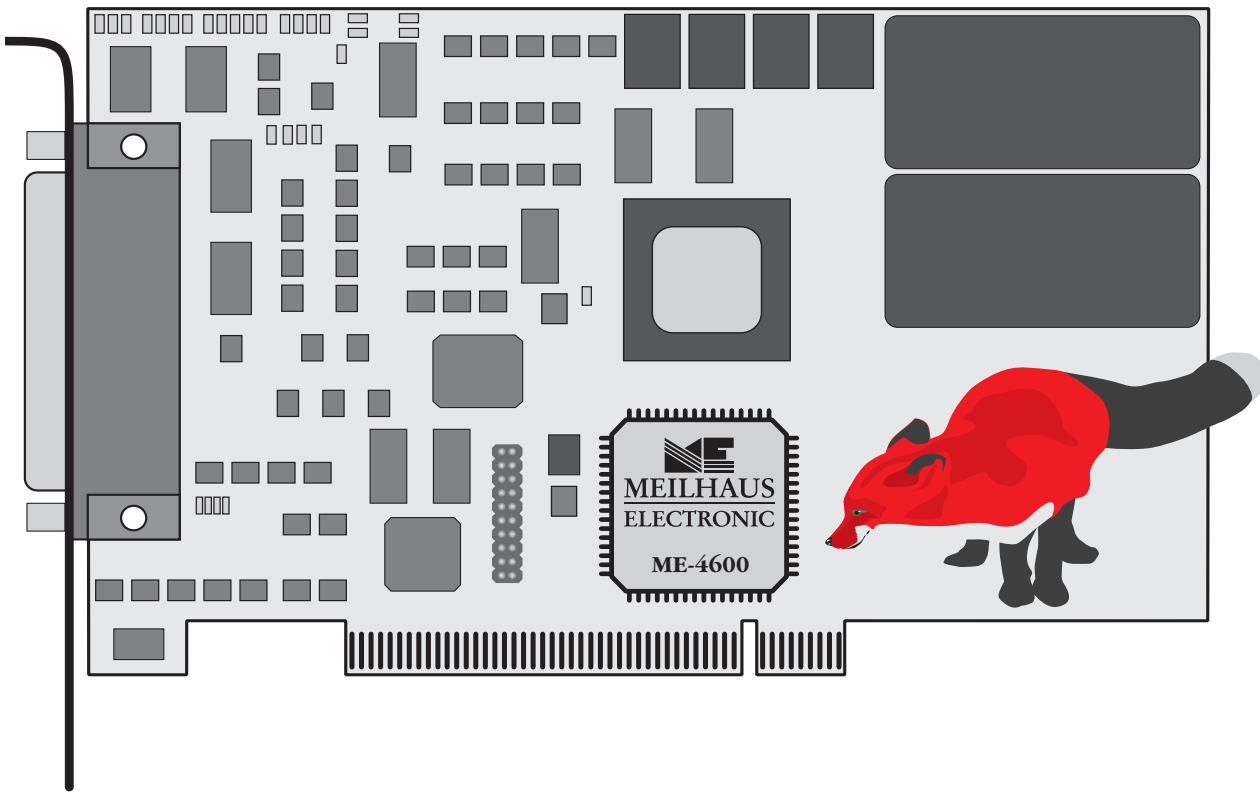
Meilhaus Electronic Handbuch

ME-4600 Serie 1.7D

(ME-4650/4660/4670/4680)

Die „ME-FoXX®“-Familie



16 Bit Multi-I/O-Karte mit bis zu 32 A/D- und 4 D/A-Kanälen
Optional: Optoisolierung und Sample & Hold-Stufe

Impressum

Handbuch ME-4650/4660/4670/4680

Revision 1.6D
Ausgabedatum: 22. Juni 2005

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

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.

ISO 9001

Deutscher Akkreditierungs Rat
TGA-ZQ-002/92-00

PCI INDUSTRIAL COMPUTERS PICMG EUROPE

Inhalt

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.

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:

- Multi-I/O-Karte der ME-4600 Serie für PCI-Bus
- Handbuch im PDF-Format auf CD-ROM
  (optional in gedruckter Form)
- Treibersoftware auf CD-ROM
- 78poliger Sub-D-Gegenstecker
- Zusatz-Slotblech ME-AK-D25F/S
- 25poliger Sub-D-Gegenstecker

1.2 Leistungsmerkmale

Übersicht	16 Bit A/D [Kanäle] single ended/differentiell	Analog-Trigger für A/D-Teil	16 Bit D/A [Kanäle]	D/A-Kanäle mit FIFOs	Digital-I/Os [Kanäle][1)]	Optoisolierung[2)]	Sample&Hold [Kanäle][3)]	Zähler	
ME-4650 „ME-LittleFoXX“	16/–	–	–	–	32	–	–	–	ME-FoXX Familie
ME-4660(i/s/is)* „ME-RedFoXX“	16/–	–	2	–	32	✔	8	3	
ME-4670(i/s/is)* „ME-SlyFoXX“	32/16	✔	4	–	32	✔	8	3	
ME-4680(i/s/is)* „ME-SylverFoXX“	32/16	✔	4	✔	32	✔	8	3	

* Beachten Sie, daß nicht alle theoretisch möglichen Varianten standardmäßig lieferbar sind (siehe www.meilhaus.com/me-foxx).

1) Digital-Port A+B sind auf 78polige Sub-D-Buchse der Karte geführt, Port C+D sind über optionales Slotblech mit 25poliger Sub-D-Buchse abgreifbar.

2) Nur „i“-Versionen: Optoisolierung von A/D- und D/A-Teil, Zähler, sowie Digital-Ports A+B (nicht Port C+D).

3) Optional mit 8 Sample&Hold-Kanälen („s“-Versionen)

Die Karten der **ME-4600 Serie** verfügen über bis zu **32 A/D-Kanäle**, die entweder als 32 single ended oder 16 differentielle Kanäle betrieben werden können (ME-4650/4660: 16 single ended Kanäle). Die Eingangskanäle werden über eine hochohmige Eingangsstufe auf einen 16 Bit 500 kHz A/D-Wandler geführt. Sie können zwischen den Eingangsbereichen 0…2,5 V, 0…10 V, ±2,5 V und ±10 V wählen.

Alle Modelle sind mit **32 Digital-I/O-Kanälen** ausgestattet, die als 4 bidirektionale Ports organisiert sind. Falls Sie die Option „Optoisolierung“ gewählt haben, ist Port A als Ausgang und Port B als Eingang festgelegt. Port C und D sind grundsätzlich nicht optoisoliert. Diese beiden Ports sind auf einen 20pol. Stiftstecker geführt und können über ein Zusatz-Slotblech abgegriffen werden.

Mit Ausnahme der ME-4650 stehen dem Anwender 3 frei programmierbare **16 Bit Zähler** zur Verfügung (1 x 8254).

Das Modell **ME-4660** verfügt über 2 und das Modell **ME-4670** über 4 hochgenaue **16 Bit D/A-Kanäle**. Die Ausgangsspannung kann im Bereich ±10 V variiert werden.

Beim Spitzenmodell **ME-4680** sind die **4 D/A-Kanäle** zusätzlich **mit FIFOs** ausgestattet. Damit können Sie Ausgaberaten von bis zu 500 kS/s pro Kanal erreichen. In der Betriebsart „AOContinuous“ können noch während der Ausgabe kontinuierlich Werte nachgeladen werden, während die Betriebsart „AOWraparound“ für die Ausgabe periodischer Signale gedacht ist.

Mit der Option **„Optoisolierung“** („i“-Versionen) haben Sie die Möglichkeit, alle Funktionsgruppen (außer Port C + D) der Karte konsequent von der PC-Masse zu entkoppeln. Dies ist vor allem zur Verhinderung von Masseschleifen und in störfeldbehafteten Umgebungen hilfreich.

Für die simultane Datenerfassung sind bei den „s“-Versionen 8 A/D-Kanäle mit einer **„Sample & Hold“**-Option ausgestattet.

Die mitgelieferte Software ermöglicht das rasche Einbinden der Karten in allen gängigen Hochsprachen unter Windows und Linux. Ebenso sind Treiber für graphische Programmierumgebungen wie Agilent VEE und LabVIEW™ erhältlich.

1.3 Systemanforderungen

Die ME-4600 setzt einen PC mit Intel® Pentium® Prozessor oder kompatiblen Rechner voraus, der über einen freien Standard-PCI Steckplatz (32 Bit, 33MHz, 5V) verfügt. Die Karte wird von allen aktuellen Windows Versionen sowie 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

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 einfachen Test der Einsteckkarte werden im ME-Software-Developer-Kit (ME-SDK) entsprechende Testprogramme mitgeliefert. Nach dem Entpacken des ME-SDK finden Sie diese in entsprechenden Unterverzeichnissen von `C:\MEILHAUS` (Default). Beachten Sie, daß der Systemtreiber installiert sein muß!

3 Hardware

3.1 Blockschaltbild

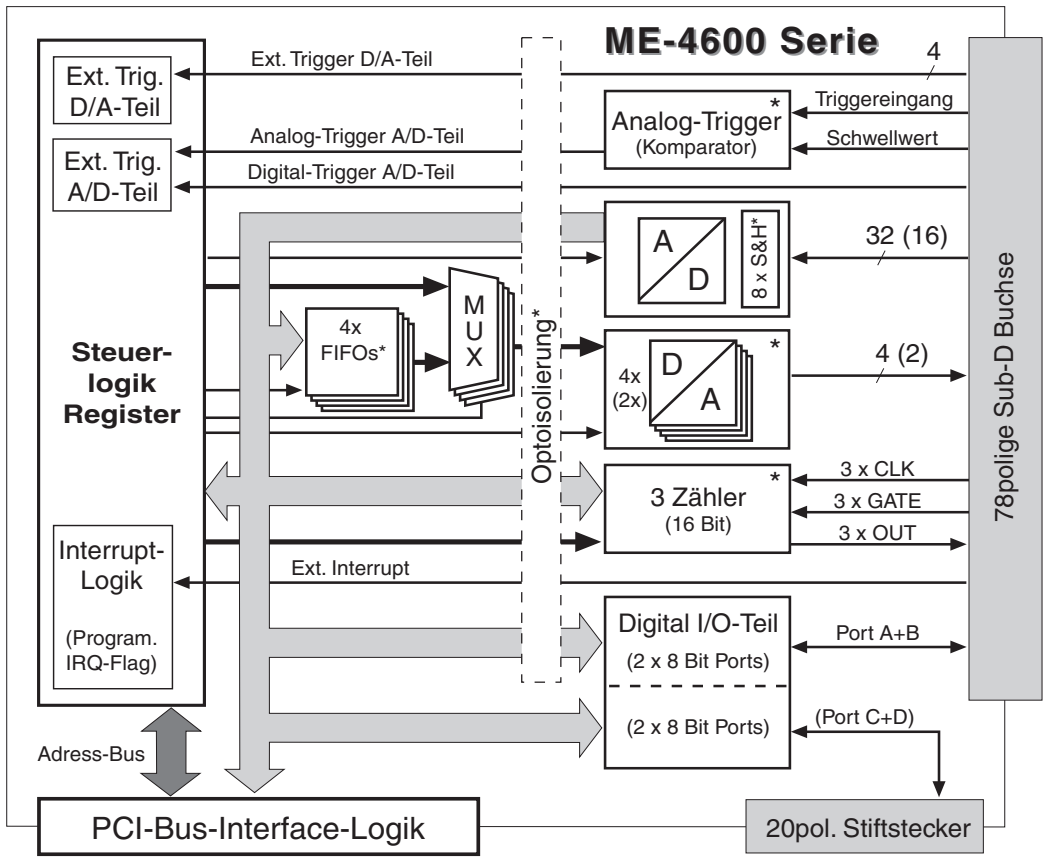


Abb. 1: Blockschaltbild der ME-4600

* Je nach Modell sind nicht alle der in obigem Blockschaltbild dargestellten Funktionsgruppen bestückt:

ME-4650: 16 A/D-Kanäle, 32 Digital-I/Os

ME-4660: 16 A/D-Kanäle, 2 D/A-Kanäle, 32 Digital-I/Os, 3 Zähler

ME-4670: 32 A/D-Kanäle, Analog-Trigger, 4 D/A-Kanäle, 32 Digital-I/Os, 3 Zähler

ME-4680: 32 A/D-Kanäle, Analog-Trigger, 4 D/A-Kanäle mit FIFO, 32 Digital-I/Os, 3 Zähler

„i"-Option: mit Optoisolierung

„s"-Option: mit 8 Sample & Hold Kanälen

3.2 Generelle Hinweise

Achtung: Sämtliche Steckverbindungen der Karte sollten grundsätzlich nur im spannungslosen Zustand hergestellt bzw. gelöst werden.

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

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!

Alle unbenutzten Eingangskanäle sind grundsätzlich auf Masse zu legen, um ein Übersprechen zwischen den Eingangskanälen zu vermeiden.Wir empfehlen die Verwendung abgeschirmter Leitungen.

Die Belegung der 78poligen Sub-D Buchse finden Sie im Anhang (siehe „Anschlußbelegungen" auf Seite 234).

In den folgenden Kapiteln finden Sie eine Beschreibung zur Beschaltung der einzelnen Funktionsgruppen. Zu Betriebsarten und Programmierung lesen Sie bitte Kapitel 4 ab Seite 31.

3.3 A/D-Teil

Mit Ausnahme der Modelle ME-4650 und ME-4660 (16 single ended Kanäle) verfügen alle Modelle der ME-4600 Serie über 32 single-ended bzw. 16 differentielle Eingangskanäle. Alle Kanäle sind über eine hochohmige Eingangsstufe entkoppelt:

- Eingangsimpedanz: R_{IN} = typ. 600MΩ, C_{IN} = typ. 3pF

Bei Karten mit Sample&Hold-Option (siehe auch Kap. 3.3.3) beträgt die Eingangsimpedanz der ersten 8 Kanäle (AD_0…7):

- R_{IN} = typ. 1MΩ, C_{IN} = typ. 5pF (dies gilt unabhängig davon, ob Sample&Hold-Option eingeschaltet ist oder nicht).

Die Spannung an den analogen Eingängen darf ±15 V nicht überschreiten!

Der Anwender kann zwischen den unipolaren Messbereichen 0…(2,5V-1LSB) und 0…(10V-1LSB) sowie den bipolaren Messbereichen -2,5V…(+2,5V-1LSB) und -10V…(+10V-1LSB) wählen.

Es gelten folgende (ideale) Kennlinien:

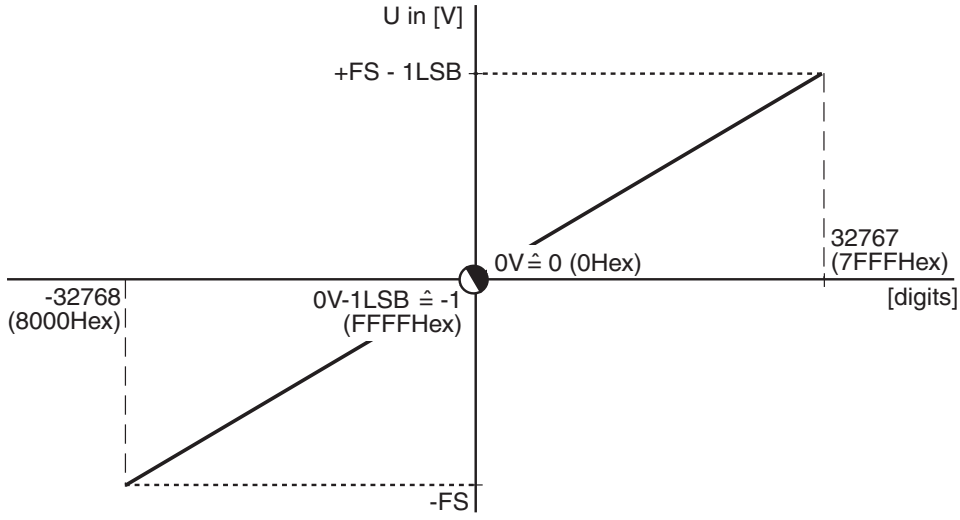


Abb. 2: Kennlinie für bipolare Eingangsbereiche

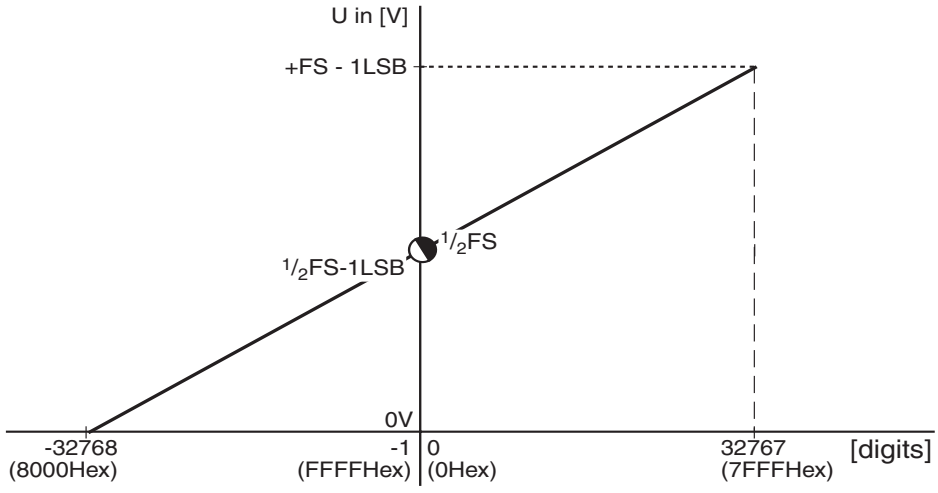


Abb. 3: Kennlinie für unipolare Eingangsbereiche

(„FS“ steht für „Full Scale“ (Vollausschlag) im jeweiligen Meßbereich; „LSB“ steht für das niederwertigste Bit der 16 Bit breiten A/D-Wandlung).

Beachten Sie, daß der theoretische Wert für Vollausschlag (Full Scale) im jeweiligen Messbereich in der Regel nur annähernd erreicht wird (siehe auch Spezifikationen auf Seite 229).

Zur **timergesteuerten Wandlung** stehen ein 32 Bit CHAN- und ein 36 Bit SCAN-Timer zur Verfügung. Die Konfiguration des A/D-Teils in den Betriebsarten „AIContinuous“ und „AIScan“ erfolgt mit der Funktion …*AIConfig*. Der für den jeweiligen Kanal

gewünschte Eingangsspannungsbereich wird in einer sog. Kanalliste abgelegt. Verwenden Sie zur Generierung der Kanallisteneinträge (max. 1024 Einträge) die Funktion *…AIMakeChannel ListEntry*. Gestartet wird die Wandlung je nach Programmierung per Software oder durch eine der zahlreichen externen Triggeroptionen.

3.3.1 Single-Ended-Betrieb

Im Single-Ended-Betrieb stehen 32 Eingangskanäle (ME-4650/ ME-4660: 16 Kanäle) in allen Eingangsbereichen zur Verfügung. Das Meßsignal wird mit dem gewünschten Eingangskanal verbunden. Jeder Eingangskanal (AD_*x*) benötigt einen möglichst niederohmigen Bezug zur Masse des A/D-Teils (A_GND). Achten Sie darauf, daß alle Minusleitungen gleiches Potential haben, um „Querströme“ und damit Meßfehler zu vermeiden.

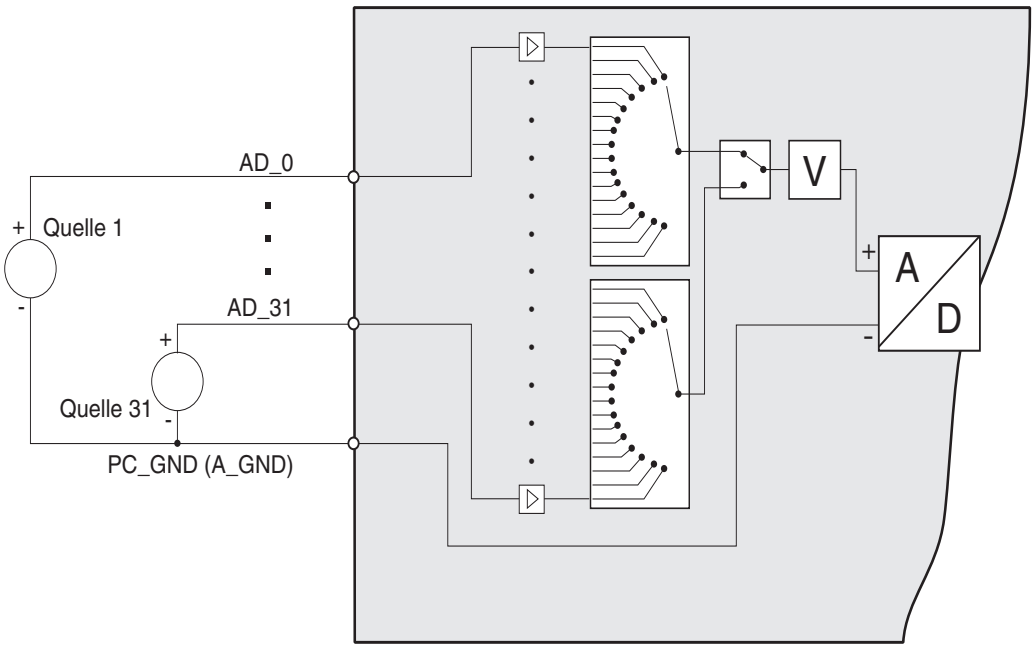


Abb. 4: Beschaltung im Single-Ended-Betrieb

3.3.2 Differentieller Betrieb

Der Vorteil der differentiellen Messung liegt in der weitgehenden Unterdrückung von Gleichtaktstörungen. Sie können bis zu 16 differentielle Eingangskanäle in den bipolaren Eingangsbereichen (±2,5V und ±10V) nutzen. Jeder Eingangskanal benötigt einen positiven und einen negativen Eingang.

Hinweis: Die ME-4650/4660 können nur single ended messen!

Die Zuordnung der Pins zu den differentiellen Kanälen entnehmen Sie bitte folgender Tabelle:

Pos. Signal		Neg. Signal		Pos. Signal		Neg. Signal	
Kanal	**Pin**	**Kanal**	**Pin**	**Kanal**	**Pin**	**Kanal**	**Pin**
AD_0	39	AD_16	15	AD_8	78	AD_24	54
AD_1	19	AD_17	34	AD_9	58	AD_25	73
AD_2	38	AD_18	14	AD_10	77	AD_26	53
AD_3	18	AD_19	33	AD_11	57	AD_27	72
AD_4	37	AD_20	13	AD_12	76	AD_28	52
AD_5	17	AD_21	32	AD_13	56	AD_29	71
AD_6	36	AD_22	12	AD_14	75	AD_30	51
AD_7	16	AD_23	31	AD_15	55	AD_31	70

Tabelle 1: Zuordnung der Kanäle bei differentiellem Betrieb

Beachten Sie bitte, daß auch im differentiellen Betrieb ein Bezug zur Analogmasse vorhanden sein muß. Diesen Bezug stellen Sie her, indem Sie die negativen Eingänge über einen Widerstand (ca. 100 k**Ω**) mit der Masse des A/D-Teils (A_GND) verbinden.

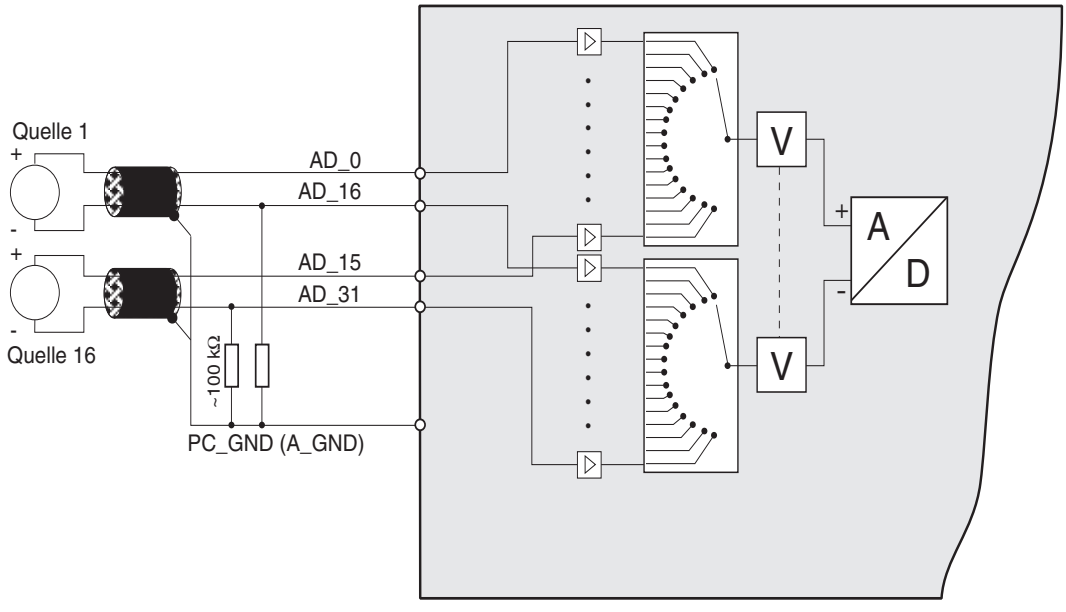


Abb. 5: Beschaltung im differentiellen Betrieb

3.3.3 Simultan-Betrieb

Bei Karten mit Sample&Hold-Option („s“-Versionen) kann die simultane Erfassung der ersten 8 Kanäle per Software aus- und eingeschaltet werden. Die Eingangsimpedanz der Sample&Hold-Kanäle beträgt: R_{IN} = typ. 1MΩ, C_{IN} = typ. 5pF. Dies gilt unabhängig davon, ob die Sample&Hold-Option eingeschaltet ist oder nicht.

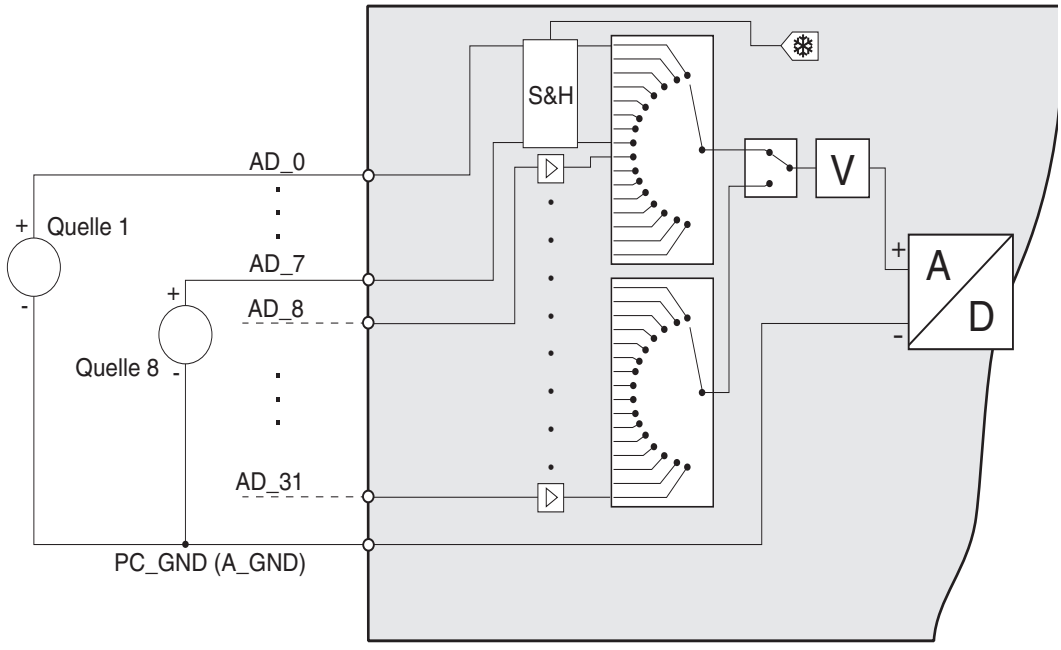


Abb. 6: Beschaltung im Simultan-Betrieb

Nach einem entsprechenden Signal der Ablaufsteuerung werden die an den Kanälen AD_0…7 anliegenden Spannungswerte „eingefroren“ und gemäß Kanalliste sequentiell „abgeholt“. Beachten Sie dabei folgende Punkte:

- Im Simultan-Betrieb, muß die Betriebsart „single ended“ (für alle Kanallisteneinträge) verwendet werden!
- Pro Kanallistenabarbeitung kann jeder S&H-Kanal nur einmal abgetastet werden. D. h. die A/D-Kanäle 0…7 dürfen nur einmal in der Kanalliste eingetragen sein.
- Sinnvolle Werte für die Anzahl der Kanallisteneinträge: 2…8
- Wir empfehlen bei simultaner Erfassung stets die schnellste Abtastrate (2 µs) einzustellen. Ansonsten „schmilzt“ der „eingefrorene“ Spannungswert mit typ. 0,08 µV/µs.

- Die minimale Zeit zwischen 2 simultanen Messungen hängt von der Anzahl der abgetasteten Kanäle und von der Erholzeit ab. Beachten Sie dies, falls Sie hier mit dem SCAN-Timer arbeiten. Für die min. SCAN-Zeit im Simultan-Betrieb gilt:

  Min. SCAN-Zeit = (Anzahl der Kanallisteneinträge x CHAN-Zeit) + Erholzeit

Beachten Sie, daß nach Abarbeitung der Kanalliste auf jeden Fall eine Erholzeit von min. 1,5 µs eingehalten werden muß!

Im folgenden Beispiel sollen 4 Kanäle simultan erfaßt werden. Die Werte sollen schnellstmöglich „abgeholt“ werden, d. h. die CHAN-Zeit soll minimal sein (2 µs). Daraus ergibt sich:

min. SCAN-Zeit = (4 x 2 µs) + 1,5 µs = 9,5 µs

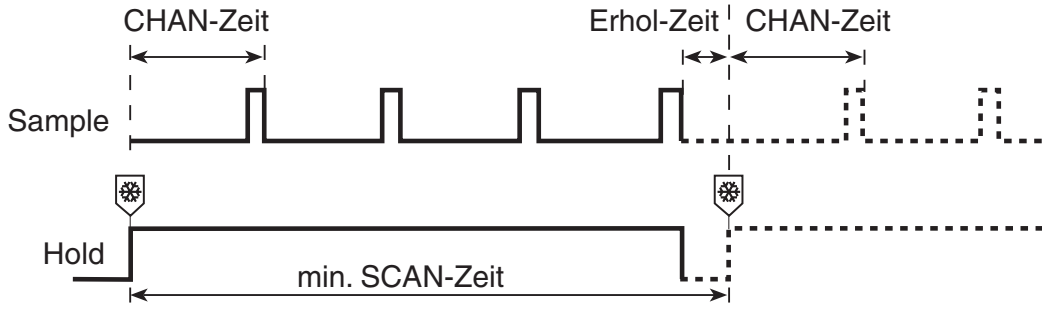


Abb. 7: Sample & Hold Timing

3.3.4 Externer Trigger A/D-Teil

Alle Modelle der ME-4600 Serie verfügen über einen digitalen A/D-Triggereingang. Die Modelle ME-4670 und ME-4680 sind zusätzlich mit einer analogen Triggereinheit ausgestattet. Je nach gewählter Flanken-Option („RISING“, „FALLING“ oder „BOTH“) wird die Wandlung durch eine entsprechende Flanke gestartet.

Abb. 8: Triggerflanken

3.3.4.1 Analog-Trigger A/D-Teil

Die analoge A/D-Trigger-Einheit verwendet einen Komparator, der die Spannungspegel an den Eingängen AD_TRIG_A+ (Pin 50) und AD_TRIG_A- (Pin 69) vergleicht.

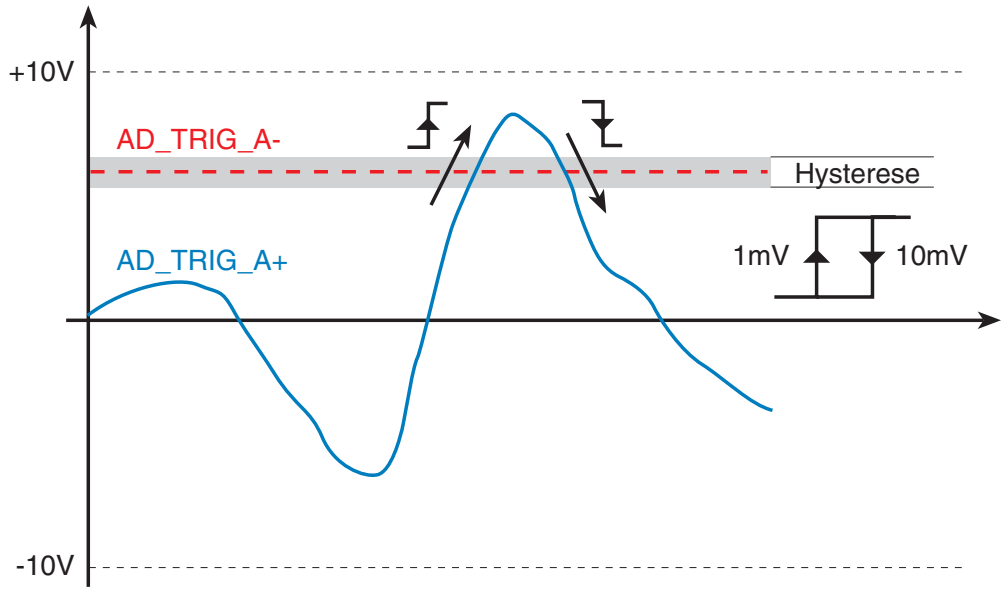


Abb. 9: Analog-Trigger

Wir empfehlen, am Minus-Eingang einen Pegel anzulegen, der als „Schwellwert“ dient. Dies kann z. B. über einen D/A-Kanal oder durch eine ext. Spannungsquelle erfolgen. Am Plus-Eingang wird nun das Signal angelegt auf das getriggert werden soll. Dies könnte z. B. ein A/D-Kanal sein, der mit dem Plus-Eingang verbunden wird (siehe auch Abb. 10). Sobald der Pegel am Plus-Eingang positiver wird als der Schwellwert am Minus-Eingang entspricht dies einer steigenden Flanke. In umgekehrter Richtung spricht man von einer negativen Flanke.

Es können dynamische Signale bis 500 kHz bei max. ±10V angelegt werden. Berücksichtigen Sie einen Massebezug der Trigger-Eingänge. Bei nicht optoisolierten Karten ist dies die PC-Masse (PC_GND). Bei optoisolierten Karten benötigt der analoge Trigger einen Bezug zur Analog-Masse (A_GND).

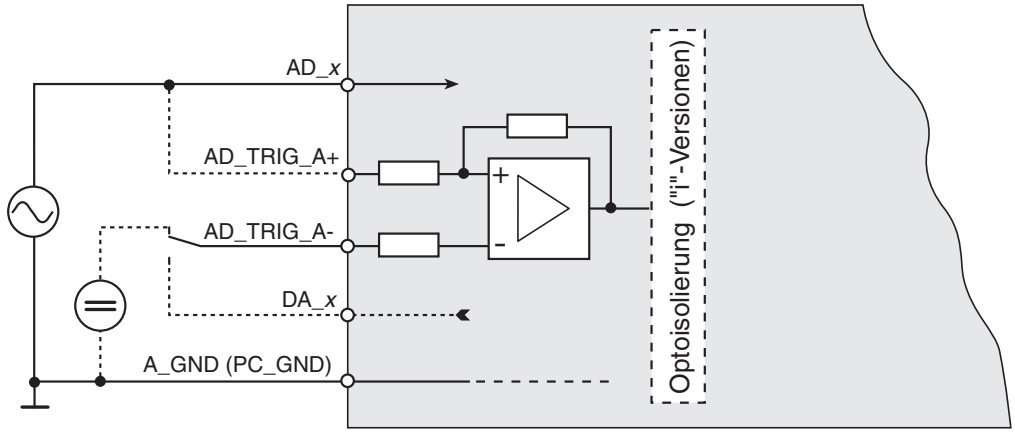


Abb. 10: Beschaltung Analog-Trigger

3.3.4.2 Digital-Trigger A/D-Teil

Der digitale Triggereingang (AD_TRIG_D) ist für einen High-Pegel von +5V ausgelegt und muß bei Varianten mit Optoisolation mit einem Strom I_F von min. 7,5 mA gespeist werden. Das Triggersignal benötigt einen Masse-Bezug (PC_GND bzw. DIO_GND).

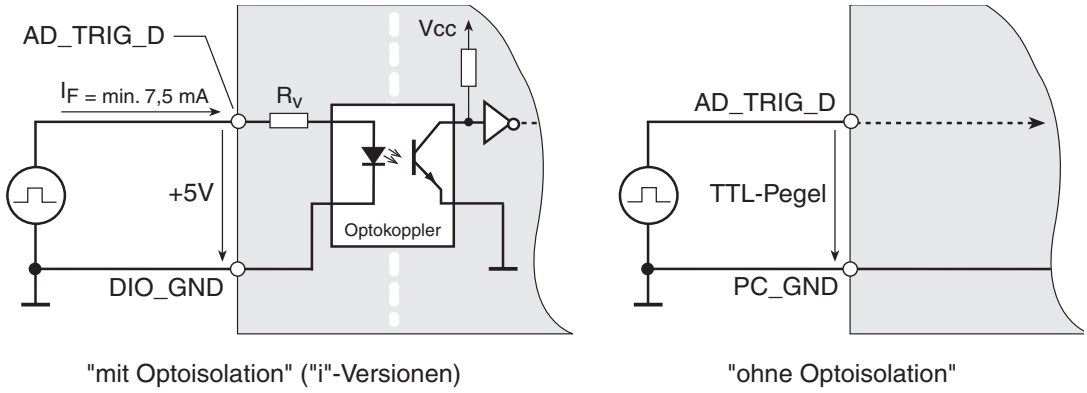


Abb. 11: Beschaltung Digital-Trigger

3.4 D/A-Teil

Die ME-4660 verfügt über 2 und die ME-4670 und ME-4680 verfügen über 4 analoge Ausgangskanäle. Jeder Kanal ist mit einem seriellen 16 Bit D/A-Wandler bestückt, der mit bis zu 500 kS/s wandeln kann. Die Ausgangsspannung kann einen Spannungsbereich von -10V…+10V-1LSB überstreichen.

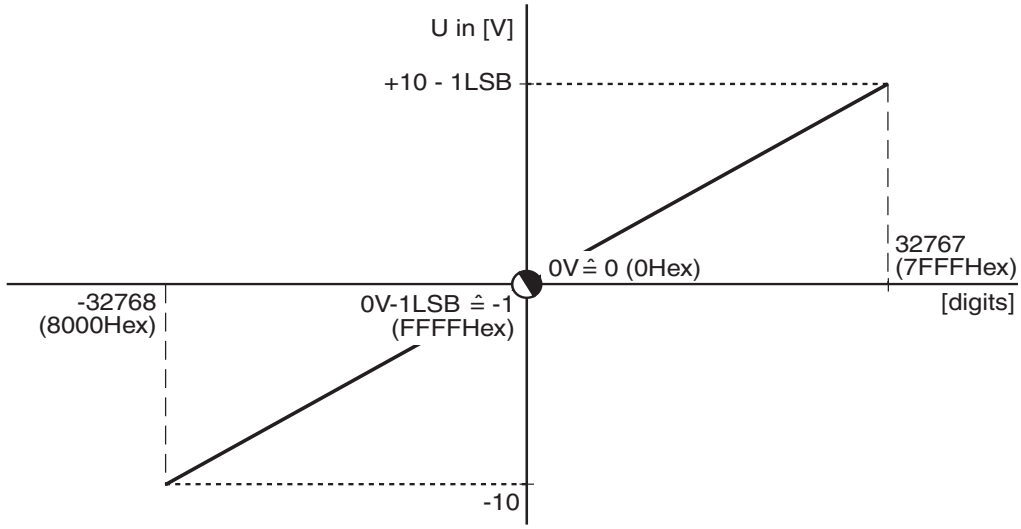


Abb. 12: Kennlinie des D/A-Wandlers

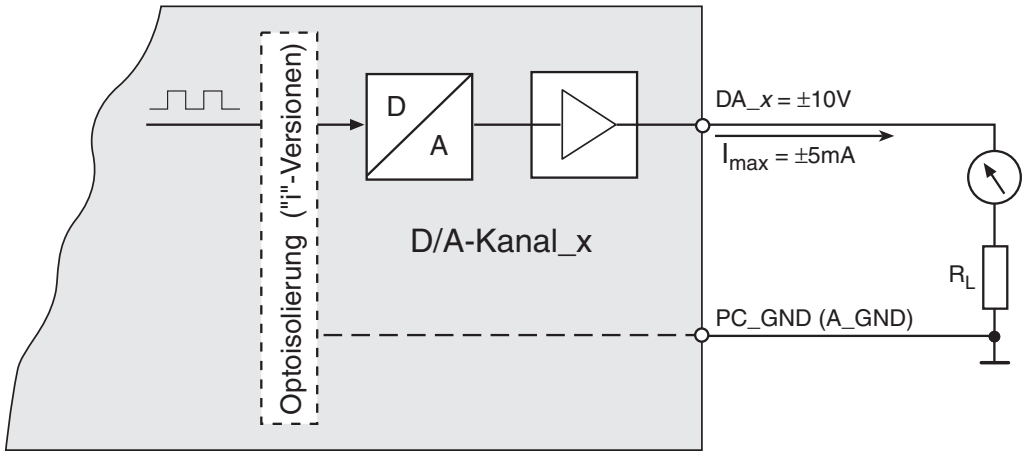


Abb. 13: Beschaltung der analogen Ausgänge

Beachten Sie, daß I_{max} = **±5 mA** pro Kanal nicht überschritten werden dürfen!

Bei den optoisolierten Modellen („i"-Versionen) sind alle D/A-Kanäle von der PC-Masse entkoppelt und beziehen sich gemeinsam auf die Analog-Masse (A_GND).

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

3.4.1 Externer Trigger D/A-Teil

Für jeden D/A-Kanal steht ein externer Triggereingang (DA_TRIG_*x*) zur Verfügung. Je nach gewählter Flanken-Option („RISING“, „FALLING“ oder „BOTH“) wird die Wandlung durch eine entsprechende Flanke gestartet.

Abb. 14: Triggerflanken

Achten Sie bei der Beschaltung der ext. Triggereingänge darauf, daß die Spannungspegel eingehalten werden (siehe Spezifikationen auf Seite 231) und ein Bezug zur PC-Masse (PC_GND) bzw. Digital-Masse (DIO_GND) bei „i“-Versionen hergestellt werden muß. Der Vorwiderstand R_V der optoisolierten Triggereingänge ist für einen High-Pegel von +5V bei I_F = 7,5 mA ausgelegt. Für nicht optoisolierte Eingänge gilt TTL-Pegel.

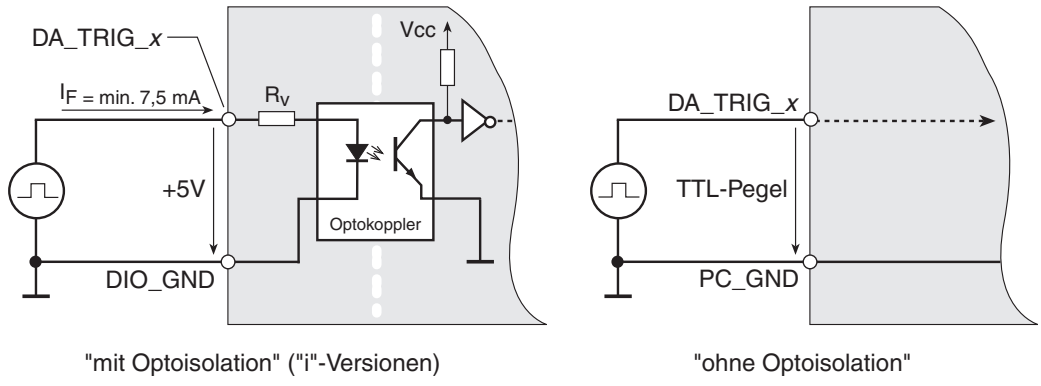


Abb. 15: Beschaltung der D/A-Triggereingänge

3.5 Digital-I/O-Teil

Die Karten der ME-4600 Serie verfügen über vier 8 Bit breite Digital-I/O-Ports. Sofern Ihre Karte keine Optoisolierung hat, kann jeder Port unabhängig als Ein- oder Ausgang konfiguriert werden. Bei Modellen mit Optoisolierung („i"-Versionen) ist Port A stets Ausgangs-Port und Port B Eingangs-Port.

