**

Benutzerdefinierter Zeiger, der an die „Terminate“-Funktion weitergegeben wird. Falls die „Terminate“-Funktion nicht genutzt wird, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<TimeoutSeconds>

Optional können Sie hier ein Zeitintervall in Sekunden angeben innerhalb dessen der erste Triggerimpuls eintreffen muß. Ansonsten wird die Operation abgebrochen. Falls Sie ohne ext. Trigger arbeiten oder kein Time-Out nutzen möchten, übergeben Sie hier die Konstante ME4000_VALUE_NOT_USED.

Bei Synchron-Start gilt der Time-Out-Wert in der Funktion *...AOSTartSynchronous*.

< Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

5.3.4 Digitale Ein-/Ausgabe

Hinweis: Die ME-6000/6100 PCI ab Hardware-Version 2.6 sowie alle CompactPCI-Modelle sind mit zwei 8 Bit breiten Digital-I/O-Ports (A, B) ausgestattet.

me4000DIOConfig

Beschreibung

ME-6000/6200	ME-6100/6300
✓ (siehe Hinweis S. 84)	✓ (siehe Hinweis S. 84)

Diese Funktion dient der Richtungsumschaltung der digitalen Ports (A, B). Die Richtung kann für jeden der 8 Bit breiten Ports unabhängig konfiguriert werden.

Diese Funktion muß für jeden Port getrennt aufgerufen werden. Ein als Ausgang konfigurierter Port kann auch rückgelesen werden!

● Definitionen

VC: me4000DIOConfig(unsigned int uiBoardNumber, unsigned int uiPortNumber, int iPortDirection);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<PortNumber>

Port auswählen:

- ME4000_DIO_PORT_A Port A
- ME4000_DIO_PORT_B Port B

<PortDirection>

Port-Richtung für Standard-Ein/Ausgabe:

- ME4000_DIO_PORT_INPUT
Eingangsport
- ME4000_DIO_PORT_OUTPUT
Ausgangsport

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000DIOGetBit

Beschreibung

ME-6000/6200	ME-6100/6300
✓ (siehe Hinweis S. 84)	✓ (siehe Hinweis S. 84)

Liefert den Zustand der selektierten Bits zurück.

Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher die Funktion ...*DIOConfig* aufgerufen werden.

● Definitionen

VC: me4000DIOGetBit(unsigned int uiBoardNumber, unsigned int uiPortNumber, unsigned int uiBitNumber, int *piBitValue);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<PortNumber>

Port auswählen:

- ME4000_DIO_PORT_A Port A
- ME4000_DIO_PORT_B Port B

<BitNumber>

Nummer der Eingangsleitung, die abgefragt werden soll; möglich sind: 0...7

<BitValue>

Zeiger auf einen Integerwert, der dem Leitungszustand entsprechend gelesen wird:

„0“: Leitung führt Low-Pegel

„1“: Leitung führt High-Pegel

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000DIOGetByte

Beschreibung

ME-6000/6200	ME-6100/6300
✓ (siehe Hinweis S. 84)	✓ (siehe Hinweis S. 84)

Liest ein Byte vom spezifizierten Port. Ausgangsports können auch rückgelesen werden!

Hinweis!

Zur Konfiguration des Ports muß vorher die Funktion ...*DIOConfig* aufgerufen werden.

● Definitionen

VC: me4000DIOGetByte(unsigned int uiBoardNumber, unsigned int uiPortNumber, unsigned char *pucByteValue);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<PortNumber>

Port auswählen:

- ME4000_DIO_PORT_A Port A
- ME4000_DIO_PORT_B Port B

<ByteValue>

Zeiger auf einen „unsigned char“ Wert, der das gelesene Byte aufnimmt.

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000DIOResetAll

Beschreibung

ME-6000/6200	ME-6100/6300
✓ (siehe Hinweis S. 84)	✓ (siehe Hinweis S. 84)

Die Richtung der digitalen Ports wird auf Eingang gesetzt.

● Definitionen

VC: int me4000DIOResetAll (unsigned int uiBoardNumber);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

◀ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000DIOSetBit

Beschreibung

ME-6000/6200	ME-6100/6300
✓ (siehe Hinweis S. 84)	✓ (siehe Hinweis S. 84)

Setzt eine digitale Ausgangsleitung in den gewünschten Zustand.

Hinweis!

Zur Konfiguration des Ports muß vorher die Funktion ...*DIOConfig* aufgerufen werden.

● Definitionen

VC: me4000DIOSetBit(unsigned int uiBoardNumber, unsigned int uiPortNumber, unsigned int uiBitNumber, int iBitValue);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<PortNumber>

Port auswählen:

- ME4000_DIO_PORT_A Port A
- ME4000_DIO_PORT_B Port B

<BitNumber>

Nummer der Ausgangsleitung, die gesetzt werden soll; möglich sind: 0...7

<BitValue>

Mögliche Werte sind:

„0“: Bit wird auf Low-Pegel gesetzt

„1“: Bit wird auf High-Pegel gesetzt

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

me4000DIOSetByte

Beschreibung

ME-6000/6200	ME-6100/6300
✓ (siehe Hinweis S. 84)	✓ (siehe Hinweis S. 84)

Schreibt ein Byte an einen als Ausgang konfigurierten digitalen Port.

