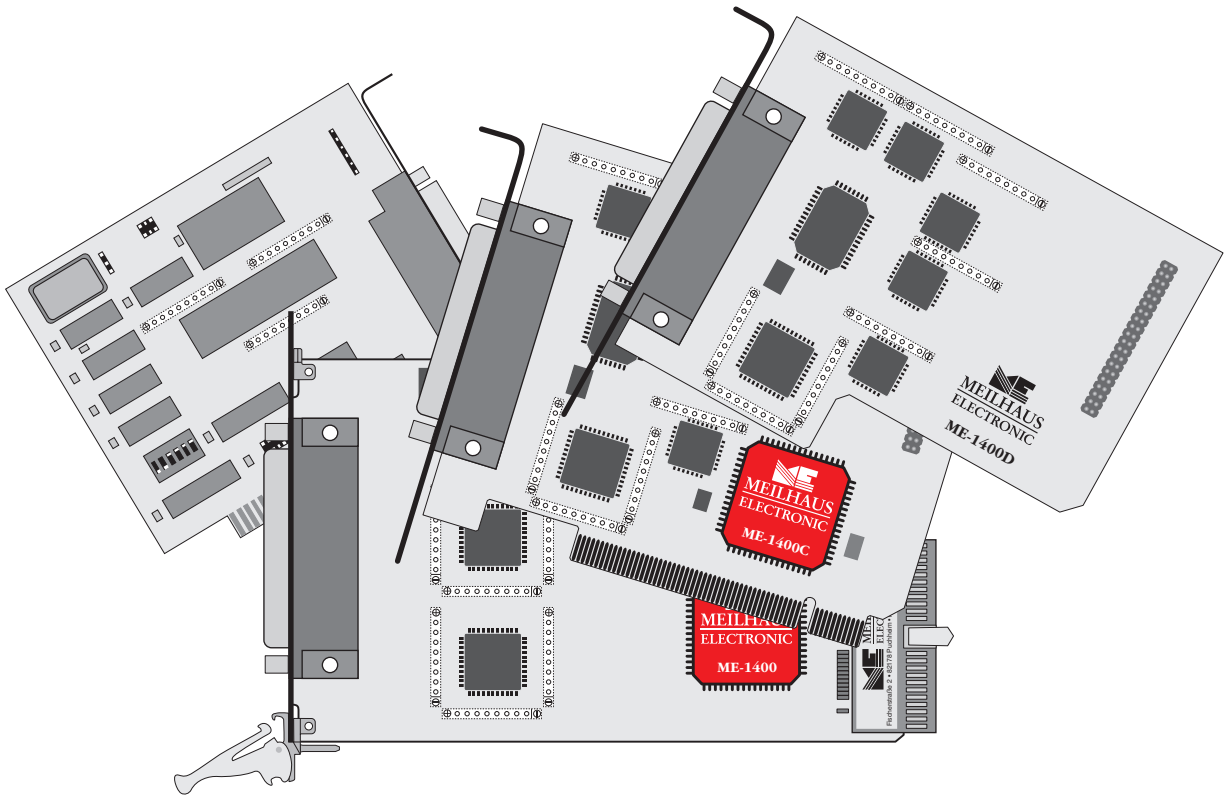
Meilhaus Electronic Handbuch

ME-14, ME-1400 1.91D

ISA, PCI- und CompactPCI-Varianten



TTL Digital I/O- und Zähler-Karten

Impressum

Handbuch ME-14, ME-1400

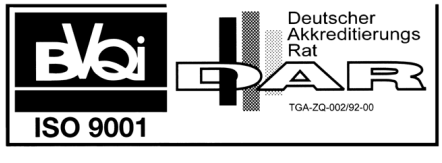
Revision 1.91D
Ausgabedatum: 17. Januar 2005

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.



Inhalt

1 Einführung

Sehr geehrte Kundin, sehr geehrter Kunde,

Mit Ihrem Kauf 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.

Wir empfehlen Ihnen, vor Installation der Karte, dieses Handbuch – insbesondere das Kapitel zur Installation – aufmerksam zu lesen. Beachten Sie bei den ISA-Versionen vor allem den Abschnitt zur Einstellung der Jumper.

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:

- Digital I/O- und Zähler-Karte der ME-14/1400 Serie
- Handbuch im PDF-Format auf CD-ROM
  (optional in gedruckter Form)
- Treiber-Software auf CD-ROM
- ME-14, ME-1400E/EA/EB: 37poliger Sub-D Gegenstecker
- ME-1400/A/B/C/D: 78poliger Sub-D-Gegenstecker
- ME-14B, ME-1400EB: Flachbandkabel von Stiftstecker auf 37polige Sub-D-Buchse an zusätzlichem Slotblech

1.2 Leistungsmerkmale

Modell-Übersicht

Modell	Stecker	TTL-IOs	Zähler
ME-14A ISA	37pol. Sub-D	24	3 x 16 Bit
ME-14B ISA	2 x 37pol. Sub-D	48	6 x 16 Bit
ME-1400 PCI/cPCI	78pol. Sub-D	24	---
ME-1400A PCI/cPCI	78pol. Sub-D	24	3 x 16 Bit
ME-1400B PCI/cPCI	78pol. Sub-D	48	6 x 16 Bit
ME-1400C PCI	78pol. Sub-D	24	15 x 16 Bit
ME-1400D EXP (Erweiterungskarte für ME-1400C)	78pol. Sub-D	24	15 x 16 Bit
ME-1400E PCI (steckerkompatibel zur ME-14)	37pol. Sub-D	24	---
ME-1400EA PCI (steckerkompatibel zur ME-14A)	37pol. Sub-D	24	3 x 16 Bit
ME-1400EB PCI (steckerkompatibel zur ME-14B)	2 x 37pol. Sub-D	48	6 x 16 Bit

Tabelle 1: Modell-Übersicht ME-14/1400 Familie

Die Karten der ME-14/1400 Serie sind frei programmierbare Digital-I/O- und Zähler-Karten. Sie verfügen, je nach Modell über 24 oder 48 TTL-kompatible Ein-/Ausgänge (8255-kompatibel) und bis zu 30 voneinander unabhängig programmierbare 16 Bit Zähler (8254-kompatibel).

Modelle, die mit Zählern austattet sind, besitzen einen vom Systemtakt des PCs unabhängigen 10 MHz Quarz-Oszillator (bei den ISA-Modellen per Jumper, bei den PCI-/cPCI-Modellen per Software auf 1 MHz umstellbar). Dieser Takt kann bei entsprechender Konfiguration auch an der Sub-D-Buchse abgegriffen werden (nicht ME-1400C/D). Die Karten verfügen über einen externen Interrupteingang (nicht ME-1400/E) und die ISA-Modelle über eine Wait-State-Logik.

Die Verbindung zur Außenbeschaltung wird über eine 37polige (ISA- und ME-1400E Modelle) bzw. eine 78polige Sub-D-Buchse (ME-1400/A/B/C/D) hergestellt. Bei der ME-14B und ME-1400EB werden die Signale für die zweite TTL-I/O- und Zählereinheit über ein Flachbandkabel auf ein zusätzliches Slotblech mit 37poliger Sub-D-Buchse geführt (im Lieferumfang enthalten).

Die Basisadresse wird bei den ISA-Modellen über DIP-Schalter eingestellt. Sie kann in einem weiten Bereich variiert werden. Bei den PCI-/cPCI-Modellen wird die Ressourcen-Vergabe vom BIOS bzw. Betriebssystem automatisch vorgenommen (Plug&Play).

1.3 Systemanforderungen

Die ME-14 ISA Modelle können in einem PC (XT oder neuer) mit einem freien 8 Bit ISA-Steckplatz eingesetzt werden. Die PCI-Modelle setzen einen Rechner mit freiem Standard-PCI- bzw. CompactPCI-Steckplatz voraus. Wir empfehlen ein System mit Intel® Pentium® Prozessor oder Kompatible.

1.4 Wichtiger Hinweis für die ISA-Versionen

Falls Sie einen PC mit PCI-Bus und einem BIOS mit Plug&Play-Funktionalität benutzen, müssen Sie für alle ISA-Einsteckkarten mit Interruptfunktion im BIOS Ihres Rechners den Interruptkanal dieser Karte für den ISA-Bus reservieren. Das entsprechende BIOS-Menü kann je nach BIOS-Hersteller etwas variieren (siehe Handbuch Ihres Motherboards). **Ansonsten ist die Interruptfunktion nicht gewährleistet!!!**

Beachten Sie, daß bei neueren Rechnern der ISA-Bus – abweichend von seiner Spezifikation – teilweise mit mehr als 8 MHz betrieben werden kann (siehe Einstellung im Setup Ihres PCs). In diesem Fall können wir jedoch eine einwandfreie Funktion der ISA-Karte nicht gewährleisten.

1.5 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.

Hinweis: Falls Sie PCI/cPCI-Modelle zusammen mit bereits vorhandener Applikationssoftware benutzen wollen, beachten Sie bitte die Hinweise in den entsprechenden README-Dateien.

Falls Sie eine ISA-Karte verwenden, nehmen Sie bitte zuerst die Jumpereinstellungen auf der Karte vor (siehe folgende Kapitel).

2.1 Hardware-Installation der ISA-Modelle

☛ **Schalten Sie Ihren Rechner aus.**

Achtung: Gefahr der Zerstörung hochempfindlicher Bauteile durch elektrostatische Entladung!