Port C und D können über den 20pol. Stiftstecker ST2 auf das mitgelieferte Zusatz-Slotblech (ME-AK-D25F/S) mit einer 25pol. Sub-D-Buchse geführt werden. Diese beiden Ports sind auch bei den „i"-Versionen nicht optoisoliert. Optional kann dies über den Anschlußadapter ME-AA4-3i erreicht werden.

Die Richtung der Ports wird per Software konfiguriert. Nach dem Einschalten der Versorgung sind alle Ports auf Eingang geschaltet mit Ausnahme von Port A (= Ausgang) bei den optoisolierten Modellen („i"-Versionen).

Zur Programmierung des Digital-I/O-Teils lesen Sie bitte Kap. 4.3 "Digital-I/O-Teil".

3.5.1 Digitale Eingänge

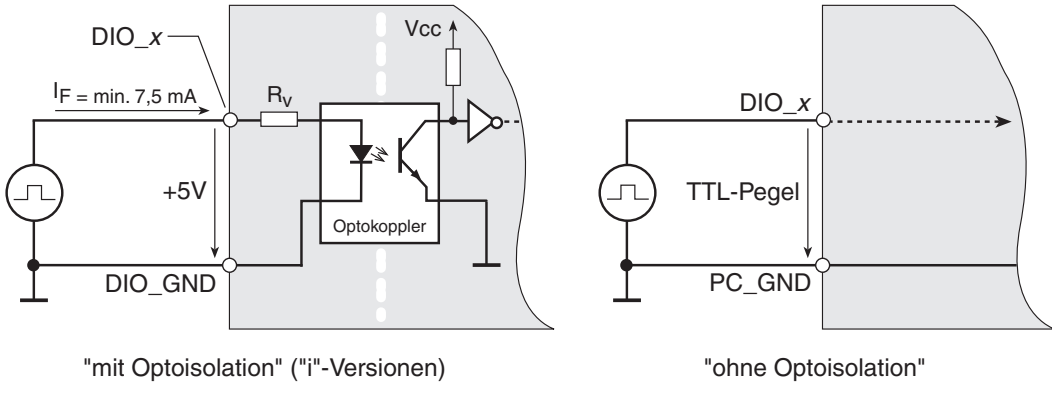


Abb. 16: Beschaltung der digitalen Eingänge

Achten Sie bei der Beschaltung der Eingänge darauf, daß die Spannungspegel eingehalten werden (siehe Spezifikationen auf Seite 231) und ein Bezug zur PC-Masse (PC_GND) bzw. Digital-Masse (DIO_GND) bei „i"-Versionen hergestellt werden muß. Der Vorwiderstand R_V der optoisolierten Eingänge ist für einen High-Pegel von +5V bei 7,5 mA ausgelegt. Für nicht optoisolierte Eingänge gilt TTL-Pegel.

3.5.2 Digitale Ausgänge

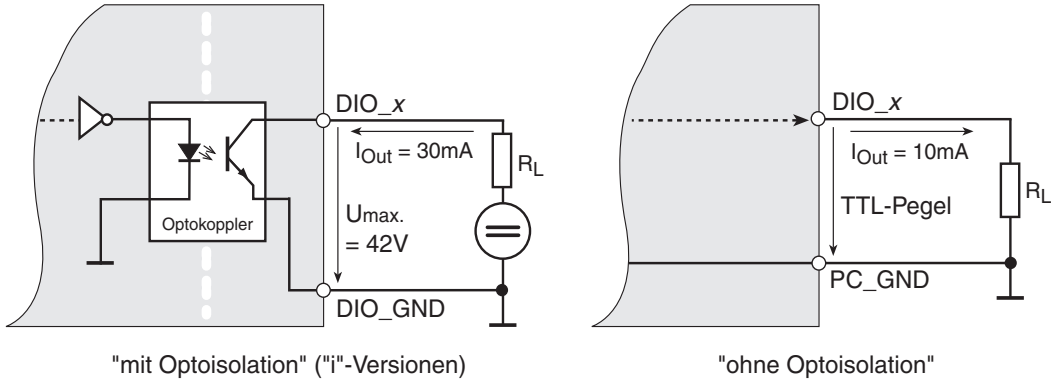


Abb. 17: Beschaltung der digitalen Ausgänge

Achten Sie bei der Beschaltung der Ausgänge darauf, daß die Spannungspegel eingehalten werden (siehe Spezifikationen auf Seite 231) und ein Bezug zur PC-Masse (PC_GND) bzw. Digital-Masse bei „i"-Versionen (DIO_GND) hergestellt werden muß. Bei optoisolierten Versionen darf die Spannung U_{max} bis zu 42V betragen. Der max. Ausgangsstrom bei TTL-Versionen beträgt $I_{Out} = I_{OL} = I_{OH} = 10$ mA; bei optoisolierten Versionen darf I_{Out} max. 30 mA sein.

3.6 Zähler

3.6.1 Zähler-Baustein

Auf den Karten der **ME-4600 Serie** (nicht ME-4650) kommt ein Standard-Zähler-Baustein vom Typ 82C54 zum Einsatz. Dies ist ein sehr vielseitiger Baustein, der über 3 unabhängige 16 Bit (Abwärts-) Zähler verfügt. Alle Zähler-Signale stehen an der Sub-D-Buchse zur Verfügung. Nach geeigneter Freigabe des GATE-Eingangs (TTL: 5V/Opto: 0V) zählt der entsprechende Zähler negativ flankengesteuert abwärts. Der Zählertakt (CLK) zur Speisung der Zähler muß extern eingespeist werden und darf max. 10 MHz betragen. Durch geeignete externe Beschaltung ist eine Kaskadierung der Zähler jederzeit möglich.

Die Zählersignale von nicht optoisolierten Karten arbeiten mit TTL-Pegel (siehe Anhang A "Spezifikationen") und benötigen einen Bezug zur PC-Masse (PC_GND). Der max. Ausgangsstrom beträgt im Low-Pegel I_{OL} = 7,8mA und im High-Pegel I_{OH} = 6mA.

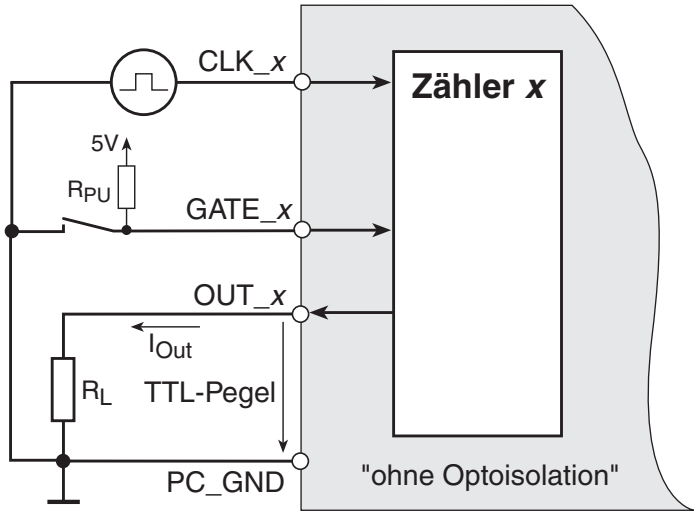
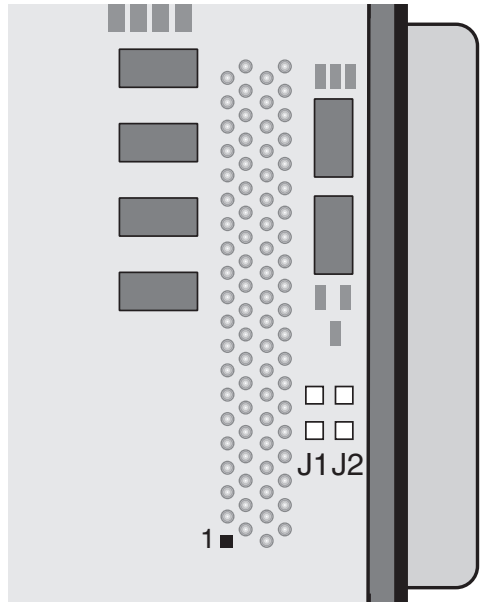


Abb. 18: Zähler-Beschaltung ohne Optoisolierung



Zur Versorgung der Optokoppler können Sie wählen, ob Sie eine externe Versorgungsspannung von +5V/30mA an Pin 1 (CNT_VCC_IN) zuführen, oder die interne Versorgungsspannung (A_VCC) des Analog-Teils verwenden möchten. Im ersten Fall dürfen J1 und J2 nicht gebrückt werden (Auslieferungszustand), im zweiten Fall müssen J1 und J2 gebrückt werden. Beachten Sie, daß die galvanische Trennung zwischen Analog-Masse (A_GND) und Zähler-Masse (CNT_GND) dadurch aufgehoben wird.

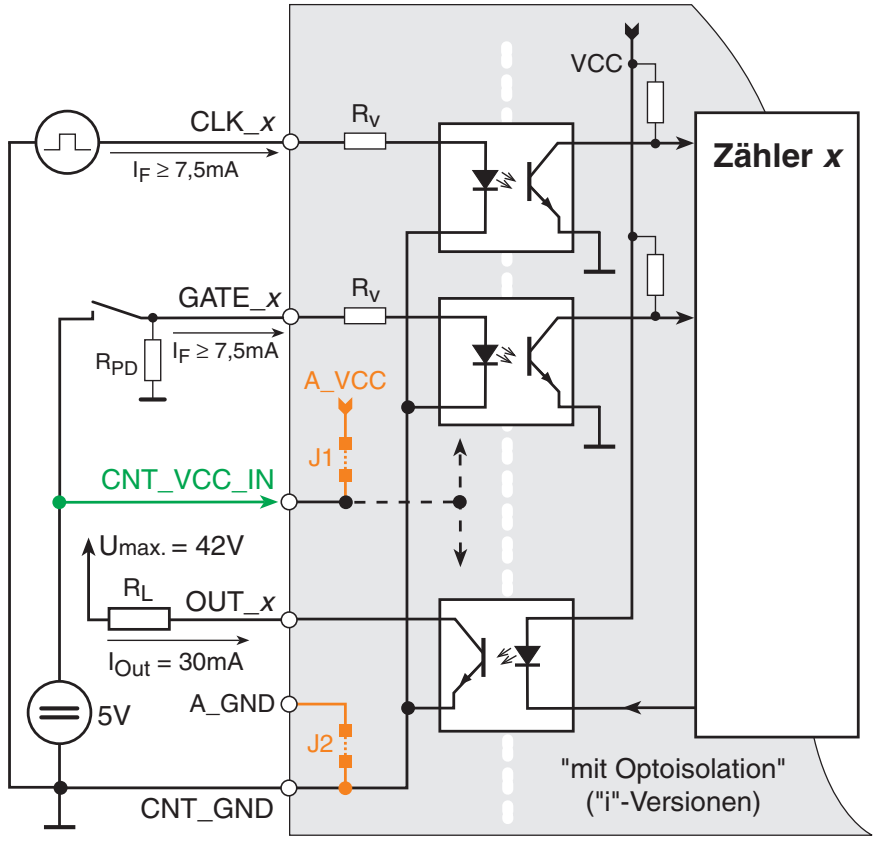


Abb. 19: Beschaltung der Zähler mit Optoisolierung

Beachten Sie, daß die Polarität der Eingangssignale (CLK_*x* und GATE_*x*) auf den optoisolierten Versionen durch die Optokoppler-Beschaltung invertiert wird. Alle Zähler-Signale benötigen einen Bezug zur Zähler-Masse (CNT_GND). Die Spannung U_{max}

darf 42V nicht überschreiten! Der max. Ausgangsstrom bei opto-isolierten Varianten darf I_{Out} = 30mA nicht überschreiten.

Zur Programmierung der Zähler lesen sie bitte Kap. 4.4.

3.6.2 Pulsweiten-Modulation

Durch geeignete externe Beschaltung kann mit Hilfe der Zähler 0…2 ein Signal mit variablem Tastverhältnis ausgegeben werden. Das Tastverhältnis kann zwischen 1…99% in 1%-Schritten variiert werden. Der Vorteiler muß mit einem externen Basistakt von max. 10MHz gespeist werden. Dies ergibt eine maximale Frequenz des Ausgangssignals von 50kHz. Bei Verwendung der in Abb. 20 gezeigten Beschaltung können Sie mit den Funktionen *me4000CntPWMStart/Stop* die Programmierung stark vereinfachen (siehe Seite 78 und 175ff).

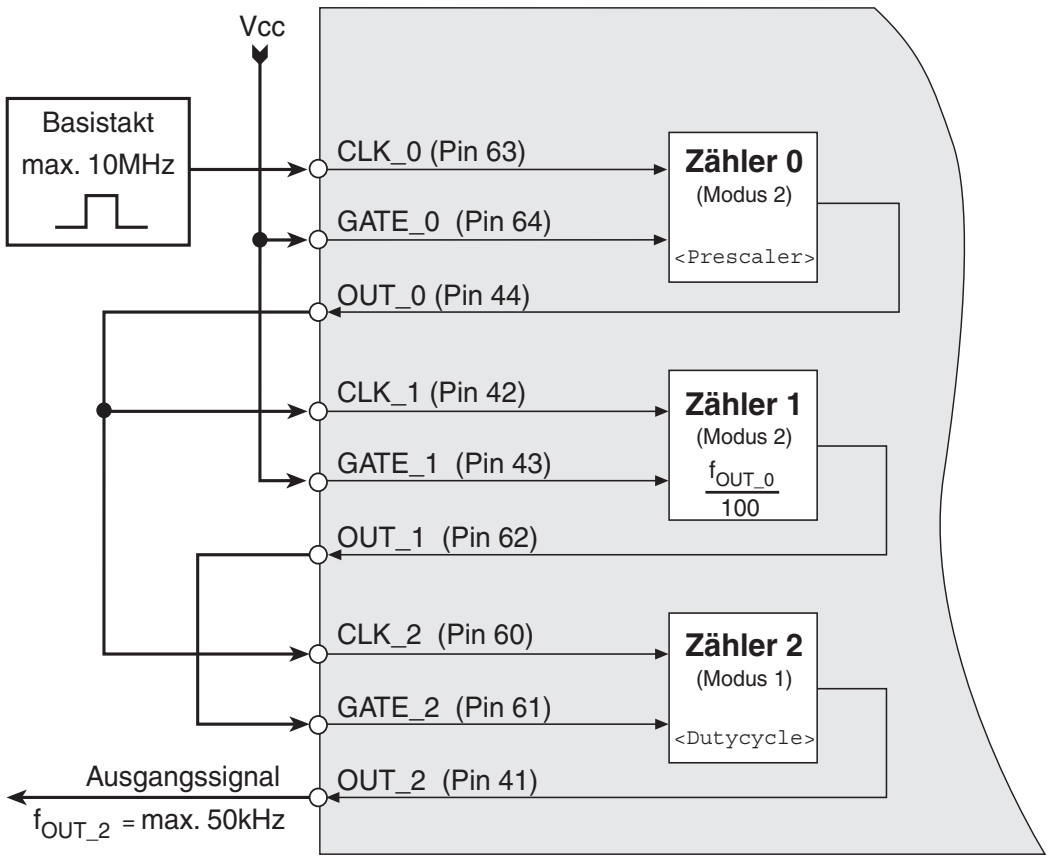


Abb. 20: Beschaltung Pulsweiten-Modulation

Für die Berechnung der Frequenz f_{OUT_2} gilt:

$$f_{OUT\_2} = \frac{\text{Basistakt}}{\texttt{<Prescaler>} \cdot 100} \qquad (\text{mit } \texttt{<Prescaler>} = 2\ldots(2^{16} - 1))$$

Tip: Für die ext. Beschaltung können Sie den optionalen Anschluß-Adapter ME-AA4-3(i) verwenden, der auch über einen 10MHz Quarzoszillator verfügt.

3.7 Externer Interrupt

Am externen Interrupt-Eingang (EXT_IRQ, Pin 48) können sie mit einer positiven Flanke einen Interrupt auslösen, der direkt an den PCI-Bus weitergeleitet wird. Vorraussetzung ist die Freischaltung der Interruptfunktionalität für die Karte mittels der Funktion *me4000ExtIrqEnable*.

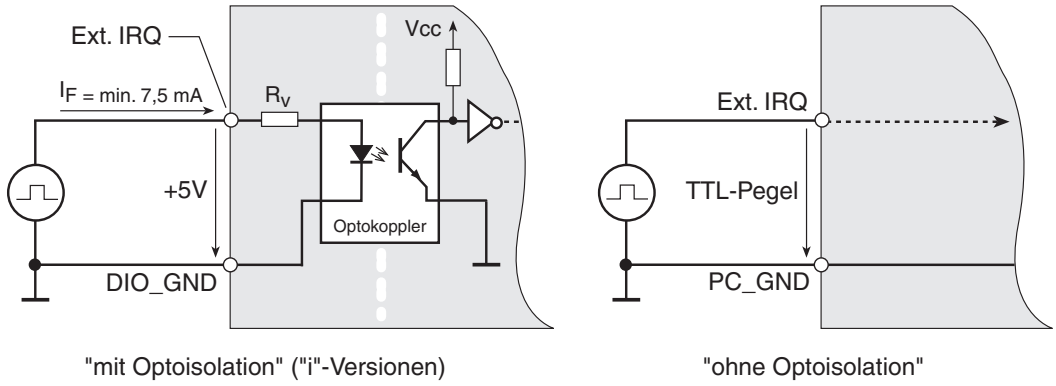


Abb. 21: Beschaltung ext. Interrupteingang

Beachten Sie, daß die Spannungspegel am Interrupteingang eingehalten werden (siehe Anhang A "Spezifikationen") und ein Bezug zur PC-Masse (ohne Optoisolation) bzw. Digital-Masse (mit Optoisolation) hergestellt wird.

4 Programmierung

4.1 A/D-Teil

Betriebsart	Anwendung	Trigger	Timing
AISingle (siehe Seite 31)	1 Kanal, 1 Meßwert	Software-Start, ext. digital, analog (opt.)	–
AIContinuous (siehe Seite 34)	Erfassung einer unbekannten Anzahl an Messwerten gemäß Kanalliste	Software-Start, ext. digital, analog (opt.)	CHAN-Timer, SCAN-Timer (…AIConfig)
AIScan (siehe Seite 34)	Erfassung einer bekannten Anzahl an Messwerten gemäß Kanalliste	Software-Start, ext. digital, analog (opt.)	CHAN-Timer, SCAN-Timer (…AIConfig)
AISimultaneous (siehe Seite 32)	Optional für AIScan und AIContinuous: Simultane Erfassung mehrerer Kanäle		

Tabelle 2: A/D-Betriebsarten

4.1.1 Betriebsart „AISingle“

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Betriebsart dient zur Erfassung eines einzelnen Wertes vom gewählten Kanal. Sie haben folgende Parameter zur Verfügung:

- Kanalnummer 0…31 (ME-4650/4660: 0…15)
- Eingangsspannungsbereich: 0…2,5V; 0…10V; ±2,5V; ±10V (Beachten Sie, daß die differentielle Betriebsart nur mit den bipolaren Eingangsbereichen kombiniert werden kann).
- Betriebsart single ended oder differentiell (ME-4650/4660: nur single ended).
- Triggermodi: per Software, externem Digital-Trigger oder ext. Analog-Trigger (nur ME-4670/4680).
- Externer Trigger: reagiert auf fallende, steigende oder beide Flanken.
- Time-Out: falls externes Triggersignal ausbleibt.

Es muß keine Kanalliste erzeugt werden.

Das folgende Diagramm soll den Programmfluss kurz erläutern:

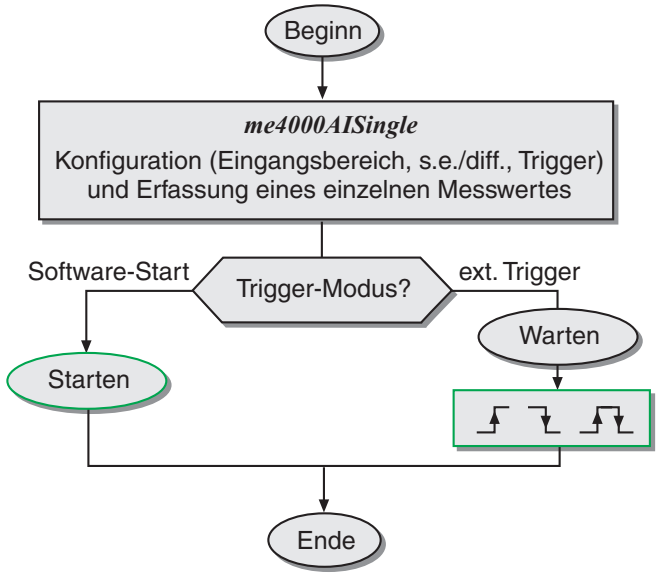


Abb. 22: Einzelwert-Erfassung

Beachten Sie auch die Beispiele im ME-SDK und die Funktionsbeschreibung auf Seite 128.

4.1.2 Betriebsart „AISimultaneous“

ME-4650	ME-4660	ME-4670	ME-4680
–	nur mit „s“-Option		

Bei Modellen mit der Option „s“ für Sample&Hold, können Sie bei Bedarf die A/D-Kanäle 0…7 auch simultan abtasten. Siehe auch Kap. 3.3.3)

Nach einem entsprechenden Signal der Ablaufsteuerung werden die an den Kanälen AD_0…7 anliegenden Spannungswerte „eingefroren“ und gemäß Kanalliste sequentiell „abgeholt“. Beachten Sie dabei folgende Punkte:

- Im Simultan-Betrieb, muß die Betriebsart „single ended“ (für alle Kanallisteneinträge) verwendet werden!
- Pro Kanallistenabarbeitung kann jeder S&H-Kanal nur einmal abgetastet werden. D. h. die A/D-Kanäle 0…7 dürfen nur einmal in der Kanalliste eingetragen sein.

- Sinnvolle Werte für die Anzahl der Kanallisteneinträge: 2…8
- Wir empfehlen bei simultaner Erfassung stets die max. Abtastrate (500 kS/s) einzustellen. Ansonsten „schmilzt“ der „eingefrorene“ Spannungswert mit typ. 0,08 µV/µs (max. 0,5 µV/µs).
- Beachten Sie, daß die minimale Zeit zwischen 2 simultanen Messungen von der Anzahl der abgetasteten Kanäle und von der Erholzeit abhängt. Berücksichtigen Sie dies bei der Berechnung der SCAN-Zeit.
- Für die min. SCAN-Zeit im Simultan-Betrieb gilt:

  min. SCAN-Zeit = (Anzahl der Kanallisteneinträge x CHAN-Zeit) + Erholzeit

  Minimale Erholzeit: 1,5 µs

Im folgenden Beispiel sollen 4 Kanäle simultan erfaßt werden. Die Werte sollen schnellstmöglich abgeholt werden, d. h. die CHAN-Zeit soll minimal sein (2 µs). Daraus ergibt sich:

min. SCAN-Zeit = (4 x 2 µs) + 1,5 µs = 9,5 µs

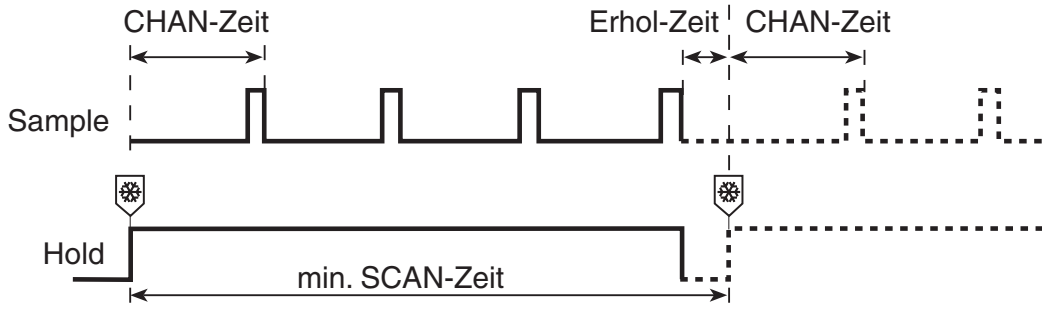


Abb. 23: Sample & Hold Timing

4.1.3 Timergesteuerte „AI-Betriebsarten“

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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

1. Erzeugen einer benutzerdefinierten Kanalliste mit …*AIMakeChannelListEntry* und Konfiguration des A/D-Teils mit …*AIConfig* (siehe Kap. 4.1.3.1)
2. Vorbereitung der Software mit …*AIContinuous* oder …*AIScan* (siehe Kap. 4.1.3.2)
3. Start der Erfassung mit …*AIStart* (siehe Kap. 4.1.3.3)

Vor Beginn der Erfassung müssen Sie sich entscheiden ob Sie eine bekannte Anzahl an Messwerten erfassen möchten („AIScan“) oder kontinuierlich („AIContinuous“) bis die Erfassung vom Anwender gestoppt wird. Nach Konfiguration von Hardware und Software kann die Erfassung per Software oder durch ein externes Triggersignal gestartet werden.

Abbildung 24 soll die grundsätzliche Vorgehensweise in den Betriebsarten **„AIContinuous“** und **„AIScan“** veranschaulichen. Zur weiteren Vorgehensweise bzgl. Konfiguration, Daten-Handling und Ausführungsmodi beachten Sie bitte die folgenden Kapitel.

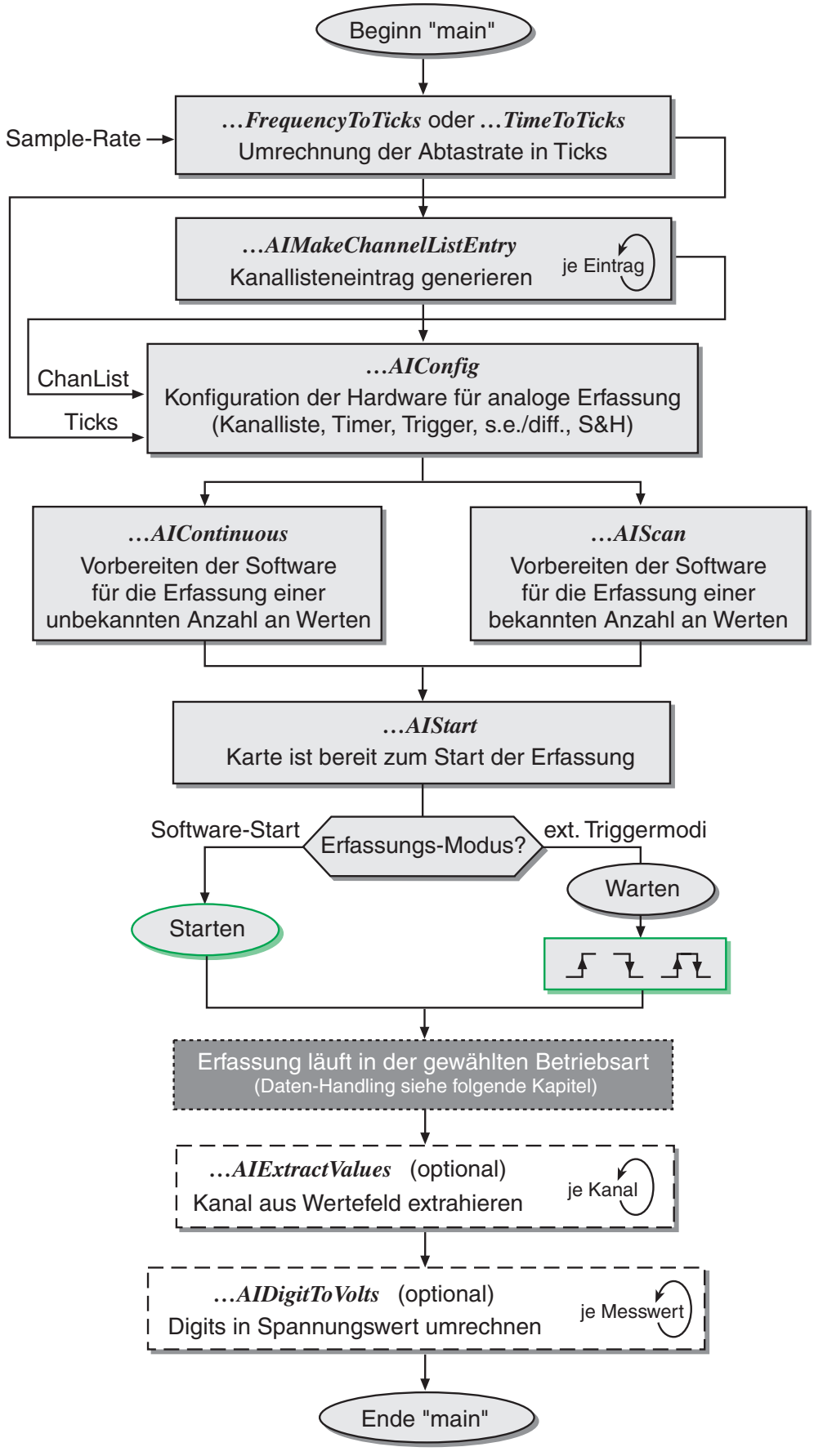


Abb. 24: Vorgehensweise Programmierung AI-Teil

4.1.3.1 Konfiguration des A/D-Teils

Vor der eigentlichen Konfiguration müssen Sie eine benutzerdefinierte A/D-Kanalliste zur Steuerung von Kanal-Nummer und Eingangsspannungsbereich generieren. Die Kanalliste kann max. 1024 Einträge enthalten. Allokieren Sie dazu ein Wertefeld definierter Größe, in das Sie durch wiederholten Aufruf der Funktion *…AIMakeChannelListEntry* die einzelnen Einträge schreiben. Sie haben folgende Parameter zur Verfügung (Funktionsbeschreibung siehe Seite 123):

- Kanalnummer 0…31 (ME-4650/4660: 0…15)
- Eingangsspannungsbereich: 0…2,5V; 0…10V; ±2,5V; ±10V (Beachten Sie, daß die differentielle Betriebsart nur mit den bipolaren Eingangsbereichen kombiniert werden kann).

Nachdem Sie die Kanalliste generiert haben, wird die Hardware mit der Funktion *…AIConfig* konfiguriert. Sie haben folgende Parameter zur Verfügung (Funktionsbeschreibung siehe Seite 112).

- Betriebsart single ended oder differentiell (ME-4650/4660: nur single ended)
- Falls gewünscht: simultane Erfassung der Kanäle 0…7 ein- bzw. ausschalten (nur für Versionen mit Sample & Hold-Option; siehe auch Kap. 3.3.3)
- Erfassungsmodi (siehe auch Kap .4.1.4 "Externer Trigger A/D-Teil"):
  - ***Erfassungsmodus „Software-Start“***

    Die Erfassung wird unmittelbar nach Aufruf der Funktion *me4000AIStart* gestartet. Die Abarbeitung erfolgt gemäß Timer-Einstellungen.

  - ***Erfassungsmodus „Extern-Standard“***

    Bereit zur Erfassung nach Aufruf der Funktion *…AIStart.* Erfassung beginnt mit dem ersten ext. Triggerimpuls. Abarbeitung gemäß Timer-Einstellungen. Weitere Triggerimpulse bleiben ohne Wirkung.

- ***Erfassungsmodus „Extern-Einzelwert“***

  Bereit zur Erfassung nach Aufruf der Funktion *…AIStart*. Mit jedem ext. Triggerimpuls wird genau **ein** Wert gemäß Kanalliste gewandelt. Nach Eintreffen des ersten Triggerimpulses bis zur ersten Wandlung verstreicht einmalig die CHAN-Zeit.

- ***Erfassungsmodus „Extern-Kanalliste“***

  Bereit zur Erfassung nach Aufruf der Funktion *…AIStart*. Mit jedem ext. Triggerimpuls wird die Kanalliste einmal abgearbeitet. Die Abarbeitung erfolgt gemäß Timer-Einstellungen. Der SCAN-Timer bleibt ohne Wirkung!

- Externe Triggermodi: analoge oder digitale Triggerquelle (fallende, steigende oder beide Flanken).

- Als Zeitgeber dienen 2 programmierbare Zähler. Ein 32 Bit breiter CHAN-Timer sowie ein 36 Bit breiter SCAN-Timer. Als gemeinsame Zeitbasis nutzen alle Timer einen 33MHz Takt. Daraus ergibt sich eine Periodendauer von 30,30ns, die als kleinste Zeiteinheit definiert wird und im Folgenden „1 Tick“ genannt wird. Zur einfachen Umrechnung können Sie die Funktionen *…FrequencyToTicks* oder *…TimeToTicks* verwenden.

  - Der CHAN-Timer bestimmt die Abtastrate (Sample-Rate) innerhalb einer Kanalliste (Zeitdifferenz zwischen zwei aufeinanderfolgenden Kanallisten-Einträgen). Es sind CHAN-Zeiten im Bereich 2 µs…130 s einstellbar.

  - Der SCAN-Timer bestimmt die Zeit zwischen dem Start zwei aufeinander folgender Kanallistenabarbeitungen. Die Verwendung ist optional. Es sind SCAN-Zeiten bis ca. 30 Minuten möglich. Die SCAN-Zeit errechnet sich aus:

    (Anzahl der Kanallisten-Einträge x CHAN-Zeit) + „Pause“

    Die „Pause“ und damit die SCAN-Zeit, kann in Schritten von 30,3 ns (1 Tick) eingestellt werden. Die Pausenzeit muß min. 1 Tick betragen.

Die folgende Graphik zeigt das A/D-Timing für eine typische Erfassung mit …*AIScan*. Es gilt:

- Anzahl der Kanallistenabarbeitungen = 2
- Anzahl der Kanallisteneinträge = 3 (max. 1024 Einträge)

⇒ Anzahl der Messwerte = 6

Gestartet wird per Software:
(Erfassungsmodus: ME4000_AI_ACQ_MODE_SOFTWARE).

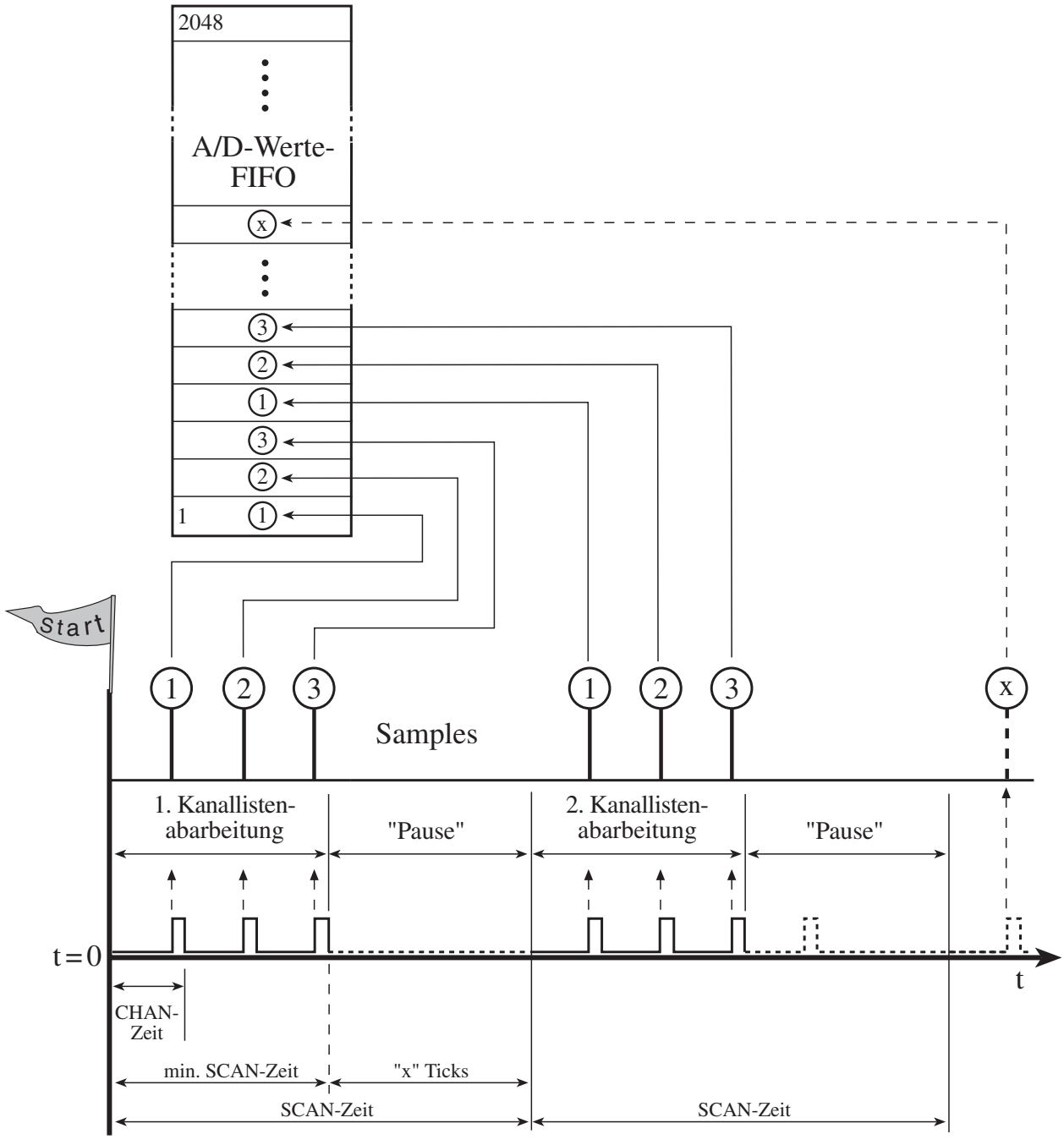


Abb. 25: A/D-Timing

4.1.3.2 Vorbereitung der Software

In einem zweiten Schritt wird die Software für die Erfassung vorbereitet. Je nachdem ob Sie eine bekannte Anzahl an Messwerten erfassen möchten (siehe Kap. 4.1.3.2.2) oder kontinuierlich erfassen möchten (siehe Kap. 4.1.3.2.1), stehen unterschiedliche Funktionalitäten zur Verfügung, die in den folgenden Kapiteln beschrieben werden.

Intern arbeitet der Treiber mit einem Ringpuffer, der mit den Messdaten gefüllt wird. Bei Bedarf hat der Anwender die Möglichkeit das interruptgesteuerte Schreiben der Messwerte in den Ringpuffer zu beeinflussen. In dem Parameter `<RefreshFrequency>` können Sie dazu einen Richtwert übergeben. Der Wert gibt stets die Anzahl der Kanallistenabarbeitungen an, nach denen das A/D-Werte FIFO ausgelesen werden soll. Wenn der Anwender keine Vorgabe macht wird vom Treiber ein sinnvoller Wert verwendet.

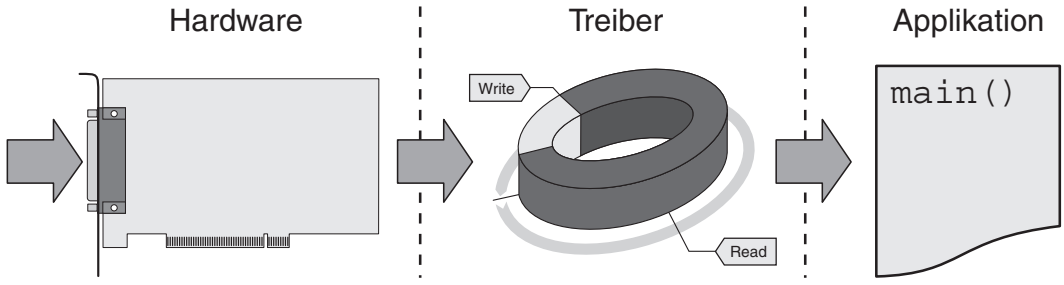


Abb. 26: Ringpuffer AI-Betrieb

4.1.3.2.1 Betriebsart „AIContinuous“

Die Funktion …*AIContinuous* dient der kontinuierlichen Erfassung einer unbekannten Anzahl an Messwerten. Das „Abholen“ der Daten kann entweder mit einer benutzerdefinierten Callback-Funktion oder durch wiederholten Aufruf der Funktion …*AIGetNewValues* erfolgen. Die Erfassung wird in dieser Betriebsart grundsätzlich als Hintergrundprozeß gestartet, d. h. durch Aufruf der Funktion …*AIStart* wird automatisch ein neuer „Thread“ erzeugt. Parallel dazu können andere Aufgaben („Threads“) abgearbeitet werden.

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

a. Das „Abholen“ der Daten erfolgt im Hintergrund (asynchron) mit einer benutzerdefinierten Callback-Funktion.

b. Das „Abholen“ der Daten erfolgt im Hintergrund (asynchron) mit der Funktion …*AIGetNewValues*.

Beachten Sie auch die Programmierbeispiele im ME-SDK und die Funktionsbeschreibung auf Seite 115.