Hinweis!

Zur Konfiguration des Ports muß vorher die Funktion ...*DIOConfig* aufgerufen werden.

● Definitionen

VC: me4000DIOSetByte(unsigned int uiBoardNumber, unsigned int uiPortNumber, unsigned char ucByteValue);

LV: me4000LV_... (siehe me4000LV.h)

VB: me4000VB_... (siehe me4000.bas)

VEE: me4000VEE_... (siehe me4000VEE.h)

→ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-46xx oder ME-6x00 (0...31)

<PortNumber>

Port auswählen:

- ME4000_DIO_PORT_A Port A
- ME4000_DIO_PORT_B Port B

<ByteValue>

Ausgabewert; mögliche Werte sind: 0...255 (00Hex...FFHex).

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 0 (ME4000_NO_ERROR) zurückgegeben. Im Fehlerfall wird ein Fehlercode ungleich 0 zurückgegeben. Die genaue Fehlerursache kann mit den Funktionen zum Fehler-Handling ermittelt werden.

Anhang

A Spezifikationen

(Umgebungstemperatur 25°C)

PC-Interface

Standard-PCI- bzw. CompactPCI-Bus (32 Bit, 33MHz, 5V);
 PCI Local Bus Spezifikation Version 2.1;
 CompactPCI Spezifikation PICMG 2.0 R3.0;
 Automatische Ressourcen-Zuweisung (Plug&Play)

Spannungsausgänge

(für „U-Plus“-Kanal gelten zum Teil abweichende Spezifikationen
 – siehe separater Abschnitt)

Anzahl der Kanäle	je nach Modell 4, 5, 8, 9 oder 16
D/A-Wandler	1 serieller Wandler (500kHz) pro Kanal
Auflösung	16 Bit
Ausgangsbereich	±10V
Ausgangsstrom	Ohne externe Spannungsversorgung: je nach Anzahl der bestückten bzw. genutzten Kanäle:

Kanäle	I _{max} pro Kanal
4	15mA
8	15mA
12	10mA
16	3mA

Mit ext. Spannungsversorgung (±15V) nur in Verbindung mit den Optionen „Insel-Kanäle“ und „High Current“: max. ±15mA pro Kanal

Ext. Spannungsversorgung	±15 V (optional); Strom pro Kanal: 7mA + Laststrom (max. ±15mA)
Betriebsarten	„AOSingle“: Einzelwert-Ausgabe; „AOSimultaneous“: Einzelwertausgabe auf mehrere Kanäle simultan
Gesamtgenauigkeit:	
„Mit galvanischer Trennung“	max. ±20mV
„Mit Insel-Kanälen“	max. ±10mV
Einschwingzeit (DAC)	max. 2µs bei Vollausschlag (-10V → +10V)

Ausgangsstufe „U-Plus“ (Kanal-Nr. 8)

Ausgangssignal	Uout_8
Spannungsbereich	0...50V
Ausgangsstrom	max. 20mA
Offset-Fehler	typ. $\pm 5\text{mV}$; max. $\pm 20\text{mV}$
Verstärkungsfehler	$\pm 0,16\%$
Einschwingzeit	max. $25\mu\text{s}$ bei Vollausschlag (0 \rightarrow 50V) mit 20mA Last

Timergesteuerte Ausgabe (ME-6100/6300, Kanal 0...3)

Kanäle	0...3 (voneinander unabhängig)
D/A-FIFOs	8k Werte pro Kanal
Sample-Rate	max. 500kS/s
D/A-Timer	von $2\mu\text{s}$ bis 130s in Schritten von $30,30\overline{ns}$ programmierbar
Betriebsarten	„AOContinuous“: kontinuierliche Ausgabe; „AOWraparound“: periodische Ausgabe
Triggermodi	Software-Start, ext. Digital-Trigger, Synchron-Start (Software/extern)

Externer Trigger (Kanal 0...3)

Ext Triggerflanken	steigend, fallend, beide
Spannungspegel	typ. 5V
Eingangsstrom I_F	$7,5\text{mA} \leq I_F \leq 10\text{mA}$
Massebezug	Masse (GND_x)
Verzögerungszeit	max. 80ns

Galvanische Trennung, Insel-Kanäle (optional)

Überspannungsschutz	max. 500V
---------------------	-----------

Digital-I/Os

Ports	2 x 8 Bit
Massebezug	PC-Masse (PC_GND)
Port-Typ	bidirektionale TTL-Ports
Ausgangspegel	U_{OL} : max. 0,5V bei 24mA U_{OH} : min. 2,4V bei -24mA
Eingangspegel	U_{IL} : max. 0,8V bei $V_{CC} = 5\text{V}$ U_{IH} : min. 2V bei $V_{CC} = 5\text{V}$
Eingangsstrom:	$\pm 1\mu\text{A}$