Deshalb: Entladen Sie Ihre Person vor Einbau der Karte indem Sie z. B. ein blankes Gehäuseteil Ihres Rechners berühren.

☛ Ziehen Sie das Netzkabel Ihres Rechners.

☛ Öffnen Sie das Gehäuse.

2.1.1 Position der Jumper

Bei den ISA-Modellen ME-14A/B müssen vor dem Einbau Jumper-/DIP-Schalter-Einstellungen vorgenommen bzw. dahingehend überprüft werden, ob die werkseitigen Einstellungen für Ihren Rechner geeignet sind.

Die Positionen der Jumper und DIP-Schalter sind in den schematischen Darstellungen der Karten gekennzeichnet. Erläuterungen zu den Einstellungen finden Sie in den nachfolgenden Kapiteln.

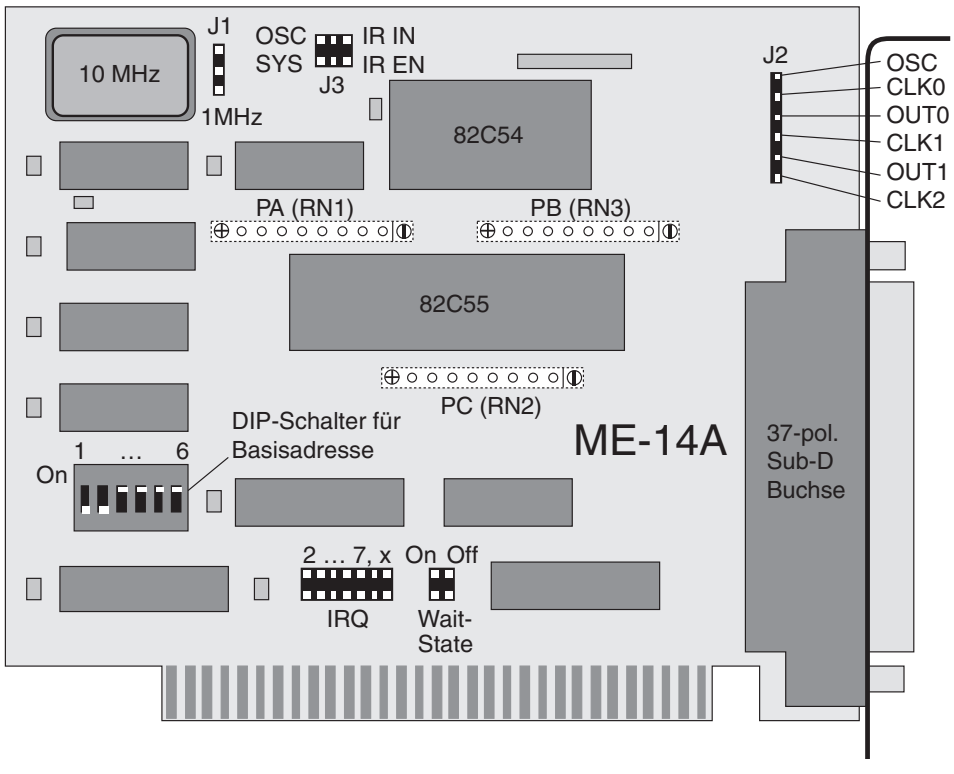


Abb. 1: Schematische Darstellung der ME-14A ISA

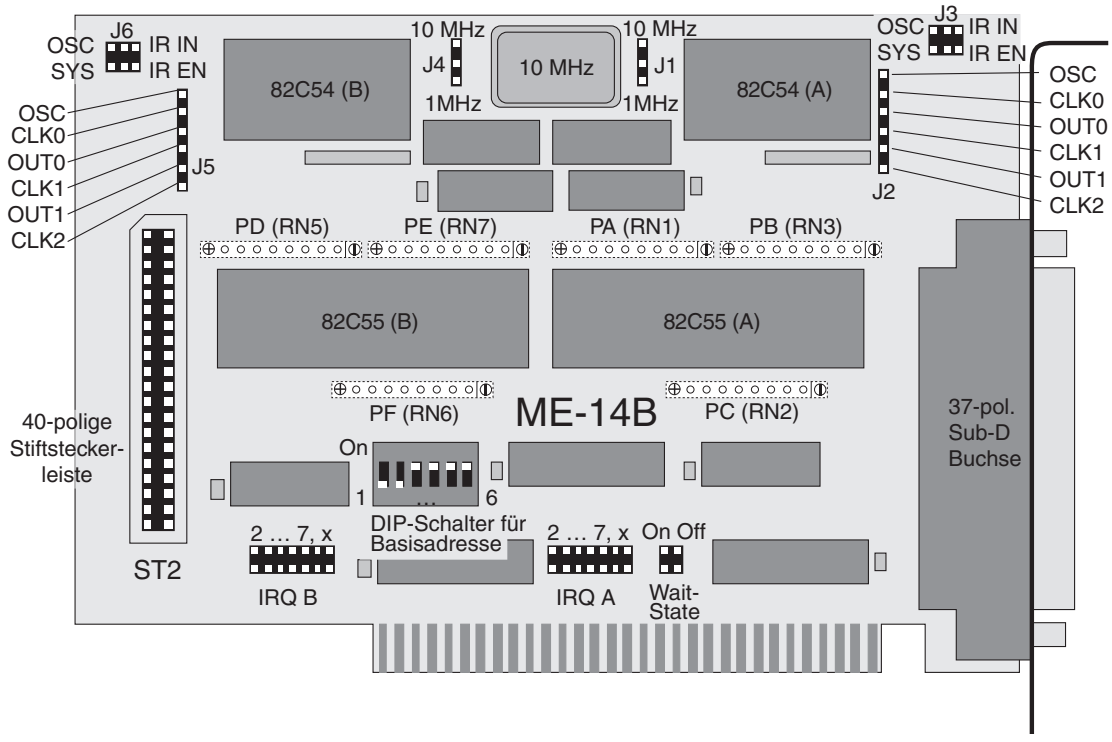


Abb. 2: Schematische Darstellung der ME-14B ISA

2.1.2 Einstellungen der DIP-Schalter/Jumper

2.1.2.1 Basisadresse

Mit dem 6fach DIP-Schalter läßt sich die Basisadresse (BA) der ME-14 im Bereich von 0Hex bis 3F0Hex in Schritten von 10Hex einstellen. Mit der Basisadresse beginnend belegt die ME-14A 8 Bytes, die ME-14B 16 Bytes des I/O-Adreßraumes. Vermeiden Sie bei der Einstellung Adreßkonflikte mit anderen Karten!

Ein geöffneter Schalter (Stellung OFF) entspricht dem logischen Zustand „1“. Die Basisadresse errechnet sich dann durch Aufsummierung der Wertigkeit der Schalter in Stellung OFF. Das Beispiel erläutert die werksseitige Grundeinstellung der Karte (300Hex).

Beachten Sie auch, daß bei den ISA-Karten nur die unteren 10 Adreßbits ausdekodiert sind. Es treten also Adreß-Spiegelungen bei BA+400Hex, BA+800Hex, BA+C00Hex, BA+1000Hex usw. auf.

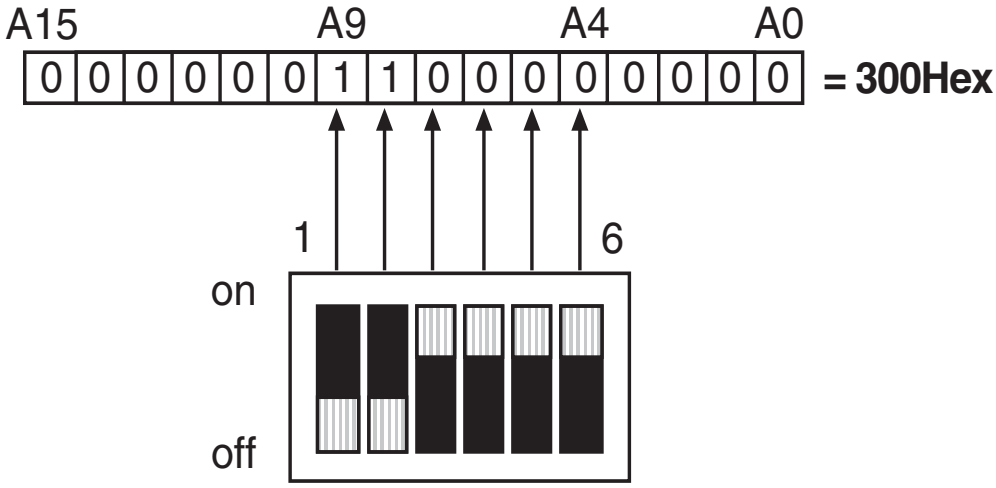


Abb. 3: Zuordnung der Adressen zu DIP-Schalterstellungen, hier Grundeinstellung: 300Hex

2.1.2.2 Interrupts

Für die Interruptsteuerung muß auf der ME-14A mittels Jumperreihe „IR“ und auf der ME-14B mit der Jumperreihe „IRQ A“ und „IRQ B“, eine Interrupt-Request-Leitung (IRQ) zwischen 2…7, bzw. X für „keine“ ausgewählt werden:

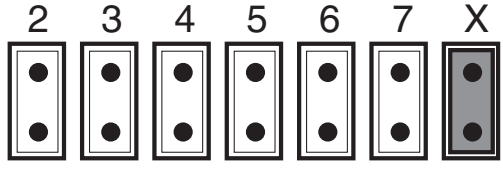
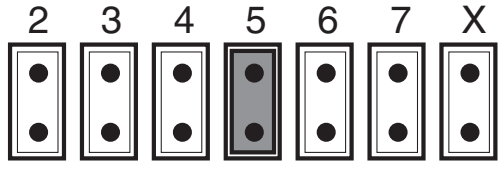
Jumper IR bzw. IRQ A und B	Funktion
	kein IRQ
	Beispiel: IRQ5 gewählt

Tabelle 2: Jumper für IRQ

2.1.2.3 Wait-States

Mit dem mit „WAIT STATE“ bezeichneten Jumper wird die Wait-State Logik der ME-14 aktiviert, die den Betrieb der Karte bei Bustaktfrequenzen von mehr als 8 MHz ermöglicht.