Zu a: Daten-Handling in der Betriebsart **„AIContinuous“** mit Hilfe einer benutzerdefinierten Callback-Funktion:

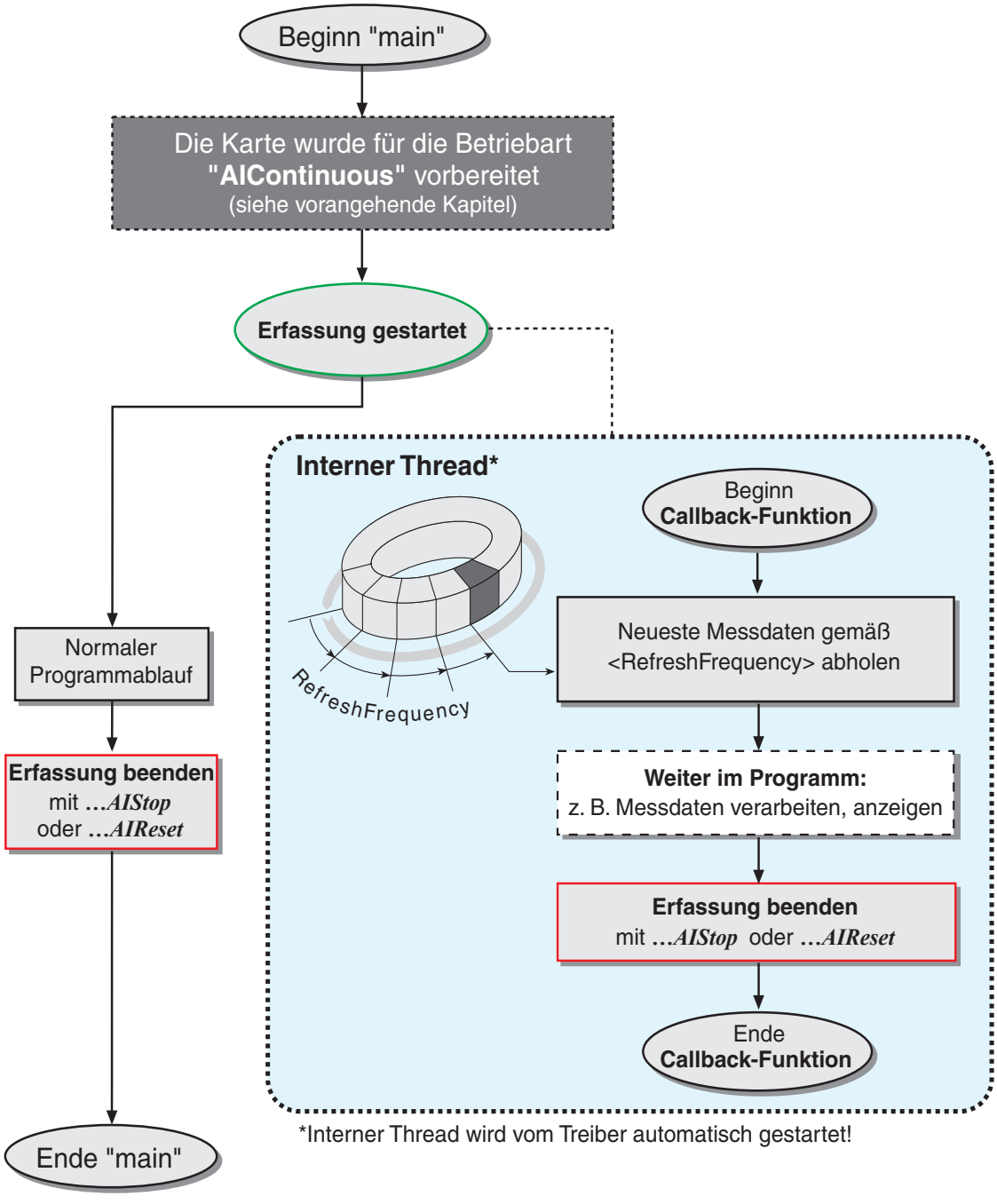


Abb. 27: Programmierung „AIContinuous“ mit Callback-Funktion

Zu b: Daten-Handling in der Betriebsart **„AIContinuous“** durch wiederholten Aufruf der Funktion *…AIGetNewValues* (BLOCKING oder NON_BLOCKING):

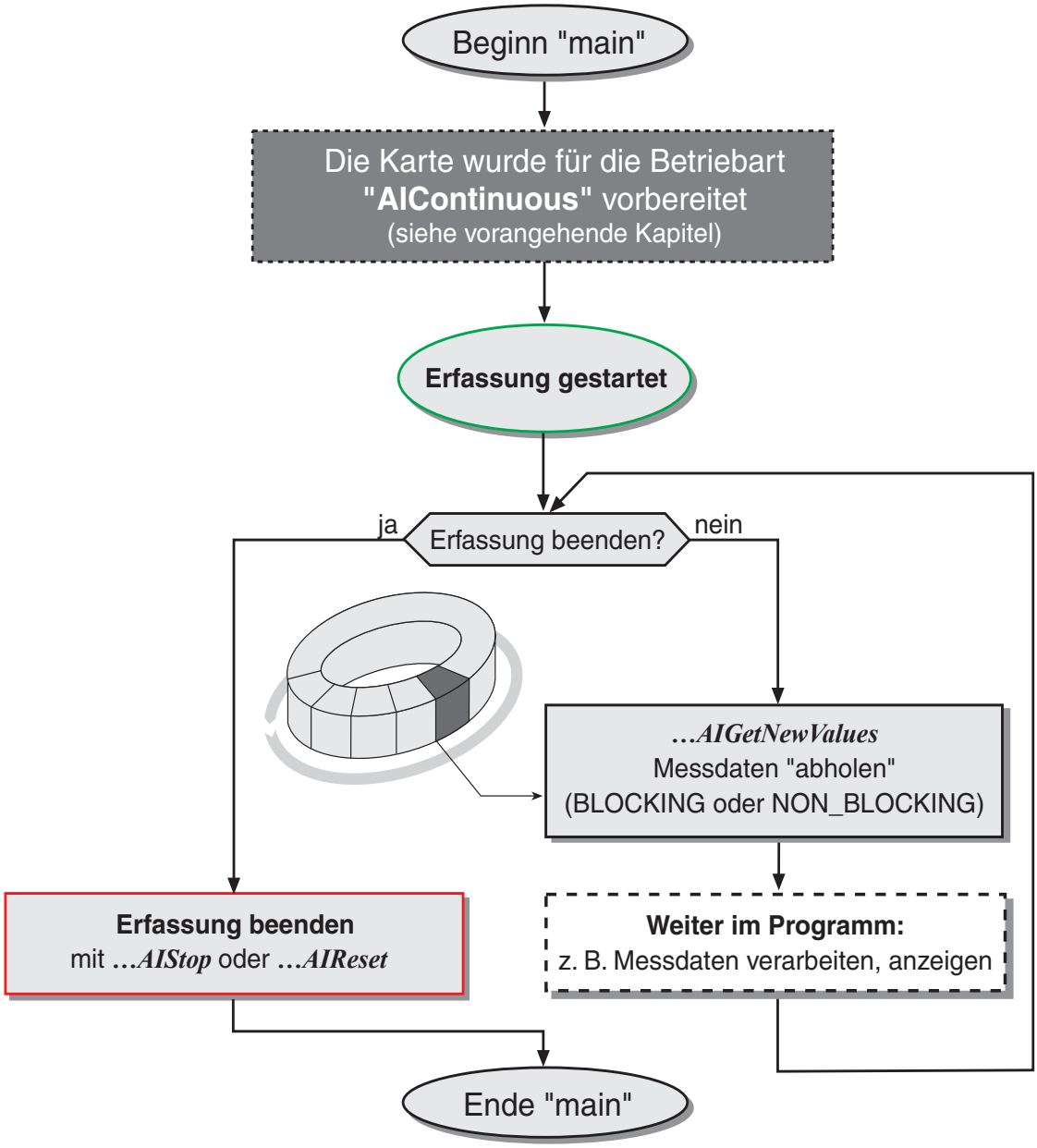


Abb. 28: Programmierung „AIContinuous“ mit …AIGetNewValues

4.1.3.2.2 Betriebsart „AIScan“

Die Funktion *…AIScan* dient der Erfassung einer bekannten Anzahl an Messwerten. Es wird ein benutzerdefinierter Datenpuffer allokiert in dem am Ende der Erfassung die Messwerte stehen. Im Ausführungsmodus „BLOCKING“ kehrt der „Thread“, in dem die Funktion *…AIStart* aufgerufen wurde, erst nach Erfassung des letzten Wertes zurück. Im Ausführungsmodus „ASYNCHRONOUS“ wird die Erfassung als Hintergrundprozeß gestartet, d. h. durch Aufruf der Funktion *…AIStart* wird automatisch ein neuer „Thread“ erzeugt. Parallel dazu können andere Aufgaben („Threads“) abgearbeitet werden. Falls gewünscht (z. B. bei einer längeren Erfassung), können Sie die Messwerte bereits während der Erfassung „einsehen“. Dies kann entweder mit einer benutzerdefinierten Callback-Funktion oder durch Aufruf der Funktion *…AIGetNewValues* erfolgen. Mit einer „Terminate“-Funktion können Sie (falls gewünscht) das Ende der Erfassung an Ihre Applikation melden lassen.

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

a. Die Erfassung der Daten blockiert den Programmfluß bis alle Werte im Datenpuffer stehen (Ausführungsmodus BLOCKING der Funktion *…AIScan*).

b. Die Erfassung der Daten erfolgt im Hintergrund mit einer benutzerdefinierten Callback-Funktion (Ausführungsmodus ASYNCHRONOUS der Funktion *…AIScan*).

c. „Einsehen“ der Daten mit der Funktion *…AIGetNewValues* während im Hintergrund die Erfassung läuft (Ausführungsmodus ASYNCHRONOUS der Funktion *…AIScan*).

Beachten Sie auch die Programmierbeispiele im ME-SDK und die Funktionsbeschreibung auf Seite 125.

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

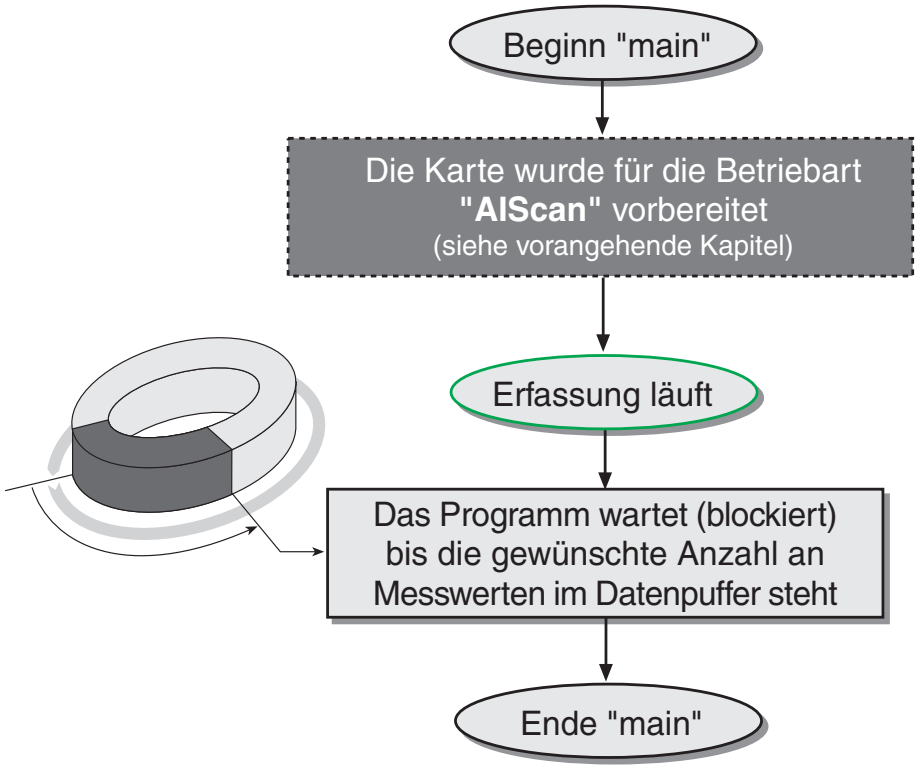


Abb. 29: Programmierung „AIScan“ im BLOCKING-Mode

Zu b: Daten-Handling in der Betriebsart **„AIScan“** im Ausführungsmodus „ASYNCHRONOUS“ mit Hilfe einer benutzerdefinierten Callback-Funktion:

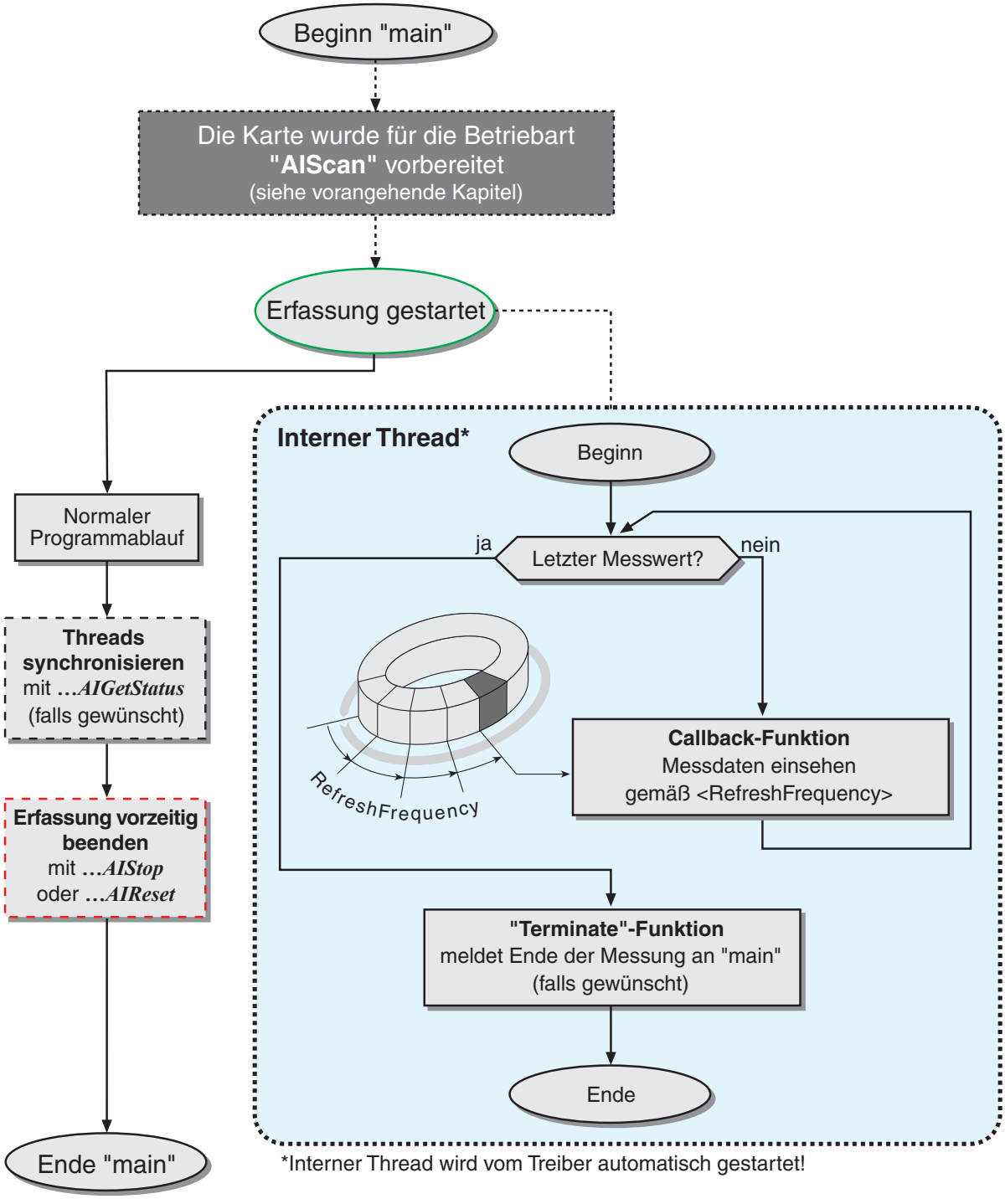


Abb. 30: Programmierung „AIScan“ mit Callback-Funktion

Zu c: Daten-Handling in der Betriebsart **„AIScan“** im Ausführungsmodus „ASYNCHRONOUS“ mit Hilfe der Funktion *…AIGetNewValues* (BLOCKING oder NON_BLOCKING).

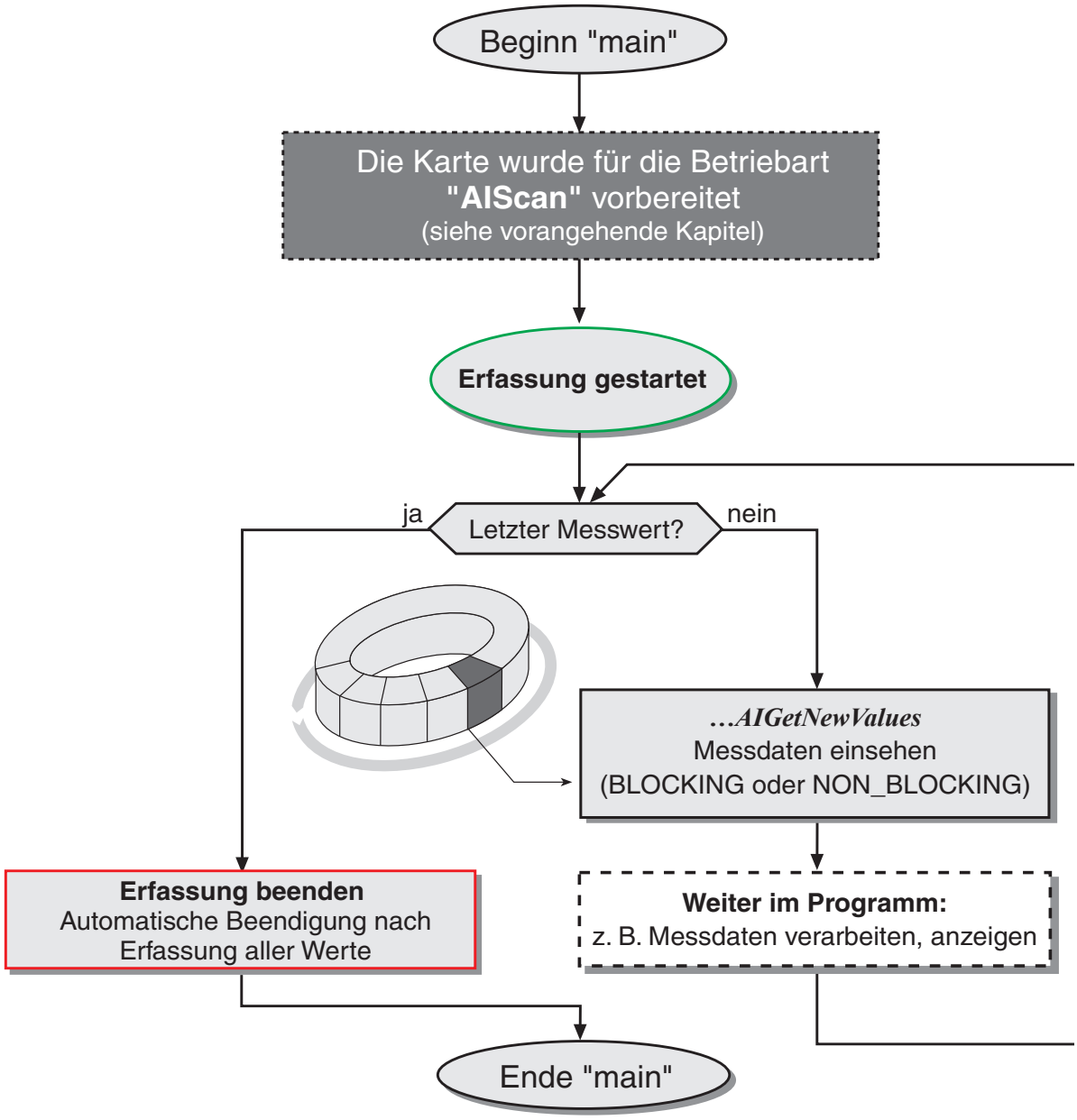


Abb. 31: Programmierung „AIScan“ mit …AIGetNewValues

4.1.3.3 Starten der Erfassung

Zum „scharfschalten“ der Erfassung muß stets die Funktion …*AIStart* aufgerufen werden. Je nach „Erfassungsmodus“ wird die Erfassung sofort nach Aufruf der Funktion gestartet (Software-Start) oder 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 Erfassung abbrechen. Die Erfassung in Abhängigkeit eines externen Triggersignals wird im Kapitel 4.1.4 näher beschrieben.

4.1.3.4 Stoppen der Erfassung

In der Betriebsart „AIContinuous“ wird die Erfassung mit der Funktion …*AIStop* sofort beendet.

In der Betriebsart „AIScan“ wird die Erfassung automatisch nach Erfassung der erwarteten Messwerte beendet.

Sofern zwischenzeitlich die Betriebsart nicht gewechselt wurde, kann die Erfassung mit der Funktion …*AIStart* jederzeit von vorne gestartet werden.

4.1.4 Externer Trigger A/D-Teil

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Je nach Applikation können Sie zwischen verschiedenen „Erfassungsmodi“ wählen, die jeweils unterschiedliche Philosophien verfolgen. Dies gilt unabhängig vom Triggermodus (analog oder digital) und Triggerflanke (auf steigende, fallende oder auf beide Flanken). Zum freischalten des externen Triggers wählen sie im Parameter `<AcqMode>` der Funktion …*AIConfig* einen der im Folgenden beschriebenen Erfassungsmodi. Vergessen Sie nicht die Karte mit der Funktion …*AIStart* für die Erfassung „scharf zu schalten“.

4.1.4.1 Erfassungsmodus „Extern-Standard“

(ME4000_AI_ACQ_MODE_EXT)

Bereit zur Erfassung nach Aufruf der Funktion …*AIStart*. Die Erfassung beginnt mit dem ersten ext. Triggerimpuls. Abarbeitung gemäß CHAN- und SCAN-Timer. Weitere Triggerimpulse bleiben ohne Wirkung. Optional können Sie ein Zeitintervall (Time-Out) angeben, in dem der externe Triggerimpuls eintreffen muß, ansonsten wird die Operation abgebrochen (Parameter `<TimeOut-Seconds>`).

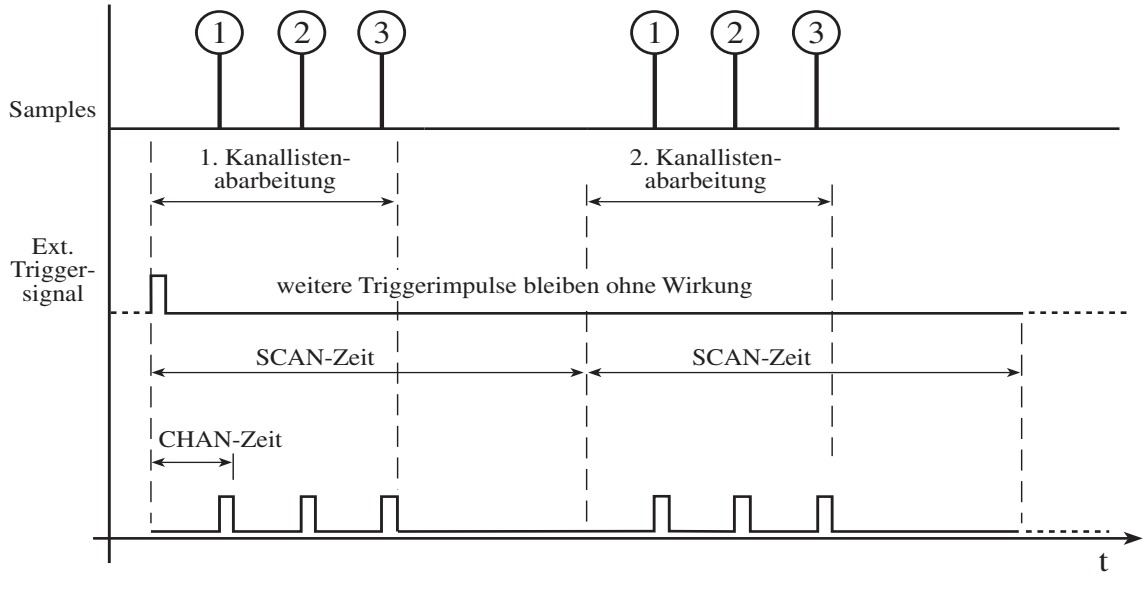


Abb. 32: Erfassungsmodus „Extern-Standard“

4.1.4.2 Erfassungsmodus „Extern-Einzelwert“

(ME4000_AI_ACQ_MODE_EXT_SINGLE_VALUE)

Bereit zur Erfassung nach Aufruf der Funktion *…AIStart.* Mit jedem ext. Triggerimpuls wird der jeweils nächste Wert gemäß Kanalliste gewandelt. Beachten Sie, daß nach Eintreffen des ersten Triggersignals bis zur ersten Wandlung einmalig die CHAN-Zeit vergeht. Ansonsten sind die Timer-Einstellungen ohne Wirkung. Die Periodendauer des externen Triggersignals darf 2 µs nicht unterschreiten.

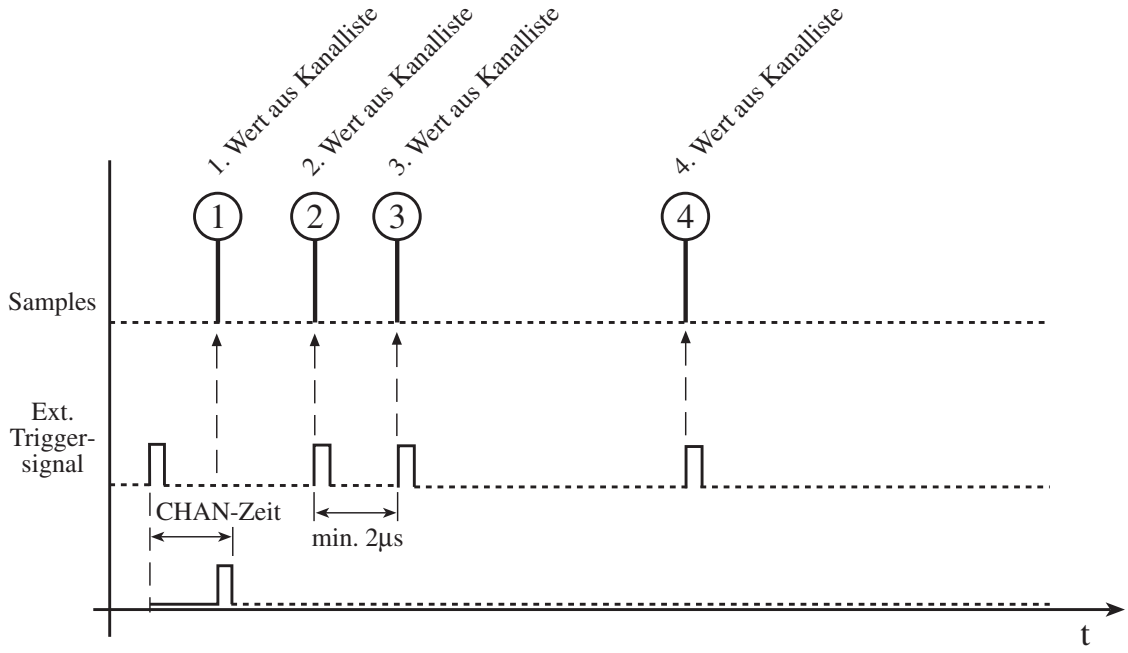


Abb. 33: Erfassungsmodus „Extern-Einzelwert“

Optional können Sie ein Zeitintervall (Time-Out) angeben, in dem der **erste** Triggerimpuls eintreffen muß, ansonsten wird die Operation abgebrochen (Parameter `<TimeOutSeconds>`). Das Ausbleiben weiterer Triggersignale wird dadurch nicht abgefangen. Berücksichtigen Sie dies gegebenenfalls bei der Wahl des Ausführungsmodus.

4.1.4.3 Erfassungsmodus „Extern-Kanalliste“

(ME4000_AI_ACQMODE_EXT_CHANNELLIST)

Bereit zur Erfassung nach Aufruf der Funktion …*AIStart*. Mit jedem ext. Triggerimpuls wird die Kanalliste einmal abgearbeitet. Die Erfassung erfolgt gemäß CHAN-Timer. Beachten Sie, daß nach Eintreffen des ersten Triggersignals bis zur ersten Wandlung einmalig die CHAN-Zeit vergeht. Der SCAN-Timer bleibt ohne Wirkung! Für die „min. Trigger-Zeit“, d. h. die minimale Periodendauer des externen Triggersignals gilt:

(Anzahl der Kanallisteneinträge x CHAN-Zeit) + 2µs

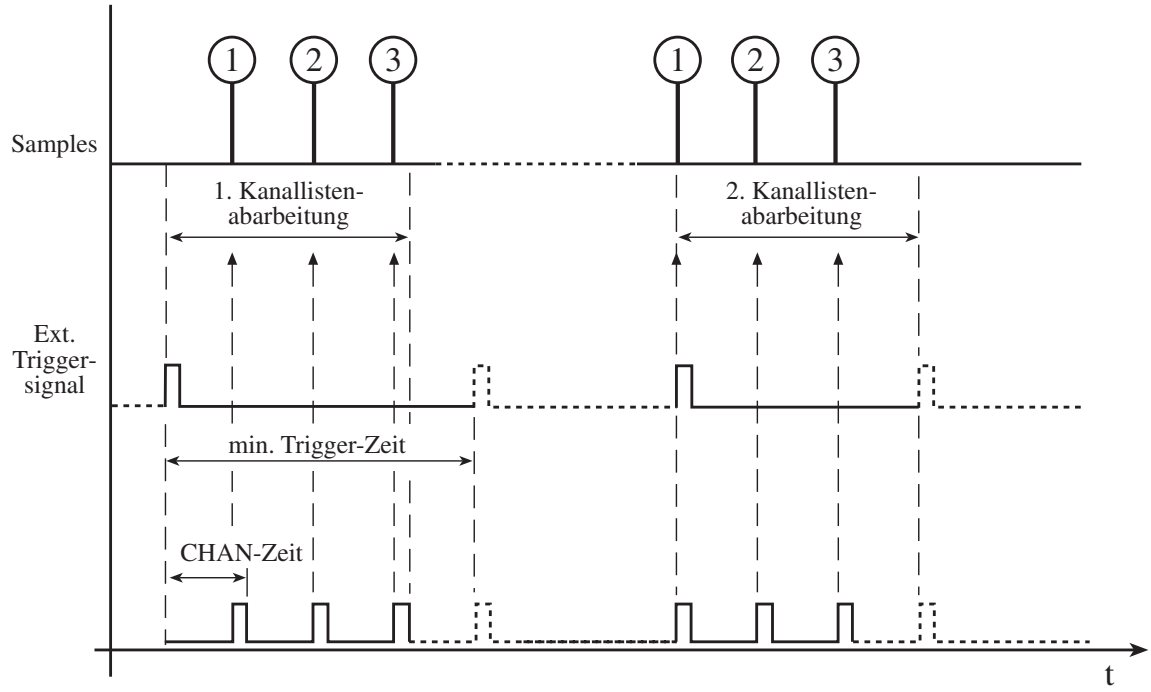


Abb. 34: Erfassungsmodus „Extern-Kanalliste“

Optional können Sie ein Zeitintervall (Time-Out) angeben, in dem der **erste** Triggerimpuls eintreffen muß, ansonsten wird die Operation abgebrochen (Parameter `<TimeOutSeconds>`). Das Ausbleiben weiterer Triggersignale wird dadurch nicht abgefangen. Berücksichtigen Sie dies gegebenenfalls bei der Wahl des Ausführungsmodus.

4.2 D/A-Teil

Betriebsart	Anwendung	Trigger	Timing
AOSingle (siehe Seite 51)	1 Wert auf 1 Kanal ausgeben	Software-Start, ext. digital	–
AOSimultaneous (siehe Seite 52)	Mehrere Kanäle simultan ausgeben	Software-Start, ext. digital	–
AOContinuous (siehe Seite 56)	Timergesteuerte Ausgabe kontinuierlich sich ändernder Werte	Software-Start, ext. digital	D/A-Timer (…AOConfig)
AOWraparound (siehe Seite 60)	Timergesteuerte Ausgabe sich wiederholender Werte	Software-Start, ext. digital	D/A-Timer (…AOConfig)

Tabelle 3: D/A-Betriebsarten

4.2.1 Betriebsart „AOSingle“

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

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. Es ist keine weitere Konfiguration nötig (siehe auch Funktionsbeschreibung auf Seite 140).

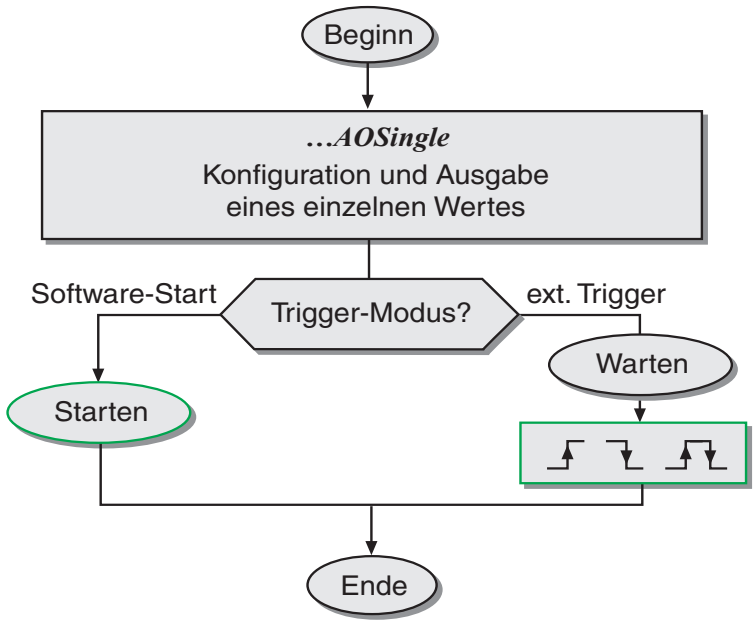


Abb. 35: Programmierung „AOSingle“

4.2.2 Betriebsart „AOSimultaneous“

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

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.

Das folgende Diagramm soll die grundsätzliche Vorgehensweise erläutern (siehe auch Funktionsbeschreibung auf Seite 142 sowie Beispielprogramme im ME-SDK):

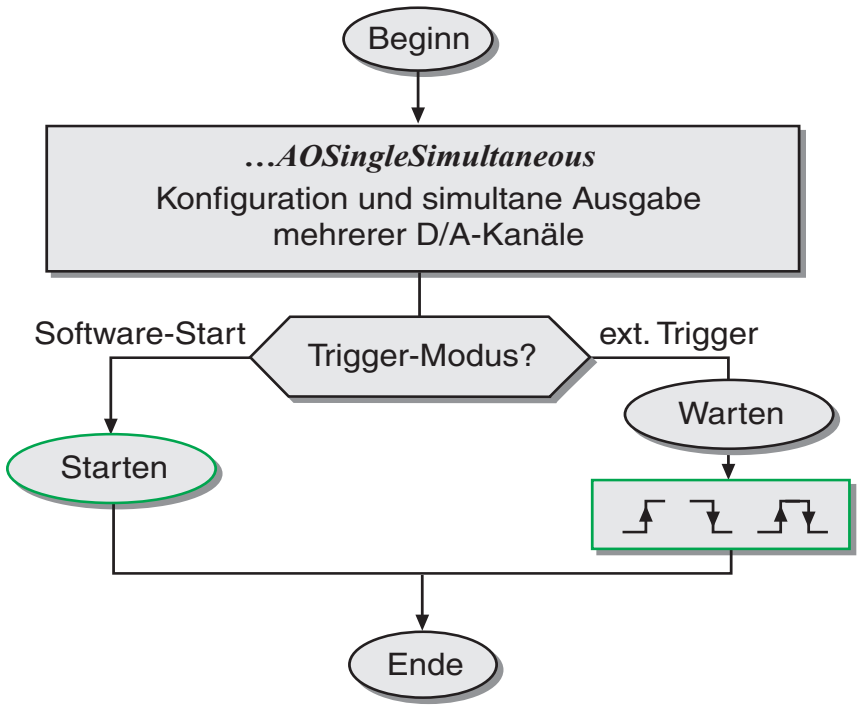


Abb. 36: Programmierung „AOSimultaneous“

4.2.3 Timergesteuerte „AO-Betriebsarten“

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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

1. Konfiguration der Hardware je Kanal mit …*AOConfig* (siehe Kap. 4.2.3.1)
2. Vorbereitung der Software je Kanal mit …*AOContinuous* oder …*AOWraparound* (siehe Kap. 4.2.3.2)
3. Start der Ausgabe mit …*AOStart* bzw. …*AOStartSynchronous* (siehe Kap. 4.2.3.3)

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 37 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.

4.2.3.1 Konfiguration des D/A-Teils

In einem ersten Schritt wird die Hardware für jeden Kanal mit der Funktion *…AOConfig* konfiguriert (Funktionsbeschreibung siehe Seite 134).

- 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 ≈ 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.

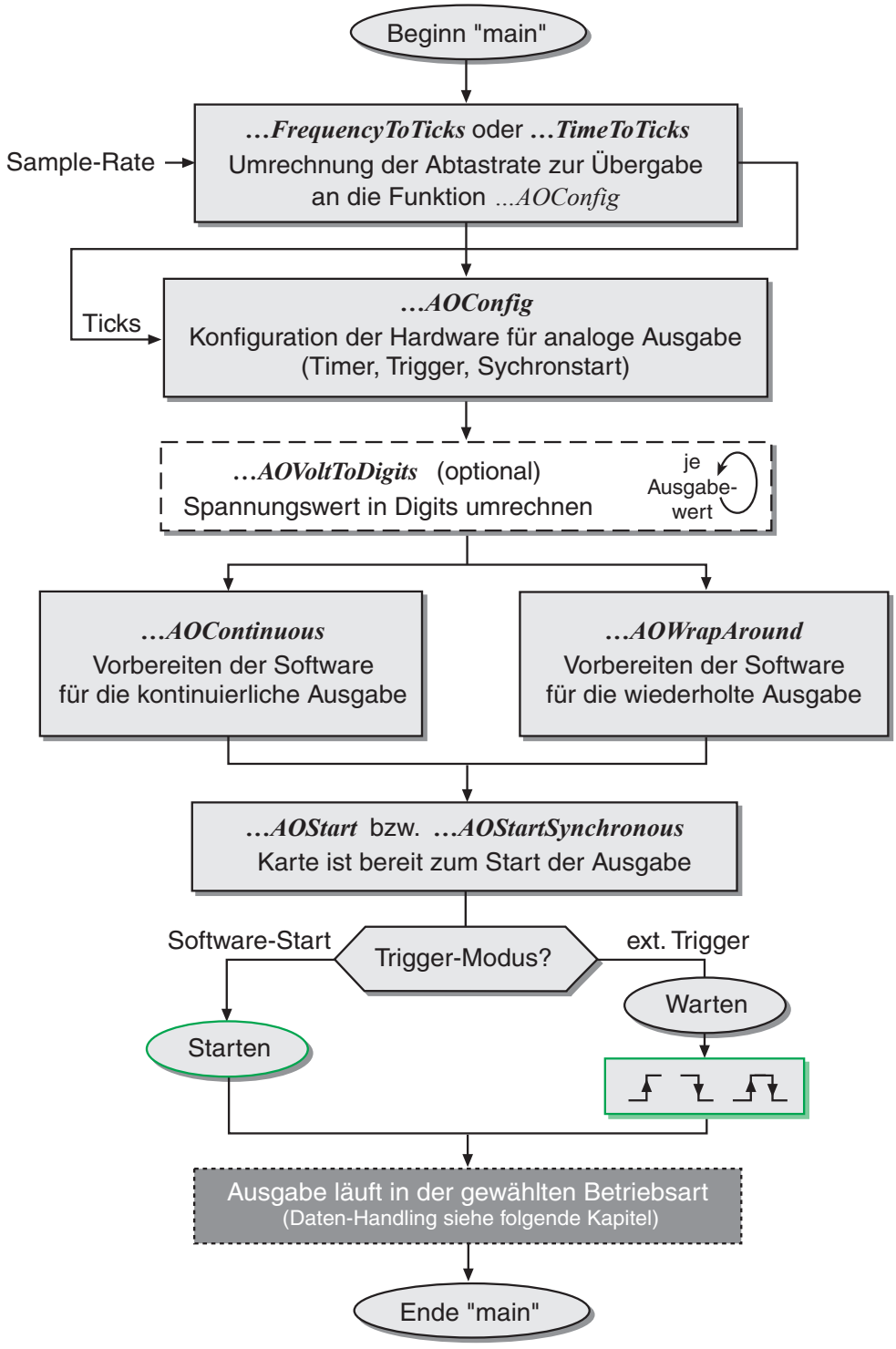


Abb. 37: Vorgehensweise Programmierung AO-Teil

4.2.3.2 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. 4.2.3.2.1) oder die gleichen Werte periodisch ausgegeben möchten (siehe Kap. 4.2.3.2.2), 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.

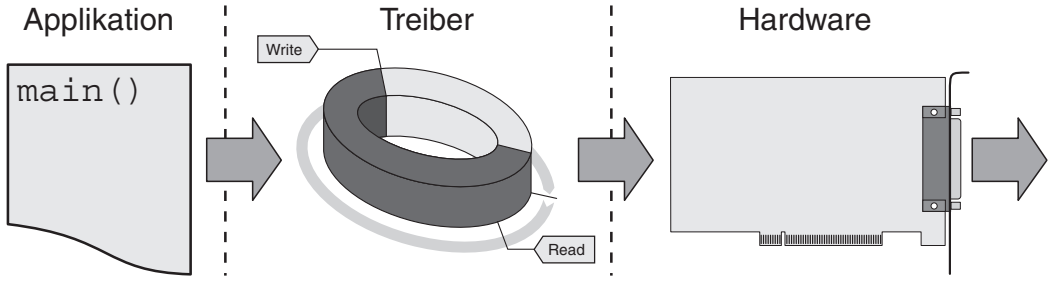


Abb. 38: Ringpuffer AO-Betrieb

4.2.3.2.1 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 „AOWraparound“). 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 136.

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

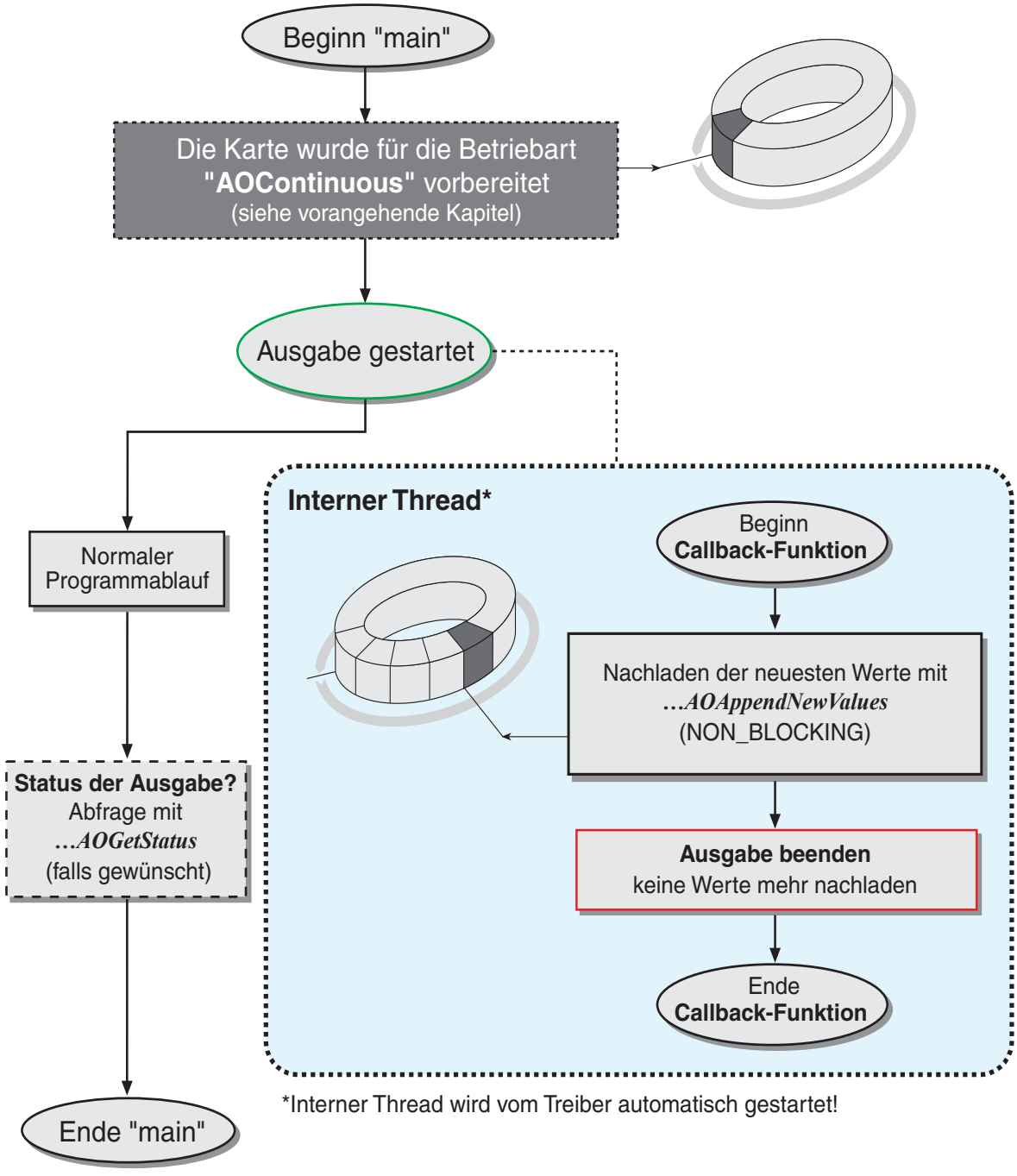


Abb. 39: Programmierung „AOContinuous“ mit Callback-Funktion

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

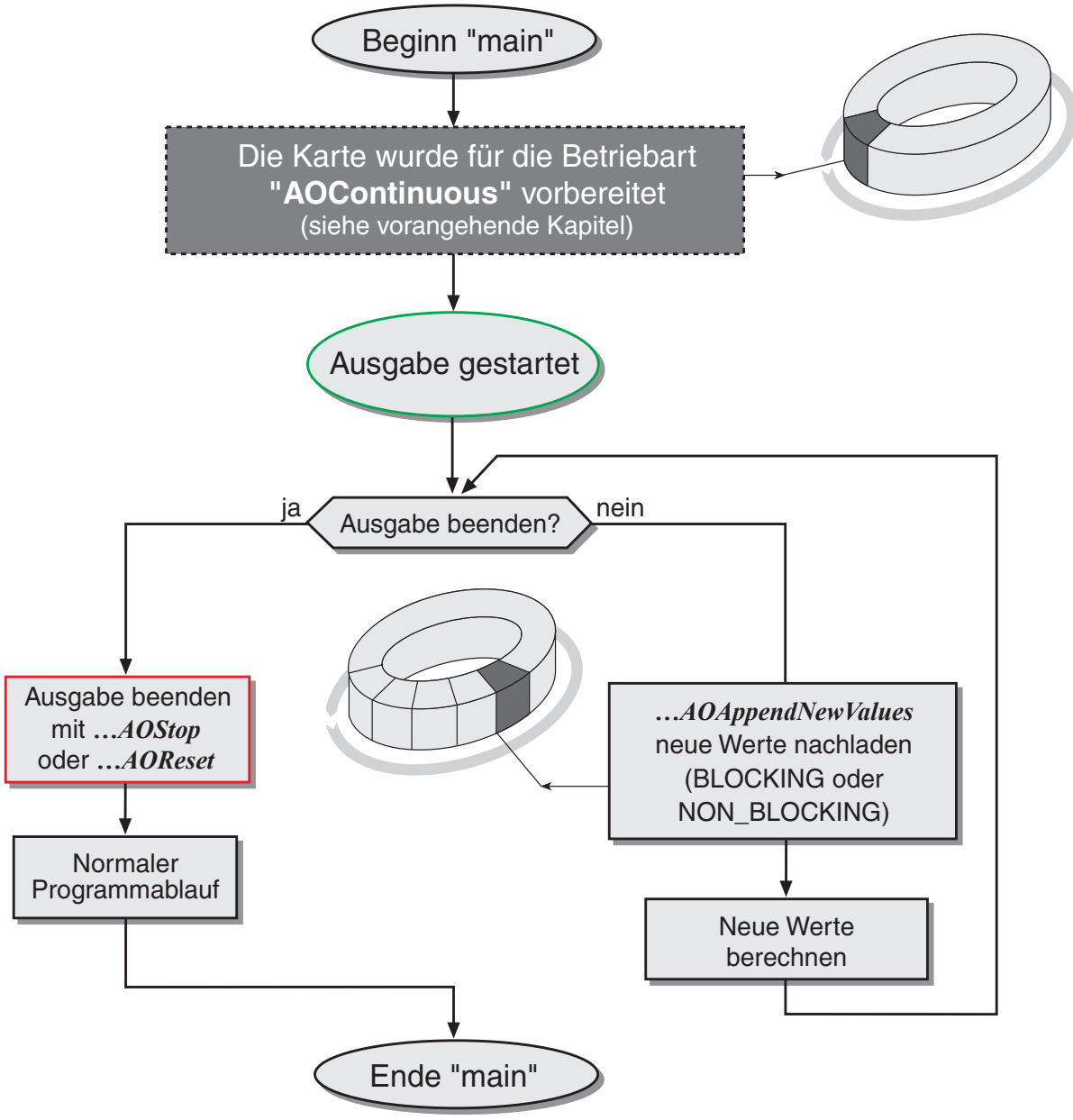


Abb. 40: Programmierung „AOContinuous“ ohne Callback-Funktion

4.2.3.2.2 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 4096 Werte nicht übersteigt und die Ausgabe „unendlich“ erfolgt, läuft die Ausgabe auf Firmware-Ebene, d. h. der Host-Rechner wird nicht belastet!