Allgemeine Daten

Stromverbrauch bei +5V (16 D/A-Kanäle; ohne ext. Last):

„Mit galvanischer Trennung“ max. 3,6A

„Mit Insel-Kanälen“ max. 1,2A

Belastbarkeit VCC_OUT max. 200mA

Kartenabmessungen PCI 174 mm x 99 mm

(ohne Slotblech und Stecker)

Kartenabmessungen cPCI 3 HE CompactPCI-Karte

Anschlüsse 78polige Sub-D-Buchse (ST1);

20poliger Stiftstecker (ST2)

Betriebstemperatur 0...70°C

Lagertemperatur -40...100 °C

Luftfeuchtigkeit 20...55% (nicht kondensierend)

CE-Zertifizierung

EG-Richtlinie 89/336/EMC

Emission EN 55022

Störfestigkeit EN 50082-2

B Anschlußbelegungen

Legende zu den Anschlußbelegungen:



Achtung: In den Optionen „High Current“ und „Insel-Kanäle“ sind die Pins $-U_x$ und $+U_x$ Eingänge für die ext. $\pm 15V$ Spannungsversorgung. In allen anderen Fällen dürfen diese Pins nicht beschaltet werden, da sie $\pm 15V$ führen. **Die Hardware würde irreversibel beschädigt werden!**

U_{out_x}	Analoge Ausgangskanäle
$+U_x$	+15V Spannungsversorgung; nur für die Optionen „High Current“ (HC) und „Insel-Kanäle“.
$-U_x$	-15V Spannungsversorgung; nur für die Optionen „High Current“ (HC) und „Insel-Kanäle“.
DA_TRIG_x	Digitaler Triggereingang für die D/A-Kanäle 0...3.
DIO_Ax	Digital-Port A
DIO_Bx	Digital-Port B
GND_x	Gemeinsame Masse aller D/A-Kanäle von der PC-Masse entkoppelt. Bei Modellen mit der Option „Inselkanäle“ sind zusätzlich die Massen der einzelnen D/A-Kanäle voneinander getrennt (Insel-Masse).
PC_GND	PC-Masse für Digital-I/O-Teil.
VCC_OUT	V_{CC} -Ausgang (+5V vom PC) bis max. 200mA belastbar
$+U_{EXT}$	ME-6200/6300 optional: Pins zur Einspeisung der positiven Versorgung der Ausgangsstufe des „U-Plus“-Kanals (U_{out_8}).
$-U_{EXT}$	ME-6200/6300 optional: Pins zur Einspeisung der negativen Versorgung der Ausgangsstufe des „U-Plus“-Kanals (U_{out_8}).
n.c.	Pin ohne Verbindung

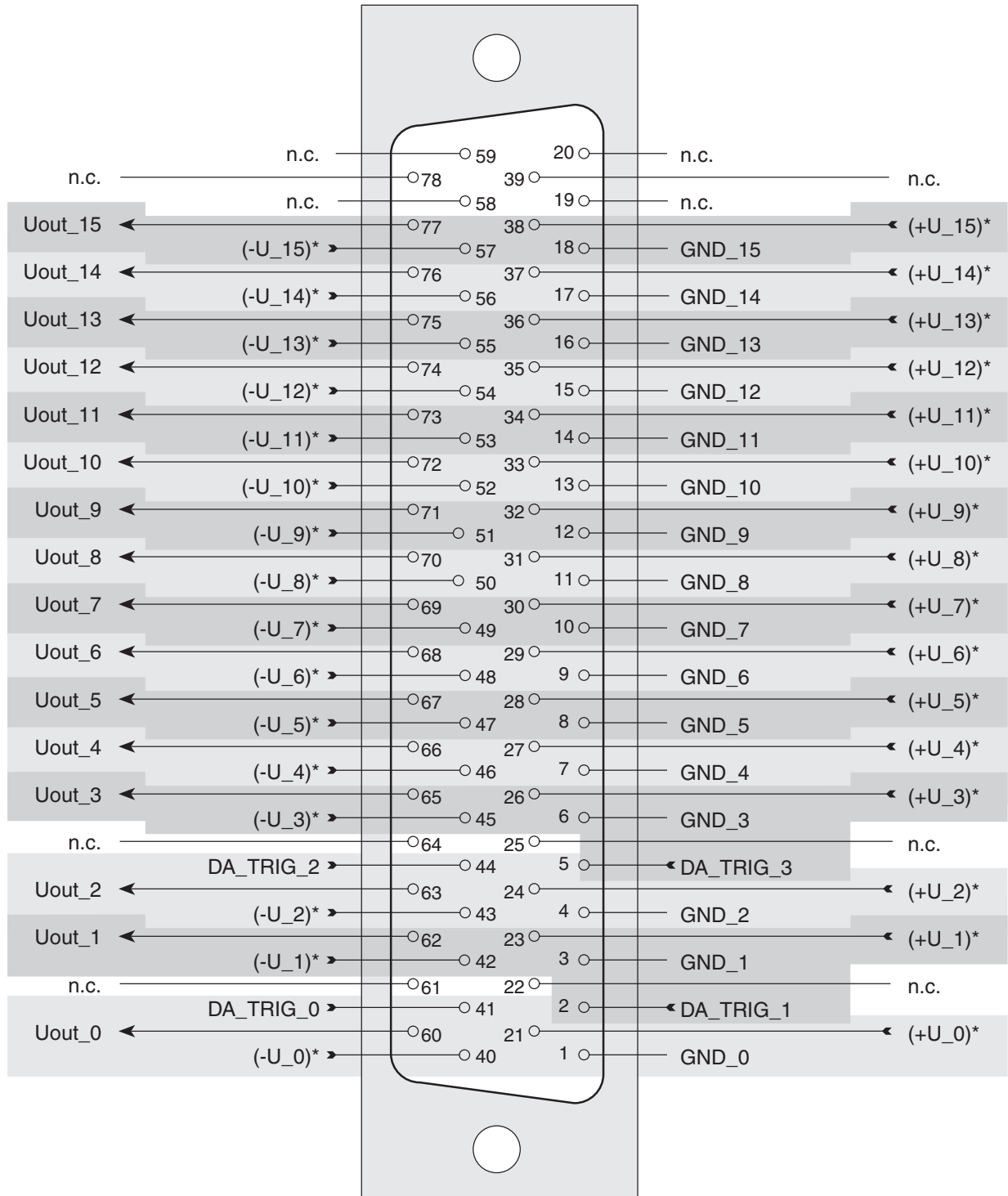
B1 78pol. Sub-D-Buchse (ST1)**B1.1 ME-6000/6100**