Jumper WAIT STATE	Funktion
	Wait-State Logik aus.
	Wait-State Logik ein.

Tabelle 3: Jumper für Wait-States

2.1.2.4 Zähler-Takt

Mit Jumperreihe J1 (und J4 bei der ME-14B) kann die Oszillator-Frequenz für den/die Zählerbaustein(e) 82C54 von 10 MHz (Standardeinstellung) auf 1 MHz umgeschaltet werden (aus Kompatibilitätsgründen zu älteren ME-14-Modellen kann dies eventuell notwendig sein).

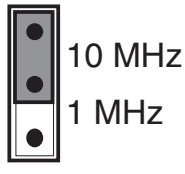
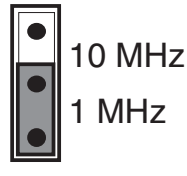
Jumper J1 bzw. J4	Funktion
	10 MHz
	1 MHz (Kompatibilität zu älteren ME-14 Modellen)

Tabelle 4: Jumper für Quarzoszillator-Takt

2.1.2.5 Kaskadierung der Zähler

Mit Jumperreihe J2 (und J5 bei der ME-14B) können die Zähler ohne externe Beschaltung kaskadiert werden. (siehe „Abb. 7: Blockschaltbild des 82C54“ auf Seite 21).

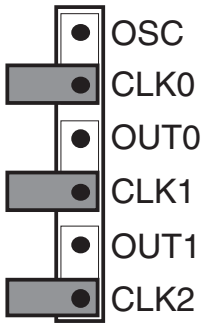
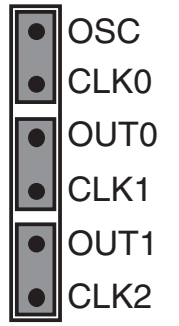
Jumper J2/J5	Funktion	Jumper J2/J5	Funktion
	Die Ein-/Ausgänge der Zähler 0…2 sind auf die Sub-D Buchse bzw. Stiftsteckverbinder geführt, sie beeinflussen sich nicht gegenseitig. (Kompatibilität zu älteren ME-14)		Beispiel: Zähler 0…2 sind in Reihe geschaltet (kaskadiert), Zähler 0 wird vom Oszillatortakt gespeist*

Tabelle 5: Kaskadierung der Zähler

* Jumperreihe J2 bzw. J5 verbindet in dieser Stellung …

- den Clock 0-Eingang des Zählers 0 mit dem Oszillatortakt
- den Out 0-Ausgang mit dem Clock 1-Eingang des Zählers 1
- den Out 1-Ausgang mit dem Clock 2-Eingang des Zählers 2.

Die Verbindung erfolgt vor der Sub-D-Buchse bzw. Stiftsteckverbinder, so daß keine externe Beschaltung zur Kaskadierung der Zähler erforderlich ist.

2.1.2.6 Taktausgabe und Interruptsteuerung

Mit der Jumperreihe J3 (und J6 auf der ME-14B) wird die Belegung der Sub-D Buchsen-Pins 18 und 36 festgelegt. Sie können diese Anschlüsse wahlweise zur Taktausgabe oder zur Interruptsteuerung (Eingänge) benutzen. Sinnvoll sind nur die beiden unten beschriebenen Jumper-Einstellungen (siehe Seite 23):

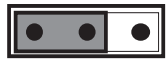
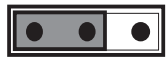
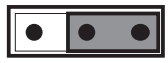
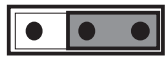
Jumper J3 bzw. J6	Funktion
OSC  IR IN SYS  IR EN	Systemtakt und Oszillatortakt der ME-14 werden auf die Pins 18 und 36 der Sub-D Buchse(n) geführt. Kompatibilität zu älteren ME-14: Pin 36 = Oszillatortakt 1 MHz (OSC); Pin 18 = Systemtakt (SYS)
OSC  IR IN SYS  IR EN	Interrupt-Enable Eingang und IRO-Eingang der ME-14 werden auf die Pins 18 und 36 der Sub-D Buchse(n) geführt: Pin 36 = Interrupt-Eingang (IR_IN); Pin 18 = Interrupt Enable Eing. (IR_EN)

Tabelle 6: Jumper Taktausgabe und Interruptsteuerung

2.1.2.7 Standardeinstellungen

Funktion	Jumper/Schalter	Einstellung
Basisadresse	DIP-Schalter	300Hex
Interrupt	IRQ (IRQ A und B)	X (keine)
Taktausgabe/Interuptsteuerung	J3 (zusätzlich J6 auf ME-14B)	OSC, SYS
Wait-States	WAIT STATE	OFF
Oszillatorfrequenz	J1 (zusätzlich J4 auf ME-14B)	1 MHz
Zählerkaskadierung	J2 (zusätzlich J5 auf ME-14B)	nicht gesteckt

Tabelle 7: Werksseitige Standardeinstellungen der ME-14A/B

3 Hardware

3.1 Blockschaltbild ME-14

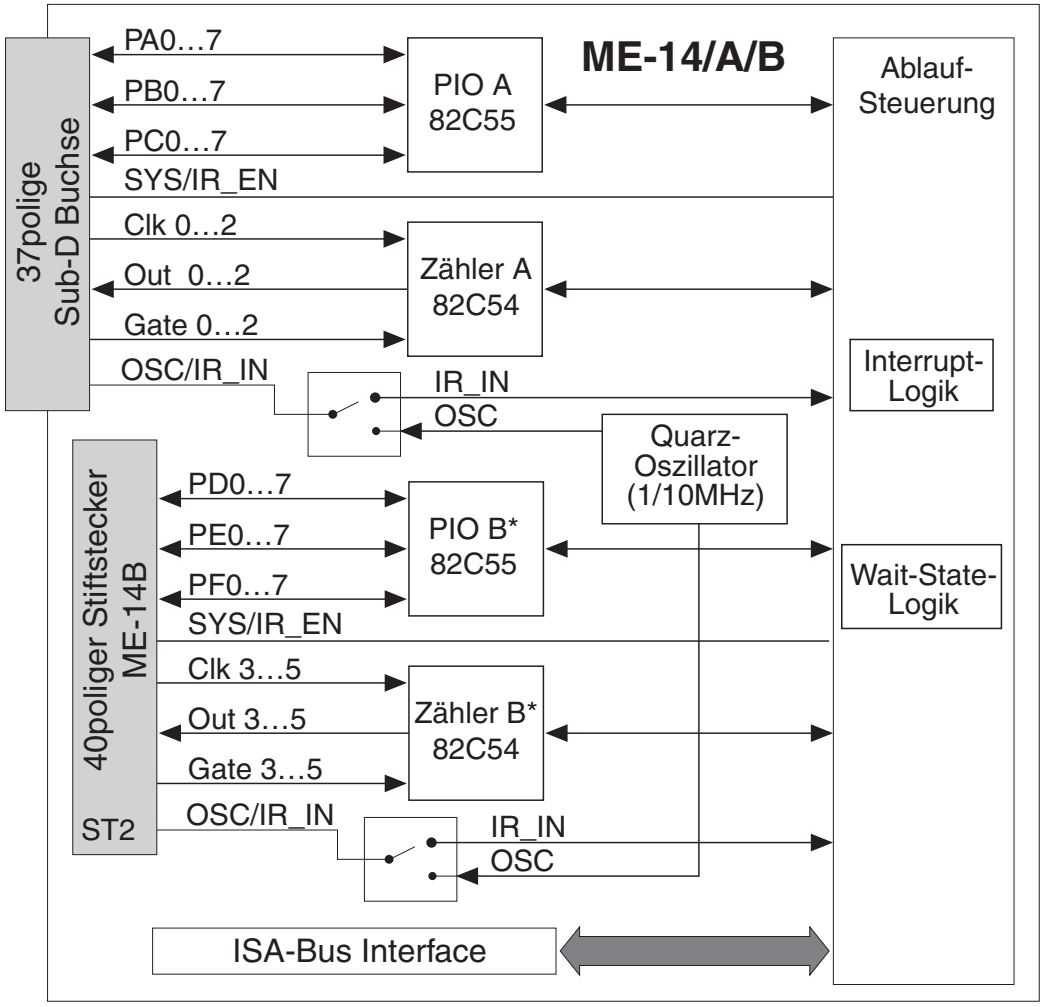


Abb. 4: Blockschaltbild der ME-14

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

ME-14A: 24 Digital-I/Os (PIO A) und 3 x 16 Bit Zähler (Zähler A) sowie Oszillator und Interrupt-Eingang.

ME-14B: 48 Digital-I/Os (PIO A, B) und 6 x 16 Bit Zähler (Zähler A, B) sowie Oszillator und Interrupt-Eingang.