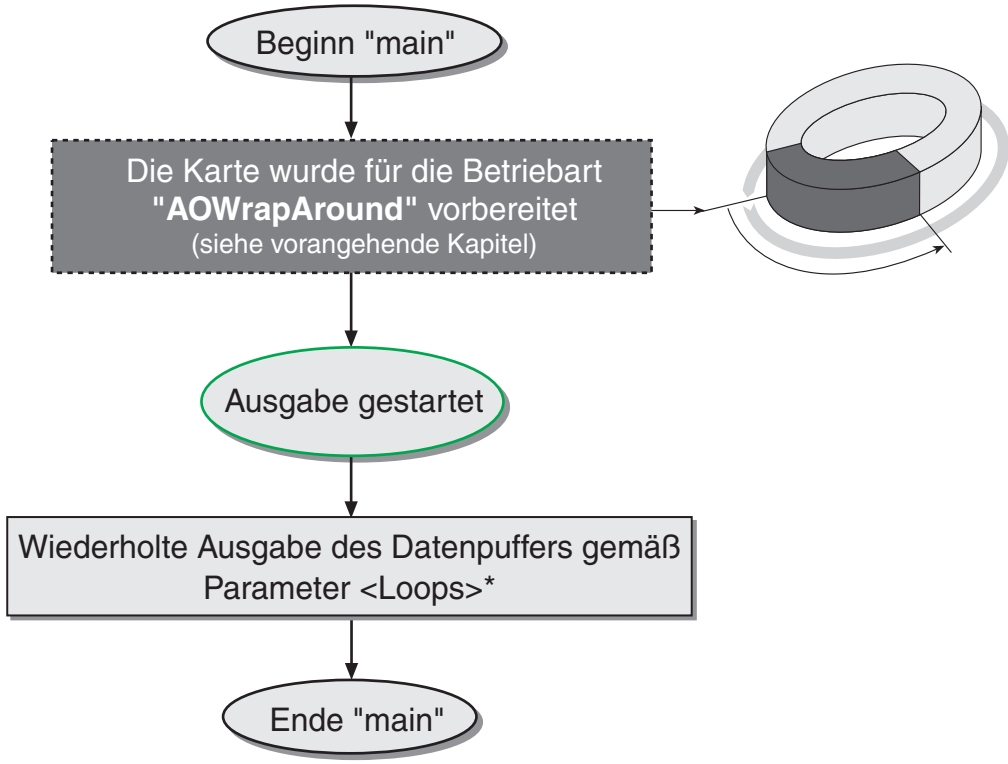
Die Diagramme auf den folgenden Seiten beschreiben den Programm-Fluss unter folgenden Bedingungen:

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 152.

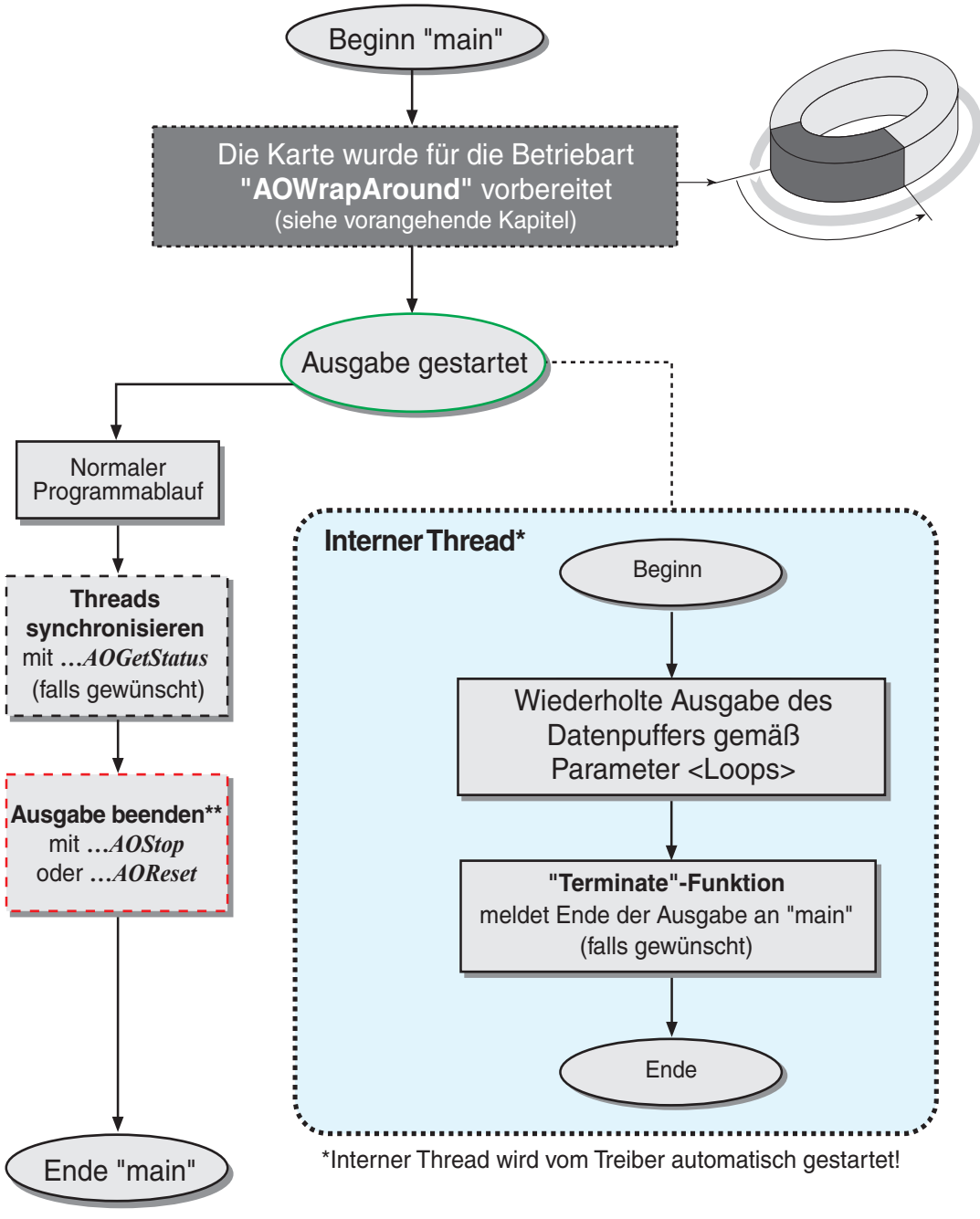
Zu a: Daten-Handling in der Betriebsart **„AOWraparound“** im Ausführungsmodus „BLOCKING“:



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

Abb. 41: 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. 42: Programmierung „AOWraparound“ im „ASYNCHRONUOUS“-Mode

4.2.3.3 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.

4.2.3.4 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.

4.3 Digital-I/O-Teil

Alle Modelle der ME-4600 Serie verfügen über vier 8 Bit breite Digital-I/O-Ports (A, B, C, D). Bei Karten mit „Optoisolierung“ ist Port A als Ausgang und Port B als Eingang festgelegt. Port C und D sind grundsätzlich nicht optoisoliert.

Zur Beschaltung des Digital-I/O-Teils lesen Sie bitte Kap. 3.5 "Digital-I/O-Teil".

4.3.1 Einfache Ein-/Ausgabe

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Sofern Ihre Karte keine Optoisolierung hat, kann jeder Port unabhängig als Ein- oder Ausgang konfiguriert werden. Nach dem Einschalten der Versorgung sind alle Ports auf Eingang geschaltet mit Ausnahme von Port A (= Ausgang) bei den optoisolierten Modellen („i“-Versionen). Verwenden Sie die Funktion *…DIOConfig* um die Port-Richtung zu definieren.

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

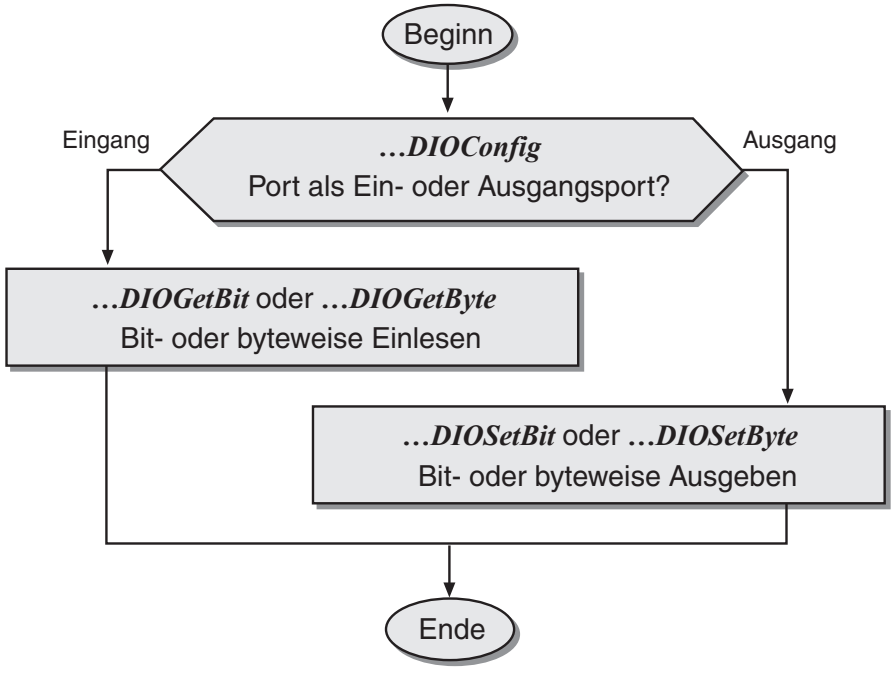


Abb. 43: Programmierung „DIO-Standard“

4.3.2 Bitmuster-Ausgabe

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Als besonderes Leistungsmerkmal bietet die ME-4680 eine timergesteuerte Bitmuster-Ausgabe. Hierzu wird das FIFO von D/A-Kanal 3 „zweckentfremdet". Getrennt nach „Low-Byte" (FIFO_LOW_BYTE) und „High-Byte" (FIFO_HIGH_BYTE) können die 16 Bit breiten FIFO-Werte (= Bitmuster) byteweise den 8 Bit breiten Digital-Ports (A, B, C, D) zugeordnet werden (siehe Funktion …*DIOBPPortConfig*). Ein für die Bitmuster-Ausgabe verwendeter Port ist automatisch Ausgangs-Port (kein …*DIO-PortConfig* nötig). Eingangsport Port B der optoisolierten Versionen kann nicht für Bitmuster-Ausgabe verwendet werden.

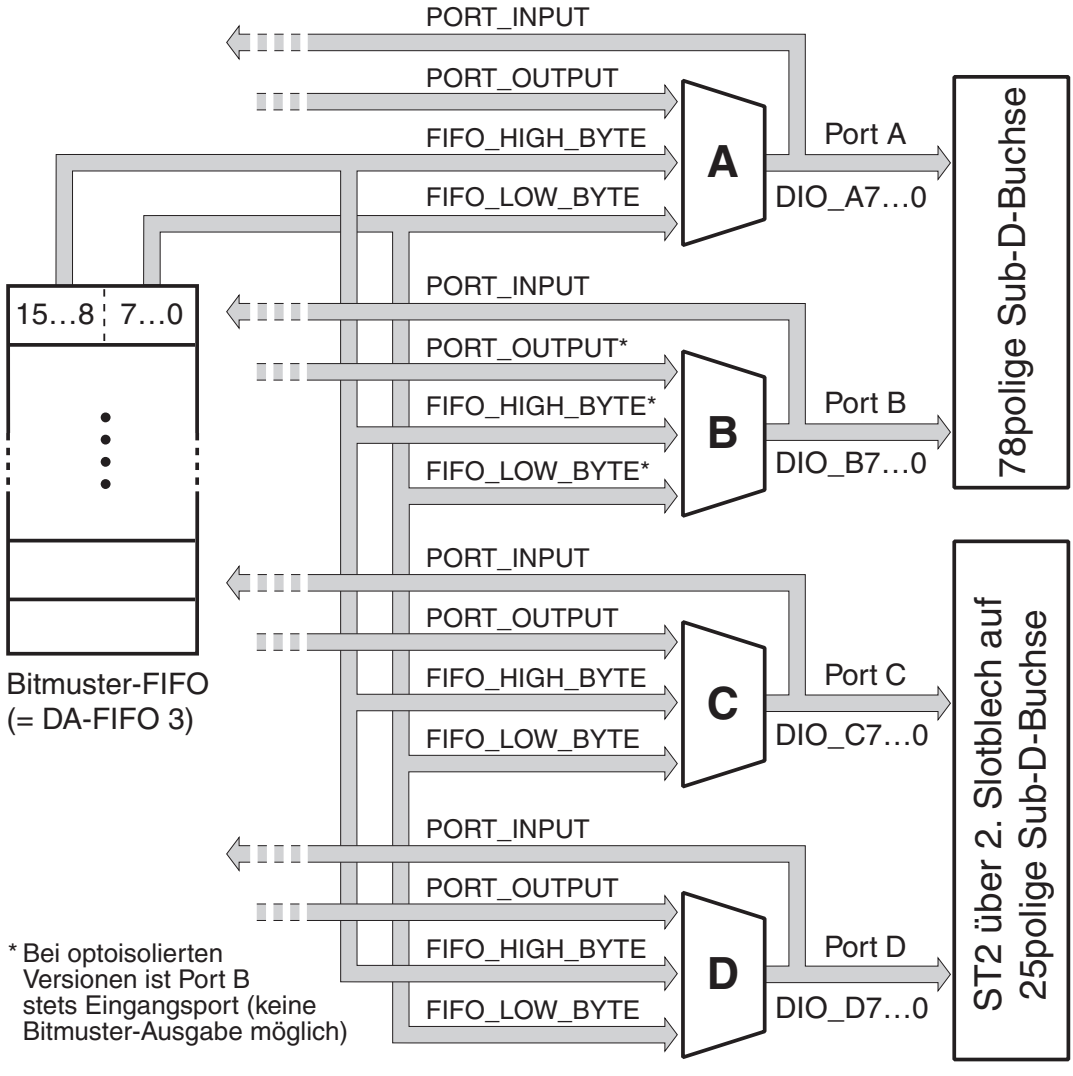


Abb. 44: Port-Mapping

Das folgende Diagramm soll die Vorgehensweise in den Betriebsarten **„BitPattern-Continuous“** und **„BitPattern-Wraparound“** veranschaulichen.

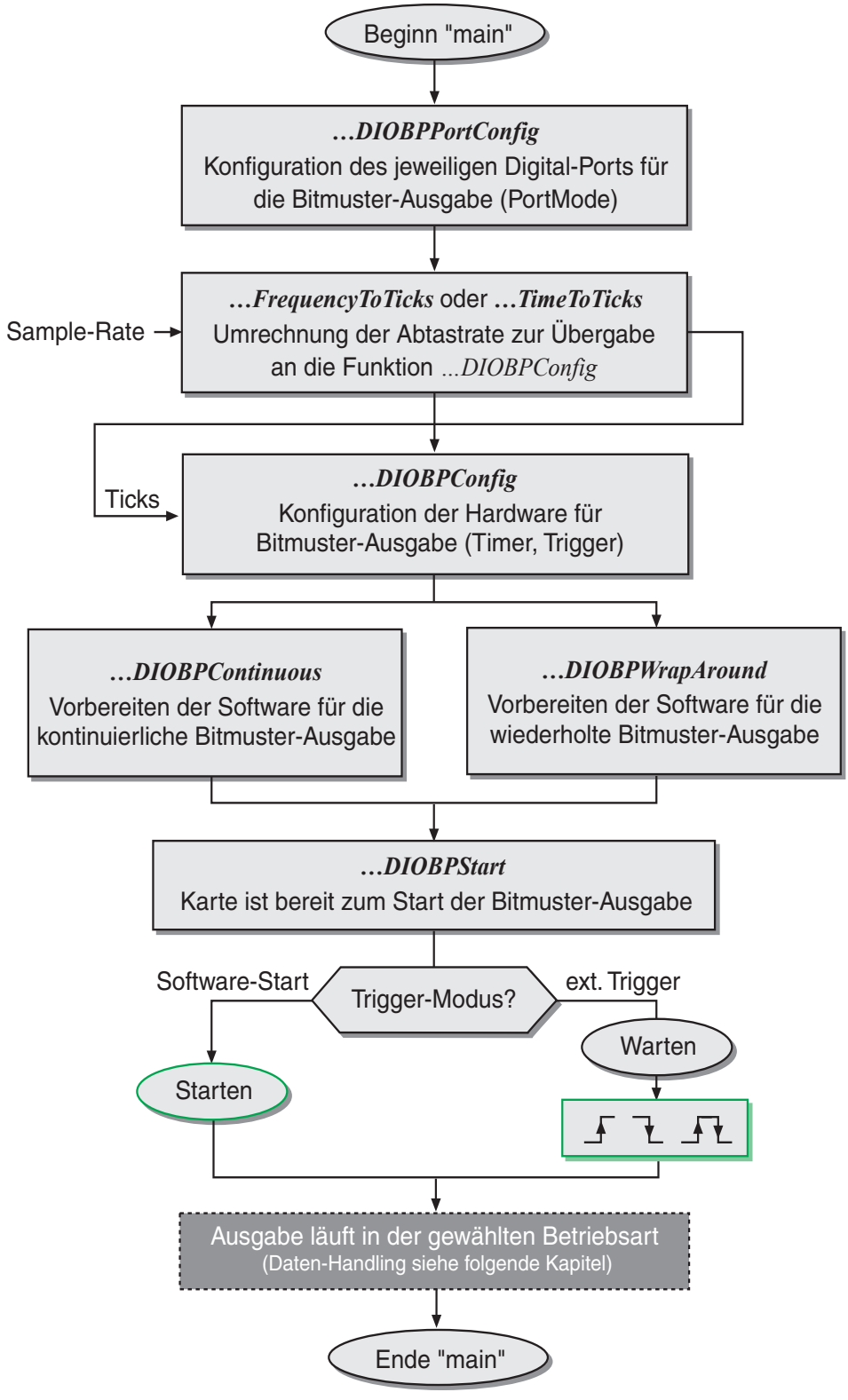


Abb. 45: Programmierung Bitmuster-Ausgabe

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

1. Konfiguration der Hardware mit *…DIOBPPortConfig und …DIOBPConfig* (siehe Kap. 4.2.3.1)
2. Vorbereitung der Software mit *…DIOBPContinuous* oder *…DIOBPWraparound* (siehe Kap. 4.2.3.2)
3. Start der Bitmuster-Ausgabe mit *…DIOBPStart* (siehe Kap. 4.2.3.3)

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

4.3.2.1 Konfiguration der Hardware

In einem ersten Schritt wird die Hardware für die Bitmusterausgabe konfiguriert.

- Wählen Sie zunächst den/die Port(s) auf welche(n) das Bitmuster ausgegeben werden soll (*…DIOBPPortConfig* auf Seite 169).
- Als Zeitgeber dient ein programmierbarer 32 Bit Zähler, der einen 33 MHz Takt als Zeitbasis verwendet. Daraus ergibt sich eine Periodendauer von 30,$\overline{30}$ns, die als kleinste Zeiteinheit definiert wird und im Folgenden „1 Tick“ genannt wird (siehe *…DIOBPConfig* auf Seite 157). Zur einfachen Umrechnung können Sie die Funktionen *…FrequencyToTicks* oder *…TimeToTicks* verwenden.
- Als Triggermodi stehen zur Verfügung (siehe *…DIOBPConfig* auf Seite 157):
  - Software-Start: Ausgabe wird unmittelbar nach Aufruf der Funktion *…DIOBPStart* gestartet.
  - Start durch externen Trigger-Impuls (steigende, fallende oder beliebige Flanke) am Eingang DA_TRIG_3 (Pin 65).

4.3.2.2 Vorbereitung der Software

In einem zweiten Schritt wird die Software für die Bitmuster-Ausgabe vorbereitet. Je nach dem ob Sie die auszugebenden Werte kontinuierlich nachladen (siehe Kap. 4.3.2.2.1) oder die gleichen Werte wiederholt ausgegeben möchten (siehe Kap. 4.3.2.2.2), stehen unterschiedliche Funktionalitäten zur Verfügung, die in den folgenden Kapiteln beschrieben sind.

Intern arbeiten die Funktionen *…DIOBPContinuous* und *…DIOBPWraparound* mit einem Ringpuffer, in den die auszugebenden Bitmuster geschrieben werden müssen (siehe Abb. 38).

4.3.2.2.1 Betriebsart „BitPattern-Continuous“

In dieser Betriebsart können Sie beliebige Bitmuster auf die digitalen Ports ausgeben, wobei sich die Bitmuster auch während der Ausgabe ändern bzw. neu berechnet werden können (im Gegensatz zur Betriebsart „BitPattern-Wraparound“). Allokieren Sie einen Datenpuffer definierter Größe mit den **ersten** auszugebenden Bitmustern. Vor Beginn der Ausgabe wird das erste Datenpaket in den Ringpuffer geschrieben. Der Timer gibt ein festes Zeitraster (Sample-Rate) für die Ausgabe der einzelnen Werte vor und muß mit der Funktion *…DIOBPConfig* vor Start der Ausgabe konfiguriert werden.

Das Nachladen erfolgt mit der Funktion *…DIOBPAppendNewValues*. 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 *…DIOBPAppendNewValues* im Rahmen einer benutzerdefinierten Callback-Funktion.

b. Die Ausgabe erfolgt im Hintergrund (asynchron) mit der Funktion *…DIOBPAppendNewValues*.

Beachten Sie auch die Beispiele im ME-SDK und die Funktionsbeschreibung auf Seite 159.

Zu a: Daten-Handling in der Betriebsart **„BitPattern-Continuous“** mit der Funktion …*DIOBPAppendNewValues* im Rahmen einer benutzerdefinierten Callback-Funktion:

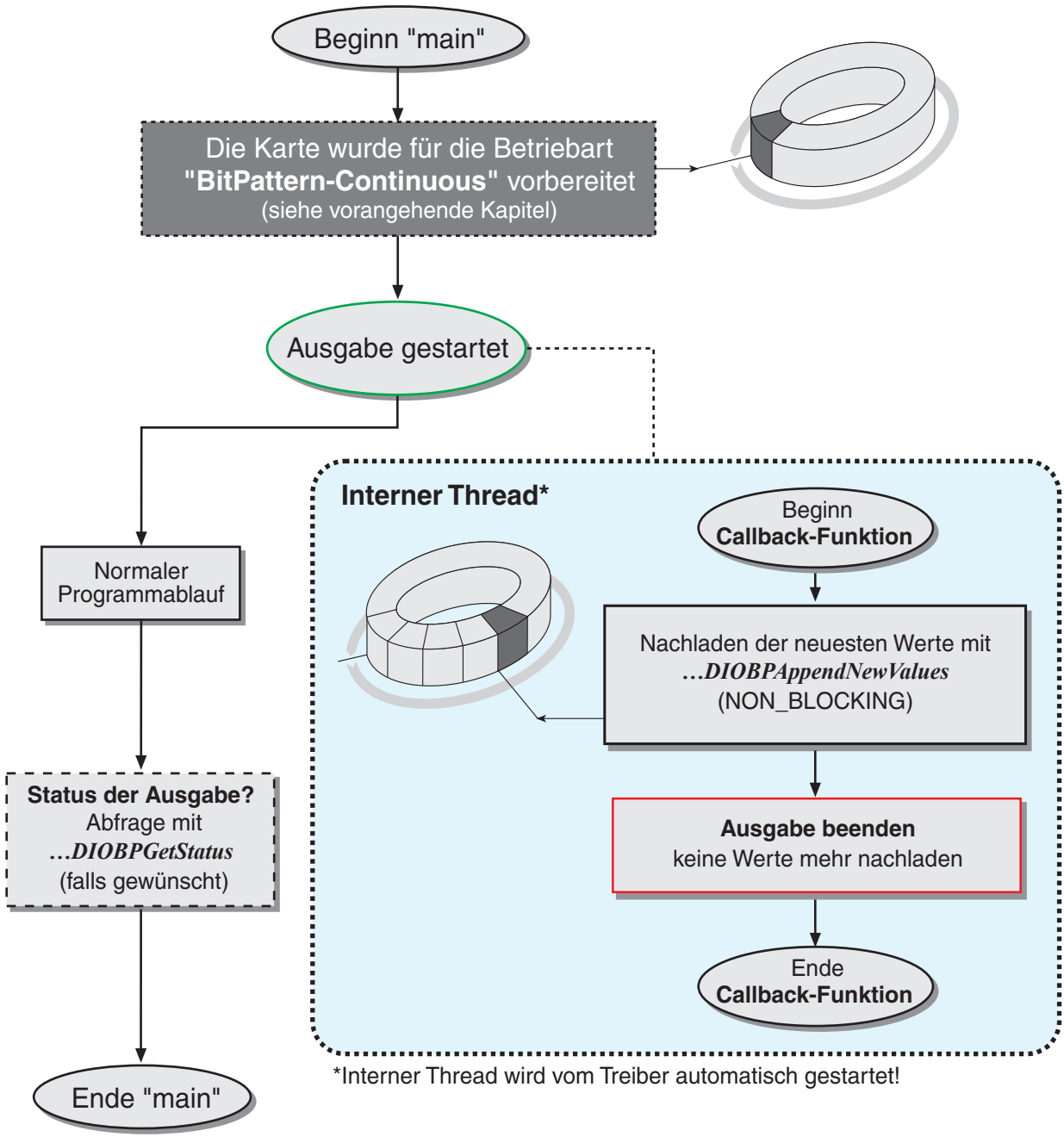


Abb. 46: Programmierung „BitPattern-Continuous“ mit Callback-Funktion

Zu b: Daten-Handling in der Betriebsart **„BitPattern-Continuous“** mit der Funktion …*DIOBPAppendNewValues*:

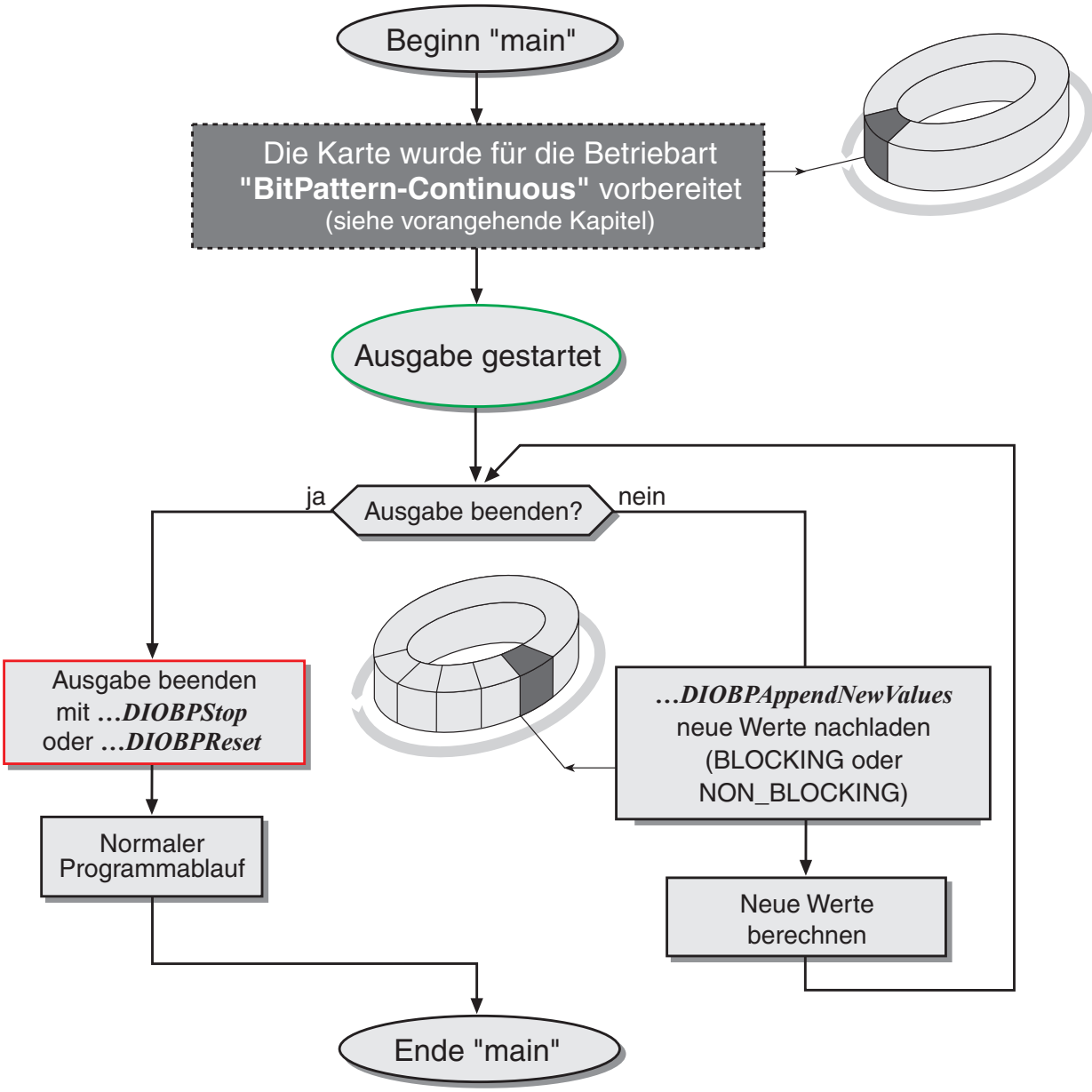


Abb. 47: Programmierung „BitPattern-Continuous“ ohne Callback-Funktion

4.3.2.2.2 Betriebsart „BitPattern-Wraparound“

In der Betriebsart „BitPattern-Wraparound“ können Sie ein stets sich wiederholendes Bitmuster ausgeben. Allokieren Sie einen Datenpuffer definierter Größe mit den wiederholt auszugebenden Bitmustern. Die Bitmuster werden beim Start **einmalig** in den Datenpuffer geschrieben. Der Timer gibt ein festes Zeitraster (Sample-Rate) für die Ausgabe der einzelnen Bitmuster vor. Die Konfiguration erfolgt mit der Funktion …*DIOBPConfig*.

Hinweis: Sofern die Größe des Datenpuffers 4096 Werte nicht übersteigt und die Ausgabe „unendlich“ erfolgt, läuft die Ausgabe auf Firmware-Ebene, d. h. der Host-Rechner wird nicht belastet!

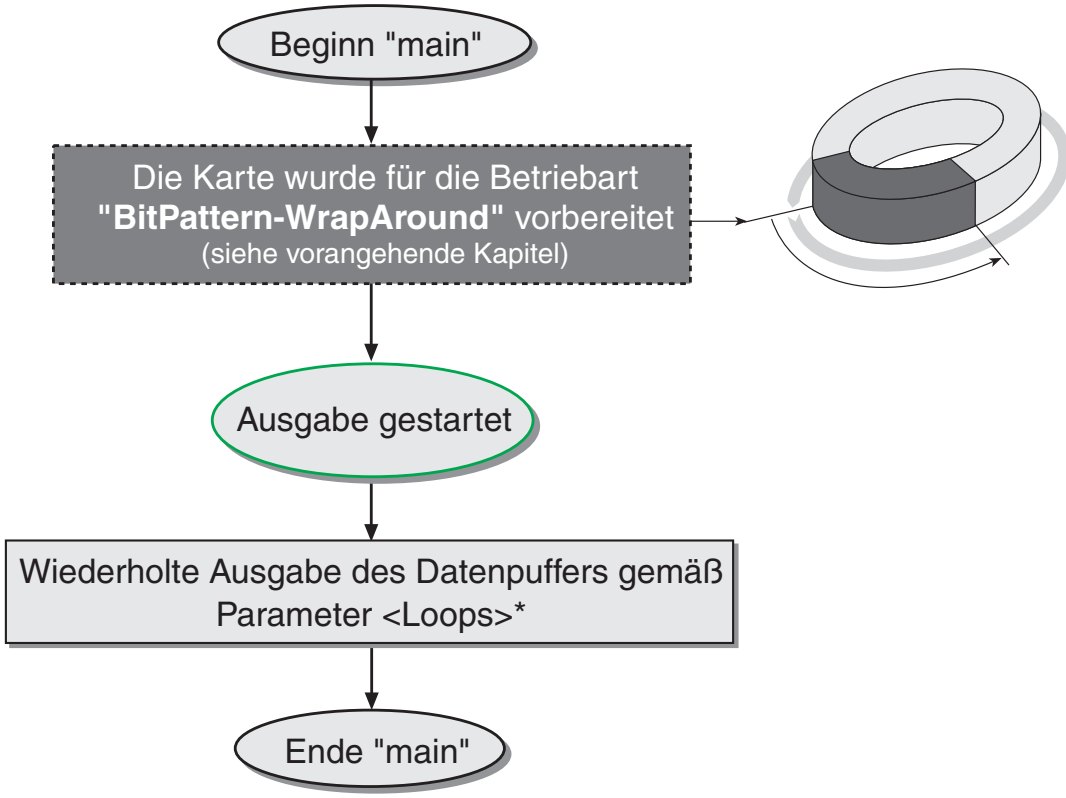
Die Diagramme auf den folgenden Seiten beschreiben den Programm-Fluss unter folgenden Bedingungen:

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 166.

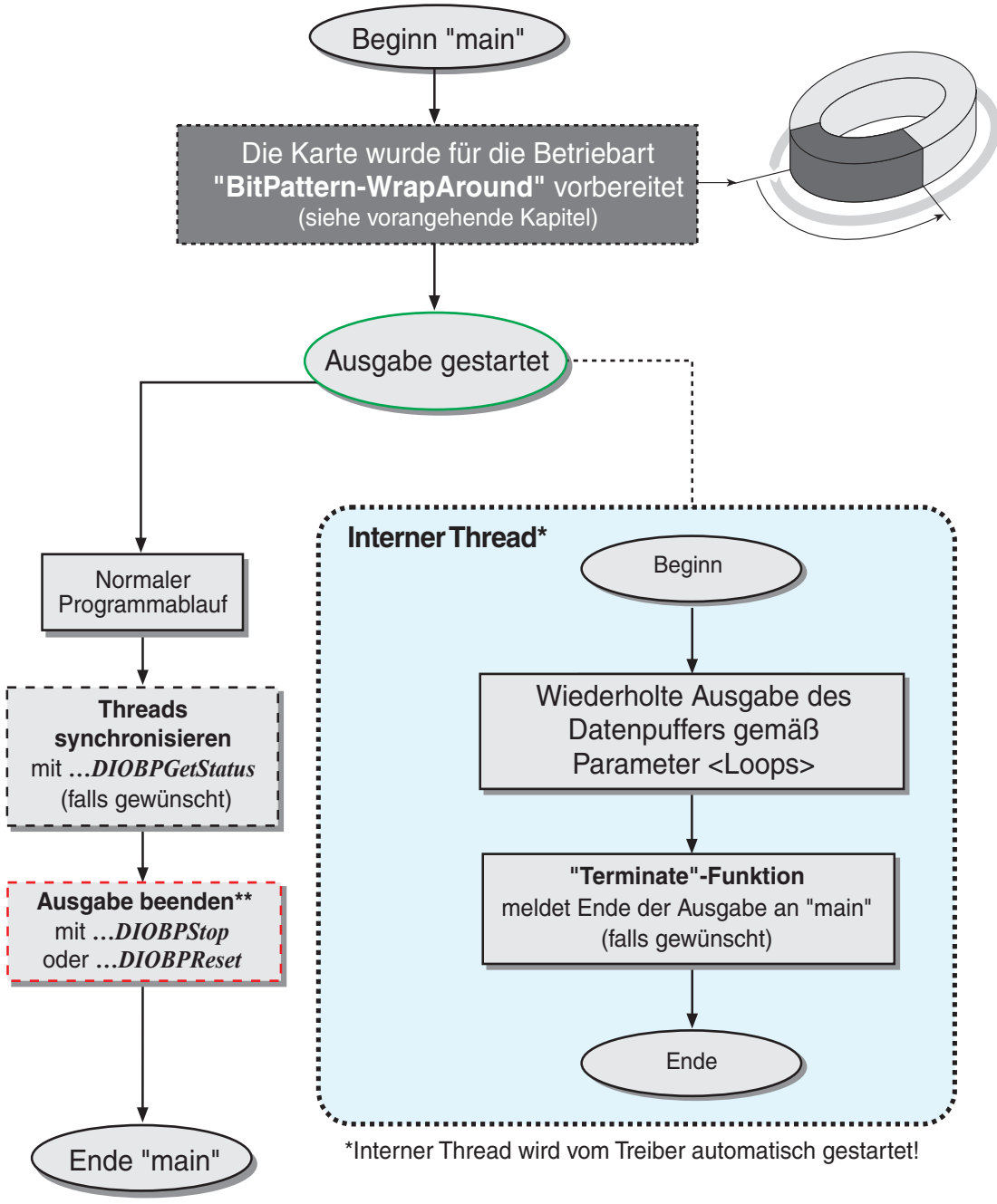
Zu a: Daten-Handling in der Betriebsart **„BitPattern-Wraparound“** im Ausführungsmodus „BLOCKING“:



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

Abb. 48: Programmierung mit …DIOBPWraparound im „BLOCKING“-Mode

Zu b: Programmierung in der Betriebsart **„BitPattern-Wraparound“** im Ausführungsmodus „ASYNCHRONOUS“:



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

Abb. 49: Programmierung mit ...DIOBPWraparound im „ASYNCHRONUOUS“-Mode

4.3.2.3 Starten der Bitmuster-Ausgabe

Zum „Scharfschalten“ der Bitmuster-Ausgabe muß stets die Funktion …*DIOBPStart* aufgerufen werden. Je nach „Trigger-Modus“ wird die Ausgabe sofort 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.

4.3.2.4 Stoppen der Ausgabe

In der Betriebsart „BitPattern-Continuous“ wird die Ausgabe in der Regel dadurch beendet, daß keine Werte mehr nachgeladen werden. Mit der Funktion …*DIOBPStop* können Sie die Ausgabe aber auch sofort beenden - danach wird das Bitmuster „0000Hex“ ausgegeben.

In der Betriebsart „BitPattern-Wraparound“ können Sie die Ausgabe mit der Funktion …*DIOBPStop* wahlweise sofort beenden (danach wird „0000Hex“ ausgegeben) oder definiert „anhalten“, d. h. die Ausgabe wird mit dem letzten Wert im Puffer und somit einem bekannten Bitmuster beendet.

Sofern zwischenzeitlich die Betriebsart nicht gewechselt wurde, kann die Ausgabe mit der Funktion …*DIOBPStart* jederzeit von vorne gestartet werden.

4.4 Zähler-Betriebsarten

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Die 3 frei verfügbaren 16 Bit Zähler (1 x 82C54) können unabhängig voneinander für folgende 6 Betriebsarten konfiguriert werden:

- Modus 0: Zustandsänderung bei Nulldurchgang
- Modus 1: Retriggerbarer „One Shot“
- Modus 2: Asymmetrischer Teiler
- Modus 3: Symmetrischer Teiler
- Modus 4: Zählerstart durch Softwaretrigger
- Modus 5: Zählerstart durch Hardwaretrigger

Verwenden Sie zur Programmierung der Zähler die Funktionen der ME-4600 Funktionsbibliothek (siehe Kap. 5.3.6 „Zählerfunktionen“ auf Seite 175).

4.4.1 Modus 0: Zustandsänderung bei Nulldurchgang

Diese Betriebsart ist z. B. zur Signalisierung eines Interrupts geeignet. Der Zähler-Ausgang (OUT_0…2) geht in den Low-Zustand (TTL: 0V/Opto: leitend), sobald der Zähler initialisiert wird oder ein neuer Startwert in den Zähler geladen wird. Zur Freigabe des Zählers muß der GATE-Eingang geeignet beschaltet werden (TTL: 5V/Opto: 0V). Sobald der Zähler geladen und freigegeben wurde, beginnt er abwärts zu zählen während der Ausgang im Low-Zustand bleibt (TTL: 0V/Opto: leitend).

Bei Erreichen des Nulldurchganges geht der Ausgang in den High-Zustand (TTL: +5V)/Opto: hochohmig) und bleibt dort, bis der Zähler neu initialisiert wird oder ein neuer Startwert geladen wird. Auch nach Erreichen des Nulldurchganges wird weiter abwärts gezählt. Sollte während des Zählvorganges ein Zählerregisters erneut geladen werden, hat dies zur Folge, daß

1. beim Schreiben des ersten Bytes der momentane Zählvorgang gestoppt wird

2. beim Schreiben des zweiten Bytes der neue Zählvorgang gestartet wird.