Abb. 25: Belegung der 78poligen Sub-D-Buchse

*** Beachten Sie den Warnhinweis auf Seite 94!**

B1.2 ME-6200/6300

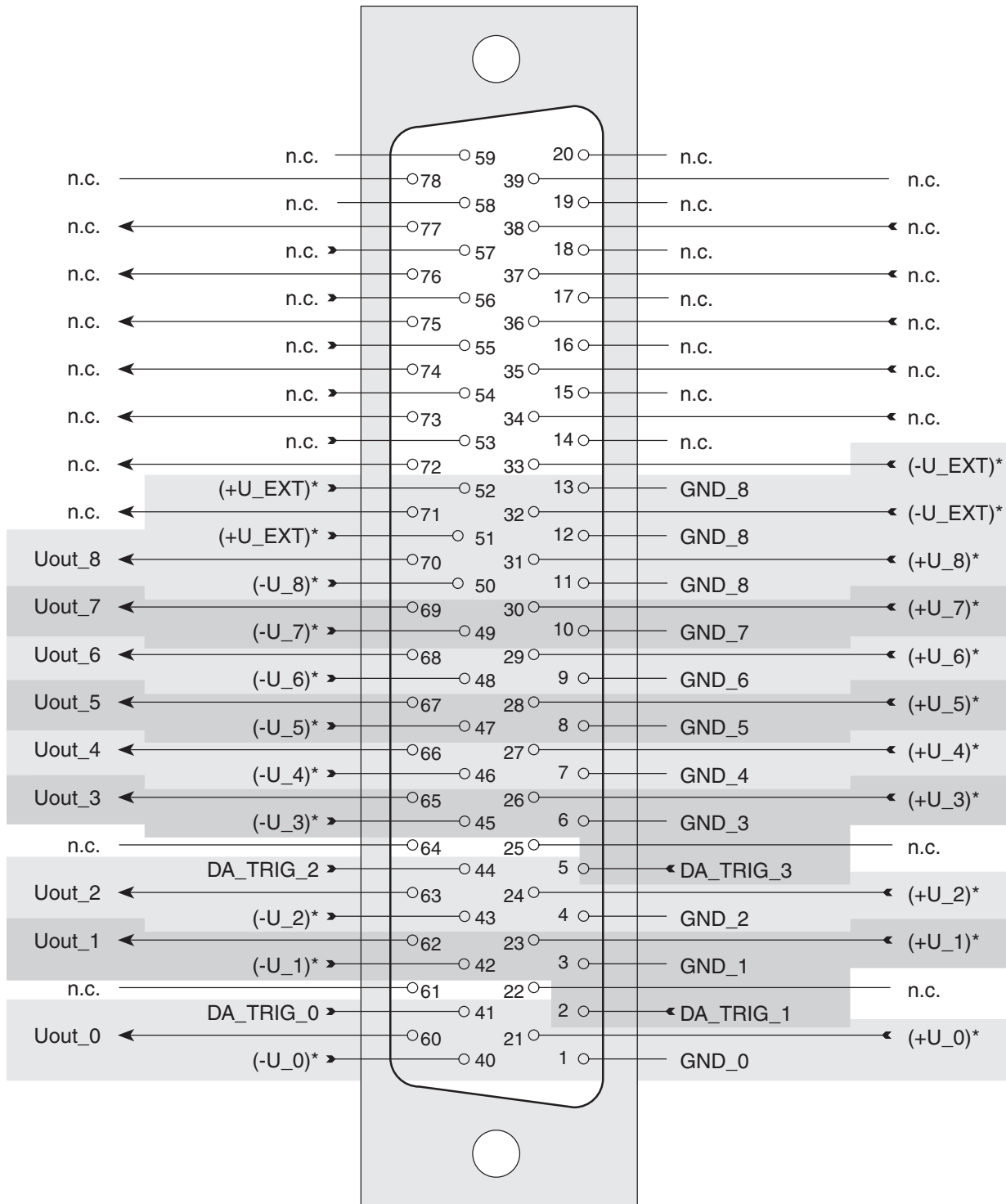


Abb. 26: Belegung der 78poligen Sub-D-Buchse

*** Beachten Sie den Warnhinweis auf Seite 94!**

B2 Zusatzstecker (ST2)