3.2 Blockschaltbild ME-1400/A/B/E/EA/EB

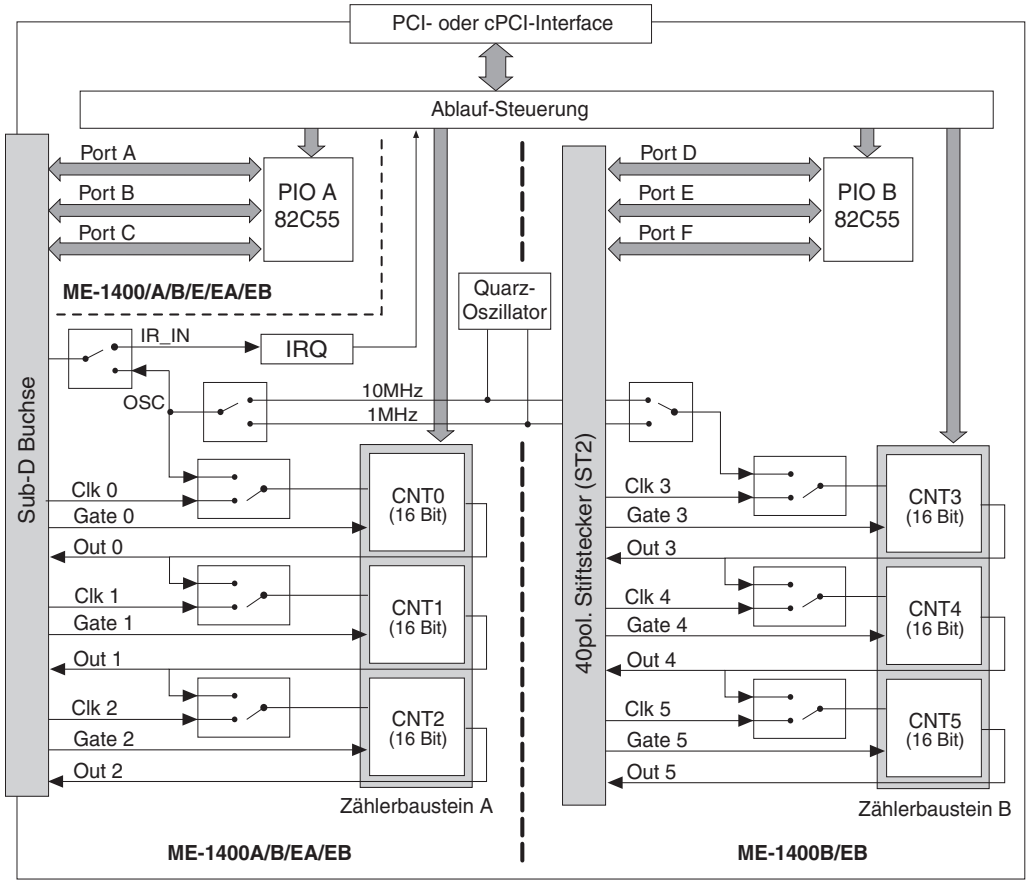


Abb. 5: Blockschaltbild der ME-1400/A/B/E/EA/EB

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

ME-1400/E: 24 Digital-I/Os (PIO A); ohne Oszillator und Interrupt-Eingang.

ME-1400A/EA: 24 Digital-I/Os (PIO A) und 3 x 16 Bit Zähler (CNT0…2) sowie Oszillator und Interrupt-Eingang.

ME-1400B/EB: 48 Digital-I/Os (PIO A, B) und 6 x 16 Bit Zähler (CNT0…5) sowie Oszillator und Interrupt-Eingang.

3.3 Blockschaltbild ME-1400C/D

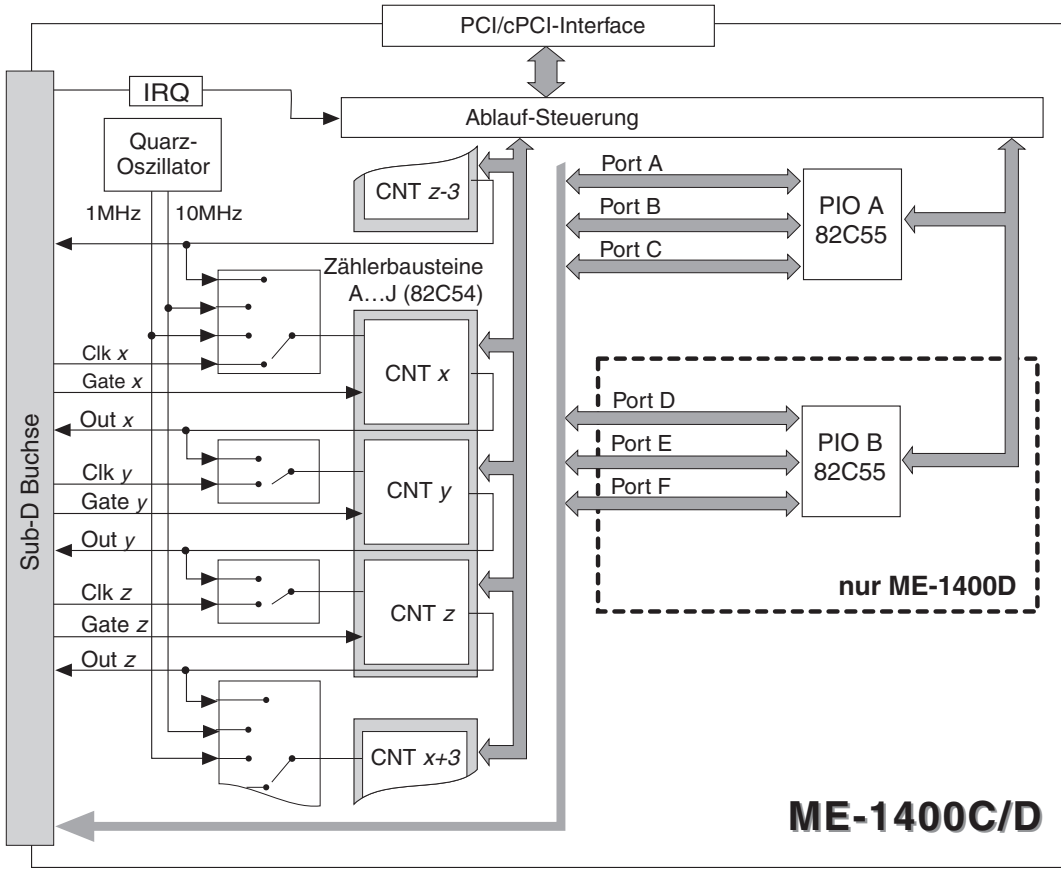


Abb. 6: Blockschaltbild der ME-1400C/D

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

ME-1400C: 24 Digital-I/Os (PIO A) und 15 x 16 Bit Zähler (CNT0…14) sowie Interrupt-Eingang.

ME-1400D: Erweiterungskarte um 24 Digital-I/Os (PIO B) und 15 x 16 Bit Zähler (CNT15…29).

Die Zähler können per Software kaskadiert werden. Der jeweils erste Zähler eines Bausteins kann vom Quarz-Oszillator gespeist werden. Es gelten folgende Indizes für die jeweils 3 Zähler (CNT *x, y, z*) pro Zählerbaustein (A…J). Siehe Tabelle 8:

	ME-1400C					ME-1400D				
Zählerbaustein ➜	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Zählernr. CNT x	0	3	6	9	12	15	18	21	24	27
CNT y	1	4	7	10	13	16	19	22	25	28
CNT z	2	5	8	11	14	17	20	23	26	29

Tabelle 8: Zähler-Indizes

3.4 Generelle Hinweise

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

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

3.5 Digitale Ein-/Ausgabe (8255)

Als programmierbarer Ein-/Ausgabe-Baustein (PIO) kommt der Standardtyp 82C55 (vollkompatible CMOS-Version des 8255A) zum Einsatz. Er verfügt über 3 x 8 Bit breite I/O-Ports und ist TTL/CMOS-kompatibel.

Für die digitalen Ports der Karte gelten folgende Zuordnungen zu den Ports (P*x*0…7) des einzelnen PIO-Bausteins:

I/Os des 8255 ➜	**PA0…7**	**PB0…7**	**PC0…7**
PIO A	Port A	Port B	Port C
PIO B	Port D	Port E	Port F

Tabelle 9: Port-Zuordnung

Der PIO-Baustein wird im Modus 0 „Einfache Ein-/Ausgabe" betrieben. Die Konfigurierung erfolgt per Software durch den Anwender. Verwenden Sie hierzu die mitgelieferte Treibersoftware siehe Kap. 5.3.2 „Digital-I/O-Funktionen" auf Seite 52ff.