4.4.2 Modus 1: Retriggerbarer „One-Shot“

Der Zähler-Ausgang (OUT_0…2) geht in den High-Zustand (TTL: +5V)/Opto: hochohmig), sobald der Zähler initialisiert wird. Nachdem ein Startwert in den Zähler geladen wurde geht der Ausgang mit dem auf den ersten Triggerimpuls am GATE-Eingang (TTL: positive Flanke/Opto: negative Flanke) folgenden Takt in den Low-Zustand (TTL: 0V)/Opto: leitend). Nach Ablauf des Zählers geht der Ausgang wieder in den High-Zustand (TTL: +5V)/Opto: hochohmig).

Mit einer geeigneten Flanke am GATE-Eingang (TTL: positive Flanke/Opto: negative Flanke) kann der Zähler jederzeit auf den Startwert zurückgesetzt („retriggered“) werden. Der Ausgang bleibt solange im Low-Zustand (TTL: 0V)/Opto: leitend) bis der Zähler den Nulldurchgang erreicht.

Der Zählerstand kann jederzeit ohne Auswirkung auf den momentanen Zählvorgang, ausgelesen werden.

4.4.3 Modus 2: Asymmetrischer Teiler

In diesem Modus arbeitet der Zähler als Frequenzteiler. Der Zähler-Ausgang (OUT_0…2) geht nach der Initialisierung in den High-Zustand (TTL: +5V)/Opto: hochohmig). Nach Freigabe des Zählers durch geeignete Beschaltung des GATE-Eingangs (TTL: +5V/Opto: 0V) wird abwärts gezählt, während der Ausgang noch im High-Zustand verbleibt. Sobald der Zähler den Wert 0001Hex erreicht hat, geht der Ausgang für die Dauer einer Taktperiode in den Low-Zustand (TTL: 0V)/Opto: leitend). Dieser Ablauf wiederholt sich periodisch solange der GATE-Eingang freigegeben ist (TTL: +5V/Opto: 0V), ansonsten geht der Ausgang sofort in den High-Zustand (TTL: +5V)/Opto: hochohmig).

Wird das Zählerregister zwischen zwei Ausgangs-Pulsen erneut geladen, so beeinflußt dies den momentanen Zählvorgang nicht, die folgende Periode arbeitet jedoch mit den neuen Werten.

4.4.4 Modus 3: Symmetrischer Teiler

Dieser Modus arbeitet ähnlich wie Modus 2 mit dem Unterschied, daß der geteilte Takt ein symmetrisches Tastverhältnis besitzt (nur für geradzahlige Zählerwerte geeignet). Der Zähler-Ausgang (OUT_0…2) geht nach der Initialisierung in den High-Zustand (TTL: +5V)/Opto: hochohmig). Nach Freigabe des Zählers durch geeignete Beschaltung des GATE-Eingangs (TTL: +5V/Opto: 0V) wird in 2er-Schritten abwärts gezählt. Nun wechselt der Ausgang, mit der Anzahl Perioden des halben Startwertes bezogen auf den Eingangstakt (beginnend mit High-Pegel). Solange der Gate-Eingang freigegeben ist (TTL: +5V/Opto: 0V), wiederholt sich dieser Ablauf periodisch, ansonsten geht der Ausgang sofort in den High-Zustand (TTL: +5V)/Opto: hochohmig).

Wird das Zählerregister zwischen zwei Ausgangs-Pulsen erneut geladen, so beeinflußt dies den momentanen Zählvorgang nicht, die folgende Periode arbeitet jedoch mit den neuen Werten.

4.4.5 Modus 4: Zählerstart durch Softwaretrigger

Der Zähler-Ausgang (OUT_0…2) geht in den High-Zustand (TTL: +5V)/Opto: hochohmig), sobald der Zähler initialisiert wird. Zur Freigabe des Zählers muß der Gate-Eingang geeignet beschaltet werden (TTL: +5V/Opto: 0V). Sobald der Zähler geladen (Software-Trigger) und freigegeben wurde, beginnt er abwärts zu zählen, während der Ausgang noch im High-Zustand (TTL: +5V)/Opto: hochohmig) bleibt.

Bei Erreichen des Nulldurchganges geht der Ausgang für die Dauer einer Takt-Periode in den Low-Zustand (TTL: 0V)/Opto: leitend). Danach geht der Ausgang wieder in den High-Zustand (TTL: +5V)/Opto: hochohmig) und bleibt dort bis der Zähler initialisiert und ein neuer Startwert geladen wird.

Wird das Zählerregister während eines Zählvorganges erneut geladen, so wird der neue Startwert mit dem nächsten Takt geladen.

4.4.6 Modus 5: Zählerstart durch Hardwaretrigger

Der Zähler-Ausgang (OUT_0…2) geht in den High-Zustand (TTL: +5V)/Opto: hochohmig), sobald der Zähler initialisiert wird. Nachdem ein Startwert in den Zähler geladen wurde beginnt der Zählvorgang mit dem auf den ersten Triggerimpuls am GATE-Eingang (TTL: positive Flanke/Opto: negative Flanke) folgenden Takt. Bei Erreichen des Nulldurchganges geht der Ausgang für die Dauer einer Takt-Periode in den Low-Zustand (TTL: 0V)/Opto: leitend). Danach geht der Ausgang wieder in den High-Zustand (TTL: +5V)/Opto: hochohmig) und bleibt dort bis ein erneuter Triggerimpuls ausgelöst wird.

Wird das Zählerregister zwischen zwei Triggerimpulsen erneut geladen, so wird der neue Startwert erst nach dem nächsten Triggerimpuls berücksichtigt.

Mit einer positiven (TTL) bzw. negativen (Opto) Flanke am Gate-Eingang kann der Zähler jederzeit auf den Startwert zurückgesetzt („retriggered“) werden. Der Ausgang bleibt solange im High-Zustand (TTL: +5V)/Opto: hochohmig) bis der Zähler den Nulldurchgang erreicht.

4.4.7 Pulsweiten-Modulation

Ein spezieller Anwendungsfall der Zähler ist die sog. Pulsweiten-Modulation (PWM). Damit können Sie an OUT_2 (Pin 41) ein Rechteck-Signal von max. 50 kHz bei variablem Tastverhältnis ausgeben. Voraussetzung ist die externe Verdrahtung der Zähler wie in Kap. 3.6.2 auf Seite 28 beschrieben. Zähler 0 wird als Vorteiler für den extern eingespeisten Basistakt verwendet. Über den Parameter `<Prescaler>` können Sie die Frequenz f_{OUT_2} folgendermaßen einstellen:

$$f_{OUT\_2} = \frac{\text{Basistakt}}{\texttt{<Prescaler>} \cdot 100} \qquad (\text{mit } \texttt{<Prescaler>} = 2\ldots(2^{16} - 1))$$

Mit dem Parameter `<DutyCycle>` kann das Tastverhältnis zwischen 1…99% in Schritten von 1% eingestellt werden. Die Ausgabe wird unmittelbar durch Aufruf der Funktion …*CntPWMStart* gestartet und mit …*CntPWMStop* gestoppt. Es ist keine weitere Programmierung der Zähler erforderlich.

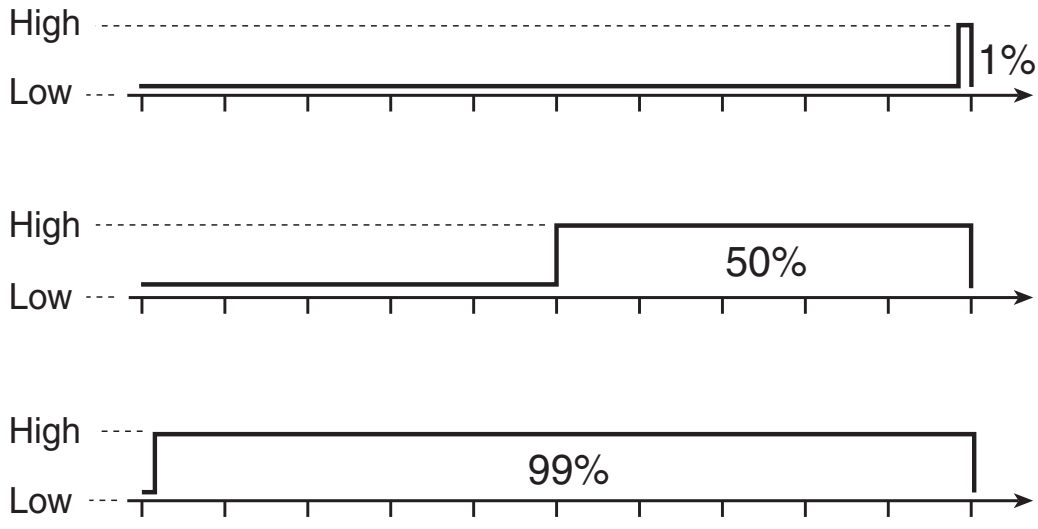


Abb. 50: Tastverhältnis PWM-Signal

Bei optoisolierten Karten ist der Ausgang OUT_2 als „Open Collector“ Ausgang ausgeführt. D. h. „High“ bedeutet Ausgang leitend und „Low“ Ausgang ist hochohmig (siehe Abb. 19 auf Seite 27).

4.5 ME-MultiSig-Steuerung

Zum Verständnis des ME-MultiSig-Systems und der im Folgenden beschriebenen Funktionalitäten empfehlen wir dringend das Handbuch des ME-MultiSig-System vorher und vollständig zu lesen!

4.5.1 „Mux“-Betrieb

Betriebsart	Anwendung	Trigger	Timing
MultiSig-AISingle (siehe Seite 205)	1 Kanal (0…255), 1 Meßwert	Software-Start, ext. digital, analog (opt.)	–
MultiSig-AIContinuous (siehe Seite 193)	Erfassung einer unbekannten Anzahl an Werten gemäß Mux-Kanalliste	Software-Start, ext. digital, analog (opt.)	CHAN-Timer, SCAN-Timer (…MultiSigAIConfig)
Multisig-AIScan (siehe Seite 205)	Erfassung einer bekannten Anzahl an Messwerten gemäß Mux-Kanalliste	Software-Start, ext. digital, analog (opt.)	CHAN-Timer, SCAN-Timer (…MultiSigAIConfig)

Tabelle 4: „Mux“-Betriebsarten

Zur Nutzung des vollen Funktionsumfangs im „Mux“-Betrieb werden 2 digitale Ausgangsports der ME-4600 benötigt. Bei Ein-

satz einer nicht optoisolierten Variante werden die Digital-Ports A und B verwendet. Bei optoisolierten Varianten („i"-Versionen) steuert der Treiber automatisch die Ausgangsports A und C an, da Port B als Eingangsport festgelegt ist. Auf diese Weise können Sie in Verbindung mit dem Anschlußadapter ME-AA4-3i (optional) die Funktionalität der Basiskarten dennoch uneingeschränkt nutzen.

Beachten Sie, daß bei Verwendung der „MultiSig-Funktionen" des ME-4600 Treibers die ME-MultiSig Basiskarten (ME-MUX32-M/S) für die Betriebsart „Single-Mux" konfiguriert sein müssen. Falls Sie einen von A/D-Kanal 0 (Auslieferungszustand) abweichenden Kanal verwenden möchten, setzen Sie Lötbrücke „A" für den gewünschten A/D-Kanal der ME-4600 (siehe Handbuch „ME-MultiSig"-System). Übergeben Sie die Nummer des auf der Hardware eingestellten A/D-Kanals im Parameter `<AIChannelNumber>` der Funktion *…MultiSigAIConfig* bzw. *…MultiSigAISingle*.

4.5.1.1 Konfiguration der Basiskarten

Zur Nutzung folgender Funktionalitäten müssen Sie in den Konfigurationsmodus wechseln (*…MultiSigOpen, …MultiSigClose):*

Das folgende Diagramm soll den Programmfluss kurz erläutern:

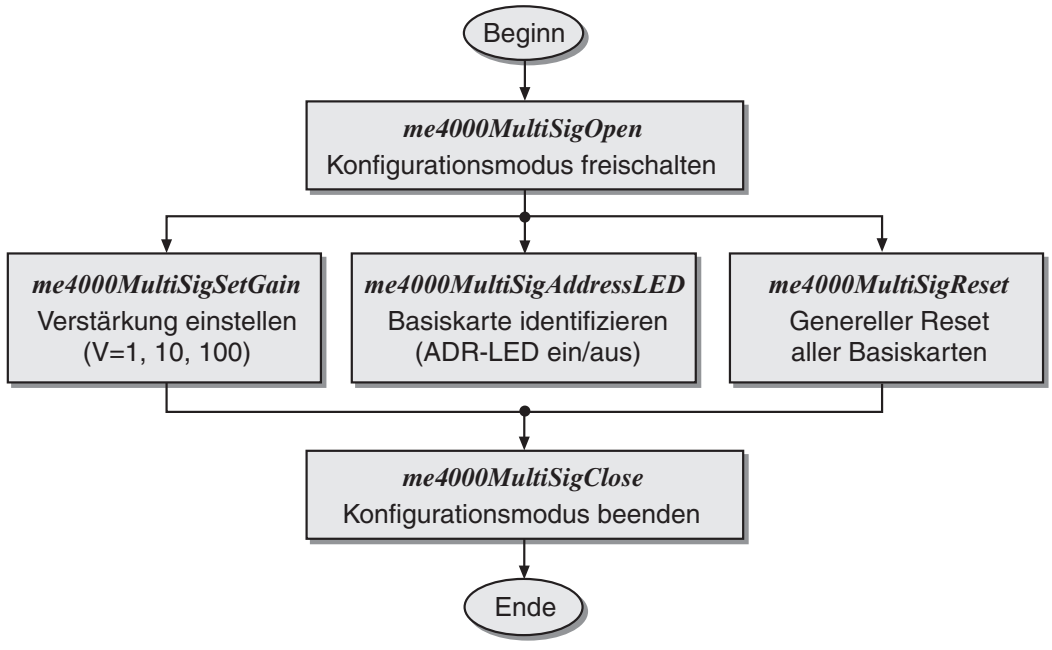


Abb. 51: Grund-Konfiguration Mux-Betrieb

4.5.1.1.1 Verstärkung einstellen

Der Verstärkungsfaktor kann für jede Kanalgruppe der Basiskarten ME-MUX32-M(aster) und ME-MUX32-S(lave) getrennt eingestellt werden. Wenn Sie mit Verstärkungsfaktor V=1 (Standard) arbeiten wollen, brauchen Sie keine Konfiguration vornehmen.

4.5.1.1.2 Adress-LED ansteuern

Zu Wartungszwecken etc. kann die Adress-LED der Basiskarten ME-MUX32-M und ME-MUX32-S gezielt angesteuert werden.

4.5.1.1.3 Genereller Reset

Mit Hilfe der Funktion …*MultiSigReset* können Sie alle Master- und Slave-Karten in den Grundzustand setzen:

- Verstärkung V=1.
- Adress-LEDs werden ausgeschaltet.

4.5.1.2 Betriebsart „MultiSig-AISingle“

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Betriebsart dient zur Erfassung eines einzelnen Wertes vom gewählten Kanal des Mux-Systems. Sie haben folgende Parameter zur Verfügung:

- Mux-Kanalnummer 0…255
- A/D-Kanalnummer 0…31 (ME-4650/4660: 0…15)
- Verstärkungsfaktor der Kanalgruppe (V=1, 10, 100).
- Triggermodi: per Software, externem Digital-Trigger oder ext. Analog-Trigger (nur ME-4670/4680).
- Ext. Trigger: fallende, steigende oder beide Flanken.
- Time-Out: falls externes Triggersignal ausbleibt.

Das folgende Diagramm soll den Programmfluss kurz erläutern:

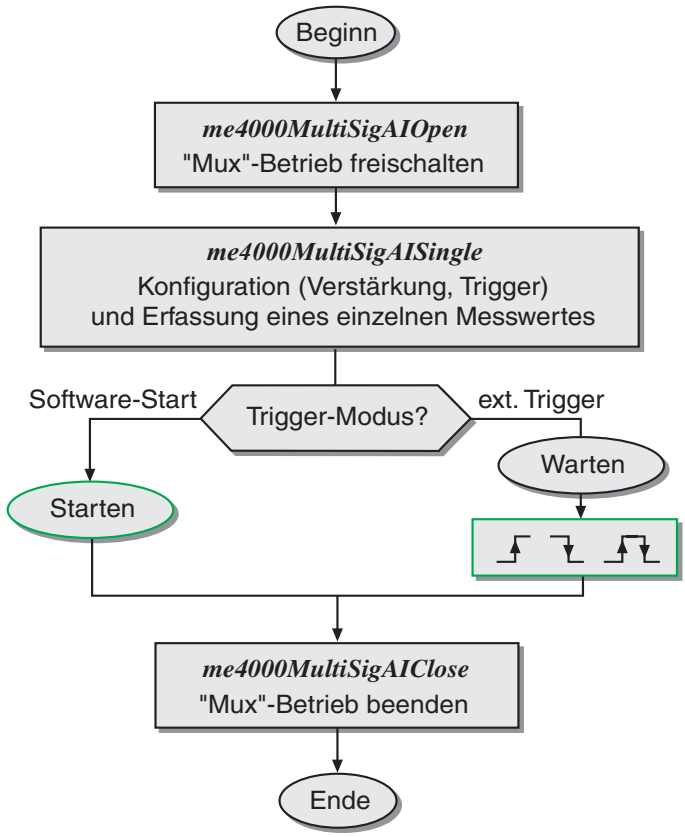


Abb. 52: Programmierung „MultiSig-AISingle“

4.5.1.3 Timergesteuerter „Mux“-Betrieb

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

In diesem Kapitel wird die timergesteuerte Erfassung über das ME-MultiSig-System in Verbindung mit einer Karte der ME-4600 Serie beschrieben. Geeignete Basiskarten sind die ME-MUX32-M (Master) mit bis zu 7 Slave-Karten vom Typ ME-MUX32-S.

Gehen Sie folgendermaßen vor:

1. Freischalten des „Mux“-Betriebs mit …*MultiSigAIOpen* (siehe Seite 203)
2. Erzeugen einer benutzerdefinierten Mux-Kanalliste zur Ansteuerung der Mux-Kanäle (0…255).
3. Konfiguration des A/D-Teils der ME-4600 mit der Funktion …*MultiSigAIConfig* (siehe Seite 190)

4. Vorbereitung der Software mit …*MultiSigAIContinuous* (siehe Seite 193) bzw. …*MultiSigAIScan* (siehe Seite 205)
5. Start der Erfassung mit …*MultiSigAIStart* (siehe Seite 210)
6. Sperren des „Mux"-Betriebs mit …*MultiSigAIClose* (siehe Seite 189)

Vor Beginn der Erfassung müssen Sie sich entscheiden, ob Sie eine bekannte Anzahl an Messwerten erfassen möchten („MultiSig-AIScan") oder kontinuierlich („MultiSig-AIContinuous") bis die Erfassung vom Anwender gestoppt wird. Nach Konfiguration von Hardware und Software kann die Erfassung per Software oder durch ein externes Triggersignal gestartet werden.

Das folgende Diagramm soll die grundsätzliche Vorgehensweise in den Betriebsarten **„MultiSig-AIContinuous"** und **„MultiSig-AIScan"** veranschaulichen. Das Daten-Handling erfolgt in Analogie zu den „normalen" AI-Betriebsarten:

- Daten-Handling „AIContinuous" siehe Seiten 40ff.
- Daten-Handling „AIScan" siehe Seiten 43ff.

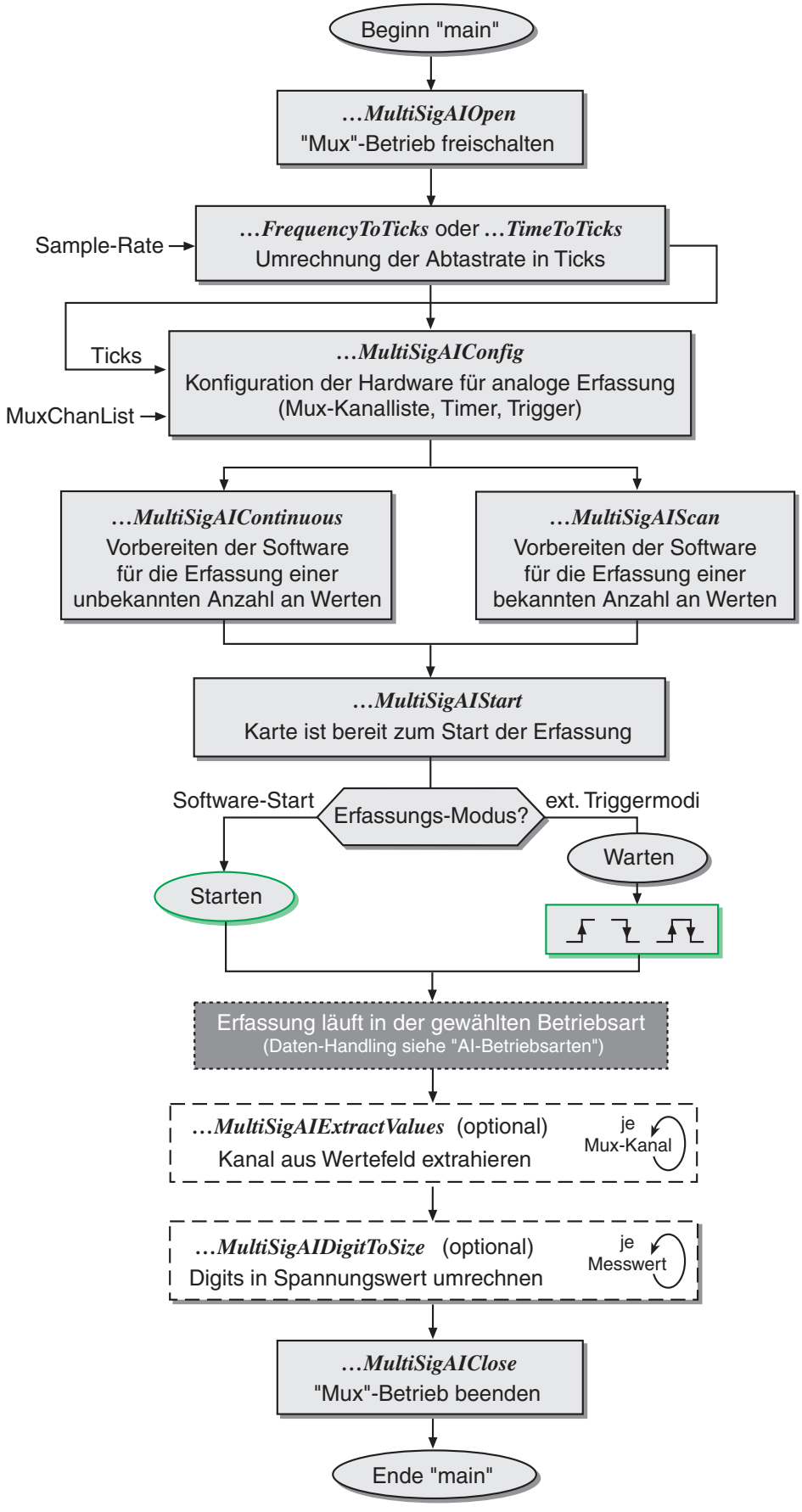


Abb. 53: Progr. „MultiSig-AIContinuous“ und „MultiSig-AIScan“

4.5.2 „Demux“-Betrieb

Betriebsart	Anwendung	Trigger	Timing
MultiSig-AOSingle (siehe Seite 214)	1 Wert auf einen Mux-Kanal (0…31)	Software-Start, ext. digital	–
MultiSig-AOContinuous (siehe Seite 216)	Ausgabe kontinuierlich sich ändernder Werte	Software-Start, ext. digital	…MultiSig AOConfig
MultiSig-AOWraparound (siehe Seite 226)	Ausgabe sich wiederholender Werte	Software-Start, ext. digital	…MultiSig AOConfig

Tabelle 5: „Demux“-Betriebsarten

Beachten Sie folgende Vorgaben:

- Zur Ansteuerung der Demultiplexer wird stets Digital-Port A verwendet.
- Es wird stets über D/A-Kanal 0 (Pin 30) der ME-4600 ausgegeben.

4.5.2.1 Betriebsart „MultiSig-AOSingle“

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Diese Betriebsart dient der Ausgabe einer Spannung auf den gewünschten Kanal des Demux-Systems (0…31). Das folgende Diagramm soll den Programmfluss kurz erläutern:

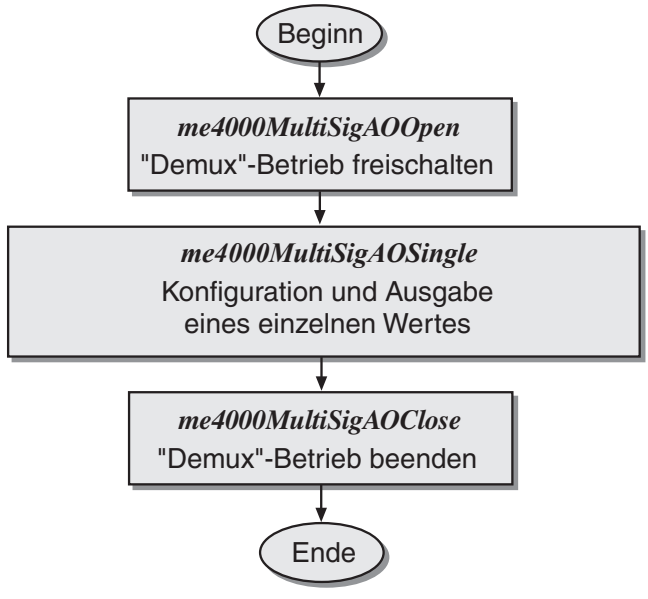


Abb. 54: Programmierung „MultiSig-AOSingle“

4.5.2.2 Timergesteuerter „Demux“-Betrieb

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

In diesem Kapitel wird die timergesteuerte Ausgabe über das ME-MultiSig-System in Verbindung mit einer Karte der ME-4600 Serie beschrieben. Verwenden Sie eine Basiskarte vom Typ ME-DEMUX32. Sie können damit auf bis zu 32 Kanäle „Demultiplexen“.

Gehen Sie folgendermaßen vor:

1. Freischalten des „Demux“-Betriebs mit …*MultiSigAOOpen* (siehe Seite 219)
2. Erzeugen einer benutzerdefinierten Demux-Kanalliste zur Ansteuerung der Demux-Kanäle (0…31).
3. Konfiguration des D/A-Teils der ME-4600 mit der Funktion …*MultiSigAOConfig* (siehe Seite 214)
4. Vorbereitung der Software mit …*MultiSigAOContinuous* (siehe Seite 216) bzw. …*MultiSigAOWraparound* (siehe Seite 226)
5. Start der Ausgabe mit …*MultiSigAOStart* (siehe Seite 223)
6. Sperren des „Demux“-Betriebs mit …*MultiSigAOClose* (siehe Seite 213)

Vor Beginn der Ausgabe müssen Sie sich entscheiden, ob die auszugebenden Werte kontinuierlich nachgeladen („MultiSig-AOContinuous“), oder stets die gleichen Werte wiederholt ausgegeben („MultiSig-AOWraparound“) werden sollen. Nach Konfiguration von Hardware und Software kann die Ausgabe per Software oder durch ein externes Triggersignal gestartet werden.

Das folgende Diagramm soll die grundsätzliche Vorgehensweise in den Betriebsarten **„MultiSig-AOContinuous“** und **„MultiSig-AOWraparound“** veranschaulichen. Das Daten-Handling erfolgt in Analogie zu den „normalen“ AO-Betriebsarten:

- Daten-Handling „AOContinuous“ siehe Seiten 56ff.
- Daten-Handling „AOWraparound“ siehe Seiten 60ff.

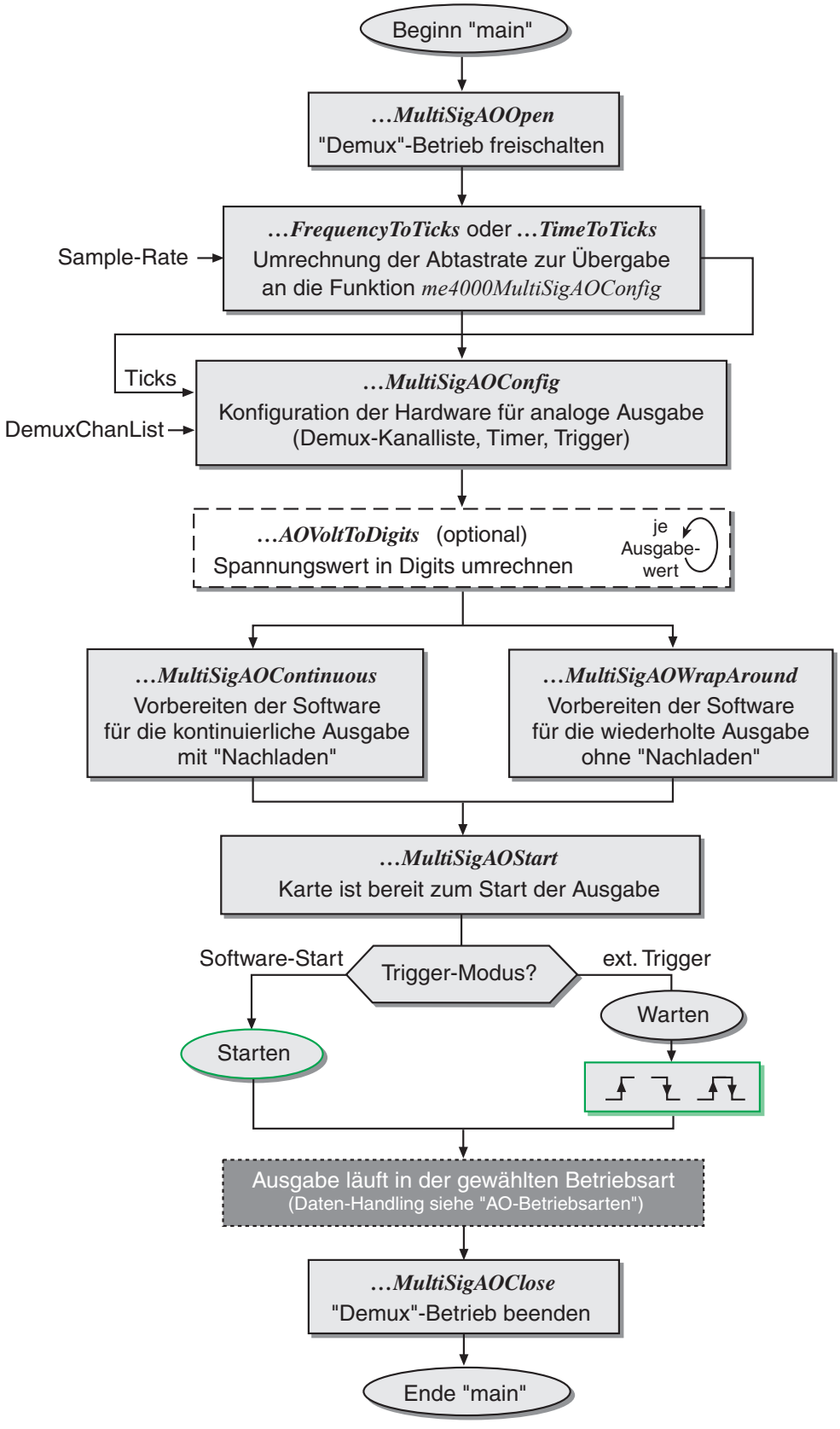


Abb. 55: Programmierung „MultiSig-AOContinuous“ und „MultiSig-AOWraparound“

4.6 Treiberkonzept

Wichtiger Hinweis: Die Karten der ME-4600 Serie (ME-4610/4650/4660/4670/4680) sowie die Karten vom Typ ME-6000 und ME-6100 verwenden 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.

4.6.1 Visual C++

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

Tabelle 6: 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.

4.6.2 Visual Basic

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

Tabelle 7: 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.

4.6.3 Delphi

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

Tabelle 8: Delphi

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

4.6.4 Agilent VEE

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

Tabelle 9: 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.

Die Funktionen *me4000VEE_AIScan* und *me4000VEE_ MultiSigAIScan* rufen automatisch die entsprechende Start-Funktion *…VEE_AIStart* bzw. *…VEE_AIMultiSigStart* auf und kehren erst nach Ende der Messung zurück.

4.6.5 LabVIEW

API-DLL	me4000Ex.dll	Systemtreiber
Funktionsprototypen	me4000LV.h*	LabVIEW-Treiber
Konstantendefinition	keine zentrale Definitionsdatei	
Funktions-Präfix	me4000LV(50)_…	siehe Hinweise im Text

Tabelle 10: 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.

Bei älteren LabVIEW-Versionen (5.0 und älter) müssen spezielle Funktionen *me4000LV50_AIScan* und *me4000LV50_ MultiSig-AIScan* aufgerufen werden, die automatisch die entsprechende Start-Funktion *…LV_AIStart* bzw. *…LV_AIMultiSigStart* aufrufen. Die „Scan“-Funktion kehrt erst nach Ende der Messung zurück.

4.6.6 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 compiliert 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.

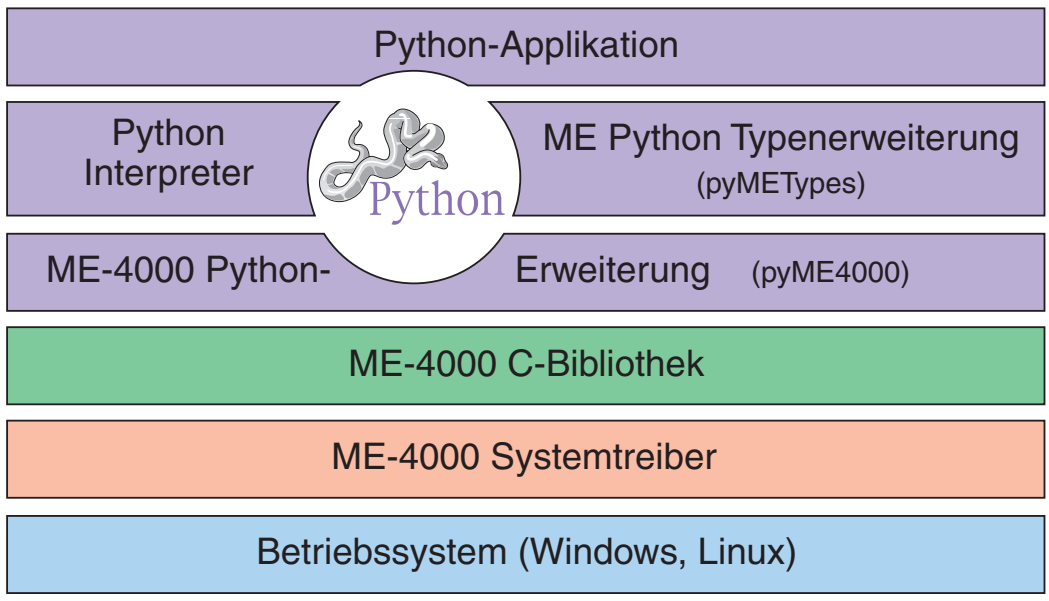


Abb. 56: Python Programmierung

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 Sofwarearchitektur 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. 4.6 ab Seite 88).

- **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 vom Typ ME-6000/6100 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 vom Typ ME-6000/6100 (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:

Funktionsnamen	werden im Fließtext kursiv geschrieben z. B. *me4000GetBoardVersion*.
`<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:

Funktion	Kurzbeschreibung	Seite
Fehler-Behandlung		
…ErrorGetMessage	Fehlernummer einen Fehlerstring zuweisen.	101
…ErrorGetLastMessage	Zuletzt aufgetretenem Fehler einen Fehlerstring zuweisen	102
…ErrorSetDefaultProc	Vordefinierte, globale Fehlerroutine für API installieren	103
…ErrorSetUserProc	Benutzerdefinierte, globale Fehlerroutine für API installieren	104
Allgemeine Funktionen		
…FrequencyToTicks	Frequenz in Ticks umrechnen	105
…GetBoardVersion	Kartenversion ermitteln	107
…GetDLLVersion	DLL-Versionsnummer ermitteln	108
…GetDriverVersion	Treiber-Versionsnummer ermitteln	108
…GetSerialNumber	Seriennummer der Karte ermitteln	109
…TimeToTicks	Periodendauer in Ticks umrechnen	110

Tabelle 11: Übersicht der Bibliotheksfunktionen

Funktion	Kurzbeschreibung	Seite
Analoge Erfassung		
…AIConfig	A/D-Teil konfigurieren	112
…AIContinuous	Erfassung einer unbekannten Anzahl an Messwerten	115
…AIDigitToVolt	Umrechnung des Digit-Wertes in Spannungswert	117
…AIExtractValues	Werte für einen Kanal aus Datenpuffer extrahieren	119
…AIGetNewValues	Daten asynchron abholen	120
…AIGetStatus	Status-Abfrage für „AIScan“	122
…AIMakeChannelListEntry	Kanallisten-Eintrag generieren	123
…AIReset	Beenden einer timergesteuerten Erfassung	124
…AIScan	Erfassung einer bekannten Anzahl an Messwerten	125
…AISingle	Einzelwert-Messung	128
…AIStart	Start einer timergesteuerten Erfassung	130
…AIStop	„Anhalten“ einer timergesteuerten Erfassung	131
Analoge Ausgabe		
…AOAppendNewValues	Ausgabepuffer nachladen	132
…AOConfig	D/A-Teil konfigurieren	134
…AOContinuous	Kontinuierliche Ausgabe	136
…AOGetStatus	Status-Abfrage für „AOContinuous“ und „AOWraparound“	138
…AOReset	Ausgabekanal rücksetzen	139
…AOSingle	Einzelwert-Ausgabe	140
…AOSingleSimultaneous	Start der simultanen Ausgabe in der Betriebsart „AOSimultaneous“	142
…AOStart	Start einer timergesteuerten Ausgabe	144
…AOStartSynchronous	Synchron-Start in den Betriebsarten „AOContinuous“ & „AOWraparound“	145
…AOStop	Ausgabe stoppen	148
…AOVoltToDigit	Umrechnung des auszugebenden Spannungswertes in Digitwert	149

Tabelle 11: Übersicht der Bibliotheksfunktionen

Funktion	Kurzbeschreibung	Seite
…AOWaveGen	Einfacher Funktionsgenerator	150
…AOWraparound	Periodische Ausgabe	152
Bitmuster-Ausgabe		
…DIOBPAppendNewValues	Datenpuffer nachladen	155
…DIOBPConfig	Hardware für Bitmuster-Ausgabe konfigurieren	157
…DIOBPPortConfig		157
…DIOBPContinuous	Kontinuierliche Bitmuster-Ausgabe	159
…DIOBPGetStatus	Status-Abfrage für „BitPattern-Continuous“ und „BitPattern-Wraparound“	161
…DIOBPReset	Bitmuster-Ausgabe rücksetzen	163
…DIOBPStart	Start einer Bitmuster-Ausgabe	164
…DIOBPStop	Bitmuster-Ausgabe stoppen	165
…DIOBPWraparound	Periodische Bitmuster-Ausgabe	166
Digitale Standard-Ein/Ausgabe		
…DIOConfig	Digital-Ports für Standard-Digital-I/O konfigurieren	169
…DIOGetBit	Bit einlesen	170
…DIOGetByte	Byte einlesen	171
…DIOResetAll	DIO-Teil rücksetzen	172
…DIOSetBit	Bit ausgeben	173
…DIOSetByte	Byte ausgeben	174
Zähler-Funktionen		
…CntPWMStart	PWM-Ausgabe starten	175
…CntPWMStop	PWM-Ausgabe beenden	176
…CntRead	Zählerstand einlesen	177
…CntWrite	Zähler konfigurieren und starten	178
Funktionen für externen Interrupt		
…ExtIrqDisable	Externen IRQ-Eingang sperren	179
…ExtIrqEnable	Externen IRQ-Eingang freigeben	180
…ExtIrqGetCount	Anzahl der ext. IRQs ermitteln	181

Tabelle 11: Übersicht der Bibliotheksfunktionen

Funktion	Kurzbeschreibung	Seite
MultiSig-Funktionen		
…MultiSigAddressLED	Adress-LED ansteuern	183
…MultiSigClose	Konfigurationsmodus beenden	184
…MultiSigOpen	Konfigurationsmodus freischalten	185
…MultiSigReset	Rücksetzen aller Master-/Slave-Karten	186
…MultiSigSetGain	Verstärkung je Kanalgruppe	187
…MultiSigAIClose	Betriebsart „Multiplexen“ abschließen	189
…MultiSigAIConfig	Hardware für analoge Erfassung konfigurieren	190
…MultiSigAIContinuous	Erfassung einer unbekannten Anzahl an Messwerten	193
…MultiSigAIDigitToSize	Umrechnung des Digit-Wertes in entsprechende physikalische Größe	195
…MultiSigAIExtractValues	Werte für einen Kanal aus Datenpuffer extrahieren	198
…MultiSigAIGetNewValues	Daten im Asynchron-Betrieb abholen	200
…MultiSigAIGetStatus	Status-Abfrage im „Mux-Betrieb“	202
…MultiSigAIOpen	Betriebsart „Multiplexen“ freischalten	203
…MultiSigAIReset	Beenden einer timergesteuerten Erfassung	204
…MultiSigAIScan	Erfassung einer bekannten Anzahl an Messwerten	205
…MultiSigAISingle	Einzelwert-Messung	208
…MultiSigAIStart	Start einer timergesteuerten Erfassung	210
…MultiSigAIStop	„Anhalten“ einer timergesteuerten Erfassung	211
…MultiSigAOAppendNewValues	Ausgabepuffer nachladen	212
…MultiSigAOClose	Betriebsart „Demultiplexen“ abschließen	213
…MultiSigAOConfig	Hardware für analoge Ausgabe konfigurieren	214

Tabelle 11: Übersicht der Bibliotheksfunktionen

Funktion	Kurzbeschreibung	Seite
…MultiSigAOContinuous	Kontinuierliche Ausgabe	216
…MultiSigAOGetStatus	Status-Abfrage im „Demux-Betrieb“	218
…MultiSigAOOpen	Betriebsart „Demultiplexen“ freischalten	219
…MultiSigAOReset	Ausgabekanal rücksetzen	220
…MultiSigAOSingle	Einzelwert-Ausgabe	221
…MultiSigAOStart	Start einer timergesteuerten Ausgabe	223
…MultiSigAOStop	Ausgabe stoppen	224
…MultiSigAOVoltToDigit	Umrechnung des auszugebenden Spannungswertes in Digitwert	225
…MultiSigAOWraparound	Periodische Ausgabe	226

Tabelle 11: Übersicht der Bibliotheksfunktionen

5.3.1 Fehler-Behandlung

me4000ErrorGetMessage

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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

● Definitionen

VC: me4000ErrorGetMessage(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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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 …*ErrorGetMessage* um dem Fehlercode 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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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:

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

Die Periodendauer läßt sich in Schritten von $30,\overline{30}$ 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:

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

Beachten Sie, daß der Parameter <`TicksHighPart`> nur für eine SCAN-Frequenz <0.0077 Hz benötigt wird (siehe Funktionen „*…AIConfig*“ und „*…MultiSigAIConfig*“). 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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Es wird die Kartenversion ermittelt (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:

4650Hex:	ME-4650	16 A/D, ohne Zähler
4660Hex:	ME-4660	16 A/D, 2 D/A
4661Hex:	ME-4660i	32 A/D, 2 D/A, opto
4662Hex:	ME-4660s	32 A/D, 2 D/A, S&H
4663Hex:	ME-4660is	32 A/D, 2 D/A, opto, S&H
4670Hex:	ME-4670	32 A/D, 4 D/A
4671Hex:	ME-4670i	32 A/D, 4 D/A, opto
4672Hex:	ME-4670s	32 A/D, 4 D/A, S&H
4673Hex:	ME-4670is	32 A/D, 4 D/A, opto, S&H
4680Hex:	ME-4680	32 A/D, 4 D/A-FIFO
4681Hex:	ME-4680i	32 A/D, 4 D/A-FIFO, opto
4682Hex:	ME-4680s	32 A/D, 4 D/A-FIFO, S&H
4683Hex:	ME-4680is	32 A/D, 4 D/A-FIFO, opto, S&H

‹ 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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Ermittelt die Seriennummer der ausgewählten Karte.

● 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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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 (`ChanTicks`).