Adapterkabel (ME-AK-D25F/S (cPCI)) von 20pol. Stiftstecker auf Slotblech mit 25poliger Sub-D-Buchse (im Lieferumfang der Karte).

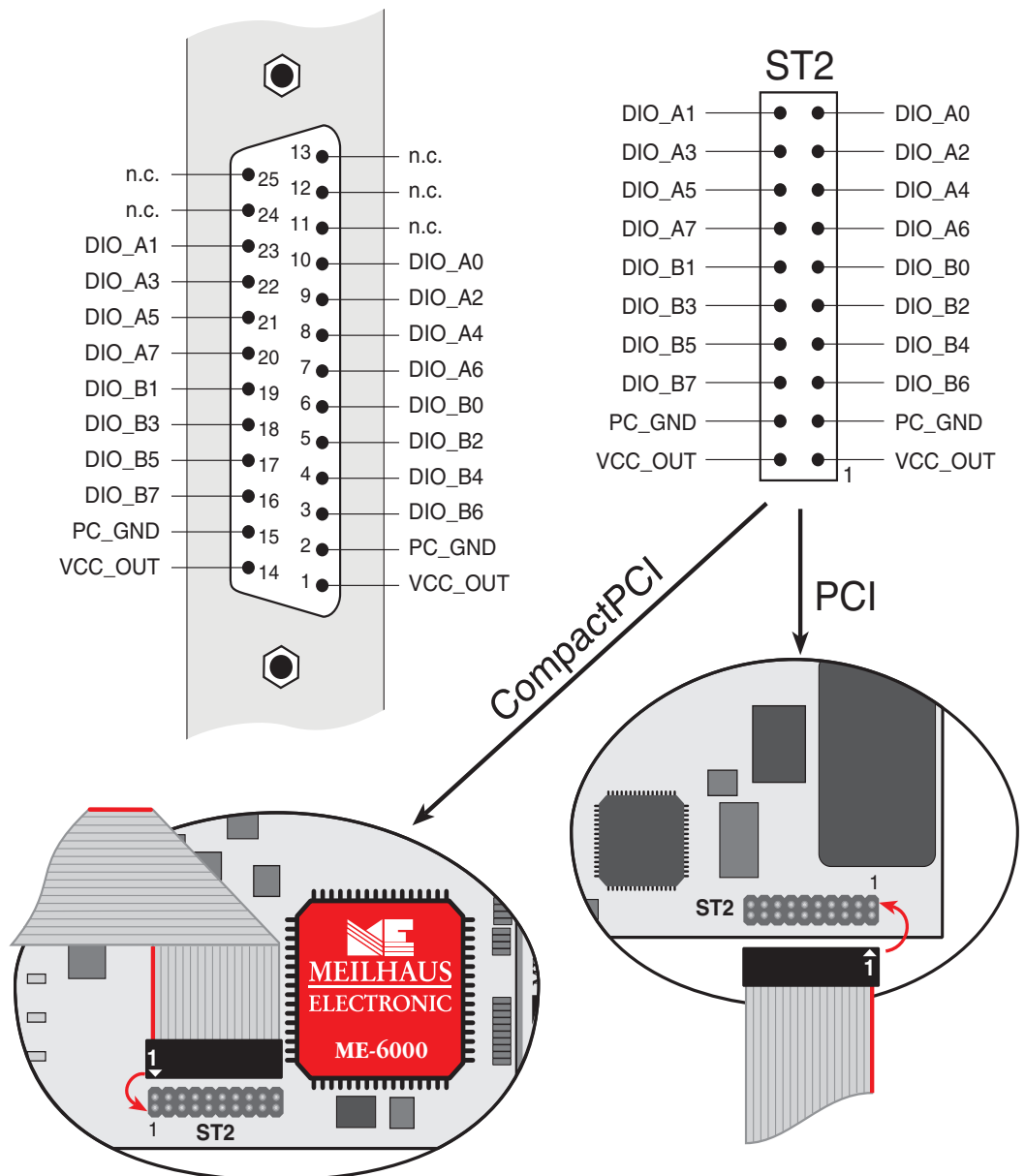


Abb. 27: Zusatzstecker ST2 der ME-6000 Serie (Draufsicht)