3.6 Zähler (8254)

Als Zähler-Baustein kommt der Standardtyp 82C54 zum Einsatz. Dies ist ein sehr vielseitiger Baustein, der über 3 unabhängige 16 Bit Zähler verfügt. Die folgende Abbildung zeigt schematisch den Aufbau des Bausteins:

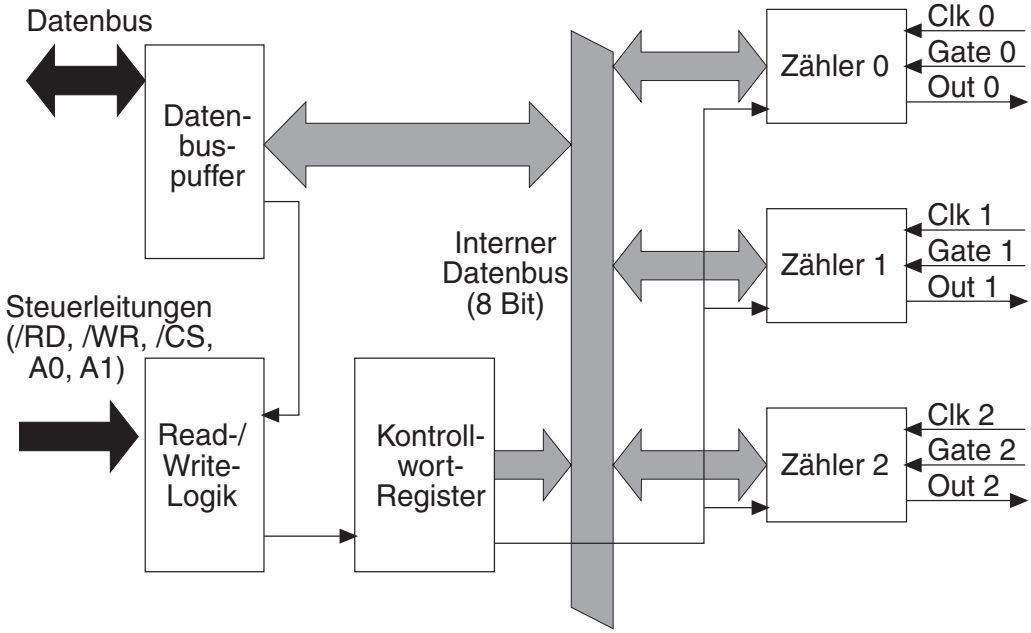


Abb. 7: Blockschaltbild des 82C54

Die Konfigurierung der Zähler erfolgt per Software. Mit der Funktion *…CntWrite* wird jeder Zähler einzeln geladen, konfiguriert und gestartet. Bei geeigneter Beschaltung des Gate-Eingangs (High-Pegel) zählt der entsprechende Zähler negativ flankengesteuert abwärts, wobei im BCD- oder Binärcode gezählt werden kann. Folgende Betriebsarten können für jeden Zähler voneinander unabhängig eingestellt werden (eine detaillierte Beschreibung der Modi finden Sie auf Seite 37ff):

- 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

Beachten Sie auch die Einstellungen für Kaskadierung und Taktquelle, etc., die für ISA-Modelle per Jumper und für PCI/cPCI-Modelle mit Hilfe der Funktion *me1400CntInitSrc* vorzunehmen sind.

Wir empfehlen die Verwendung der mitgelieferten Bibliotheksfunktionen siehe Kap. 5.3.3 „Zählerfunktionen“ auf Seite 59 (bei ISA-Modellen ist auch die Programmierung auf Registerebene siehe Kap. 4.4 möglich).

Auf der ME-14A/B sind die Clk-, Gate- und Out-Leitungen von der Sub-D-Buchse direkt mit den entsprechenden Leitungen des 82C54 verbunden. Bei den PCI/cPCI-Varianten sind die Gate- und Out-Leitungen von der Sub-D-Buchse ebenfalls direkt mit den entsprechenden Eingängen des 82C54 verbunden, in den Clk-Leitungen sind noch „Multiplexer“ zwischengeschaltet.

3.6.1 Kaskadierung der Zähler

Zur Kaskadierung der Zähler können die Ausgänge der Zähler eines Bausteins ohne externe Beschaltung „in Reihe“ geschaltet werden. Auf der ME-1400C und D ist auch eine Kaskadierung von Baustein zu Baustein möglich (Ausnahme: von Baustein E nach F). Siehe auch Blockschaltbilder Seite 17ff.

Die Verbindung der Zähler erfolgt bei den PCI-/cPCI-Modellen per Software mit der Funktion *me1400CntInitSrc* (nach dem Einschalten oder nach einem Reset werden alle Zähler vom externen Takt gespeist). Bei den ISA-Modellen wird diese Einstellung mit Jumperreihe J2 (zusätzlich J5 bei der ME-14B) vorgenommen.

Sollen zum Beispiel die Zähler 0…2 kaskadiert werden, so sind per Jumper bzw. Software folgende Einstellungen vorzunehmen:

- Den Takt-Eingang des Zählers 0 (Clk 0) mit dem externen Oszillatortakt verbinden
- Den Ausgang von Zähler 0 (Out 0) mit dem Takt-Eingang des Zählers 1 (Clk 1) verbinden
- Den Ausgang von Zähler 1 (Out 1) mit dem Takt-Eingang des Zählers 2 (Clk 2) verbinden
- Außerdem müssen die Gate-Eingänge (Gate 0…2) zur Freigabe der Zähler beschaltet werden (High-Pegel).

- Am Ausgang des Zählers 2 (Out 2) steht das kaskadierte Zählersignal zur Verfügung.

Gleichzeitig können die Ausgänge aller Zähler auch an der Sub-D Buchse abgegriffen werden.

3.6.2 Zählertakt

Der Zählertakt kann wahlweise intern (1 MHz/10 MHz), extern oder per Kaskadierung zugeführt werden (siehe auch folgende Kapitel und Blockschaltbilder Seite 17ff.)

Der interne Oszillator kann für jeden Baustein separat von 1 MHz (Default) auf 10 MHz umgeschaltet werden. Bei den PCI-/cPCI-Modellen erfolgt diese Umschaltung per Software, bei den ISA-Modellen mit Jumper J1 (und J4 bei der ME-14B).

3.6.3 Taktausgabe und Interruptsteuerung

Der Pin mit der Bezeichnung „OSC/IR_IN“ bzw. „IR_IN“ dient standardmäßig als Interrupt-Eingang. Alternativ dazu kann ein systemunabhängiger symmetrischer Takt ausgegeben werden, der von dem Quarzoszillator auf der Karte erzeugt wird (1 MHz oder 10 MHz).

Ausnahmen: Auf der ME-1400C dient der Pin nur als Interrupt-Eingang, auf der ME-1400, ME-1400D und ME-1400E hat dieser Pin keine Funktion.

OSC: **Oszillatortakt Ausgang** - mit diesem Signal wird der interne Takt, mit dem die Zähler gespeist werden (1 MHz oder 10 MHz) zur Sub-D Buchse geführt.

IR_IN: **IRQ Eingang** - Eine steigende Flanke an diesem Eingang löst einen Interrupt aus. Falls IR_IN auf High gehalten wird, nicht beschaltet wird oder aber am IRQ-Jumper der ISA-Modelle die Stellung „X“ ausgewählt wurde, so wird IR_IN ignoriert.

Bei den PCI- und cPCI-Modellen ist die Interrupt-Logik nach dem Einschalten deaktiviert und muß zunächst mit der Funktion *me1400EnableInt* freigeschaltet werden.

Das Pin „SYS/IR_EN“ steht nur bei den ISA-Modellen zur Verfügung.

SYS: **Systemtakt Ausgang** - mit diesem Signal wird der Systemtakt des ISA-Buses über einen Treiber zur Sub-D-Buchse geführt

IR_EN: **Interrupt Enable** - um die Interruptlogik der ME-14 ISA zu aktivieren, muß an diesem Eingang ein Low-Pegel angelegt werden. Wenn der Pin unbeschaltet bleibt, wird über einen internen Pull-Up-Widerstand die Interruptlogik deaktiviert.

Modell	Funktion	Default	Einstellung
ME-14A	OSC/IR_IN	OSC	J3 (s. Seite 15)
	SYS/IR_EN	SYS	
ME-14B	OSC/IR_IN	OSC	J3 und J6 (s. Seite 15)
	SYS/IR_EN	SYS	
ME-1400	n.c.	–	–
ME-1400E	n.c.	–	
ME-1400A	OSC/IR_IN	IR_IN	per Software (s. Seite 51)
ME-1400EA	OSC/IR_IN	IR_IN	
ME-1400B	OSC/IR_IN	IR_IN	per Software (s. Seite 51)
ME-1400EB	OSC/IR_IN	IR_IN	
ME-1400C	IR_IN	IR_IN	Eingang
ME-1400D	nicht verfügbar (bitte nicht beschalten)		

Tabelle 10: Übersicht Taktausgabe und Interuptsteuerung

3.7 Beschaltung

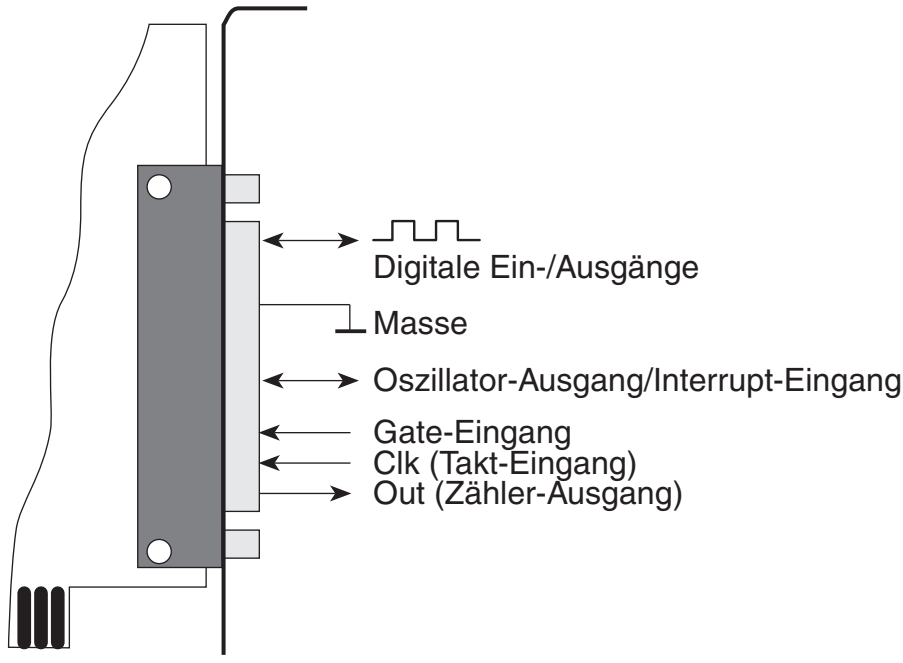


Abb. 8: Beschaltung der ME-14/1400 Serie

Hinweis: Alle Signale müssen TTL- bzw. CMOS-Pegel einhalten und einen Bezug zur PC-Masse haben.