☞ Hinweis:

Die Anzahl der Ticks errechnet sich folgendermaßen:

Allgemein:

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

Die Periodendauer läßt sich in Schritten von $30,\overline{30}$ 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:

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

Beachten Sie, daß der Parameter `<TicksHighPart>` nur für SCAN-Zeiten >130s benötigt wird (siehe Funktionen „*…AIConfig*“ und „*…MultiSigAIConfig*“). 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 Erfassung

me4000AIConfig

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion konfiguriert die Hardware des A/D-Teils für eine timergesteuerte Erfassung. Sie konfiguriert die Timer, übergibt die Kanalliste, bestimmt den Erfassungsmodus `<AcqMode>`, legt die ext. Triggerquelle und Triggerflanke fest und schaltet bei Bedarf die simultane Erfassung ein (optional).

Vorab müssen Sie mit Hilfe der Funktion *…AIMakeChannelListEntry* eine benutzerdefinierte Kanalliste generieren.

Nach Aufruf von *…AIScan* oder *…AIContinuous* wird die Erfassung stets mit der Funktion *…AIStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal gestartet.

☞ Hinweis:

Es stehen ein 32 Bit breiter CHAN-Timer sowie ein 36 Bit breiter SCAN-Timer zur Verfügung. Der CHAN-Timer bestimmt die Abtastrate (Sample-Rate) innerhalb der Kanalliste. Die max. CHAN-Zeit beträgt ca. 130s, die min. CHAN-Zeit beträgt 2 µs. Der SCAN-Timer bestimmt die Zeit zwischen dem jeweils ersten Kanallisteneintrag von zwei aufeinander folgenden Kanallistenabarbeitungen. Die Verwendung ist optional. Als gemeinsame Zeitbasis nutzen alle Timer einen 33 MHz Takt. Daraus ergibt sich eine Periodendauer von 30,30 ns, die als kleinste Zeiteinheit definiert wird und im Folgenden „1 Tick“ genannt wird. Die gewünschte Periodendauer muß nun als Vielfaches eines Ticks an die Parameter `<ChanTicks>`, `<ScanTicksLow>` und `<ScanTicksHigh>` übergeben werden. `<ScanTicksHigh>` wird nur für SCAN-Zeiten >130s benötigt.

Verwenden Sie die Funktionen *…FrequencyToTicks* und *…TimeToTicks* (siehe Seite 105ff) zur bequemen Umrechnung von Frequenz bzw. Periodendauer in Ticks zur Übergabe an die Timer.

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

● Definitionen

VC: me4000AIConfig(unsigned int uiBoardNumber, unsigned char* pucChanList, unsigned int uiChanListCount, int iSDMode, int iSimultaneous, unsigned long ulReserved, unsigned long ulChanTicks, unsigned long ulScanTicksLow, unsigned long ulScanTicksHigh, int iAcqMode, int iExtTriggerMode, 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)

<ChanList>

Zeiger auf den Anfang der Kanalliste.
(siehe Funktion …*AIMakeChannelListEntry*)

<ChanListCount>

Anzahl der Kanallisten-Einträge.

<SDMode>

Single ended oder differentielle Messung:

- ME4000_AI_INPUT_SINGLE_ENDED:
  Single ended Messung
- ME4000_AI_INPUT_DIFFERENTIAL:
  Differentielle Messung

<Simultaneous>

Simultane Erfassung der Kanäle 0…7 ein-/ausschalten (nur für Modelle mit „s“-Option). Siehe auch Kap. 3.3.3

- ME4000_AI_SIMULTANEOUS_DISABLE
  Simultanbetrieb ausschalten (Standard)
- ME4000_AI_SIMULTANEOUS_ENABLE
  Simultanbetrieb einschalten

<Reserved>

Dieser Parameter ist reserviert. Bitte übergeben Sie „0“.

<ChanTicks>

Anzahl der Ticks für den CHAN-Timer (32 Bit), der die Abtastrate festlegt. Der Wertebereich liegt zwischen 66 (42Hex) und 2^{32}-1 (FFFFFFFFHex) Ticks.

<ScanTicksLow>

Anzahl der Ticks für den niederwertigen Teil (Bits 31…0) des insgesamt 36 Bit breiten SCAN-Timers (siehe auch `ScanTicksHigh`). Er legt die Zeit zwischen der Wandlung des jeweils ersten Kanallisteneintrags von zwei aufeinanderfolgenden Kanallistenabarbeitungen fest. Wenn Sie diesen Timer nicht benutzen möchten, übergeben Sie hier ***und*** in `<ScanTicksHigh>` ME4000_VALUE_NOT_USED.

Für eine sinnvolle SCAN-Zeit gilt (siehe auch Abb. 25):
(Anzahl der Kanallisten-Einträge x CHAN-Zeit) + „x“ Ticks

Die max. erlaubte SCAN-Zeit beträgt 30 Minuten, dies entspricht 59.400.000.000 Ticks (DD4841200Hex).

<ScanTicksHigh>

Anzahl der Ticks für den höherwertigen Teil (Bits 35…32) des insgesamt 36 Bit breiten SCAN-Timers (siehe auch `ScanTicksLow`). Diesen Timer benötigen Sie nur für Scan-Zeiten über 130,15s - ansonsten übergeben Sie ME4000_VALUE_NOT_USED.

<AcqMode>

Erfassungsmodus für die timergesteuerte Erfassung:

- ME4000_AI_ACQ_MODE_SOFTWARE
  Wandlungsstart nach Aufruf der Funktion *…AIStart*. Abarbeitung gemäß Timer-Einstellungen (siehe Abb. 25).
- ME4000_AI_ACQ_MODE_EXT
  Bereit zur Erfassung nach Aufruf der Funktion *…AIStart*. Erfassung beginnt mit dem ersten ext. Triggerimpuls. Abarbeitung gemäß Timer-Einstellungen. Weitere Triggerimpulse bleiben ohne Wirkung (siehe Abb. 32).
- ME4000_AI_ACQ_MODE_EXT_SINGLE_VALUE
  Bereit zur Erfassung nach Aufruf der Funktion *…AIStart*. Mit jedem ext. Triggerimpuls wird genau **ein** Wert gemäß Kanalliste gewandelt. Vom ersten Triggersignal bis zur ersten Wandlung vergeht einmal die CHAN-Zeit. Ansonsten bleiben die Timer-Einstellungen ohne Wirkung (siehe Abb. 33).
- ME4000_AI_ACQ_MODE_EXT_SINGLE_CHANLIST
  Bereit zur Erfassung nach Aufruf der Funktion *…AIStart*. Mit jedem ext. Triggerimpuls wird die Kanalliste einmal abgearbeitet. Die Abarbeitung erfolgt gemäß Timer-Einstellungen (siehe Abb. 34). Der SCAN-Timer bleibt ohne Wirkung!

<ExtTriggerMode>

Auswahl der externen Triggerquelle für A/D-Teil. Siehe auch Parameter `<AcqMode>`.

- ME4000_AI_TRIGGER_EXT_DIGITAL
  Der digitale Triggereingang ist Triggerquelle.
- ME4000_AI_TRIGGER_EXT_ANALOG (nicht ME-4650/4660)
  Die analoge Triggereinheit ist Triggerquelle.
- ME4000_VALUE_NOT_USED
  Kein ext. Trigger verwendet.

<ExtTriggerEdge>

Auswahl der externen Triggerflanke für A/D-Teil. Siehe auch Parameter `<ExtTriggerMode>`.

- ME4000_AI_TRIGGER_EXT_EDGE_RISING
  Steigende Triggerflanken werden ausgewertet.
- ME4000_AI_TRIGGER_EXT_EDGE_FALLING
  Fallende Triggerflanken werden ausgewertet.
- ME4000_AI_TRIGGER_EXT_EDGE_BOTH
  Sowohl steigende als auch fallende Triggerflanken werden ausgewertet.
- ME4000_VALUE_NOT_USED
  Kein ext. Trigger verwendet.

‹ 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.

me4000AIContinuous

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Mit dieser Funktion wird die Software für eine timergesteuerte Erfassung vorbereitet bei der die Anzahl der Messwerte vorher unbekannt ist. Diese Funktion wird grundsätzlich asynchron ausgeführt. Die Messwerte werden entweder mit einer benutzerdefinierten Callback-Funktion oder durch wiederholten Aufruf der Funktion *…AIGetNewValues* abgeholt.

Gestartet wird die Erfassung stets mit der Funktion *…AIStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe *…AIConfig*). Falls Sie mit einem externen Triggersignal arbeiten und dieses ausbleibt können Sie mit einem geeigneten „Time-Out"-Wert die Erfassung abbrechen. Durch Aufruf der Funktion *…AIStop* oder *…AIReset* wird die Erfassung beendet.

☞ Hinweis:

Zur Vorgehensweise beachten Sie bitte Kap. 4.1.3 "Timergesteuerte „AI-Betriebsarten"" auf S. 34ff, sowie die Programmbeispiele im ME-Software-Developer-Kit (ME-SDK).

● Definitionen

Typdefinition für ME4000_P_AI_CALLBACK_PROC:

```
typedef void (_stdcall *
ME4000_P_AI_CALLBACK_PROC)(short* psValues,
unsigned int uiNumberOfValues, void* pCall-
backContext, int iLastError);
```

VC: me4000AIContinuous(unsigned int uiBoardNumber, ME4000_P_AI_CALLBACK_PROC pCallbackProc, void* pCallbackContext, unsigned int uiRefreshFrequency, 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)

<CallbackProc> **~~LV~~, ~~VB~~, ~~VEE~~**

Callback-Funktion, die während der Erfassung in regelmäßigen Abständen aufgerufen wird. Der Funktion wird ein Zeiger auf die neu hinzugekommenen Werte sowie deren Anzahl übergeben. 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 weitergegeben wird. Falls keine Callback-Funktion angegeben wurde, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<RefreshFrequency>

Anzahl der Kanallistenabarbeitungen nach denen der Ringpuffer zyklisch ausgelesen werden soll. Übergabewert dient als Richtwert, der vom Treiber gegebenenfalls angepaßt wird. Bei Übergabe von ME4000_VALUE_NOT_USED ermittelt der Treiber einen sinnvollen Wert.

<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.

‹ 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.

me4000AIDigitToVolt

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion erlaubt Ihnen die einfache Umrechnung der Messwerte [Digits] in Spannungswerte [V] unter Berücksichtigung des jeweiligen Eingangsspannungsbereiches. Die Funktion kann auf Einzelwerte oder durch wiederholten Aufruf auf ein ganzes Wertefeld angewandt werden. Die Verwendung ist optional.

Die Funktion verwendet folgende Formeln zur Umrechnung:

Bipolar, ±10V: $U[Volt] = \frac{10V}{32768} \cdot U[Digits]$	***Bipolar, ±2,5V:*** $U[Volt] = \frac{2,5V}{32768} \cdot U[Digits]$
Unipolar, 0...10V: $U[Volt] = \frac{5V}{32768} \cdot U[Digits] + 5V$	***Unipolar, 0...2,5V:*** $U[Volt] = \frac{1,25V}{32768} \cdot U[Digits] + 1,25V$

● Definitionen

VC: me4000AIDigitToVolt(short sDigit, int iRange, double* pdVolt);
LV: me4000LV_… (siehe `me4000LV.h`)
VB: me4000VB_… (siehe `me4000.bas`)
VEE: me4000VEE_… (siehe `me4000VEE.h`)

➔ Parameter

`<Digit>`

Übergabe eines „Rohwertes“, wie er nach der Erfassung im Datenpuffer steht.

`<Range>`

Auswahl des Eingangsspannungsbereichs:

- ME4000_AI_RANGE_BIPOLAR_10: ±10V
- ME4000_AI_RANGE_BIPOLAR_2_5: ±2,5V
- ME4000_AI_RANGE_UNIPOLAR_10: 0…10V
- ME4000_AI_RANGE_UNIPOLAR_2_5: 0…2,5V

`<Volt>`

Zeiger auf den errechneten Spannungswert in Volt.

❮ 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.

me4000AIExtractValues

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion extrahiert aus dem Wertefeld aller erfassten Werte die Werte des spezifizierten Kanals korrespondierend zur Kanalliste. Um die Daten für mehrere Kanäle zu extrahieren, muß die Funktion für jeden Kanal getrennt aufgerufen werden.

● Definitionen

VC: me4000AIExtractValues(unsigned int uiChannelNumber, short* psAIBuffer, unsigned long ulAIDataCount, unsigned char* pucChanList, unsigned int uiChanListCount, short* psChanBuffer, unsigned long ulChanBufferSizeValues, unsigned long* pulChanDataCount);

LV: me4000LV_… (siehe `me4000LV.h`)

VB: me4000VB_… (siehe `me4000.bas`)

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

➔ Parameter

`<ChannelNumber>`

A/D-Kanal-Nummer dessen Werte extrahiert werden sollen; mögliche Werte: 0…31; (ME-4650/4660: 0…15 single ended)

`<AIBuffer>`

Zeiger auf Datenpuffer mit allen erfassten Werten.

`<AIDataCount>`

Anzahl der Messwerte im Wertefeld `<AIBuffer>`.

`<ChanList>`

Zeiger auf Kanalliste, die mit der Funktion *…AIMakeChannelListEntry* generiert und der Funktion *…AIConfig* übergeben wurde.

`<ChanListCount>`

Anzahl der Kanallisteneinträge (`ChanList`).

`<ChanBuffer>`

Zeiger auf Wertefeld in dem die extrahierten Werte des spezifizierten Kanals abgelegt werden.

`<ChanBufferSizeValues>`

Größe des Wertefeldes `ChanBuffer` in Anzahl der Messwerte.

<ChanDataCount>

Zeiger der nach Rückkehr der Funktion die Anzahl der tatsächlich in `ChanBuffer` abgelegten Werte enthält. Die Anzahl wird nie größer als `ChanBufferSizeValues` sein, kann aber auch kleiner sein.

‹ 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.

me4000AIGetNewValues

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

In Verbindung mit der Betriebsart „AIContinuous“ können sie mit dieser Funktion die Messwerte „Abholen“. In der Betriebsart „AI-Scan“ dient sie dem „Einsehen“ der Messwerte während einer im Hintergrund laufenden Erfassung (asynchron). Ein Anwendungsfall besteht z. B. darin, während einer längeren Erfassung die Messwerte einzulesen und anzuzeigen.

☞ Hinweis:

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 39, sowie in den Beispielprogrammen, die im ME-SDK enthalten sind.

● Definitionen

VC: me4000AIGetNewValues(unsigned int uiBoardNumber, short* psBuffer, unsigned long ulNumberOfValuesToRead, int iExecutionMode, unsigned long* pulNumberOfValuesRead, int* piLastError);

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)

<Buffer>

Zeiger auf Datenpuffer, der die neuesten Messwerte (linearisiert) der laufenden Erfassung enthält. Verwenden Sie die Funktion …*AIDigitToVolt* zur einfachen Umrechnung in Spannungswerte.

<NumberOfValuesToRead>

Größe des Datenpuffers in Anzahl der Messwerte. Die Puffergröße sollte ein Vielfaches der Kanallistenlänge betragen. Bei Übergabe von „0“ können Sie im Parameter `<NumberOfValuesRead>` die Anzahl der Werte abfragen, die zur Abholung bereitstehen.

<ExecutionMode>

Ausführungsmodus für diese Funktion wählen:

- ME4000_AI_GET_NEW_VALUES_BLOCKING:
  Das Programm ist blockiert bis alle Messwerte abgeholt wurden.
- ME4000_AI_GET_NEW_VALUES_NON_BLOCKING:
  Es werden nur die aktuell vorhandenen Messwerte abgeholt.

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

<NumberOfValuesRead>

Zeiger, der nach Rückkehr der Funktion die Anzahl der tatsächlich im Datenpuffer abgelegten Werte enthält. Die Anzahl wird nie größer als `<NumberOfValuesToRead>` sein, kann aber auch kleiner sein, falls noch nicht so viele neue Messwerte vorliegen.

<LastError>

Dieser Parameter enthält den letzten Fehler, der seit dem letzten Aufruf dieser Funktion auftrat. Mögliche Fehler sind FIFO-Überlauf oder Datenpuffer-Überlauf. Ist kein Fehler aufgetreten, so wird „0“ (ME4000_NO_ERROR) zurückgegeben.

❮ 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.

me4000AIGetStatus

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Funktion dient der Abfrage ob eine Erfassung in der Betriebsart „AI-Scan“ im Ausführungsmodus „ASYNCHRONOUS“ noch läuft oder bereits alle der erwarteten Messwerte erfasst wurden.

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

● Definitionen

VC: me4000AIGetStatus(unsigned int uiBoardNumber, 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)

`<WaitIdle>`

„Rückkehr-Verhalten“ dieser Funktion:

- ME4000_AI_WAIT_NONE
  Funktion gibt im Parameter `<Status>` den aktuellen Betriebszustand sofort zurück.
- ME4000_AI_WAIT_IDLE
  Funktion kehrt erst zurück nachdem alle Werte erfasst wurden. In diesem Fall enthält der Parameter `<Status>` stets den Wert ME4000_AI_STATUS_IDLE.

`<Status>`

Aktueller Betriebszustand:

- ME4000_AI_STATUS_IDLE
  Die Erfassung ist beendet.
- ME4000_AI_STATUS_BUSY
  Die Erfassung 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.

me4000AIMakeChannelListEntry

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion generiert aus den Parametern `<ChannelNumber>` und `<Range>` einen Kanallisteneintrag und schreibt diesen in ein Wertefeld zur späteren Übergabe an die Funktion *…AIConfig*. Die Funktion muß für jeden Kanallisteneintrag getrennt aufgerufen werden.

☞ Hinweis!

Beachten Sie, daß unipolare Eingangsbereiche nicht mit der Betriebsart differentiell kombiniert werden können!

● Definitionen

VC: me4000AIMakeChannelListEntry(unsigned int uiChannelNumber, int iRange, unsigned char* pucChanListEntry);

LV: me4000LV_… (siehe `me4000LV.h`)

VB: me4000VB_… (siehe `me4000.bas`)

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

➔ Parameter

`<ChannelNumber>`

A/D-Kanal-Nummer; mögliche Werte im single ended Betrieb: 0…31; im differentiellen Betrieb 0…15; (ME-4650/4660: 0…15 single ended)

`<Range>`

Auswahl des Eingangsspannungsbereichs:

- ME4000_AI_RANGE_BIPOLAR_10: ±10V
- ME4000_AI_RANGE_BIPOLAR_2_5: ±2,5V
- ME4000_AI_RANGE_UNIPOLAR_10: 0…10V
- ME4000_AI_RANGE_UNIPOLAR_2_5: 0…2,5V

<ChanListEntry>

Zeiger auf einzelnes Element eines Wertefeldes vom Typ „unsigned char“ in dem der Kanallisteneintrag abgelegt wird. Wertefeld muß zuvor vom Anwender allokiert werden. Später wird im Parameter <ChanList> der Funktion …*AIConfig* ein Zeiger auf dieses Wertefeld übergeben.

‹ 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.

me4000AIReset

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Die Erfassung wird sofort und vollständig beendet. Alle bis dahin erfaßten Messwerte gehen verloren. Der A/D-Teil muß für eine erneute Erfassung neu konfiguriert werden (…*AIConfig*, …*AIScan*, …*AIContinuous*).

● Definitionen

VC: me4000AIReset(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.

me4000AIScan

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Mit dieser Funktion wird die Software für die Erfassung einer von vornherein bekannten Anzahl an Messwerten vorbereitet. Es wird ein benutzerdefinierter Datenpuffer allokiert in dem die Messwerte abgelegt werden. Im Ausführungsmodus „BLOCKING“ kehrt der „Thread“, in dem die Funktion *…AIStart* aufgerufen wurde, erst nach Erfassung des letzten Wertes zurück. Im Ausführungsmodus „ASYNCHRONOUS“ wird die Erfassung als Hintergrundprozeß gestartet, d. h. durch Aufruf der Funktion *…AIStart* wird automatisch ein neuer „Thread“ erzeugt. Parallel dazu können andere Aufgaben („Threads“) abgearbeitet werden. Falls gewünscht (z. B. bei einer längeren Erfassung), können Sie die Messwerte bereits während der Erfassung „einsehen“. Dies können Sie entweder mit einer benutzerdefinierten Callback-Funktion oder mit Hilfe der Funktion *…AIGetNewValues* machen. Mit einer weiteren Callback-Funktion „Terminate“ können Sie (falls gewünscht) das Ende der Erfassung an Ihre Applikation melden lassen.

Gestartet wird die Erfassung stets mit der Funktion *…AIStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe *…AIConfig*). Falls Sie mit einem externen Triggersignal arbeiten und dieses ausbleibt können Sie mit einem geeigneten „Time-Out“-Wert die Erfassung abbrechen. Beendet wird die Erfassung automatisch nach Erfassung der erwarteten Messwerte.

☞ Hinweis:

Zur Vorgehensweise beachten Sie bitte Kap. 4.1.3 "Timergesteuerte „AI-Betriebsarten“" auf S. 34ff, sowie die Programmbeispiele im ME-Software-Developer-Kit (ME-SDK).

● Definitionen

Typdefinition für ME4000_P_AI_CALLBACK_PROC:

```
typedef void (_stdcall *
ME4000_P_AI_CALLBACK_PROC)(short* psValues,
unsigned int uiNumberOfValues, void* pCall-
backContext, int iLastError);
```

Typdefinition für ME4000_P_AI_TERMINATE_PROC:

```
typedef void (_stdcall *
    ME4000_P_AI_TERMINATE_PROC)(short*psValues,
    unsigned int uiNumberOfValues, void*
    pTerminateContext, int iLastError);
```

VC: me4000AIScan(unsigned int uiBoardNumber, unsigned int uiNumberOfChanLists, short* psBuffer, unsigned long ulBufferSizeValues, int iExecutionMode, ME4000_P_AI_CALLBACK_PROC pCallbackProc, void* pCallbackContext, unsigned int uiRefreshFrequency, ME4000_P_AI_TERMINATE_PROC pTerminateProc, void* pTerminateContext, unsigned long ulTimeOutSeconds);

LV50: me4000LV50_… (siehe `me4000LV.h`)

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)

<NumberOfChanLists>

Anzahl der Kanallistenabarbeitungen.

<Buffer>

Zeiger auf Datenpuffer, der nach Ende der Erfassung die linearisierten Messwerte enthält. Verwenden Sie die Funktion …*AIDigitToVolt* zur einfachen Umrechnung in Spannungswerte.

<BufferSizeValues>

In diesem Parameter muß die Größe des Datenpuffers in Anzahl der Messwerte übergeben werden.

<ExecutionMode>

Ausführungsmodus für diese Funktion wählen:

- ME4000_AI_SCAN_BLOCKING:
  Das Programm ist blockiert bis alle Messwerte erfasst wurden.
- ME4000_AI_SCAN_ASYNCHRONOUS:
  Der folgende Aufruf von …*AIStart* erzeugt automatisch einen neuen Thread, sodaß der aufrufende Thread nicht blockiert wird.

<CallbackProc> **~~LV~~, ~~VB~~, ~~VEE~~**

Callback-Funktion, die während der Erfassung in regelmäßigen Abständen aufgerufen wird. Der Funktion wird ein Zeiger auf die neu hinzugekommenen Werte sowie deren Anzahl übergeben. 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 weitergegeben wird. Falls keine Callback-Funktion angegeben wurde, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<RefreshFrequency>

Anzahl der Kanallistenabarbeitungen nach denen der Ringpuffer zyklisch ausgelesen werden soll. Übergabewert dient als Richtwert, der vom Treiber gegebenenfalls angepaßt wird. Bei Übergabe von ME4000_VALUE_NOT_USED ermittelt der Treiber einen sinnvollen Wert.

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

Callback-Funktion, die am Ende der Erfassung aufgerufen wird. Der Funktion wird ein Zeiger auf den Anfang des Datenpuffers sowie die Gesamtzahl der Werte übergeben. 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.

❮ 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.

me4000AISingle

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion wandelt einen einzelnen Wert. Der Wandlungsstart erfolgt wahlweise per Software oder auf ein externes Triggersignal (analog/digital). Es werden keine weiteren Funktionen für Konfiguration und Start der Erfassung benötigt.

☞ Hinweis!

Bei differentieller Messung können nur bipolare Eingangsbereiche verwendet werden! Diese Funktion wird immer im „Blocking"-Mode ausgeführt. D. h. der Programmfluß wird blockiert bis die Funktion zurückkehrt. In der Praxis ist dies nur relevant falls die Messung durch ein externes Triggersignal ausgelöst werden soll.

● Definitionen

VC: me4000AISingle(unsigned int uiBoardNumber, unsigned int uiChannelNumber, int iRange, int iSDMode, int iTriggerMode, int iExtTriggerEdge, unsigned long ulTimeOutSeconds, short* psDigitalValue);

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>`

A/D-Kanal-Nummer; mögliche Werte im single ended Betrieb: 0…31; im differentiellen Betrieb 0…15; (ME-4650/4660: 0…15 single ended)

`<Range>`

Auswahl des Eingangsspannungsbereichs:

- ME4000_AI_RANGE_BIPOLAR_10: ±10V
- ME4000_AI_RANGE_BIPOLAR_2_5: ±2,5V
- ME4000_AI_RANGE_UNIPOLAR_10: 0…10V
- ME4000_AI_RANGE_UNIPOLAR_2_5: 0…2,5V

<SDMode>

Single ended oder differentielle Messung:

- ME4000_AI_INPUT_SINGLE_ENDED:
  Single ended Messung
- ME4000_AI_INPUT_DIFFERENTIAL:
  Differentielle Messung

<TriggerMode>

Trigger-Ereignis für A/D-Teil:

- ME4000_AI_TRIGGER_SOFTWARE
  Wandlungsstart unmittelbar nach Aufruf dieser Funktion.
- ME4000_AI_TRIGGER_EXT_DIGITAL
  Bereit zur Wandlung nach Aufruf dieser Funktion.
  Wandlungsstart durch digitales Trigger-Signal.
- ME4000_AI_TRIGGER_EXT_ANALOG (nicht ME-4650/4660)
  Bereit zur Wandlung nach Aufruf dieser Funktion.
  Wandlungsstart durch analoges Trigger-Signal.

<ExtTriggerEdge>

Auswahl der externen Triggerflanke für A/D-Teil.

- ME4000_AI_TRIGGER_EXT_EDGE_RISING
  Start durch steigende Triggerflanke.
- ME4000_AI_TRIGGER_EXT_EDGE_FALLING
  Start durch fallende Triggerflanke.
- ME4000_AI_TRIGGER_EXT_EDGE_BOTH
  Start durch steigende oder fallende Triggerflanke.
- 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.

<DigitalValue>

Zeiger auf den linearisierten Messwert (vorzeichenbehaftet). Verwenden Sie die Funktion …*AIDigitToVolt* zur einfachen Umrechnung in Spannungswerte.

❮ 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.

me4000AIStart

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Mit Aufruf dieser Funktion wird die Karte je nach Konfiguration von Hardware und Software für die Erfassung „scharfgemacht“. In der Betriebsart „Software-Start“ wird die Erfassung unmittelbar nach Aufruf dieser Funktion gestartet. Bei Verwendung des externen Triggers hängt der Start vom jeweiligen Trigger-Ereignis ab (siehe Kap. 3.3.4 auf Seite 19).

Sofern nicht mit der Funktion …*AIReset* beendet wurde, kann nach Ende der Erfassung durch Aufruf dieser Funktion jederzeit von vorne begonnen werden, ohne daß vorher der A/D-Teil neu konfiguriert werden muß.

Falls Sie eine Erfassung in der Betriebsart „AIContinuous“ beenden wollen oder eine Erfassung in der Betriebsart „AIScan“ vorzeitig beenden wollen verwenden Sie die Funktionen …*AIStop* oder …*AIReset*.

☞ Hinweis!

Sollte bei Aufruf von …*AIStart* der A/D-Teil 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: me4000AIStart(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.

me4000AIStop

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Die Daten-Erfassung wird sofort gestoppt. Die Messwerte, die seit dem letzten „Abholen“ erfaßt wurden gehen verloren. Die Konfiguration des A/D-Teils bleibt erhalten (Kanalliste, Timer, etc.). Ein erneutes Starten mit der Funktion …*AIStart* ist jederzeit möglich.

● Definitionen

VC: me4000AIStop(unsigned int uiBoardNumber, int iReserved);
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)

`<Reserved>`

Dieser Parameter ist reserviert. Bitte übergeben Sie „0“.

‹ 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 Analoge Ausgabe

me4000AOAppendNewValues

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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 22, 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. 105ff).

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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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. 4.2 auf Seite 51 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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

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…3

<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 Triggereingang 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 `<TriggerMode>`.

<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; der Wertebereich liegt zwischen -32768 (-10 V) und +32767 (+10 V - LSB)

❮ 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-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

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:

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 jener Kanäle, die simultan ausgegeben werden sollen.

<Count>

Anzahl der im Wertefeld <ChannelNumber> gelisteten Kanäle. Gilt auch für die Parameter <ExtTriggerEnable>, <Ext-TriggerEdge> 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_*x*). 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 <Channel-Number>.

- 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); der Wertebereich liegt zwischen -32768 (-10 V) und +32767 (+10 V - LSB). 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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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 145).

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 53, 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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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 53, 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 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>.

- 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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

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. 12 auf Seite 22):

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

● 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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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 60.

Siehe Programmierbeispiele, die im ME-SDK enthalten sind.

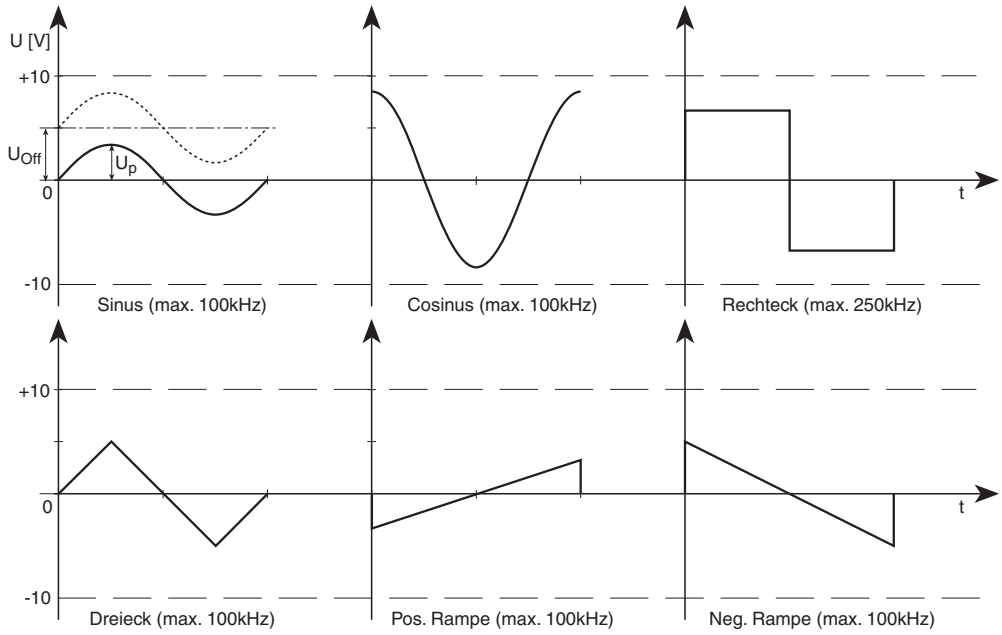


Abb. 57: Offset, Amplitude, Signalformen

☞ 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-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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 51, sowie in den Programmbeispielen, die im ME-Software-Developer-Kit (ME-SDK) enthalten sind.

☞ Hinweis:

Sofern die Größe des Datenpuffers 4096 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_ASYNCHRONOUS:
  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.

5.3.5 Digitale Ein-/Ausgabe

5.3.5.1 Bitpattern-Ausgabe

me4000DIOBPAppendNewValues

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Diese Funktion dient dem kontinuierlichen Nachladen des Ringpuffers während einer laufenden Bitmuster-Ausgabe.

Verwenden Sie die Funktionen *…DIOBPStop* oder *…DIOBPReset* zum Beenden der Ausgabe.

☞ Hinweis!

Sie müssen nicht den Gleichen, wie in *…DIOBPContinuous* verwendeten Datenpuffer verwenden.

● Definitionen

VC: me4000DIOBPAppendNewValues(unsigned int uiBoardNumber, 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)

`<Buffer>`

Zeiger auf den Datenpuffer, der mit dem aktuell nachzuladenden Bitmusterstrom gefüllt sein muss.

`<NumberOfValuesToAppend>`

Anzahl der Werte im Datenpuffer. Bei Übergabe von „0“ können Sie im Parameter `<NumberOfValuesAppended>` die Anzahl der Werte abfragen, die bei Rückkehr der Funktion im Datenpuffer Platz finden würden.

<ExecutionMode>

Ausführungsmodus für diese Funktion wählen:

- ME4000_DIOBP_APPEND_NEW_VALUES_BLOCKING:
  Das Programm ist blockiert bis alle Werte im Ringpuffer Platz gefunden haben.
- ME4000_DIOBP_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 ins FIFO 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.

me4000DIOBPConfig

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Diese Funktion konfiguriert die Karte für eine timergesteuerte Bitmuster-Ausgabe über die digitalen Ports. Sie können zwischen den Betriebsarten „BitPattern-Continuous“ und „BitPattern-Wraparound“ wählen. Gestartet wird die Ausgabe stets mit der Funktion *…DIOBP-Start* entweder sofort (Software-Start) oder 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 Bitmuster-Ausgabe muß nun 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 500 kS/s für TTL-Ports und 172 kS/s für den optoisolierten Port A der „i“-Versionen.

Beachten Sie, daß zu Beginn für jeden Port, der in die Ausgabe einbezogen werden soll, die Funktion *…DIOBP**Port**Config* aufgerufen werden muß (siehe auch Abb. 44, Seite 65).

☞ Hinweis:

Die Funktion *…FrequencyToTicks* bzw. *…TimeToTicks* bietet Ihnen eine bequeme Umrechnungsmöglichkeit von Frequenz bzw. Periodendauer in Ticks zur Übergabe an den Timer (siehe Seite 105ff).

Beachten Sie, daß während dieser Betriebsart keine timergesteuerte Ausgabe auf D/A-Kanal 3 möglich ist. Sollten benötigte Hardware-Ressourcen bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

● Definitionen

VC: me4000DIOBPConfig(unsigned int uiBoardNumber, 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)

<Ticks>

Anzahl der Ticks für den 32 Bit Timer, der die Sample-Rate bestimmt. Der Wertebereich liegt zwischen 66 (42Hex) für TTL-Ports bzw. 192 (C0Hex) für optoisolierte Ports und 2^{32}-1 (FFFFFFFFHex) Ticks.

<TriggerMode>

Trigger-Ereignis zum Start der Bitpattern-Ausgabe:

- ME4000_DIOBP_TRIGGER_SOFTWARE
  Start per Software nach Aufruf der Funktion *…DIOBPStart*.
- ME4000_DIOBP_TRIGGER_EXT_DIGITAL
  Bereit zur Ausgabe nach Aufruf der Funktion *…DIOBPStart*.
  Ausgabe wird durch externes Trigger-Signal gestartet.

<ExtTriggerEdge>

Auswahl der Triggerflanke (Triggereingang ist DA_TRIG_3).

- ME4000_DIOBP_TRIGGER_EXT_EDGE_RISING
  Start durch steigende Flanke.
- ME4000_DIOBP_TRIGGER_EXT_EDGE_FALLING
  Start durch fallende Flanke.
- ME4000_DIOBP_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.

me4000DIOBPContinuous

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Diese Funktion ermöglicht eine timergesteuerte Bitmuster-Ausgabe bis 500 kS/s über die digitalen Ports. Sie können damit beliebige Bitmuster ausgeben, die sich nach Beginn der Ausgabe auch ändern können (im Gegensatz zur Betriebsart „BitPattern-Wraparound“). Der Timer gibt ein festes Zeitraster (Sample-Rate) für die Ausgabe der Bitmuster vor (siehe *…DIOBPConfig*). Allokieren sie einen Datenpuffer definierter Größe, der das erste auszugebende Bitmusterpaket enthält. Verwenden Sie die Funktion *…DIOBPAppendNewValues* zum kontinuierlichen Nachladen. Dies kann mit oder ohne Callback-Funktion geschehen.

Gestartet wird die Ausgabe stets mit der Funktion *…DIOBPStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe *…DIOBPConfig*). Mit der Funktion *…DIOBPStop* können Sie die Ausgabe sofort beenden. Sofern zwischenzeitlich die Betriebsart für diesen Kanal nicht gewechselt wurde, kann die Ausgabe mit der Funktion *…DIOBPStart* jederzeit von vorne gestartet werden. Mit der Funktion *…DIOBPReset* wird im Vergleich zu *…DIOBPStop* auch das FIFO gelöscht und damit die Ausgabe vollständig beendet.

☞ Hinweis!

Beachten Sie, daß während dieser Betriebsart keine timergesteuerte Ausgabe auf D/A-Kanal 3 möglich ist. Sollten benötigte Hardware-Ressourcen bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

● Definitionen

Typdefinition für ME4000_P_DIOBP_CALLBACK_PROC:

```
typedef void (_stdcall *
ME4000_P_DIOBP_CALLBACK_PROC)
(unsigned long ulBufferAvailable,
void* pCallbackContext);
```

VC: me4000DIOBPContinuous(unsigned int uiBoardNumber, short* psBuffer, unsigned long ulDataCount, ME4000_P_DIOBP_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)

<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.

<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.

me4000DIOBPGetStatus

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Funktion dient der Abfrage ob eine Bitmuster-Ausgabe in den Betriebsarten „Bitpattern-Continuous“ und „BitPattern-Wraparound“ noch läuft oder das FIFO bereits „leer gelaufen“ ist. Dies ist dann der Fall wenn Sie das FIFO entweder 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: me4000DIOBPGetStatus(unsigned int uiBoardNumber, 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)

`<WaitIdle>`

„Rückkehr-Verhalten“ dieser Funktion:

- ME4000_DIOBP_WAIT_NONE
  Funktion gibt im Parameter `<Status>` den aktuellen Betriebszustand sofort zurück.
- ME4000_DIOBP_WAIT_IDLE
  Funktion kehrt erst dann Ende der Ausgabe zurück (FIFO leer). In diesem Fall enthält der Parameter `<Status>` stets den Wert ME4000_DIOBP_STATUS_IDLE.

`<Status>`

Aktueller Betriebszustand:

- ME4000_DIOBP_STATUS_IDLE
  Die Ausgabe ist beendet, d. h. das FIFO ist leer.
- ME4000_DIOBP_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.

me4000DIOBPPortConfig

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Diese Funktion dient der Portzuordnung für die Bitmuster-Ausgabe.

Das 16 Bit breite Bitmuster wird nach „Low-Byte“ (ME4000_DIOBP_OUTPUT_BYTE_LOW) und „High-Byte“ (ME4000_DIOBP_OUTPUT_BYTE_HIGH) getrennt und kann byteweise den 8 Bit breiten Digital-Ports A, B, C und/oder D) zugeordnet werden (siehe Abb. 44, Seite 65).

Diese Funktion muß für jeden Port, der in die Bitmuster-Ausgabe einbezogen werden soll getrennt aufgerufen werden. Verwenden Sie Funktion …*DIOBPConfig* für die weitere Konfiguration. Die Funktion …*DIOConfig* wird für diese Betriebsart nicht benötigt.

☞ Hinweis!

Bei optoisolierten Karten („i“-Versionen) ist die Richtung von Port A und B durch die Hardware vorgegeben. In diesem Fall ist Port A Ausgangsport und Port B Eingangsport. Port B kann in diesem Fall nicht für die Bitmuster-Ausgabe verwendet werden, bleibt jedoch weiterhin als Eingangsport nutzbar. Grundsätzlich sind nicht für die Bitmuster-Ausgabe verwendete Ports als „normale“ Digital-I/O-Ports nutzbar.

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

● Definitionen

VC: me4000DIOBPPortConfig(unsigned int uiBoardNumber, unsigned int uiPortNumber, int iOutputMode);

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
- ME4000_DIO_PORT_C: Port C
- ME4000_DIO_PORT_D: Port D

<OutputMode>

Port-Konfiguration für Bitmuster-Ausgabe (siehe Abb. 44, Seite 65):

- ME4000_DIOBP_OUTPUT_BYTE_LOW: Low-Byte des FIFOs (Bit 7…0)
- ME4000_DIOBP_OUTPUT_BYTE_HIGH: High-Byte des FIFOs (Bit 15…8)

‹ 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.

me4000DIOBPReset

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Diese Funktion beendet eine laufende Ausgabe in den Betriebsarten „BitPattern-Continuous“ oder „BitPattern-Wraparound“. Die Ausgabe wird sofort und vollständig beendet. Danach wird der korrespondierende Ringpuffer gelöscht und die betroffenen Digital-Ports geben das Bitmuster 0000Hex aus.

● Definitionen

VC: me4000DIOBPReset (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.

me4000DIOBPStart

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Funktion zum Starten der Bitmuster-Ausgabe in den Betriebsarten „BitPattern-Continuous“ und „BitPattern-Wraparound“. Falls Sie zuvor die Option „Externer Trigger“ gewählt haben, wird die Ausgabe durch eine entsprechende Flanke an Pin 65 (DA_TRIG_3) gestartet.

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 65, sowie in den Beispielprogrammen, die im ME-SDK enthalten sind.

☞ Hinweis:

Sollten benötigte Hardware-Ressourcen 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: me4000DIOBPStart(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.

me4000DIOBPStop

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Funktion zum Beenden der Bitmuster-Ausgabe in den Betriebsarten „BitPattern-Continuous“ und „BitPattern-Wraparound“. In der Betriebsart „BitPattern-Continuous“ wird die Ausgabe sofort und vollständig beendet. Nach Beendigung der Ausgabe wird das Bitmuster 0000Hex ausgegeben und der Ringpuffer gelöscht. In der Betriebsart „BitPattern-Wraparound“ können Sie mit dem Parameter `<StopMode>` selbst bestimmen, ob die Ausgabe sofort beendet und das Bitmuster 0000Hex ausgegeben werden soll oder mit dem letzten (bekannten) Bitmuster im Datenpuffer gestoppt werden soll. Sofern die Betriebsart für diesen Kanal nicht gewechselt wurde kann die Ausgabe mit der Funktion *…DIOBPStart* jederzeit von vorne gestartet werden.

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 65, sowie in den Beispielprogrammen, die im ME-SDK enthalten sind.

● Definitionen

VC: me4000DIOBPStop(unsigned int uiBoardNumber, 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)

<StopMode>

- ME4000_DIOBP_STOP_MODE_LAST_VALUE
  Ausgabe mit letztem Wert im Ringpuffer definiert beenden (nur sinnvoll in Verbindung mit der Funktion *…DIOBPWraparound*).
- ME4000_DIOBP_STOP_MODE_IMMEDIATE
  Ausgabe sofort beenden und Bitmuster 0000Hex 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.

me4000DIOBPWraparound

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Mit dieser Funktion wird der betreffende Kanal für die Betriebsart „BitPattern-Wraparound“ vorbereitet. Sie können in diesem Modus beliebige Bitmuster wiederholt ausgeben. Vor Beginn der Bitmusterausgabe muß der Ringpuffer einmalig beladen werden. Erzeugen sie einen Datenpuffer definierter Größe mit dem auszugebenden Bitmusterstrom. Der Timer gibt ein festes Zeitraster (Sample-Rate) für die Ausgabe der Bitmuster vor (siehe *…DIOBPConfig*).

Gestartet wird die Ausgabe stets mit der Funktion *…DIOBPStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe *…DIOBPConfig*).

Mit der Funktion *…DIOBPStop* 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 Bitmuster beendet. Sofern zwischenzeitlich die Betriebsart für diesen Kanal nicht gewechselt wurde, kann die Ausgabe mit der Funktion *…DIOBPStart* jederzeit von vorne gestartet werden. Mit der Funktion *…DIOBPReset* wird im Vergleich zu *…DIOBPStop* auch das FIFO gelöscht und damit die Ausgabe **vollständig** beendet.

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 65, sowie in den Programmbeispielen, die im ME Software-Developer-Kit (ME-SDK) enthalten sind.