Beachten Sie beim Anschließen des Adapterkabels, daß Sie Pin 1 des Flachbandkabels (rot markierte Leitung) wie oben gezeigt auf den Stiftsteckers ST2 stecken.

C Zubehör

Wir empfehlen die Verwendung qualitativ hochwertiger Anschlußkabel mit getrennter Schirmung pro Kanal. Im Lieferumfang befindet sich das Spezial-Anschlußkabel ME-AK-D78/6000M-OE (Länge: 1 m).

ME-AK-D78/6000M-OE

Spezial-Anschlußkabel von 78poligem Sub-D-Stecker auf 16 einzeln geschirmte Leitungen mit offenem Ende.

Beachten Sie den Warnhinweis auf Seite 94!

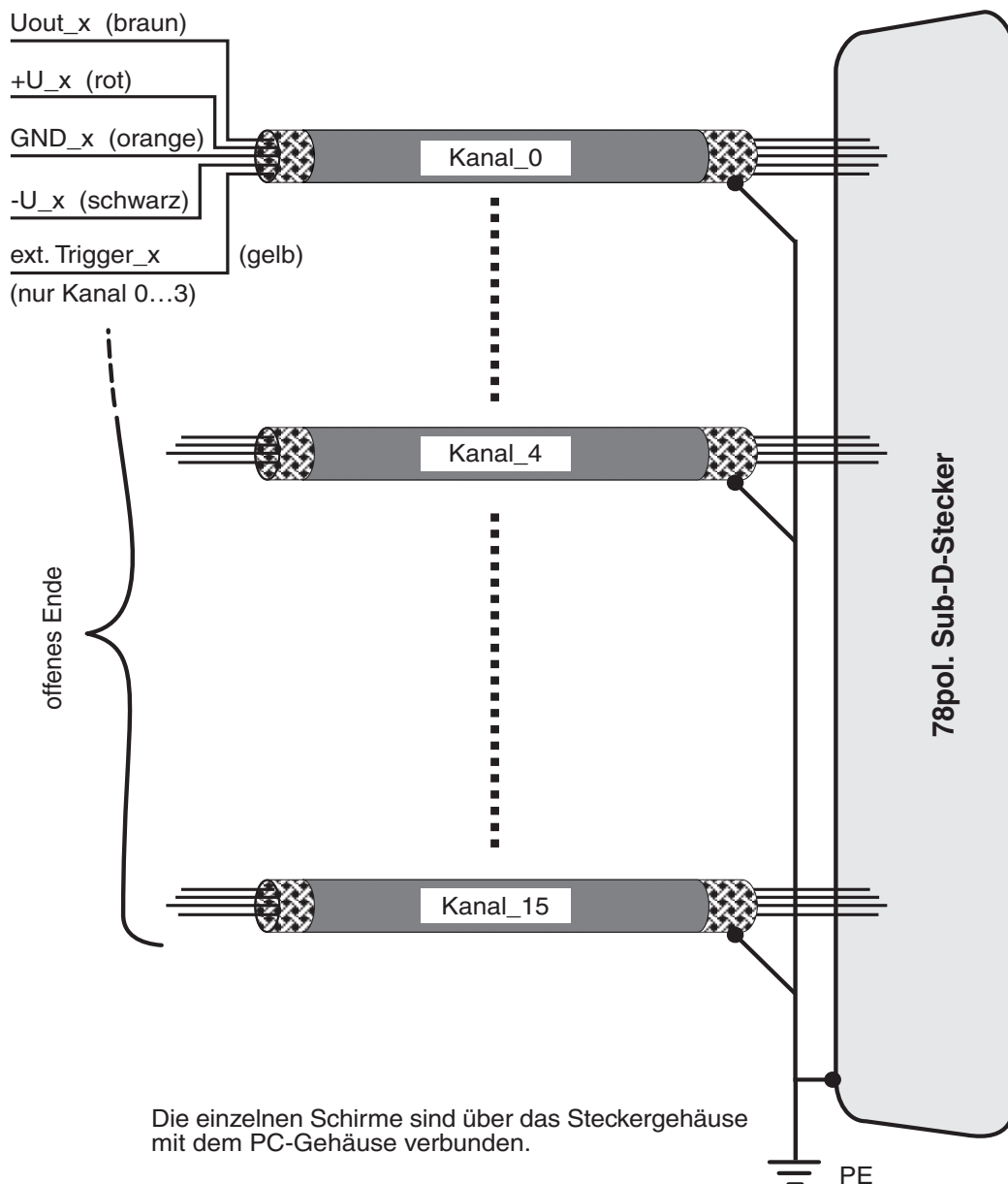


Abb. 28: Spezialkabel für ME-6000-Serie

ME-63Xtend-Serie

Externe Relais- und Digital-I/O-Karten (für DIN-Hutschienen-Montage geeignet). Anschluss über ST2 mit Zusatz-Slotblech ME AK-D25F/S und Spezial-Anschlusskabel ME AK-D2578/4000.