3.7.1 Pull-Up/Pull-Down Widerstände

Da nach dem Einschalten der Versorgungsspannung alle digitalen Ports als Eingänge konfiguriert werden, sind die zugehörigen Pins (ohne externe Beschaltung) zunächst hochohmig. Je nach Anwendungsfall kann jedoch ein definierter Einschaltzustand der I/O-Leitungen erforderlich sein. Zu diesem Zweck bietet die ME-14/1400 die Möglichkeit direkt auf der Platine Pull-Up bzw. Pull-Down Widerstände zu bestücken. Dies kann mit geeigneten Widerstandsarrays (4,7 kΩ empfohlen) portweise erfolgen. Beachten Sie, daß sich bei Verwendung von Pull-Up Widerständen die Strombelastbarkeit des Ausgangs entsprechend verringert (z. B. bei R_{up}=4,7 kΩ I_{max}=1,6 mA).

Durch entsprechendes Bestücken der Widerstandsarrays können Sie die Widerstände als Pull-Up- oder Pull-Down-Widerstände verschalten. Für Pull-Up-Widerstände müssen Sie den gemeinsammen Pin des Arrays auf das Plus-Symbol stecken, für Pull-Down-Widerstände dementsprechend auf das Minussymbol (siehe Abb. 9: bis 13).

Achtung:
Beachten sie unbedingt die ESD-Bestimmungen zum Schutz der Karte vor statischer Entladung.

Port	Array-Nr. ME-14A/B	Array-Nr. ME-1400/A/B /E/EA/EB	Array-Nr. ME-1400C/D
Port A	RN1	RN3	RN1
Port B	RN3	RN2	RN2
Port C	RN2	RN1	RN3
Port D	RN5	RN4	RN1
Port E	RN7	RN5	RN2
Port F	RN6	RN6	RN3