☞ Hinweis:

Beachten Sie, daß während dieser Betriebsart keine timergesteuerte Ausgabe auf D/A-Kanal 3 möglich ist. Sollten benötigte Hardware-Ressourcen bereits aktiv sein, wird eine Fehlermeldung ausgegeben.

Sofern die Größe des Datenpuffers 4096 Werte nicht übersteigt und die Ausgabe „unendlich“ erfolgt, läuft die Ausgabe auf Firmware-Ebene, d. h. der Host-Rechner wird nicht belastet!

● Definitionen

Typdefinition für ME4000_P_DIOBP_TERMINATE_PROC:

```
typedef void (_stdcall *
ME4000_P_DIOBP_TERMINATE_PROC)
(void* pTerminateContext);
```

VC: me4000DIOBPWraparound(unsigned int uiBoardNumber, short* psBuffer, unsigned long ulDataCount, unsigned long ulLoops, int iExecutionMode, ME4000_P_DIOBP_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)

`<Buffer>`

Zeiger auf benutzerallokierten Datenpuffer mit dem auszugebenden Bitmuster.

`<DataCount>`

Anzahl der Werte im Datenpuffer `<Buffer>`.

`<Loops>`

Dieser Parameter gibt an, wie oft das Bitmuster im Datenpuffer ausgegeben werden sollen. Für unendlich übergeben Sie die Konstante: ME4000_DIOBP_WRAPAROUND_INFINITE.

<ExecutionMode>

Ausführungsmodus für diese Funktion wählen:

- ME4000_DIOBP_WRAPAROUND_BLOCKING:
  Das Programm ist blockiert bis alle Werte ausgegeben wurden. In Verbindung mit einer „unendlichen“ Ausgabe (siehe Parameter <Loops>) ist diese Konstante nicht möglich.
- ME4000_DIOBP_WRAPAROUND_ASYNCHRONOUS:
  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.

❮ 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.

5.3.5.2 Digitale Standard-Ein-/Ausgabe

me4000DIOConfig

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion dient der Richtungsumschaltung der Ports für eine digitale Standard-Ein-/Ausgabe. Die Richtung kann für jeden der 8 Bit breiten Ports unabhängig konfiguriert werden (siehe auch „Hinweis“).

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

☞ Hinweis!

Bei Karten mit optoisoliertem Digital-I/O-Teil („i“-Versionen) ist die Richtung von Port A und B durch die Hardware vorgegeben. In diesem Fall ist Port A Ausgangsport und Port B Eingangsport. Port C und D sind weiterhin frei konfigurierbar.

Die Konfiguration der Ports für die Bitmuster-Ausgabe erfolgt mit der Funktion *…DIOBPPortConfig* (siehe Seite 162).

● 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
- ME4000_DIO_PORT_C: Port C
- ME4000_DIO_PORT_D: Port D

<PortDirection>

Port-Richtung für Standard-Ein/Ausgabe:

- ME4000_DIO_PORT_INPUT:
  Eingangsport (Port B bei „i“-Versionen)
- ME4000_DIO_PORT_OUTPUT:
  Ausgangsport (Port A bei „i“-Versionen)

‹ 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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Liefert den Zustand der selektierten Bits zurück. Ausgangsports können mit dieser Funktion auch rückgelesen werden.

☞ 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
- ME4000_DIO_PORT_C: Port C
- ME4000_DIO_PORT_D: Port D

<BitNumber>

Nummer des Bits (0…7), das abgefragt werden soll

<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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Liest ein Byte vom spezfizierten Port. Ausgangsports können mit dieser Funktion 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
- ME4000_DIO_PORT_C: Port C
- ME4000_DIO_PORT_D: Port D

<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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Die Richtung der Ports wird auf Eingang gesetzt mit Ausnahme von Port A bei optoisolierten Varianten.

Eine evtl. laufende Bitmuster-Ausgabe oder der Betrieb des ME-MultiSig-Systems wird durch Aufruf dieser Funktion nicht gestört. Es wird eine entsprechende Fehlermeldung ausgegeben.

● 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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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
- ME4000_DIO_PORT_C: Port C
- ME4000_DIO_PORT_D: Port D

<BitNumber>

Nummer des Bits (0…7), das gesetzt werden soll

<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-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

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
- ME4000_DIO_PORT_C: Port C
- ME4000_DIO_PORT_D: Port D

<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.

5.3.6 Zählerfunktionen

me4000CntPWMStart

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Diese Funktion konfiguriert den Zählerbaustein (8254) für die Betriebsart „Pulsweiten-Modulation“ (PWM) und startet die Ausgabe. Eine anderweitige Nutzung der Zähler 0…2 ist in dieser Betriebsart nicht möglich. Eine vorausgehende Programmierung dieser Zähler wird überschrieben. Das Signal steht an Pin 41 (OUT_2) der Sub-D-Buchse zur Verfügung. Ein Basistakt (max. 10 MHz) muß von außen zugeführt werden. Zähler 0 kann als Vorteiler verwendet werden (siehe Abb. 20 auf Seite 28). Die Frequenz des Ausgangssignals kann max. 50 kHz betragen und errechnet sich folgendermaßen:

$$f_{OUT\_2} = \frac{\text{Basistakt}}{\texttt{<Prescaler>} \cdot 100} \qquad (\text{mit } \texttt{<Prescaler>} = 2\ldots(2^{16} - 1))$$

Das Tastverhältnis kann zwischen 1…99% in Schritten von 1% eingestellt werden (siehe Abb. 50 auf Seite 79).

☞ Hinweis!

Die Verwendung dieser Funktion ist nur sinnvoll in Verbindung mit der in Abb. 20 auf Seite 28 gezeigten externen Beschaltung.

● Definitionen

VC: me4000CntPWMStart(unsigned int uiBoardNumber, int iPrescaler, int iDutyCycle);
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)

`<Prescaler>`

Wert für Vorteiler (Zähler 0) im Bereich 2…65535.

`<DutyCycle>`

Tastverhältnis des Ausgangssignals von 1% –99% in 1%-Schritten einstellbar.

❮ 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.

me4000CntPWMStop

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Mit dieser Funktion beendet eine mit der Funktion …*CntPWMStart* gestartete Ausgabe.

● Definitionen

VC: me4000CntPWMStop(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.

me4000CntRead

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Puffert den aktuellen Zählerstand eines Zählers und liest diesen Wert.

● Definitionen

VC: me4000CntRead(unsigned int uiBoardNumber, unsigned int uiCounterNumber, unsigned short* pusValue);

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)

<CounterNumber>

Zähler, dessen Zählerstand eingelesen werden soll, mögliche Werte sind: 0, 1, 2

<Value>

Zählerstand als 16-Bit 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.

me4000CntWrite

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Konfiguriert einen Zähler mit der gewünschten Betriebsart und lädt einen 16 Bit Startwert. Der Start des Zählvorgangs erfolgt unmittelbar nach Aufruf dieser Funktion.

Für weitere Informationen zur Programmierung des Zählerbausteins 8254 beachten Sie bitte die Datenblätter der jeweiligen Hersteller (z. B. NEC, Intel, Harris) im Internet.

● Definitionen

VC: me4000CntWrite(unsigned int uiBoardNumber, unsigned int uiCounterNumber, int iMode, unsigned short usValue);

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)

`<CounterNumber>`

Zähler, der konfiguriert werden soll, mögliche Werte sind: 0, 1, 2

`<Mode>`

Betriebsart des Zählers (eine Beschreibung der Modi finden Sie im Kapitel 4.4 "Zähler-Betriebsarten" auf Seite 75).

- ME4000_CNT_MODE_0
  "Zustandsänderung bei Nulldurchgang"
- ME4000_CNT_MODE_1
  "Retriggerbarer One-Shot"
- ME4000_CNT_MODE_2
  "Asymmetrischer Teiler"
- ME4000_CNT_MODE_3
  "Symmetrischer Teiler"
- ME4000_CNT_MODE_4
  "Zählerstart durch Softwaretrigger"
- ME4000_CNT_MODE_5
  "Zählerstart durch Hardwaretrigger"

<Value>

16-Bit Startwert für spezifizierten Zähler; Wertebereich: 0…65535 (0000Hex…FFFFHex)

‹ 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.

5.3.7 Funktionen für externern Interrupt

me4000ExtIrqDisable

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Mit dieser Funktion deaktivieren Sie die externe Interruptfunktion.

● Definitionen

VC: int me4000ExtIrqDisable (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.

me4000ExtIrqEnable

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Mit dieser Funktion aktivieren Sie die externe Interruptfunktion. Ein ankommendes Interruptsignal wird direkt ans System weitergeleitet.

● Definitionen

Typdefinition für ME4000_P_EXT_IRQ_PROC:

```
typedef void (_stdcall *
ME4000_P_EXT_IRQ_PROC)
(void* pExtIrqContext);
```

VC: me4000ExtIrqEnable(unsigned int uiBoardNumber, ME4000_P_EXT_IRQ_PROC pExtIrqProc, void* pExtIrqContext);

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)

<ExtIrqProc> **~~LV~~, ~~VB~~, ~~VEE~~**

Zeiger auf anwenderdefinierte Callback-Routine. Falls diese Funktionalität nicht erwünscht ist, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<ExtIrqContext> **~~LV~~, ~~VB~~, ~~VEE~~**

Benutzerdefinierter Zeiger, der an die Callback-Funktion weitergegeben wird. Falls keine Callback-Funktion angegeben wurde, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

❮ 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.

me4000ExtIrqGetCount

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion ermittelt die Anzahl aller externen Interrupts seit dem Starten des Gerätes. Zweck dieser Funktion ist es, auch unter grafischen Programmieroberflächen wie Agilent VEE oder LabVIEW™ Interruptfunktionen zur Verfügung stellen zu können. Wie üblich, muß auch hier die Interruptfunktionalität mit den Funktionen *…ExtIrqEnable* und *…ExtIrqDisable* aktiviert bzw. deaktiviert werden. Durch Abfrage des Zahlenwertes im Parameter `IrqCount` ist es möglich, relativ zu einer vorherigen Abfrage festzustellen, ob ein Interrupt eingetroffen ist oder nicht.

● Definitionen

VC: me4000ExtIrqGetCount(unsigned int uiBoardNumber, unsigned int *puiIrqCount);

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)

<IrqCount>

Anzahl der seit dem Gerätestart eingetroffenen Interrupts.

☞ Beispiel

```
me4000ExtIrqEnable(0, ME4000_POINTER_NOT_USED,
   ME4000_POINTER_NOT_USED);
me4000ExtIrqGetCount(0, &iIrqCountBefore);

Sleep(1000);           //auf Interrupts warten

me4000ExtIrqGetCount(0, &iIrqCountAfter);
me4000ExtIrqDisable(0);
IrqCount = (iIrqCountAfter-iIrqCountBefore);
```

‹ 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.

5.3.8 MultiSig-Funktionen

me4000MultiSigAddressLED

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Identifikation der spezifizierten Basiskarte durch Ansteuerung der jeweiligen Adress-LED z. B. zu Wartungszwecken.

☞ Hinweis!

Zur Nutzung dieser Funktionalität in Verbindung mit optoisolierten Versionen der ME-4600 Serie empfehlen wir einen Anschlußadapter vom Typ ME-AA4-3i.

● Definitionen

VC: me4000MultiSigAddressLED(unsigned int uiBoardNumber, unsigned int uiBase, int iLEDStatus);

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)

`<Base>`

Adresse der Basiskarte; mögliche Werte: 0…7
(Masterkarte hat immer die Adresse „0“)

`<LEDStatus>`

LED auf Basiskarte ein/ausschalten:

- ME4000_MULTISIG_LED_OFF
  Adress-LED ausschalten
- ME4000_MULTISIG_LED_ON
  Adress-LED einschalten

‹ 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.

me4000MultiSigClose

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion schließt den mit …*MultiSigOpen* geöffneten Konfigurationsmodus ab. Reservierte Hardware-Ressourcen werden wieder freigegeben. Beachten Sie auch Kap. 4.5 „ME-MultiSig-Steuerung“ auf Seite 79ff.

● Definitionen

VC: int me4000MultiSigClose (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.

me4000MultiSigOpen

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion eröffnet den Konfigurationsmodus für folgende Funktionalitäten der Multiplexer-Basiskarten ME-MUX32-M/S:

- Verstärkung einstellen (siehe …*MultiSigSetGain*)
- Adress-LED ansteuern (siehe …*MultiSigAddressLED*)
- Genereller Reset (siehe …*MultiSigReset*)

Hinweis: Zur Nutzung dieser Funktionalitäten in Verbindung mit optoisolierten Versionen der ME-4600 Serie empfehlen wir einen Anschlußadapter vom Typ ME-AA4-3i.

Folgende Hardware-Ressourcen werden reserviert:

- Ohne Optoisolierung: Digital-Port A und B.
- Mit Optoisolierung: Digital-Port A und C.

Beachten Sie auch Kap. 4.5 „ME-MultiSig-Steuerung“ auf Seite 79ff.

● Definitionen

VC: int me4000MultiSigOpen (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.

me4000MultiSigReset

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Rücksetzen aller Master- und Slavekarten in den Grundzustand. Verstärkung wird auf V=1 gesetzt und die Adress-LEDs werden ausgeschaltet.

☞ Hinweis!

Zur Nutzung dieser Funktionalität in Verbindung mit optoisolierten Versionen der ME-4600 Serie empfehlen wir einen Anschlußadapter vom Typ ME-AA4-3i.

● Definitionen

VC: me4000MultiSigReset(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.

me4000MultiSigSetGain

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion dient der Verstärkungseinstellung für die spezifizierte Kanalgruppe. Standardeinstellung ist der Verstärkungsfaktor V=1.

☞ Hinweis!

Zur Nutzung dieser Funktionalität in Verbindung mit optoisolierten Versionen der ME-4600 Serie empfehlen wir einen Anschlußadapter vom Typ ME-AA4-3i.

● Definitionen

VC: me4000MultiSigSetGain(unsigned int uiBoardNumber, unsigned int uiBase, int iChannelGroup, int iGain);

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)

`<Base>`

Adresse der Basiskarte; mögliche Werte: 0…7
(Masterkarte hat immer die Adresse „0“)

`<ChannelGroup>`

Auswahl der Kanalgruppe:

- ME4000_MULTISIG_GROUP_A:
  Kanalgruppe A
- ME4000_MULTISIG_GROUP_B:
  Kanalgruppe B

<Gain>

Einstellung des Verstärkungsfaktors für spezifizierte Kanalgruppe (ein mit der Funktion …*MultiSigAISingle* eingestellter Verstärkungsfaktor wird überschrieben):

- ME4000_MULTISIG_GAIN_1
  Verstärkungsfaktor 1 (Standard)
- ME4000_MULTISIG_GAIN_10
  Verstärkungsfaktor 10
- ME4000_MULTISIG_GAIN_100
  Verstärkungsfaktor 100

❮ 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.

5.3.8.1 „Mux“-Funktionen

Die folgenden Funktionen dienen der analogen Erfassung über das ME-MultiSig-System in Verbindung mit einer Karte der ME-4600 Serie (siehe auch Kap. „Mux“-Betrieb auf Seite 79 und Handbuch „ME-MultiSig“-System).

me4000MultiSigAIClose

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion schließt die mit …*MultiSigAIOpen* freigegebene Funktionalität ab. Reservierte Hardware-Ressourcen werden wieder freigegeben. Beachten Sie auch Kap. 4.5 „ME-MultiSig-Steuerung“ auf Seite 79ff.

● Definitionen

VC: int me4000MultiSigAIClose (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.

me4000MultiSigAIConfig

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Diese Funktion konfiguriert die Hardware der ME-4600 für eine timergesteuerte Erfassung in Verbindung mit dem ME-MultiSig-System in der Betriebsart „Single-Mux“ (siehe Handbuch „ME-MultiSig“-System).

Sie konfiguriert die Timer, übergibt die Mux-Kanalliste, bestimmt den Erfassungsmodus `<AcqMode>` und legt die ext. Triggerquelle und Triggerflanke fest. Die Kanalliste zur Konfiguration des A/D-Teils der ME-4600 wird automatisch erstellt. Die ME-4600 verwendet hier immer den Eingangsspannungsbereich ±10V und arbeitet single ended.

Nach Aufruf von *…MultiSigAIScan* oder *…MultiSigAIContinuous* wird die Erfassung stets mit der Funktion *…MultiSigAIStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal gestartet.

☞ Hinweis:

Es stehen ein 32 Bit breiter CHAN-Timer sowie ein 36 Bit breiter SCAN-Timer zur Verfügung. Der CHAN-Timer bestimmt die Abtastrate (Sample-Rate) innerhalb der „Mux“-Kanalliste. Die max. CHAN-Zeit beträgt ca. 130s, die min. CHAN-Zeit beträgt 2 µs für TTL-Versionen bzw. 5,8µs für optoisolierte Versionen. Der SCAN-Timer bestimmt die Zeit zwischen dem jeweils ersten Eintrag in der „Mux“-Kanalliste von zwei aufeinander folgenden Kanallistenabarbeitungen. Die Verwendung ist optional. Als gemeinsame Zeitbasis nutzen alle Timer einen 33 MHz Takt. Daraus ergibt sich eine Periodendauer von 30,30 ns, die als kleinste Zeiteinheit definiert wird und im Folgenden „1 Tick“ genannt wird. Die gewünschte Periodendauer muß nun als Vielfaches eines Ticks übergeben werden an die Parameter `<ChanTicks>`, `<ScanTicksLow>` und `<ScanTicksHigh>` übergeben werden. `<ScanTicksHigh>` wird nur für SCAN-Zeiten >130s benötigt.

Verwenden Sie die Funktionen *…FrequencyToTicks* und *…TimeToTicks* (siehe Seite 105ff) zur bequemen Umrechnung von Frequenz bzw. Periodendauer in Ticks zur Übergabe an die Timer.

● Definitionen

VC: me4000MultiSigAIConfig(unsigned int uiBoardNumber, unsigned int uiAIChannelNumber, unsigned char *pucMuxChanList, unsigned int uiMuxChanListCount, unsigned long ulReserved, unsigned long ulChanTicks, unsigned long ulScanTicksLow, unsigned long ulScanTicksHigh, int iAcqMode, int iExtTriggerMode, 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)

`<AIChannelNumber>`

A/D-Kanal-Nummer für den die „Mux"-Kette konfiguriert wurde. Die Kanal-Nummer muß mit der Lötbrücke „A" auf der Master-Karte korrespondieren; mögliche Werte: 0…31 (ME-4650/4660: 0…15)

`<MuxChanList>`

Zeiger auf den Anfang der Mux-Kanalliste zur Steuerung der Multiplexer. Allokieren Sie vorab ein Wertefeld definierter Größe, in das Sie die Mux-Kanalnummern (0…255) in der gewünschten Reihenfolge eintragen.

`<MuxChanListCount>`

Anzahl der Mux-Kanallisten-Einträge.

`<Reserved>`

Dieser Parameter ist reserviert. Übergeben Sie den Wert „0".

`<ChanTicks>`

Anzahl der Ticks für den CHAN-Timer (32 Bit), der die Abtastrate festlegt. Der Wertebereich liegt zwischen 66 (42Hex) für TTL-Versionen bzw. 192 (C0Hex) für optoisolierte Versionen und 2^{32}-1 (FFFFFFFFHex) Ticks.

`<ScanTicksLow>`

Anzahl der Ticks für den niederwertigen Teil (Bits 31…0) des insgesamt 36 Bit breiten SCAN-Timers (siehe auch `ScanTicksHigh`). Er legt die Zeit zwischen der Wandlung des jeweils ersten Eintrags in der „Mux"-Kanalliste von zwei aufeinanderfolgenden Kanallistenabarbeitungen fest. Wenn Sie diesen Timer nicht benutzen möchten, übergeben Sie hier ***und*** in `<ScanTicksHigh>` ME4000_VALUE_NOT_USED.

Für eine sinnvolle SCAN-Zeit gilt (siehe auch Abb. 25):
(Anzahl der „Mux"-Kanallisten-Einträge x CHAN-Zeit) + „x" Ticks

Die max. erlaubte SCAN-Zeit beträgt 30 Minuten, dies entspricht 59.400.000.000 Ticks (DD4841200Hex).

<ScanTicksHigh>

Anzahl der Ticks für den höherwertigen Teil (Bits 35…32) des insgesamt 36 Bit breiten SCAN-Timers (siehe auch `ScanTicksLow`). Diesen Timer benötigen Sie nur für Scan-Zeiten über 130,15s - ansonsten übergeben Sie ME4000_VALUE_NOT_USED.

<AcqMode>

Erfassungsmodus für die timergesteuerte Erfassung:

- ME4000_AI_ACQ_MODE_SOFTWARE
  Wandlungsstart nach Aufruf der Funktion *…MultiSigAIStart.* Abarbeitung gemäß Timer-Einstellungen (siehe Abb. 25).
- ME4000_AI_ACQ_MODE_EXT
  Bereit zur Erfassung nach Aufruf der Funktion *…MultiSigAI-Start.* Erfassung beginnt mit dem ersten ext. Triggerimpuls. Abarbeitung gemäß Timer-Einstellungen. Weitere Triggerimpulse bleiben ohne Wirkung (siehe Abb. 32).
- ME4000_AI_ACQ_MODE_EXT_SINGLE_VALUE
  Bereit zur Erfassung nach Aufruf der Funktion *…MultiSigAI-Start.* Mit jedem ext. Triggerimpuls wird genau **ein** Wert gemäß Kanalliste gewandelt. Vom ersten Triggersignal bis zur ersten Wandlung vergeht einmal die CHAN-Zeit. Ansonsten bleiben die Timer-Einstellungen ohne Wirkung (siehe Abb. 33).
- ME4000_AI_ACQ_MODE_EXT_SINGLE_CHANLIST
  Bereit zur Erfassung nach Aufruf der Funktion *…MultiSigAI-Start.* Mit jedem ext. Triggerimpuls wird die Kanalliste einmal abgearbeitet. Die Abarbeitung erfolgt gemäß Timer-Einstellungen (siehe Abb. 34). Der SCAN-Timer bleibt jedoch ohne Wirkung!

<ExtTriggerMode>

Auswahl der externen Triggerquelle für den A/D-Teil.

- ME4000_AI_TRIGGER_EXT_DIGITAL
  Der digitale Triggereingang ist Triggerquelle.
- ME4000_AI_TRIGGER_EXT_ANALOG (nicht ME-4650/4660)
  Die analoge Triggereinheit ist Triggerquelle.
- ME4000_VALUE_NOT_USED
  Kein ext. Trigger verwendet. Siehe Parameter `<AcqMode>`.

<ExtTriggerEdge>

Auswahl der Triggerflanke für den A/D-Teil.

- ME4000_AI_TRIGGER_EXT_EDGE_RISING
  Eine steigende Triggerflanke wird ausgewertet.
- ME4000_AI_TRIGGER_EXT_EDGE_FALLING
  Eine fallende Triggerflanke wird ausgewertet.
- ME4000_AI_TRIGGER_EXT_EDGE_BOTH
  Sowohl steigende als auch fallende Triggerflanken werden ausgewertet.
- ME4000_VALUE_NOT_USED
  Kein ext. Trigger verwendet. Siehe Parameter <AcqMode>.

❮ 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.

me4000MultiSigAIContinuous

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Mit dieser Funktion wird die Software für eine timergesteuerte Erfassung vorbereitet bei der die Anzahl der Messwerte vorher unbekannt ist. Diese Funktion wird grundsätzlich asynchron ausgeführt. Die Messwerte werden entweder mit einer benutzerdefinierten Callback-Funktion oder durch wiederholten Aufruf der Funktion *…MultiSigAIGetNewValues* abgeholt.

Gestartet wird die Erfassung stets mit der Funktion *…MultiSigAIStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe *…MultiSigAIConfig*). Falls Sie mit einem externen Triggersignal arbeiten und dieses ausbleibt können Sie mit einem geeigneten „Time-Out"-Wert die Erfassung abbrechen. Durch Aufruf der Funktion *…MultiSigAIStop* oder *…MultiSigAIReset* wird die Erfassung beendet.

☞ Hinweis:

Zur Vorgehensweise beachten Sie bitte Kap. 4.1.3 "Timergesteuerte „AI-Betriebsarten"" auf S. 34ff, sowie die Programmbeispiele im ME-Software-Developer-Kit (ME-SDK).

● Definitionen

Typdefinition für ME4000_P_AI_CALLBACK_PROC:

```
typedef void (_stdcall *
ME4000_P_AI_CALLBACK_PROC)(short* psValues,
unsigned int uiNumberOfValues, void* pCall-
backContext, int iLastError);
```

VC: me4000MultiSigAIContinuous(unsigned int uiBoardNumber, ME4000_P_AI_CALLBACK_PROC pCallbackProc, void* pCallbackContext, unsigned int uiRefreshFrequency, 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)

<CallbackProc> **~~LV~~, ~~VB~~, ~~VEE~~**

Callback-Funktion, die während der Erfassung in regelmäßigen Abständen aufgerufen wird. Der Funktion wird ein Zeiger auf die neu hinzugekommenen Werte sowie deren Anzahl übergeben. 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 weitergegeben wird. Falls keine Callback-Funktion angegeben wurde, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<RefreshFrequency>

Anzahl der Kanallistenabarbeitungen nach denen der Ringpuffer zyklisch ausgelesen werden soll. Übergabewert dient als Richtwert, der vom Treiber gegebenenfalls angepaßt wird. Bei Übergabe von ME4000_VALUE_NOT_USED ermittelt der Treiber einen sinnvollen Wert.

<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.

‹ 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.

me4000MultiSigAIDigitToSize

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion erlaubt Ihnen die einfache Umrechnung der „Rohwerte“ [Digits] in die jeweilige physikalische Dimension unter Berücksichtigung des auf den Basiskarten eingestellten Verstärkungsfaktors (1, 10, 100) und eines optional verwendeten Aufsteck-Moduls zur Signalkonditionierung. Die Funktion kann auf Einzelwerte oder durch wiederholten Aufruf auf ein ganzes Wertefeld angewandt werden. Die Verwendung ist optional.

☞ Hinweis:

Verwenden Sie für timergesteuerten „Mux“-Betrieb stets die Funktion *…MultiSigAIExtractValues* vor Aufruf dieser Funktion. Nur so ist gewährleistet, daß unterschiedliche Verstärkungsfaktoren der Kanalgruppen und eine Bestückung mit unterschiedlichen Aufsteck-Modulen bei der Berechnung berücksichtigt werden können!

Diese Funktion setzt einen Eingangsspannungsbereich von ±10V der ME-4600 voraus. Sofern Sie mit den „MultiSig“-Funktionen arbeiten wird automatisch der Spannungsbereich ±10V verwendet.

Die Berechnung der Temperatur für Widerstandssensoren erfolgt nach DIN EN 60751, die für Thermoelemente nach DIN EN 60584. Weitere Informationen zur Temperaturberechnung finden Sie im ME-MultiSig-Handbuch.

● Definitionen

VC: me4000MultiSigAIDigitToSize(short sDigit, int iGain, int iModuleType, double dRefValue, double* pdSize);

LV: me4000LV_… (siehe `me4000LV.h`)

VB: me4000VB_… (siehe `me4000.bas`)

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

➔ Parameter

<Digit>

Übergabe eines „Rohwertes“, wie er nach der Erfassung im Datenpuffer steht.

<Gain>

Übergeben Sie hier denselben Verstärkungsfaktor wie in der Funktion …*MultiSigSetGain* bzw. …*MultiSigAISingle*:

- ME4000_MULTISIG_GAIN_1
  Verstärkungsfaktor 1 (Standard)
- ME4000_MULTISIG_GAIN_10
  Verstärkungsfaktor 10
- ME4000_MULTISIG_GAIN_100
  Verstärkungsfaktor 100

<ModuleType>

Falls Sie ein Aufsteckmodul zur Signalkonditionierung einsetzen, wählen Sie hier den Modul-Typ und verwenden Sie im Parameter `<Gain>` den Verstärkungsfaktor 1. Dies ist wichtig, damit der Messwert richtig berechnet werden kann. Falls Sie ohne Aufsteckmodul arbeiten, übergeben Sie die erste Konstante:

- ME4000_MULTISIG_MODULE_NONE
  Kein Aufsteckmodul verwendet (Standard)

- ME4000_MULTISIG_MODULE_DIFF16_10V
  Aufsteckmodul ME-Diff16 mit Eingangsbereich 10V
- ME4000_MULTISIG_MODULE_DIFF16_20V
  Aufsteckmodul ME-Diff16 mit Eingangsbereich 20V
- ME4000_MULTISIG_MODULE_DIFF16_50V
  Aufsteckmodul ME-Diff16 mit Eingangsbereich 50V

- ME4000_MULTISIG_MODULE_CURRENT16_0_20MA
  Aufsteckmodul ME-Current16 mit Eingangsbereich 0…20mA

- ME4000_MULTISIG_MODULE_RTD8_PT100
  Aufsteckmodul ME-RTD8 für RTDs vom Typ Pt100 (0,4Ω/K)
- ME4000_MULTISIG_MODULE_RTD8_PT500
  Aufsteckmodul ME-RTD8 für RTDs vom Typ Pt500 (2,0Ω/K)
- ME4000_MULTISIG_MODULE_RTD8_PT1000
  Aufsteckmodul ME-RTD8 für RTDs vom Typ Pt1000 (4,0Ω/K)

- ME4000_MULTISIG_MODULE_TE8_TYPE_B
  Aufsteckmodul ME-TE8, Kanal mit Thermoelement Typ B
- ME4000_MULTISIG_MODULE_TE8_TYPE_E
  Aufsteckmodul ME-TE8, Kanal mit Thermoelement Typ E
- ME4000_MULTISIG_MODULE_TE8_TYPE_J
  Aufsteckmodul ME-TE8, Kanal mit Thermoelement Typ J
- ME4000_MULTISIG_MODULE_TE8_TYPE_K
  Aufsteckmodul ME-TE8, Kanal mit Thermoelement Typ K
- ME4000_MULTISIG_MODULE_TE8_TYPE_N
  Aufsteckmodul ME-TE8, Kanal mit Thermoelement Typ N
- ME4000_MULTISIG_MODULE_TE8_TYPE_R
  Aufsteckmodul ME-TE8, Kanal mit Thermoelement Typ R
- ME4000_MULTISIG_MODULE_TE8_TYPE_S
  Aufsteckmodul ME-TE8, Kanal mit Thermoelement Typ S
- ME4000_MULTISIG_MODULE_TE8_TYPE_T
  Aufsteckmodul ME-TE8, Kanal mit Thermoelement Typ T
- ME4000_MULTISIG_MODULE_TE8_TEMP_SENSOR
  Aufsteckmodul ME-TE8, Kanal für Vergleichsstellentemperatur an Klemmleiste des Moduls

<RefValue>

- Falls Sie im Parameter <ModuleType> ein RTD-Modul ausgewählt haben:

  Zur exakten Berechnung der Temperatur müssen Sie hier den Konstant-Meßstrom I_M in Ampere [A] übergeben. Dieser muß zuvor mit einem hochgenauen Amperemeter gemessen werden (siehe Handbuch ME-MultiSig-System).

  Falls Sie die Meßtoleranz vernachlässigen möchten, rechnet die Funktion mit einem typischen Konstant-Meßstrom von $I_M = 500 \times 10^{-6}$ A. Übergeben Sie in diesem Fall die Konstante: ME4000_MULTISIG_I_MEASURED_DEFAULT.

- Falls Sie im Parameter <ModuleType> ein Thermoelementen-Modul ausgewählt haben:

  Da sich die Berechnung auf eine Vergleichsstellentemperatur von 0°C bezieht, müssen Sie mit einem integrierten Sensor vor Beginn der Messreihe die Temperatur an der Klemmeleiste des Aufsteckmoduls messen (siehe Parameter <ModuleType>). Anschließend wird der ermittelte Wert in diesem Parameter übergeben (in °C). Paramter <Size> gibt einen Zeiger auf den kompensierten Temperaturwert in °C zurück.

- In allen anderen Fällen übergeben Sie die Konstante: ME4000_VALUE_NOT_USED.

<Size>

Zeiger auf den errechneten Wert in der jeweiligen physikalischen Dimension: [V], [A], [°C] (je nach <ModuleType>)

❮ 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.

me4000MultiSigAIExtractValues

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Diese Funktion extrahiert aus dem Wertefeld aller erfassten Werte die Werte des spezifizierten Mux-Kanals korrespondierend zur Mux-Kanalliste. Um die Daten für mehrere Kanäle zu extrahieren, muß die Funktion für jeden Kanal getrennt aufgerufen werden.

● Definitionen

VC: me4000MultiSigAIExtractValues(unsigned int uiMuxChannelNumber, short* psAIBuffer, unsigned long ulAIDataCount, unsigned char *pucMuxChanList, unsigned int uiMuxChanListCount, short* psChanBuffer, unsigned long ulChanBufferSizeValues, unsigned long* pulChanDataCount);

LV: me4000LV_… (siehe me4000LV.h)

VB: me4000VB_… (siehe me4000.bas)

VEE: me4000VEE_… (siehe me4000VEE.h)

➔ Parameter

<MuxChannelNumber>

Nummer des Mux-Kanals innerhalb „Mux"-Kette, dessen Werte extrahiert werden sollen; mögliche Werte: 0…255

<AIBuffer>

Zeiger auf Datenpuffer mit allen erfassten Werten.

<AIDataCount>

Anzahl der Messwerte im Wertefeld <AIBuffer>.

<MuxChanList>

Zeiger auf „Mux"-Kanalliste, die mit der Funktion …*MultiSigAIConfig* übergeben wurde.

<MuxChanListCount>

Anzahl der Kanallisteneinträge von <MuxChanList>.

<ChanBuffer>

Zeiger auf Wertefeld in dem die extrahierten Werte des spezifizierten Kanals abgelegt werden.

<ChanBufferSizeValues>

Anzahl der extrahierten Werte im Wertefeld <ChanBuffer>.

<ChanDataCount>

Zeiger der nach Rückkehr der Funktion die Anzahl der tatsächlich in ChanBuffer abgelegten Werte enthält. Die Anzahl wird nie größer als ChanBufferSize sein, kann aber auch kleiner sein.

❮ 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.

me4000MultiSigAIGetNewValues

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

In Verbindung mit der Betriebsart „MultiSig-AIContinuous“ können sie mit dieser Funktion die Messwerte „Abholen“. In der Betriebsart „MultiSig-AIScan“ dient sie dem „Einsehen“ der Messwerte während einer im Hintergrund laufenden Erfassung (asynchron). Ein Anwendungsfall besteht z. B. darin, während einer längeren Erfassung die Messwerte einzulesen und anzuzeigen.

☞ Hinweis:

Ein Beispiel zur Vorgehensweise finden Sie im Abschnitt „Programmierung“ auf Seite 39, sowie in den Beispielprogrammen, die im ME-SDK enthalten sind.

● Definitionen

VC: me4000MultiSigAIGetNewValues(unsigned int uiBoardNumber, short* psBuffer, unsigned long ulNumberOfValuesToRead, int iExecutionMode, unsigned long* pulNumberOfValuesRead, int* piLastError);

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)

`<Buffer>`

Zeiger auf Datenpuffer, der die neuesten Daten der laufenden Erfassung enthält. Verwenden Sie die Funktion …*MultiSigAI-DigitToSize* zur einfachen Umrechnung in Spannungswerte.

`<NumberOfValuesToRead>`

Größe des Datenpuffers in Anzahl der Messwerte. Die Puffergröße sollte ein Vielfaches der Kanallistenlänge betragen. Bei Übergabe von „0“ können Sie im Parameter `<NumberOf-ValuesRead>` die Anzahl der Werte abfragen, die zur Abholung bereitstehen.

<ExecutionMode>

Ausführungsmodus für diese Funktion wählen:

- ME4000_AI_GET_NEW_VALUES_BLOCKING:
  Das Programm ist blockiert bis alle Messwerte abgeholt wurden.
- ME4000_AI_GET_NEW_VALUES_NON_BLOCKING:
  Es werden nur die aktuell vorhandenen Messwerte abgeholt.

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

<NumberOfValuesRead>

Zeiger, der nach Rückkehr der Funktion die Anzahl der tatsächlich im Datenpuffer abgelegten Werte enthält. Die Anzahl wird nie größer als `<NumberOfValuesToRead>` sein, kann aber auch kleiner sein, falls noch nicht so viele neue Messwerte vorliegen.

<LastError>

Dieser Parameter enthält den letzten Fehler, der seit dem letzten Aufruf dieser Funktion auftrat. Mögliche Fehler sind FIFO-Überlauf oder Datenpuffer-Überlauf. Ist kein Fehler aufgetreten, so wird „0“ (ME4000_NO_ERROR) zurückgegeben.

❮ 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.

me4000MultiSigAIGetStatus

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Funktion dient der Abfrage ob eine Erfassung in der Betriebsart „MultiSig-AIScan“ im Ausführungsmodus „ASYNCHRONOUS“ noch läuft oder bereits alle der erwarteten Messwerte erfasst wurden.

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

● Definitionen

VC: me4000MultiSigAIGetStatus(unsigned int uiBoardNumber, 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)

<WaitIdle>

„Rückkehr-Verhalten“ dieser Funktion:

- ME4000_AI_WAIT_NONE
  Funktion gibt im Parameter <Status> den aktuellen Betriebszustand sofort zurück.
- ME4000_AI_WAIT_IDLE
  Funktion kehrt erst zurück nachdem alle Werte erfasst wurden. In diesem Fall enthält der Parameter <Status> stets den Wert ME4000_AI_STATUS_IDLE.

<Status>

Aktueller Betriebszustand:

- ME4000_AI_STATUS_IDLE
  Die Erfassung ist beendet.
- ME4000_AI_STATUS_BUSY
  Die Erfassung 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.

me4000MultiSigAIOpen

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion bereitet den „Mux"-Betrieb vor. Entsprechende Hardware-Ressourcen werden reserviert:

- Ohne Optoisolierung: Digital-Port A und B.
- Mit Optoisolierung: Digital-Port A und C.
- D/A-Kanal 3 wird für analoge Ausgabe gesperrt (nur für ME-4680 relevant).
- Der A/D-Teil wird für die Erfassung mit den „normalen" AI-Funktionen (*me4000AI…*) gesperrt.

Beachten Sie auch Kap. 4.5 „ME-MultiSig-Steuerung" auf Seite 79ff.

● Definitionen

VC: int me4000MultiSigAIOpen (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.

me4000MultiSigAIReset

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Die Erfassung wird sofort und vollständig beendet. Alle bis dahin erfaßten Messwerte gehen verloren. Die Karte muß für eine erneute Erfassung neu konfiguriert werden (*…MultiSigAIConfig*, *…MultiSigAIScan*, *…MultiSigAIContinuous*).

● Definitionen

VC: me4000MultiSigAIReset (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.

me4000MultiSigAIScan

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Mit dieser Funktion wird die Software für die Erfassung einer von vornherein bekannten Anzahl an Messwerten vorbereitet. Es wird ein benutzerdefinierter Datenpuffer allokiert in dem die Messwerte abgelegt werden. Im Ausführungsmodus „BLOCKING“ kehrt der „Thread“, in dem die Funktion *…MultiSigAIStart* aufgerufen wurde erst nach Erfassung des letzten Wertes zurück. Im Ausführungsmodus „ASYNCHRONOUS“ wird die Erfassung als Hintergrundprozeß gestartet, d. h. durch Aufruf der Funktion *…MultiSigAIStart* wird automatisch ein neuer „Thread“ erzeugt. Parallel dazu können andere Aufgaben („Threads“) abgearbeitet werden. Falls gewünscht (z. B. bei einer längeren Erfassung), können Sie die Messwerte bereits während der Erfassung „einsehen“. Dies können Sie entweder mit einer benutzerdefinierten Callback-Funktion oder mit Hilfe der Funktion *…MultiSigAIGetNewValues* machen. Mit einer weiteren Callback-Funktion „Terminate“ können Sie (falls gewünscht) das Ende der Erfassung an Ihre Applikation melden lassen.

Gestartet wird die Erfassung stets mit der Funktion *…MultiSigAIStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe *…MultiSigAIConfig*). Falls Sie mit einem externen Triggersignal arbeiten und dieses ausbleibt können Sie mit einem geeigneten „Time-Out“-Wert die Erfassung abbrechen. Beendet wird die Erfassung automatisch nach Erfassung der erwarteten Messwerte.

☞ Hinweis:

Zur Vorgehensweise beachten Sie bitte Kap. 4.1.3 "Timergesteuerte „AI-Betriebsarten“" auf S. 34ff, sowie die Programmbeispiele im ME-Software-Developer-Kit (ME-SDK).

● Definitionen

Typdefinition für ME4000_P_AI_CALLBACK_PROC:

```
typedef void (_stdcall *
ME4000_P_AI_CALLBACK_PROC)(short* psValues,
unsigned int uiNumberOfValues, void* pCall-
backContext, int iLastError);
```

Typdefinition für ME4000_P_AI_TERMINATE_PROC:

```
typedef void (_stdcall *
ME4000_P_AI_TERMINATE_PROC)(short*psValues,
unsigned int uiNumberOfValues, void*
pTerminateContext, int iLastError);
```

VC: me4000MultiSigAIScan(unsigned int uiBoardNumber, unsigned int uiNumberOfMuxLists, short* psBuffer, unsigned long ulBufferSizeValues, int iExecutionMode, ME4000_P_AI_CALLBACK_PROC pCallbackProc, void* pCallbackContext, unsigned int uiRefreshFrequency, ME4000_P_AI_TERMINATE_PROC pTerminateProc, void* pTerminateContext, unsigned long ulTimeOutSeconds);

LV50: me4000LV50_… (siehe `me4000LV.h`)

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)

<NumberOfMuxLists>

Anzahl der „Mux"-Kanallistenabarbeitungen.

<Buffer>

Zeiger auf Datenpuffer, der die Daten der Erfassung enthält. Verwenden Sie die Funktion …*MultiSigAIDigitToSize* zur einfachen Umrechnung in die entsprechende physikalische Einheit.

<BufferSizeValues>

In diesem Parameter muß die Größe des Datenpuffers in Anzahl der Messwerte übergeben werden.

<ExecutionMode>

Ausführungsmodus für diese Funktion wählen:

- ME4000_AI_SCAN_BLOCKING:
  Das Programm ist blockiert bis alle Messwerte erfasst wurden.
- ME4000_AI_SCAN_ASYNCHRONOUS:
  Der folgende Aufruf von …*MultiSigAIStart* erzeugt automatisch einen neuen Thread, sodaß der aufrufende Thread nicht blockiert wird.

<CallbackProc> **~~LV~~, ~~VB~~, ~~VEE~~**

Callback-Funktion, die während der Erfassung in regelmäßigen Abständen aufgerufen wird. Der Funktion wird ein Zeiger auf die neu hinzugekommenen Werte sowie deren Anzahl übergeben.

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 weitergegeben wird. Falls keine Callback-Funktion angegeben wurde, übergeben Sie die Konstante ME4000_POINTER_NOT_USED.

<RefreshFrequency>

Anzahl der Kanallistenabarbeitungen nach denen der Ringpuffer zyklisch ausgelesen werden soll. Übergabewert dient als Richtwert, der vom Treiber gegebenenfalls angepaßt wird. Bei Übergabe von ME4000_VALUE_NOT_USED ermittelt der Treiber einen sinnvollen Wert.

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

„Terminate"-Funktion, die am Ende der Erfassung aufgerufen wird. Der Funktion wird ein Zeiger auf den Anfang des Datenpuffers sowie die Gesamtzahl der Werte übergeben. 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.

❮ 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.

me4000MultiSigAISingle

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
✔	✔	✔	✔

Diese Funktion wandelt einen einzelnen Wert von einem der bis zu 256 Kanäle der „Mux"-Kette. Der Wandlungsstart erfolgt wahlweise per Software oder auf ein externes Triggersignal (analog/digital). Es werden keine weiteren Funktionen für Konfiguration und Start der Erfassung benötigt.

☞ Hinweis!

Die ME-4600 verwendet hier stets den Eingangsspannungsbereich ±10V und arbeitet single ended.