ME-UB-Serie

Desktop-Relais- und Digital-I/O-Boxen. Anschluss über ST2 mit Zusatz-Slotblech ME AK-D25F/S und Spezial-Anschlusskabel ME AK-D2515/4000.

Weitere Informationen über Zubehör entnehmen Sie bitte dem aktuellen Meilhaus Katalog.

D Technische Fragen

D1 Fax-Hotline

Sollten Sie technische Fragen oder Probleme haben, die auf die Karte zurückzuführen sind, dann schicken Sie bitte eine ausführliche Problembeschreibung an unsere Hotline:

Fax-Hotline: (++49) (0)89 - 89 01 66-28

eMail: support@meilhaus.de

D2 Serviceadresse

Wir hoffen, daß Sie diesen Teil des Handbuches nie benötigen werden. Sollte bei Ihrer Karte jedoch ein technischer Defekt auftreten, wenden Sie sich bitte an:

Meilhaus Electronic GmbH

Abteilung Reparaturen

Fischerstraße 2

D-82178 Puchheim

Falls Sie Ihre Karte zur Reparatur an uns zurücksenden wollen, legen Sie bitte unbedingt eine ausführliche Fehlerbeschreibung bei, inkl. Angaben zu Ihrem Rechner/System und verwendeter Software!

D3 Treiber-Update

Unter www.meilhaus.de stehen Ihnen stets die aktuellen Treiber für Meilhaus-Karten sowie unsere Handbücher im PDF-Format zur Verfügung.

E Konstanten-Definitionen

Hinweis: Die folgenden Konstantendefinitionen gelten für Windows. Bitte beachten Sie auch die aktuelle Definitionsdatei (me4000defs.h) im Meilhaus Developer Kit (ME-SDK). Der Linux-Treiber verwendet eigenständige Konstantendefinitionen (siehe Linux-Treiber).

Konstante	Wert
Allgemein	
ME4000_MAX_DEVICES	32
ME4000_VALUE_NOT_USED	0
ME4000_POINTER_NOT_USED	NULL
ME4000_NO_ERROR	0x00000000
Fehlermeldungen	
ME4000_ERROR_DEFAULT_PROC_ENABLE	0x00010101
ME4000_ERROR_DEFAULT_PROC_DISABLE	0x00010102
Analoge Ausgabe	
ME4000_AO_TRIGGER_SOFTWARE	0x00030101
ME4000_AO_TRIGGER_EXT_DIGITAL	0x00030103
ME4000_AO_TRIGGER_EXT_EDGE_RISING	0x00030201
ME4000_AO_TRIGGER_EXT_EDGE_FALLING	0x00030202
ME4000_AO_TRIGGER_EXT_EDGE_BOTH	0x00030203
ME4000_AO_WRAPAROUND_BLOCKING	0x00030301
ME4000_AO_WRAPAROUND_ASYNCHRONOUS	0x00030302
ME4000_AO_WRAPAROUND_INFINITE	0x00
ME4000_AO_APPEND_NEW_VALUES_BLOCKING	0x00030401
ME4000_AO_APPEND_NEW_VALUES_NON_BLOCKING	0x00030402
ME4000_AO_WAIT_IDLE	0x00030501
ME4000_AO_WAIT_NONE	0x00030502
ME4000_AO_STATUS_IDLE	0x00030601
ME4000_AO_STATUS_BUSY	0x00030602
ME4000_AO_STOP_MODE_LAST_VALUE	0x00030701
ME4000_AO_STOP_MODE_IMMEDIATE	0x00030702
ME4000_AO_SHAPE_RECTANGLE	0x00030801
ME4000_AO_SHAPE_TRIANGLE	0x00030802
ME4000_AO_SHAPE_SINUS	0x00030803
ME4000_AO_SHAPE_COSINUS	0x00030804

Tabelle 11: Konstanten-Definition

Konstante	Wert
ME4000_AO_SHAPE_POS_RAMP	0x00030805
ME4000_AO_SHAPE_NEG_RAMP	0x00030806
ME4000_AO_TRIGGER_EXT_DISABLE	0x00030901
ME4000_AO_TRIGGER_EXT_ENABLE	0x00030902
<i>Digitale Ein-/Ausgabe</i>	
ME4000_DIO_PORT_A	0
ME4000_DIO_PORT_B	1
ME4000_DIO_PORT_C	2
ME4000_DIO_PORT_D	3
ME4000_DIO_PORT_INPUT	0x00040201
ME4000_DIO_PORT_OUTPUT	0x00040202