Tabelle 11: Zuordnung der Widerstandsarrays

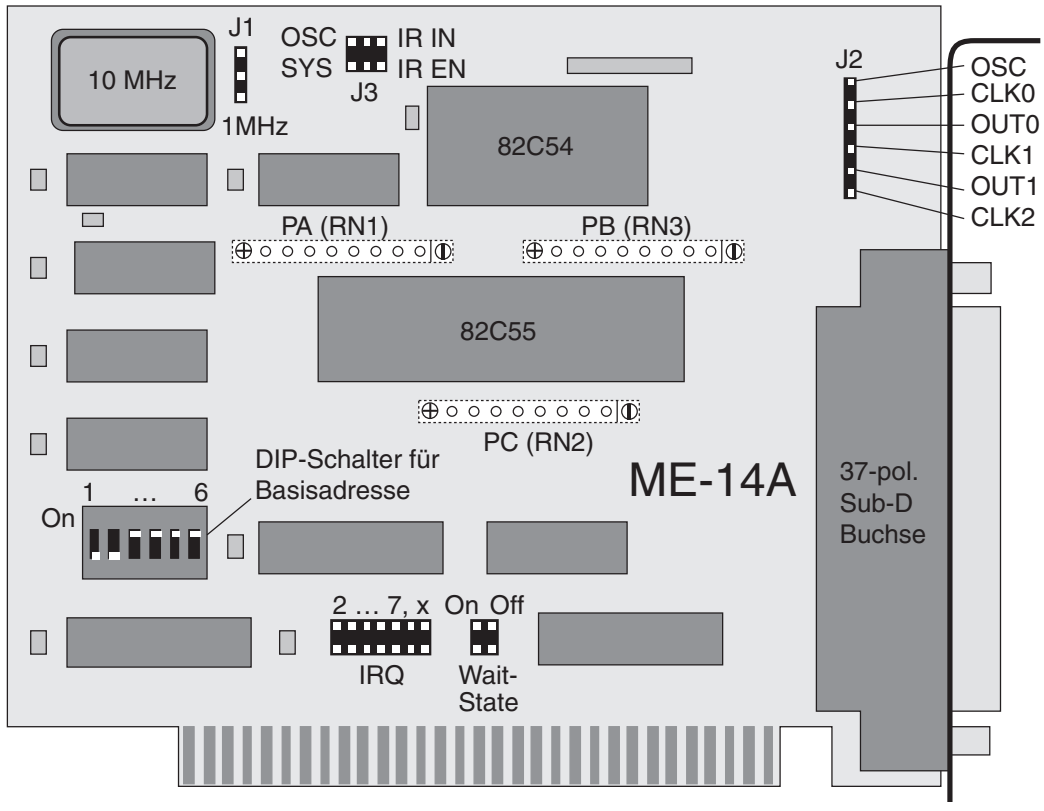


Abb. 9: Anordnung der Widerstandsarrays ME-14A ISA

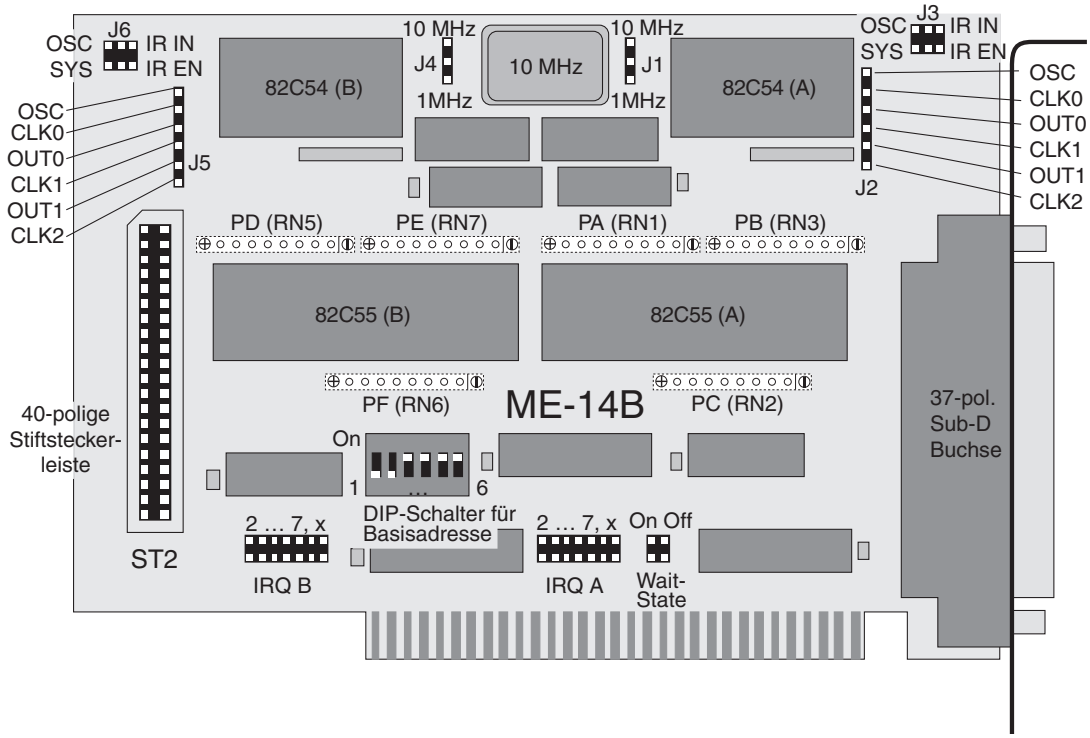


Abb. 10: Anordnung der Widerstandsarrays ME-14B ISA

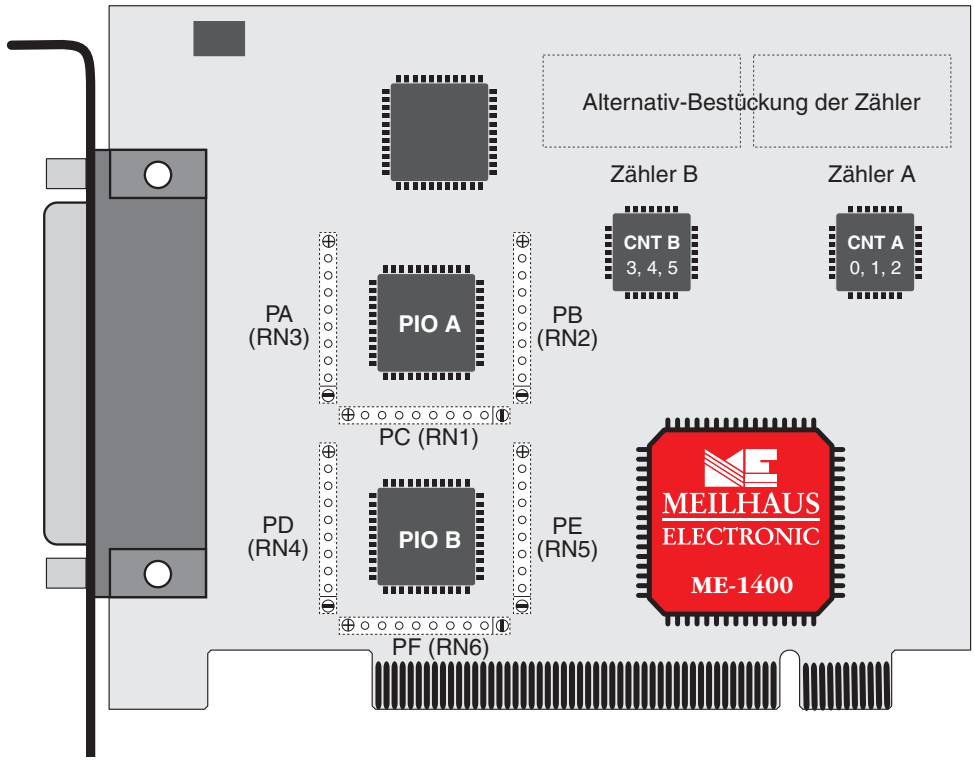


Abb. 11: Anordnung der Widerstandsarrays ME-1400/A/B PCI

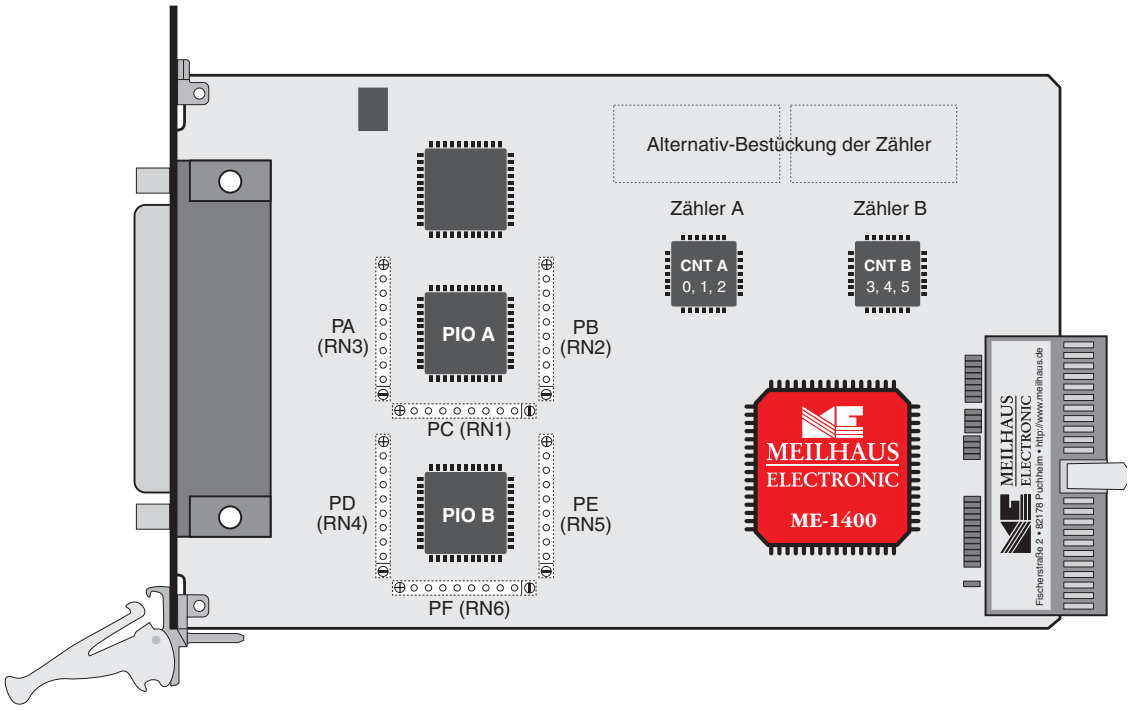


Abb. 12: Anordnung der Widerstandsarrays ME-1400/A/B cPCI

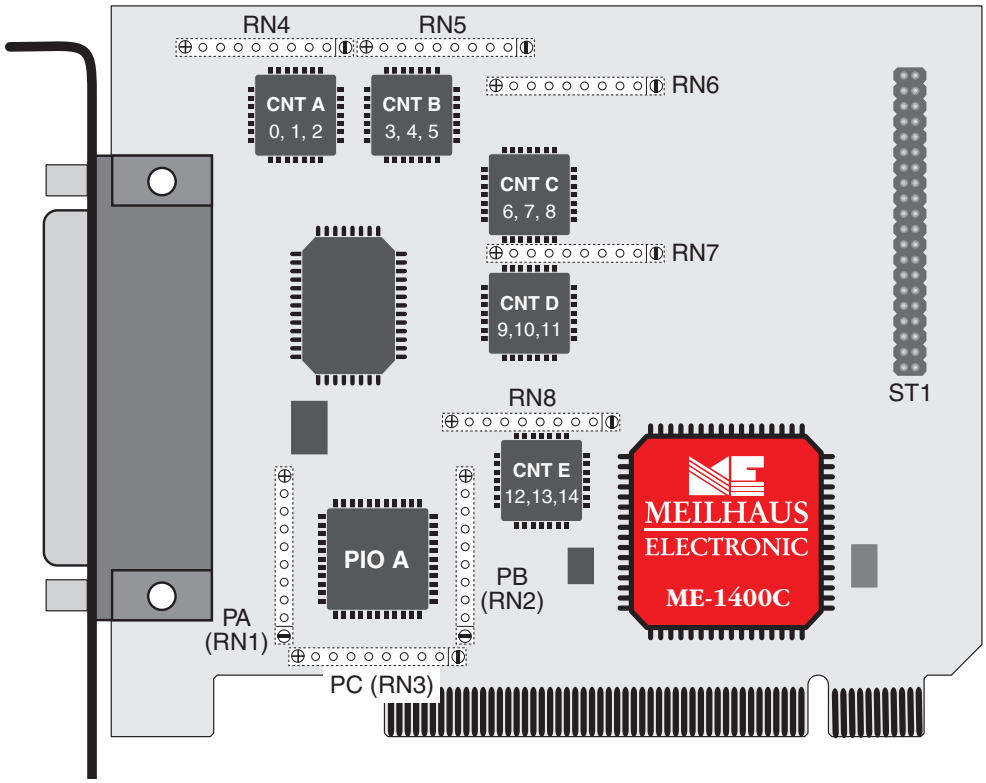


Abb. 13: Anordnung der Widerstandsarrays ME-1400C PCI

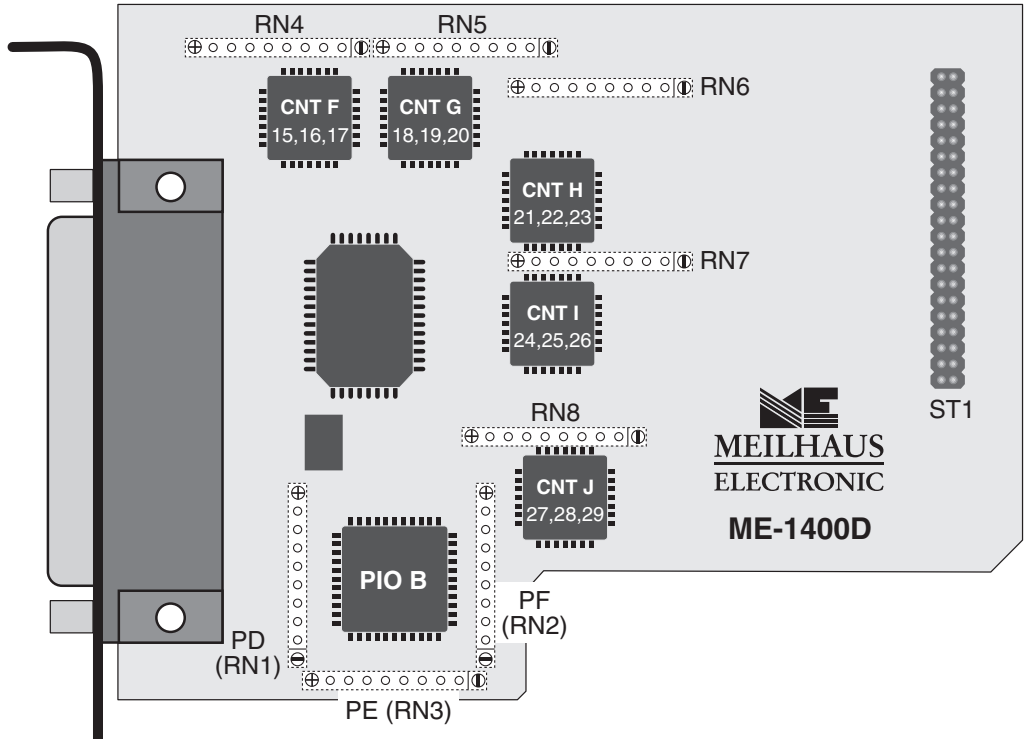


Abb. 14: Anordnung der Widerstandsarrays ME-1400D EXP

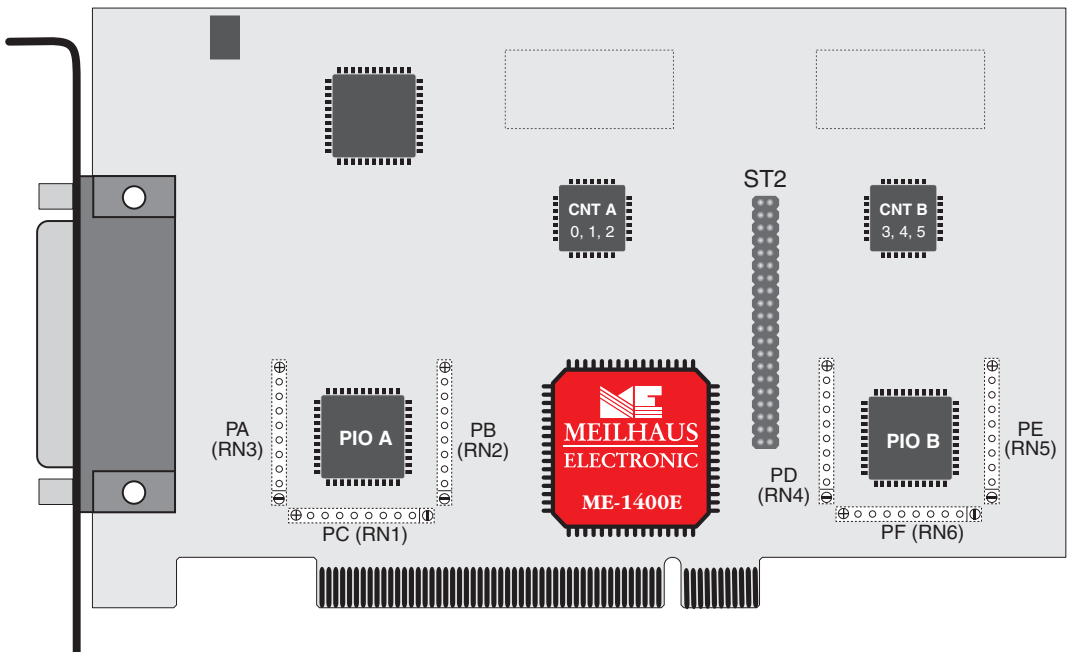


Abb. 15: Anordnung der Widerstandsarrays ME-1400E/EA/EB

3.8 Testprogramm

Zum Test der Einsteckkarte wird ein einfaches Testprogramm mit komfortabler Bedienoberfläche mitgeliefert. Sie finden das jeweilige Testprogram in einem entsprechenden Unterverzeichnis von `C:\Meilhaus\` (Default). Das Testprogramm kann durch Doppelklick gestartet werden. (Vorraussetzung: Systemtreiber korrekt installiert).

4 Programmierung

Das Treiberkonzept für die ME-14 bzw. ME-1400 Familie bietet Ihnen die Möglichkeit, ohne Änderung bereits vorhandener Applikationssoftware anstatt einer „alten“ ISA-Karte der ME-14 Familie eine äquivalente PCI-Karte zu verwenden. Da Funktionsumfang und Syntax identisch sind, ist dies im Idealfall ohne Neucompilierung möglich. (Voraussetzung: „alte“ ISA-Karte und neue PCI-Karte verwenden den gleichen Wert im Parameter <Board-Number>). Zur genauen Vorgehensweise beachten Sie bitte die entsprechenden README-Dateien des ME-1400 Treibers.

Dies gilt nur, wenn Sie zur Programmierung Ihrer ISA-Karte die Funktionsbibliothek des ME-14 Treibersystems verwendet haben!

4.1 Hochsprachenprogrammierung

Folgende Hochsprachen werden standardmäßig unterstützt:

- Visual C++ ab Version 4.0.
- Delphi ab Version 2.0.
- VisualBASIC ab Version 4.0.
- Für weitere Infos beachten Sie bitte die entsprechenden README-Dateien auf der ME-Power-CD.

Es ist darauf zu achten, daß für den Compiler und Linker die Pfade auf diese Dateien richtig gesetzt sind.

Durch Einbinden der hochsprachenspezifischen Definitionsdatei in Ihr Projekt können Sie viele Parameter in Form vordefinierter Konstanten und Makros übergeben. Alternativ ist die direkte Übergabe des entsprechenden Hex-Wertes jederzeit möglich.

4.1.1 Beispielprogramme

Zum leichteren Verständnis der Programmierung werden einfache Beispiele und kleine Projekte im Source-Code mitgeliefert. Die Beispielprogramme finden Sie im ME Software Developer Kit (ME-SDK), das standardmäßig ins Verzeichnis `C:\Meilhaus\`

me-sdk installiert wird. Bitte beachten Sie die Hinweise in den entsprechenden README-Dateien.

4.2 Agilent VEE-Programmierung

Die Agilent VEE-Komponenten für Ihre Karte finden Sie auf der „ME-Power-CD“ oder zum Download unter www.meilhaus.de.

Das Meilhaus VEE Treibersystem unterstützt die HP VEE Vollversionen 4.x und 5.x, HP VEE Lab, Agilent VEE Pro und Agilent VEE OneLab. Zur Installation der VEE-Komponenten und für weitere Infos beachten Sie bitte die Dokumentation, die Sie mit dem VEE Treibersystem erhalten. Zu den Grundlagen der VEE-Programmierung benutzen Sie bitte Ihre VEE Dokumentation und die VEE Online-Hilfe.

4.2.1 User Objects

Zur komfortableren Handhabung des Treibers wurden vordefinierte User Objects erstellt, welche intern API-Funktionen aufrufen. Diese sind über den zusätzlichen Menüpunkt „ME Board“ in der VEE-Entwicklungsumgebung aufrufbar und können – wie andere Standard-Funktionen von VEE auch – in der Entwicklungsumgebung plaziert und in einer Applikation „verdrahtet“ werden.