Diese Funktion wird immer im „Blocking"-Mode ausgeführt. D. h. der Programmfluß wird unterbrochen bis die Funktion zurückkehrt. In der Praxis ist dies nur relevant falls die Messung durch ein externes Triggersignal ausgelöst werden soll.

Beachten Sie, daß ein möglicherweise mit der Funktion …*MultiSigSetGain* eingestellter Verstärkungsfaktor der betreffenden Kanalgruppe überschrieben wird!

● Definitionen

VC: me4000MultiSigAISingle(unsigned int uiBoardNumber, unsigned int uiAIChannelNumber, unsigned int uiMuxChannelNumber, int iGain, int iTriggerMode, int iExtTriggerEdge, unsigned long ulTimeOutSeconds, short* psDigitalValue);

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)

`<AIChannelNumber>`

A/D-Kanal-Nummer für den die „Mux"-Kette konfiguriert wurde. Die Kanal-Nummer muß mit der Lötbrücke „A" auf der Master-Karte korrespondieren; mögliche Werte: 0…31 (ME-4650/4660: 0…15)

<MuxChannelNumber>

Fortlaufende Nummer des Mux-Kanals in der „Mux"-Kette, dessen Wert erfasst werden soll; mögliche Werte: 0…255

<Gain>

Einstellung des Verstärkungsfaktors für die Kanalgruppe, welcher der gewünschte Mux-Kanal angehört. Der hier eingestellte Verstärkungsfaktor gilt ab sofort für alle Kanäle der betreffenden Kanalgruppe (ein mit der Funktion …*MultiSigSetGain* eingestellter Verstärkungsfaktor wird überschrieben):

- ME4000_MULTISIG_GAIN_1
  Verstärkungsfaktor 1 (Standard)
- ME4000_MULTISIG_GAIN_10
  Verstärkungsfaktor 10
- ME4000_MULTISIG_GAIN_100
  Verstärkungsfaktor 100

<TriggerMode>

Trigger-Ereignis für den A/D-Teil:

- ME4000_AI_TRIGGER_SOFTWARE
  Wandlungsstart unmittelbar nach Aufruf dieser Funktion.
- ME4000_AI_TRIGGER_EXT_DIGITAL
  Bereit zur Wandlung nach Aufruf dieser Funktion.
  Wandlungsstart durch digitales Trigger-Signal.
- ME4000_AI_TRIGGER_EXT_ANALOG (nicht ME-4650/4660)
  Bereit zur Wandlung nach Aufruf dieser Funktion.
  Wandlungsstart durch analoges Trigger-Signal.

<ExtTriggerEdge>

Auswahl der Triggerflanke für den A/D-Teil.

- ME4000_AI_TRIGGER_EXT_EDGE_RISING
  Eine positive Triggerflanke wird ausgewertet.
- ME4000_AI_TRIGGER_EXT_EDGE_FALLING
  Eine negative Triggerflanke wird ausgewertet.
- ME4000_AI_TRIGGER_EXT_EDGE_BOTH
  Sowohl positive als auch negative Triggerflanken werden ausgewertet.
- ME4000_VALUE_NOT_USED
  Kein ext. Trigger verwendet. Siehe Parameter `<TriggerMode>`.

<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.

<DigitalValue>

Zeiger auf einen 16 Bit-Wert (vorzeichenbehaftet). Verwenden Sie die Funktion …*MultiSigAIDigitToSize* zur einfachen Umrechnung in die entsprechende physikalische Einheit.

‹ 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.

me4000MultiSigAIStart

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Mit Aufruf dieser Funktion wird die Karte je nach Konfiguration von Hardware und Software für die Erfassung „scharfgemacht“. In der Betriebsart „Software-Start“ wird die Erfassung unmittelbar nach Aufruf dieser Funktion gestartet. Bei Verwendung des externen Triggers hängt der Start vom jeweiligen Trigger-Ereignis ab (siehe Kap. 3.3.4 auf Seite 19). Sofern nicht mit der Funktion …*MultiSigAIReset* beendet wurde, kann nach Ende der Erfassung durch Aufruf von …*MultiSigAIStart* jederzeit von vorne gestartet werden, ohne daß vorher der A/D-Teil neu konfiguriert werden muß.

Falls Sie eine Erfassung in der Betriebsart „MultiSig-AIContinuous“ beenden wollen oder eine Erfassung in der Betriebsart „MultiSig-AIScan“ vorzeitig beenden wollen verwenden Sie die Funktionen …*MultiSigAIStop* oder …*MultiSigAIReset*.

☞ Hinweis!

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: me4000MultiSigAIStart (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.

me4000MultiSigAIStop

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Die Erfassung wird sofort gestoppt. Die Messwerte, die seit dem letzten „Abholen“ erfaßt wurden gehen verloren. Die Konfiguration der Karte bleibt erhalten („Mux“-Kanalliste, Timer, etc.). Ein erneutes Starten mit der Funktion …*MultiSigAIStart* ist jederzeit möglich.

● Definitionen

VC: me4000MultiSigAIStop(unsigned int uiBoardNumber, int iReserved);

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)

<Reserved>

Dieser Parameter ist reserviert. Bitte übergeben Sie „0“.

❮ 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.

5.3.8.2 „Demux“-Funktionen

Die folgenden Funktionen dienen der analogen Ausgabe über das ME-MultiSig-System in Verbindung mit einer Karte der ME-4600 Serie (siehe auch Kap. „Demux“-Betrieb auf Seite 85 und Handbuch „ME-MultiSig“-System).

me4000MultiSigAOAppendNewValues

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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

Siehe auch Kap. 4.2 auf Seite 85 und Programmierbeispiele im ME-Software-Developer-Kit (ME-SDK).

☞ Hinweis!

Sie müssen nicht den gleichen, wie in *…MultiSigAOContinuous* verwendeten Datenpuffer verwenden. Es wird stets D/A-Kanal 0 der ME-4600 verwendet.

● Definitionen

VC: me4000MultiSigAOAppendNewValues(unsigned int uiBoardNumber, 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)

`<Buffer>`

Zeiger auf den gewünschten Datenpuffer, der mit den nachzuladenden Werten gefüllt sein muss.

<NumberOfValuesToAppend>

Anzahl der Werte im Datenpuffer. Bei Übergabe von „0“ können Sie im Parameter <NumberOfValuesAppended> die Anzahl der Werte abfragen, die bei Rückkehr der Funktion 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.

me4000MultiSigAOClose

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Diese Funktion schließt die mit *…MultiSigAOOpen* freigegebene Funktionalität ab. Reservierte Hardware-Ressourcen werden wieder freigegeben. Beachten Sie auch Kap. 4.5 „ME-MultiSig-Steuerung“ auf Seite 79ff.

● Definitionen

VC: me4000MultiSigAOClose (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.

me4000MultiSigAOConfig

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Diese Funktion konfiguriert die Hardware für eine timergesteuerte Ausgabe in den Betriebsarten „MultiSig-AOContinuous“ und „MultiSigAOWraparound“. Gestartet wird die Ausgabe stets mit der Funktion *…MultiSigAOStart* entweder sofort (Software-Start) oder 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 für TTL-Versionen 500 kS/s und für optoisolierte Versionen 172 kS/s („i“-Versionen).

☞ Hinweis:

Die Funktion *…FrequencyToTicks* bzw. *…TimeToTicks* bietet Ihnen eine bequeme Umrechnungsmöglichkeit von Frequenz bzw. Periodendauer in Ticks zur Übergabe an den Timer (siehe Seite 105ff).

Für die auszugebenden Analog-Werte wird stets D/A-FIFO 0 verwendet (siehe auch Kap. 4.5 „ME-MultiSig-Steuerung“ auf Seite 79ff).

● Definitionen

VC: me4000MultiSigAOConfig(unsigned int uiBoardNumber, unsigned char *pucDemuxChanList, unsigned int uiDemuxChanListCount, 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)

`<DemuxChanList>`

Zeiger auf den Anfang der Demux-Kanalliste zur Steuerung der Demultiplexer. Allokieren Sie vorab ein Wertefeld definierter Größe, in das Sie die Demux-Kanalnummern (0…31) in der gewünschten Reihenfolge eintragen.

`<DemuxChanListCount>`

Anzahl der Demux-Kanallisten-Einträge.

`<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) für TTL-Versionen bzw. 192 (C0Hex) für Opto-Versionen und 2^{32}-1 (FFFFFFFFHex) Ticks.

`<TriggerMode>`

Trigger-Ereignis zum Start der analogen Ausgabe:

- ME4000_AO_TRIGGER_SOFTWARE
  Start per Software nach Aufruf der Funktion *…MultiSigAOStart*.
- ME4000_AO_TRIGGER_EXT_DIGITAL
  Bereit zur Ausgabe nach Aufruf der Funktion *…MultiSigAOStart*. Ausgabe wird durch externes Trigger-Signal an Pin 47 (DA_TRIG_0) gestartet.

<ExtTriggerEdge>

Auswahl der Triggerflanke für den Trigger-Eingang DA_TRIG_0.

- 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.

me4000MultiSigAOContinuous

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

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

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

☞ Hinweis!

Beachten Sie, daß die Reihenfolge der auszugebenden Werte innerhalb des Datenpuffers `<Buffer>` mit der Reihenfolge der Kanäle in der Demux-Kanalliste (siehe *…MultiSigAOConfig*) korrespondieren muß.

● Definitionen

Typdefinition für ME4000_P_AO_CALLBACK_PROC:

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

VC: me4000MultiSigAOContinuous(unsigned int uiBoardNumber, 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)

`<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.

<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.

me4000MultiSigAOGetStatus

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Funktion dient der Abfrage, ob eine analoge Ausgabe in den Betriebsarten „MultiSig-AOContinuous“ und „MultiSig-AOWraparound“ noch läuft oder der Ringpuffer bereits „leer gelaufen“ ist. Dies ist dann der Fall wenn Sie den Puffer entweder 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: me4000MultiSigAOGetStatus(unsigned int uiBoardNumber, 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)

<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 dann Ende der Ausgabe zurück (FIFO leer). In diesem Fall enthält der Parameter <Status> stets den Wert ME4000_AO_STATUS_IDLE.

<Status>

Aktueller Betriebszustand:

- ME4000_AO_STATUS_IDLE
  Die Ausgabe ist beendet, d. h. der Ringpuffer 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.

me4000MultiSigAOOpen

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Diese Funktion bereitet den „Demux“-Betrieb vor. Entsprechende Hardware-Ressourcen werden reserviert:

- Digital-Port A
- D/A-Kanal 0 wird für Ausgabe der Analog-Werte genutzt.
- D/A-Kanal 3 wird für analoge Ausgabe gesperrt.

Beachten Sie auch Kap. 4.5 „ME-MultiSig-Steuerung“ auf Seite 79ff.

● Definitionen

VC: me4000MultiSigAOOpen (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.

me4000MultiSigAOReset

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Funktion zum Beenden der Betriebsarten „MultiSig-AOContinuous“ und „MultiSig-AOWraparound“. Die Ausgabe wird sofort und vollständig beendet. Danach wird der Ringpuffer gelöscht und auf Demux-Kanal 0 wird 0V ausgegeben.

● Definitionen

VC: me4000MultiSigAOReset(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.

me4000MultiSigAOSingle

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

Diese Funktion dient der Ausgabe einer Spannung auf dem spezifizierten Demux-Kanal.

Es ist keine weitere Konfiguration mit anderen Funktionen nötig.

● Definitionen

VC: me4000MultiSigAOSingle(unsigned int uiBoardNumber, unsigned int uiDemuxChannelNumber, 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)

`<DemuxChannelNumber>`

Nummer des Demux-Kanals, auf den ausgegeben werden soll; mögliche Werte: 0…31

`<TriggerMode>`

Trigger-Ereignis zum Start der analogen Ausgabe:

- ME4000_AO_TRIGGER_SOFTWARE
  Einzelwert wird unmittelbar nach Aufruf dieser Funktion ausgegeben (Software-Start).
- ME4000_AO_TRIGGER_EXT_DIGITAL
  Bereit zur Ausgabe nach Aufruf dieser Funktion. Ausgabe wird durch externes Trigger-Signal an Pin 47 (DA_TRIG_0) gestartet.

<ExtTriggerEdge>

Auswahl der Triggerflanke am Trigger-Eingang DA_TRIG_0.

- 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; der Wertebereich liegt zwischen: -32768 (-10 V) und +32767 (+10 V - LSB).

❮ 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.

me4000MultiSigAOStart

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Funktion zum Starten der Betriebsarten „MultiSig-AOContinuous“ und „MultiSig-AOWraparound“. Falls Sie zuvor die Option „Externer Trigger“ gewählt haben, wird die Ausgabe durch eine entsprechende Flanke am ext. Triggereingang DA_TRIG_0 (Pin 47) gestartet.

☞ Hinweis!

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: me4000MultiSigAOStart (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.

me4000MultiSigAOStop

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Funktion zum Beenden der Betriebsarten „MultiSig-AOContinuous“ und „MultiSig-AOWraparound“. In der Betriebsart „MultiSig-AOContinuous“ wird die Ausgabe sofort und vollständig beendet. Die Demultiplexer-Steuerung bleibt nach Beendigung der Ausgabe auf dem gerade angesteuerten Demux-Kanal stehen und gibt 0V aus. In der Betriebsart „MultiSig-AOWraparound“ können Sie mit dem Parameter `<StopMode>` selbst bestimmen, ob die Ausgabe sofort und vollständig beendet werden soll (siehe „MultiSig-AOContinuous“) oder mit dem zuletzt in der Demux-Kanalliste eingetragenen Demux-Kanal gestoppt werden soll. Sofern die Betriebsart nicht gewechselt wurde kann die Ausgabe mit der Funktion *…MultiSigAOStart* jederzeit von vorne gestartet werden.

● Definitionen

VC: me4000MultiSigAOStop(unsigned int uiBoardNumber, 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)

`<StopMode>`

- ME4000_AO_STOP_MODE_LAST_VALUE
  Ausgabe wird mit letztem Wert der Demux-Kanalliste definiert beendet (nur sinnvoll in der Betriebsart „MultiSig-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.

me4000MultiSigAOVoltToDigit

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	✔	✔	✔

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. 12 auf Seite 22):

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

● Definitionen

VC: me4000MultiSigAOVoltToDigit(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.

me4000MultiSigAOWraparound

✎ Beschreibung

ME-4650	ME-4660	ME-4670	ME-4680
–	–	–	✔

Mit dieser Funktion wird die Karte für die Betriebsart „MultiSig-AO-Wraparound“ vorbereitet. Sie können in diesem Modus beliebige periodische Signale ausgeben. Vor Beginn der Ausgabe muß D/A-FIFO 0 einmalig beladen werden. Erzeugen sie einen Datenpuffer definierter Größe mit den auszugebenden Werten.

Der Timer gibt ein festes Zeitraster (Sample-Rate) für die Ausgabe vor (siehe *…MultiSigAOConfig*).

Gestartet wird die Ausgabe stets mit der Funktion *…MultiSigAOStart* entweder sofort (Software-Start) oder durch ein externes Triggersignal (siehe *…MultiSigAOConfig*). Mit der Funktion *…MultiSigAO-Stop* können Sie die Ausgabe wahlweise sofort beenden oder definiert „anhalten“, d. h. die Ausgabe wird mit dem letzten Wert der Demux-Kanalliste und somit einem bekannten Demux-Kanal gestoppt. Sofern zwischenzeitlich die Betriebsart für diesen Kanal nicht gewechselt wurde, kann die Ausgabe mit der Funktion *…MultiSigAOStart* jederzeit von vorne gestartet werden. Mit der Funktion *…MultiSigAOReset* wird im Vergleich zu *…MultiSigAOStop* auch das D/A-FIFO gelöscht und damit die Ausgabe vollständig beendet.

☞ Hinweis!

Beachten Sie, daß die Reihenfolge der auszugebenden Werte im Datenpuffer `<Buffer>` mit der Reihenfolge der Kanäle in der Demux-Kanalliste (siehe *…MultiSigAOConfig*) korrespondieren muß.

Sofern die Größe des Datenpuffers 4096 Werte nicht übersteigt und die Ausgabe „unendlich“ erfolgt, läuft die Ausgabe auf Firmware-Ebene, d. h. der Host-Rechner wird nicht belastet!

● Definitionen

Typdefinition für ME4000_P_AO_TERMINATE_PROC:

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

VC: me4000MultiSigAOWraparound(unsigned int uiBoardNumber, 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)

`<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_ASYNCHRONOUS:
  Die Ausgabe erfolgt im Hintergrund (asynchron). Der Programmfluß wird nicht unterbrochen.

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

„Terminate“-Funktion, die am Ende der Ausgabe aufgerufen wird. Der Funktion wird ein Zeiger auf den Anfang des Datenpuffers sowie die Gesamtzahl der Werte übergeben. 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.

‹ 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)

PCI-Interface

Standard-PCI-Karte (32 Bit, 33MHz, 5V);
PCI Local Bus Spezifikation Version 2.1 konform;
Automatische Ressourcen-Zuweisung (Plug&Play)

Spannungs-Eingänge

Anzahl A/D-Kanäle gesamt	ME-4650/4660: 16 single ended ME-4670/4680: 32 single ended/ 16 differentiell
„Sample&Hold"-Kanäle	optional: 8 single-ended simultanabtastend
A/D-Wandler	500kHz 16 Bit A/D-Wandler
Eingangsbereiche	unipolar: 0V…+2,5V - 1LSB (38µV); 0V…+10V - 1LSB (152µV); bipolar: -2,5V…+2,5V - 1LSB (76µV); -10V…+10V - 1LSB (305µV)
Fehler bei Vollausschlag (Full Scale Error)	unipolar: 0V+10LSB, +FS-10LSB bipolar: -FS+10LSB, +FS-10LSB
Eingänge geschützt bis	±15 V
Eingangsimpedanz	R_{IN} = typ. 600 MΩ; C_{IN} = typ. 3 pF Kanäle mit Sample&Hold-Option: R_{IN} = typ. 1 MΩ; C_{IN} = typ. 5 pF
Summenabtastrate	500 kS/s
„Sample&Hold"-Abtastrate („s"-Versionen) – es gilt:	Anzahl der Kanallisteneinträge x CHAN-Zeit + Erholzeit
Erholzeit (S&H)	1,5 µs
Gesamtgenauigkeit	typ. ±4 LSB, max. ±10 LSB bei Vollausschlag im Eingangsbereich ±10 V
Optoisolierung („i"-Versionen)	A/D- und D/A-Teil mit gemeinsamer Masse (A_GND) von der PC-Masse und vom Rest der Karte entkoppelt
A/D-FIFO	2 k Werte-FIFO
Kanalliste	max. 1024 Einträge (Kanal-Nummer, Verstärkungsfaktor, uni-/bipolar, s. e./diff.)
Kleinste Zeit-Einheit zur Einstellung von CHAN- und SCAN-Timer:	1 Tick ≙ 30,30 ns ≙ 33 MHz

CHAN-Timer	32 Bit-Zähler bestimmt die Zeit zwischen zwei aufeinander folgenden Kanallisten-Einträgen: von 2µs bis ~130s in Schritten von 30,30ns programmierbar.
SCAN-Timer	36 Bit-Zähler bestimmt die Zeit zwischen dem Beginn von zwei aufeinander folgenden Kanallistenabarbeitungen. Sinnvoller SCAN-Zeit Bereich (min. 2 Kanäle): von 4 µs bis ~30 Minuten in Schritten von 30,30ns programmierbar.
Betriebsarten	„AISingle“, „AIContinuous“, „AIScan“ optional: „AISimultaneous“
Erfassungsmodi	„Software-Start“, „Extern-Standard“, „Extern-Einzelwert“, „Extern-Kanalliste“
Ext. Triggermodi	ext. Analog-Trigger; ext. Digital-Trigger
Ext. Triggerflanken	steigend, fallend, beide

Ext. Trigger ohne Optoisolierung

Massebezug		PC-Masse (PC_GND)
Eingangspegel	U_{IL}:	max. 0,9V bei Vcc=4,5V
	U_{IH}:	min. 3,15V bei Vcc=4,5V
Verzögerungszeit		max. 30ns

Ext. Trigger mit Optoisolierung

Massebezug	Digital-I/O-Masse (DIO_GND)
Eingangsstrom I_F	$7{,}5mA \leq I_F \leq 10mA$
Spannungspegel	typ. 5V
Verzögerungszeit	typ. 80ns

Spannungs-Ausgänge (ME-4660, ME-4670, ME-4680)

Anzahl D/A-Kanäle	ME-4660: 2; ME-4670/4680: 4
D/A-Wandler	1 serieller Wandler pro Kanal
Auflösung	16 Bit
Ausgangsbereich	±10V
Ausgangsstrom	max. 5mA pro Kanal
Einschwingzeit (DAC)	max. 2µs bei Vollausschlag (-10V → +10V)
Gesamtgenauigkeit:	max. ±10mV
Optoisolierung („i“-Versionen)	A/D- und D/A-Teil mit gemeinsamer Masse (A_GND) von der PC-Masse und vom Rest der Karte entkoppelt
Betriebsarten	„AOSingle“, „AOSimultaneous“
Triggermodi	Software-Start, ext. Digital-Trigger
Ext. Triggerflanken	steigend, fallend, beide

Timergesteuerte Ausgabe (nur ME-4680)

Kanäle	0…3 (voneinander unabhängig)
D/A-FIFOs	4k Werte pro Kanal
Sample-Rate	max. 500kS/s
D/A-Timer	von 2µs bis 130s in Schritten von 30,$\overline{30}$ns programmierbar
Betriebsarten	„AOContinuous“, „AOWraparound“
Triggermodi	Software-Start, ext. Digital-Trigger, Synchron-Start (Software/extern)
Ext Triggerflanken	steigend, fallend, beide

Ext. Trigger ohne Optoisolierung

Massebezug		PC-Masse (PC_GND)
Eingangspegel	U_{IL}:	max. 0,9V bei Vcc = 4,5V
	U_{IH}:	min. 3,15V bei Vcc = 4,5V
Verzögerungszeit		max. 30ns

Ext. Trigger mit Optoisolierung

Massebezug	Digital-I/O-Masse (DIO_GND)
Eingangsstrom I_F	$7{,}5mA \le I_F \le 10mA$
Spannungspegel	typ. 5V
Verzögerungszeit	typ. 80ns

Digital-I/Os

Ports	4 x 8 Bit

…ohne Optoisolierung

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 Vcc = 5V
	U_{IH}:	min. 2V bei Vcc = 5V
	Eingangsstrom:	±1µA
Sample-Rate		max. 500kS/s (2µs)

…mit Optoisolierung („i“-Versionen):

Masse-Bezug		Digital-I/O-Masse (DIO_GND) von der PC-Masse und vom Rest der Karte entkoppelt
Port-Typ	Port A:	Ausgangsport
	Port B:	Eingangsport
	Port C, D:	bidirektionale TTL-Ports (es gelten die Pegel „ohne Optoisolierung“)

Ausgangs-Pegel Port A, B:

U_{max}:	42V (von ext. Spannungsquelle abhängig)
I_{Out}:	max. 30mA

Eingangs-Pegel Port A,B:

Eingangsstrom I_F:	$7{,}5mA \leq I_F \leq 10mA$
U_{IL}:	max. 0,8V
U_{IH}:	min. 4,5V, max. 5V (optional höhere Eingangsspannungen möglich - bitte wenden Sie sich an unsere Support-Abteilung)
Sample-Rate	max. 172kS/s (5,8µs)

Bitmuster-Ausgabe

Ports	flexibles Portmapping auf alle digitalen Ausgangs-Ports (A, B, C, D)
Betriebsarten	„BitPattern-Continuous“, „BitPattern-Wrap-around“
Bitmuster-FIFO	4k Werte (identisch mit D/A-FIFO 3)
Sample-Rate	
TTL-Port:	max. 500kS/s (2µs)
Optoisolierter Port:	max. 172kS/s (5,8µs)
Bitmuster-Timer	von 2µs bis 130s in Schritten von $30{,}\overline{30}$ns programmierbar
Externer Trigger	Digitaler Triggereingang DA_TRIG_3
Eingangspegel:	siehe ext. Trigger D/A-Teil
Verzögerungszeit:	ohne Optoisolierung: max. 30ns mit Optoisolierung: typ. 80ns
Triggermodi	Software-Start, ext. Digital-Trigger
Ext Triggerflanken	steigend, fallend, beide

Zähler

Anzahl	3 x 16 Bit (1 x 82C54)
Zählertakt	extern bis max. 10 MHz

…ohne Optoisolierung

Massebezug	PC-Masse (PC_GND)

Pegel für Zählerausgang (OUT_*x*)

U_{OL}:	max. +0,45V (I_{OL}=+7,8mA)
U_{OH}:	min. +2,4V (I_{OH}=-6mA)

Pegel für Zählereingänge (CLK_*x*, GATE_*x*)

U_{IL}:	-0,5V…+0,8V (I_{ILmax}=±10µA)
U_{IH}:	+2,2V…+6V (I_{IHmax}=±10µA)

…mit Optoisolierung („i"-Versionen):

Masse-Bezug:	Zähler-Masse (CNT_GND) von der PC-Masse und vom Rest der Karte entkoppelt
Ext. Versorgung für Optokoppler (CNT_VCC_IN)	+5V/30mA
Pegel für Zählerausgang (OUT_*x*):	
U_{max}:	42V
I_{Out}:	max. 30mA
Pegel für Zählereingänge (CLK_*x*, GATE_*x*):	
I_F:	min. 7,5mA
U_{IL}:	max. 0,8V
U_{IH}:	min. 4,5V, max. 5V (optional höhere Eingangsspannungen möglich - bitte wenden Sie sich an unsere Support-Abteilung)

Optional: Versorgung der Optokoppler mit VCC des Analog-Teils (A_VCC). Beachten sie, daß die galvanische Trennung zwischen Analog- und Zähler-Teil dadurch aufgehoben wird (CNT_GND = A_GND), siehe Abb. 19.

Externer Interrupt

Ext. Interrupt-Eingang	wird direkt ans System weitergeleitet (falls durch Treiber freigegeben)
Masse-Bezug	„TTL": PC-Masse (PC_GND); „Opto": Digital-I/O-Masse (DIO_GND)
Eingangspegel	siehe Digital-I/Os

Allgemeine Daten

DC/DC-Wandler A/D-Teil	±5V und ±15 V (2 x 3W)
Stromverbrauch (ohne ext. Last):	
„Ohne Optoisolierung"	typ. 2,8A
„Mit Optoisolierung"	typ. 2,8A
Belastbarkeit VCC_OUT	max. 200mA
Kartenabmessungen	175mm x 107mm (ohne Slotblech und Stecker)
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:

AD_*x*	Analoge Eingangskanäle
AD_TRIG_D	Digitaler Triggereingang für A/D-Teil
AD_TRIG_A+	Analoger Triggereingang für A/D-Teil (positiver Komparator-Eingang)
AD_TRIG_A-	Analoger Triggereingang für A/D-Teil (negativer Komparator-Eingang)
DA_*x*	Analoge Ausgangskanäle
DA_TRIG_*x*	Digitaler Triggereingang für jeden D/A-Kanal getrennt.
DIO_A*x*	Digitaler Ein/Ausgang Port A
DIO_B*x*	Digitaler Ein/Ausgang Port B
DIO_C*x*	Digitaler Ein/Ausgang Port C
DIO_D*x*	Digitaler Ein/Ausgang Port D
EXT_IRQ	Externer Interrupt-Eingang
CLK_*x*	Takt-Eingang für Zähler
GATE_*x*	Gate-Eingang für Zähler
OUT_*x*	Zähler-Ausgang
PC_GND	***Nicht-optoisolierte*** Modelle: Gemeinsame Masse aller Funktionsgruppen (= PC-Masse).
VCC_OUT	***Nicht-optoisolierte*** Modelle: V_{CC}-Ausgang (+5V vom PC) bis max. 200mA belastbar
n.c.	Pin ohne Verbindung

Gilt für optoisolierte Modelle:

A_GND	Masse für A/D- und D/A-Teil
DIO_GND	Masse für Digital-I/O-Teil
CNT_GND	Masse für Zähler
CNT_VCC_IN	Auslieferungszustand: Eingang für externe Versorgungsspannung (+5V±10%) der Zähler-Optokoppler.
A_VCC	Optional (siehe Abb. 19 auf Seite 27): Versorgung der Zähler-Optokoppler über den Analog-Teil (A_VCC). **Keine externe Beschaltung an Pin 1!**

B1 78pol. Sub-D-Buchse (ST1)

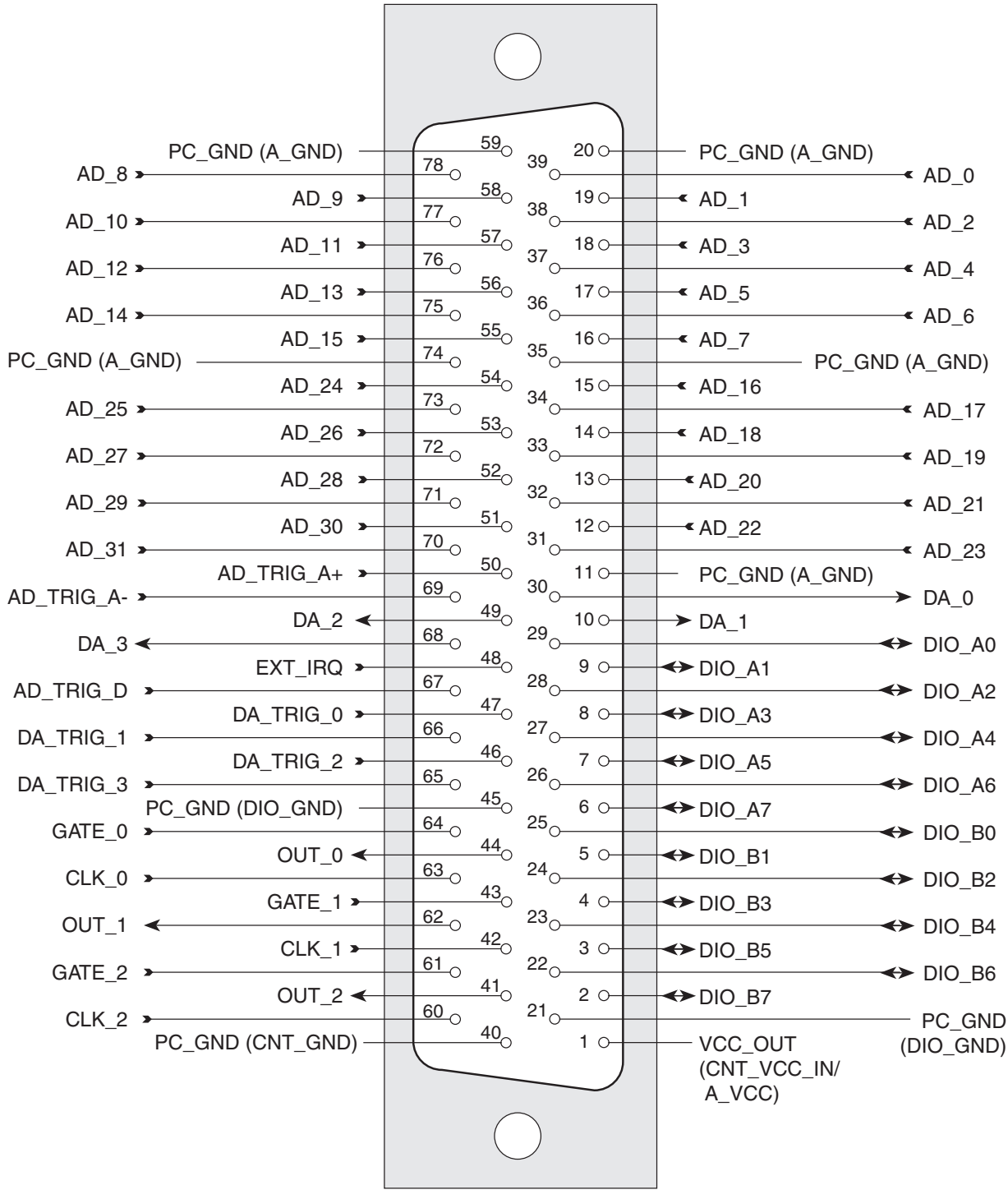


Abb. 58: Belegung der 78poligen Sub-D-Buchse ME-4600 (ST1)

Je nach Modell, sind nicht alle Pins der 78poligen Sub-D-Buchse belegt. Die Bezeichnungen in Klammern gelten für die optoisolierten Varianten („i"-Versionen).

B2 Zusatzstecker (ST2)

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

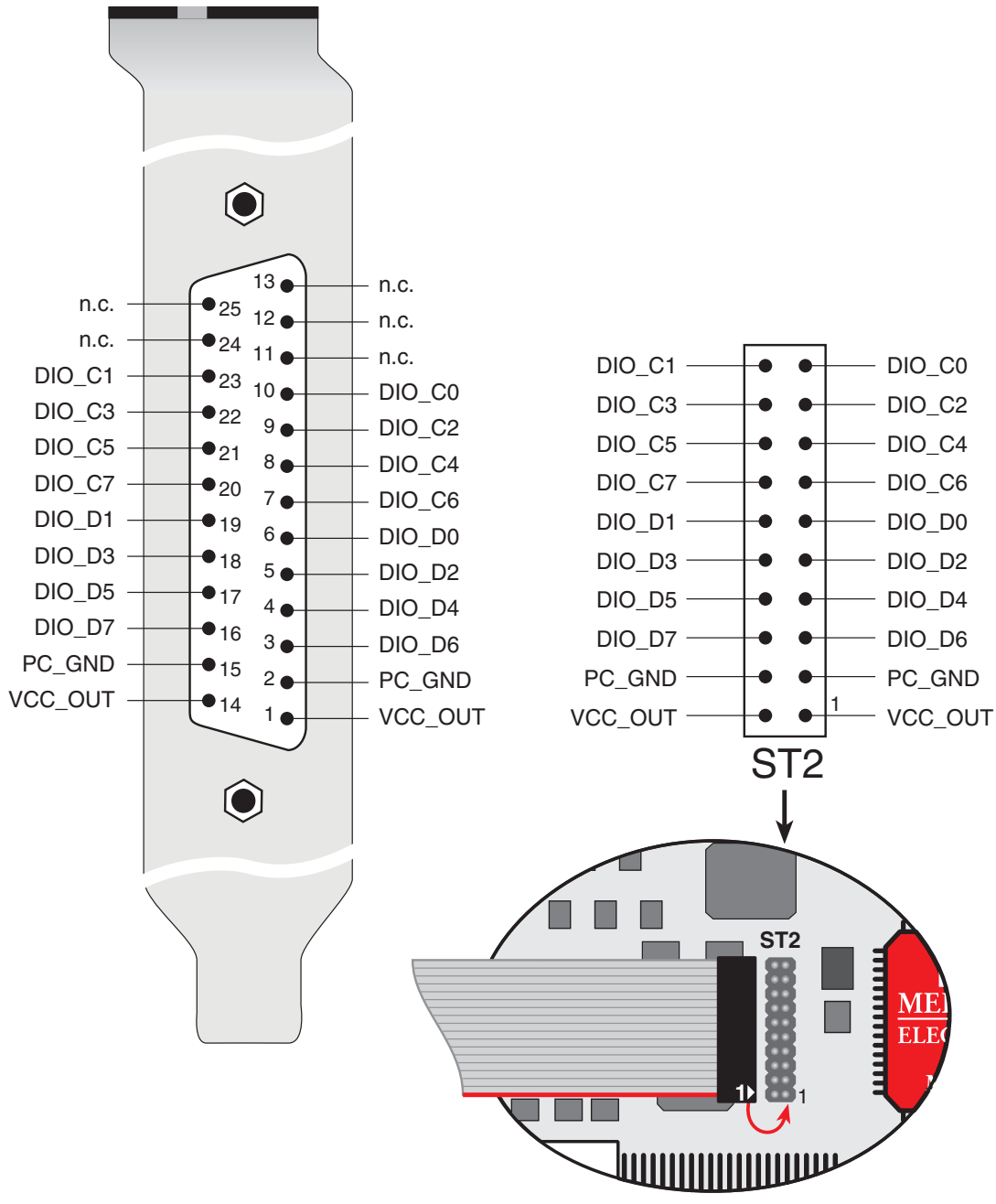


Abb. 59: Zusatzstecker ST2 der ME-4600-Serie (Draufsicht)

Beachten Sie beim Anschließen des Slotblechs, 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.

ME-AK-D78/4000M-F

Spezial-Anschluß-Kabel (1:1) für ME-4600-Serie (78pol. Sub-D Stecker auf 78pol. Sub-D-Buchse), Länge: 1 m

ME-AB-D78M

78poliger Sub-D Anschluß-Block (Stecker)

ME-AA4-3(i)

Anschlußadapter zur Anschaltung einer ME-2x00/3000-Applikation an eine Karte der ME-4600 Serie („i"-Version für optoisolierte Modelle incl. Optoisolierung für Digital-Port C + D).

ME-MultiSig-System

Umfangreiches Multiplex- und Signalkonditionierungssystem:

- Analoges Multiplexen bis 8192 Kanäle (timergesteuert bis 256 Kanäle)
- Analoges Demultiplexen bis 32 Kanäle
- Signalkonditionierung (Spannung, Strom, RTDs)

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 Eingabe	
ME4000_AI_ACQ_MODE_SOFTWARE	0x00020101
ME4000_AI_ACQ_MODE_EXT	0x00020102
ME4000_AI_ACQ_MODE_EXT_SINGLE_VALUE	0x00020103
ME4000_AI_ACQ_MODE_EXT_SINGLE_CHANLIST	0x00020104
ME4000_AI_RANGE_BIPOLAR_10	0x00020201
ME4000_AI_RANGE_BIPOLAR_2_5	0x00020202
ME4000_AI_RANGE_UNIPOLAR_10	0x00020203
ME4000_AI_RANGE_UNIPOLAR_2_5	0x00020204
ME4000_AI_INPUT_SINGLE_ENDED	0x00020301
ME4000_AI_INPUT_DIFFERENTIAL	0x00020302
ME4000_AI_TRIGGER_SOFTWARE	0x00020401
ME4000_AI_TRIGGER_EXT_DIGITAL	0x00020402
ME4000_AI_TRIGGER_EXT_ANALOG	0x00020403
ME4000_AI_TRIGGER_EXT_EDGE_RISING	0x00020501
ME4000_AI_TRIGGER_EXT_EDGE_FALLING	0x00020502
ME4000_AI_TRIGGER_EXT_EDGE_BOTH	0x00020503
ME4000_AI_SIMULTANEOUS_DISABLE	0x00020601
ME4000_AI_SIMULTANEOUS_ENABLE	0x00020602
ME4000_AI_SCAN_BLOCKING	0x00020701
ME4000_AI_SCAN_ASYNCHRONOUS	0x00020702

Tabelle 12: Konstanten-Definition

Konstante	Wert
ME4000_AI_GET_NEW_VALUES_BLOCKING	0x00020801
ME4000_AI_GET_NEW_VALUES_NON_BLOCKING	0x00020802
ME4000_AI_WAIT_IDLE	0x00020901
ME4000_AI_WAIT_NONE	0x00020902
ME4000_AI_STATUS_IDLE	0x00020A01
ME4000_AI_STATUS_BUSY	0x00020A02

Analoge Ausgabe (ME-4660/4670/4680)

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
ME4000_AO_SHAPE_POS_RAMP	0x00030805
ME4000_AO_SHAPE_NEG_RAMP	0x00030806
ME4000_AO_TRIGGER_EXT_DISABLE	0x00030901
ME4000_AO_TRIGGER_EXT_ENABLE	0x00030902

Digitale Standard Ein-/Ausgabe

ME4000_DIO_PORT_A	0
ME4000_DIO_PORT_B	1
ME4000_DIO_PORT_C	2
ME4000_DIO_PORT_D	3

Tabelle 12: Konstanten-Definition

Konstante	Wert
ME4000_DIO_PORT_INPUT	0x00040201
ME4000_DIO_PORT_OUTPUT	0x00040202

Bitmuster-Ausgabe (ME-4680)

ME4000_DIOBP_PORT_A	0
ME4000_DIOBP_PORT_B	1
ME4000_DIOBP_PORT_C	2
ME4000_DIOBP_PORT_D	3
ME4000_DIOBP_OUTPUT_MODE_BYTE_LOW	0x00060101
ME4000_DIOBP_OUTPUT_MODE_BYTE_HIGH	0x00060102
ME4000_DIOBP_TRIGGER_SOFTWARE	0x00060201
ME4000_DIOBP_TRIGGER_EXT_DIGITAL	0x00060202
ME4000_DIOBP_TRIGGER_EXT_EDGE_RISING	0x00060301
ME4000_DIOBP_TRIGGER_EXT_EDGE_FALLING	0x00060302
ME4000_DIOBP_TRIGGER_EXT_EDGE_BOTH	0x00060303
ME4000_DIOBP_WRAPAROUND_BLOCKING	0x00060401
ME4000_DIOBP_WRAPAROUND_ASYNCHRONOUS	0x00060402
ME4000_DIOBP_WRAPAROUND_INFINITE	0x00
ME4000_DIOBP_APPEND_NEW_VALUES_BLOCKING	0x00060501
ME4000_DIOBP_APPEND_NEW_VALUES_NON_BLOCKING	0x00060502
ME4000_DIOBP_WAIT_IDLE	0x00060601
ME4000_DIOBP_WAIT_NONE	0x00060602
ME4000_DIOBP_STATUS_IDLE	0x00060701
ME4000_DIOBP_STATUS_BUSY	0x00060702
ME4000_DIOBP_STOP_MODE_LAST_VALUE	0x00060801
ME4000_DIOBP_STOP_MODE_IMMEDIATE	0x00060802

Zähler (ME-4660/4670/4680)

ME4000_CNT_MODE_0	0x00050101
ME4000_CNT_MODE_1	0x00050102
ME4000_CNT_MODE_2	0x00050103
ME4000_CNT_MODE_3	0x00050104
ME4000_CNT_MODE_4	0x00050105
ME4000_CNT_MODE_5	0x00050106

Fortsetzung nächste Seite

Tabelle 12: Konstanten-Definition

Konstante	Wert
ME-MultiSig-Funktionen	
ME4000_MULTISIG_LED_OFF	0x00070101
ME4000_MULTISIG_LED_ON	0x00070102
ME4000_MULTISIG_GROUP_A	0x00070201
ME4000_MULTISIG_GROUP_B	0x00070202
ME4000_MULTISIG_GAIN_1	0x00070301
ME4000_MULTISIG_GAIN_10	0x00070302
ME4000_MULTISIG_GAIN_100	0x00070303
ME4000_MULTISIG_MODULE_NONE	0x00070401
ME4000_MULTISIG_MODULE_DIFF16_10V	0x00070402
ME4000_MULTISIG_MODULE_DIFF16_20V	0x00070403
ME4000_MULTISIG_MODULE_DIFF16_50V	0x00070404
ME4000_MULTISIG_MODULE_CURRENT16_0_20MA	0x00070405
ME4000_MULTISIG_MODULE_RTD8_PT100	0x00070406
ME4000_MULTISIG_MODULE_RTD8_PT500	0x00070407
ME4000_MULTISIG_MODULE_RTD8_PT1000	0x00070408
ME4000_MULTISIG_MODULE_TE8_TYPE_B	0x00070409
ME4000_MULTISIG_MODULE_TE8_TYPE_E	0x0007040A
ME4000_MULTISIG_MODULE_TE8_TYPE_J	0x0007040B
ME4000_MULTISIG_MODULE_TE8_TYPE_K	0x0007040C
ME4000_MULTISIG_MODULE_TE8_TYPE_N	0x0007040D
ME4000_MULTISIG_MODULE_TE8_TYPE_R	0x0007040E
ME4000_MULTISIG_MODULE_TE8_TYPE_S	0x0007040F
ME4000_MULTISIG_MODULE_TE8_TYPE_T	0x00070410
ME4000_MULTISIG_MODULE_TE8_TEMP_SENSOR	0x00070411
ME4000_MULTISIG_I_MEASURED_DEFAULT	0.0005

Tabelle 12: Konstanten-Definition

F Index

Funktionsreferenz