Tabelle 11: Konstanten-Definition

F Index

Funktionsreferenz

me4000AOAppendNewValues 61
 me4000AOConfig 63
 me4000AOContinuous 65
 me4000AOGetStatus 67
 me4000AOReset 68
 me4000AOSingle 69
 me4000AOSingleSimultaneous 71
 me4000AOSTart 73
 me4000AOSTartSynchronous 74
 me4000AOSTop 77
 me4000AOWaveGen 79
 me4000AOWrapAround 81
 me4000DIOConfig 84
 me4000DIOGetBit 85
 me4000DIOGetByte 86
 me4000DIOResetAll 87
 me4000DIOSetBit 88
 me4000ErrorGetLastMessage 50
 me4000ErrorGetMessage 49
 me4000ErrorSetDefaultProc 51
 me4000ErrorSetUserProc 52
 me4000FrequencyToTicks 53
 me4000GetBoardVersion 55
 me4000GetDLLVersion 57
 me4000GetDriverVersion 57
 me4000GetSerialNumber 58
 me4000TimeToTicks 59
 me4000VoltToDigit 78

A

Adapterkabel 97
 Agilent VEE 40
 Allgemeine Funktionen
 me4000FrequencyToTicks 53
 me4000GetBoardVersion 55
 me4000GetDLLVersion 57
 me4000GetDriverVersion 57

me4000GetSerialNumber 58

me4000TimeToTicks 59

Analoge Ausgabe

me4000AOAppendNewValues 61
 me4000AOConfig 63
 me4000AOContinuous 65
 me4000AOGetStatus 67
 me4000AOReset 68
 me4000AOSingle 69
 me4000AOSingleSimultaneous 71
 me4000AOSTart 73
 me4000AOSTartSynchronous 74
 me4000AOSTop 77
 me4000AOVoltToDigit 78
 me4000AOWaveGen 79
 me4000AOWrapAround 81

Anhang 91

Anschlußbelegungen 94

API-DLL 38

API-Funktionen 47

B

Beispielprogramme 38

Beschreibung der API-Funktionen
 47

Betriebsarten

AOContinuous 26, 29

AOSimultaneous 25

AOSingle 24

AOWrapAround 26, 33

Blockschaltbilder 13

D

D/A-Teil

Externer Trigger 18

Hardware-Beschreibung 16

Programmierung 21

Delphi 40

- Digitale Ein-/Ausgabe
 - me4000DIOConfig 84
 - me4000DIOGetBit 85
 - me4000DIOGetByte 86
 - me4000DIOResetAll 87
 - me4000DIOSetBit 88
 - me4000DIOSetByte 89
- Digital-I/O
 - Beschaltung 20
 - Hardware-Beschreibung 20
 - Programmierung 37
- Digital-I/O-Teil 20
- E**
 - Einführung 5
- F**
 - Fehler-Behandlung
 - me4000ErrorGetLastMessage 50
 - me4000ErrorGetMessage 49
 - me4000ErrorSetDefaultProc 51
 - me4000ErrorSetUserProc 52
 - Funktionsreferenz 45
- G**
 - Galvanische Trennung 16
- H**
 - Hardware-Beschreibung 13
 - Galvanische Trennung 16
 - High Current Option 18
 - Insel-Kanäle 17
 - Hochsprachenprogrammierung 38
- I**
 - Insel-Kanäle 17
- K**
 - Kernel-Treiber 38
 - Konstantendefinitionen 101
 - Kontinuierliche Analog-Ausgabe 29
- L**
 - LabVIEW 41
 - LabVIEW™
 - Programmierung 41
 - Leistungsmerkmale 6
- Lieferumfang 5
- M**
 - Modell-Übersicht 6
- P**
 - Periodische Analog-Ausgabe 33
 - Programmierung 21
 - D/A-Teil 21
 - Digital-I/O-Teil 37
 - unter Delphi 40
 - unter LabVIEW 41
 - unter Python 42
 - unter VEE 40
 - unter Visual Basic 39
 - unter Visual C++ 39
 - Python 42
- S**
 - Service und Support 100
 - Softwareunterstützung 8
 - Spezifikationen 91
 - Sub-D Buchse 95, 96
 - Systemanforderungen 8
 - Systemtreiber 38
- T**
 - Technische Fragen 100
 - Testprogramm 11, 21
 - Treiber allgemein 45
 - Treiberkonzept 38
 - Treiber-Update 100
 - Triggerflanken 18
 - Triggerung extern 18
- V**
 - VEE
 - Programmierung 40
 - Visual Basic 39
 - Visual C++ 39
- W**
 - WDM-Treiber 38
- Z**
 - Zubehör 98
 - Zusatzstecker ST2 97