Die User Objects sind weitgehend selbsterklärend und basieren auf den im Kap. „Funktionsreferenz“ dokumentierten API-Funktionen. Zusätzlich gibt es noch sog. „Expanded User Objects“, um Ihnen das Programmieren so bequem wie möglich zu machen. Eine Kurzbeschreibung zum jeweiligen User Object finden Sie auch unter „Description“ indem Sie den Mauszeiger über das entsprechende UO bewegen und die rechte Maustaste drücken.

Die User Objects können für eigene Bedürfnisse jederzeit geändert, angepaßt und bei Bedarf als kundenspezifisches Objekt abgespeichert werden.

4.2.2 Demoprogramme

Zur Demonstration und zum leichteren Verständnis wurden kleine Demoprogramme erstellt, die alle wichtigen User Objects enthalten. Die Demoprogramme sind über den Menüpunkt „ME Board – Demos“ aufrufbar.

Die VEE-Demoprogramme enthalten teilweise auch Ergänzungen der „normalen“ User Objects und tragen zur leichteren Unterscheidung von diesen das Präfix "x…" im Dateinamen.

4.2.3 Das "ME Board"-Menü

Das Installationsprogramm erweitert die Menüleiste von VEE automatisch um den Eintrag „ME Board“. Dadurch ist eine komfortable Nutzung aller unter VEE zur Verfügung stehenden Treiberfunktionen möglich. Über das „ME Board“-Menü können Sie nach Kartenfamilien geordnet, die ent-sprechenden Treiber- und Demo-User Objects aufrufen.

Hinweis:
Der Installationsumfang der User Objects (UOs) richtet sich nach der von Ihnen gewählten Kartenfamilie(n) zu Beginn der VEE Treiber-Installation. Sollten Sie UOs im „ME Board“-Menü aufrufen, die jedoch nicht installiert wurden, so führt dies zur Fehlermeldung:

File ´*filename*´ was not found. Error number: 700

Bei Bedarf können Sie die benötigten VEE Komponenten jederzeit nachinstallieren (siehe „ME-Power-CD“).

4.3 LabVIEW™-Programmierung

Die LabVIEW™-Komponenten für Ihre Karte finden Sie auf der „ME-Power-CD“ oder zum Download unter www.meilhaus.de.

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.

4.3.1 Virtual Instruments

Zur komfortableren Handhabung des Treibers wurden vordefinierte Virtual Instruments (VIs) erstellt. Diese sind über das Menü „Datei - Öffnen“ in LabVIEW™ aufrufbar und können – wie andere Standard-VIs von LabVIEW™ auch – in der Entwicklungsumgebung plaziert und in einer Applikation „verdrahtet“ werden.

Die „Source VIs“ sind weitgehend selbsterklärend und basieren auf den im Kap. „Funktionsreferenz“ dokumentierten API-Funktionen. Zusätzlich gibt es noch sog. „Expanded Virtual Instruments“, um Ihnen das Programmieren so bequem wie möglich zu machen.

Eine Kurzbeschreibung zum jeweiligen VI finden Sie auch im VI „…Function Tree“. Dieses VI dient nur der Dokumentation und kann über das Menü „Datei - Öffnen“ aufgerufen werden. Unter „Description“ finden Sie eine Kurzbeschreibung zum jeweiligen VI.

Die VIs können für eigene Bedürfnisse jederzeit geändert, angepaßt und bei Bedarf als kundenspezifisches VI abgespeichert werden.

4.3.2 Demoprogramme

Zur Demonstration und zum leichteren Verständnis wurden kleine Demoprogramme erstellt, die alle wichtigen „Virtual Instruments“ (VIs) enthalten. Die Demoprogramme sind über das Menü „Datei – Öffnen“ aufrufbar.

4.4 Pulsweiten-Modulation

Durch geeignete externe Beschaltung (siehe Abb. 17) kann mit Hilfe der drei Zähler eines jeden Zählerbausteins ein Signal mit variablem Tastverhältnis ausgegeben werden. Das Tastverhältnis kann zwischen 1…99% in 1%-Schritten variiert werden.

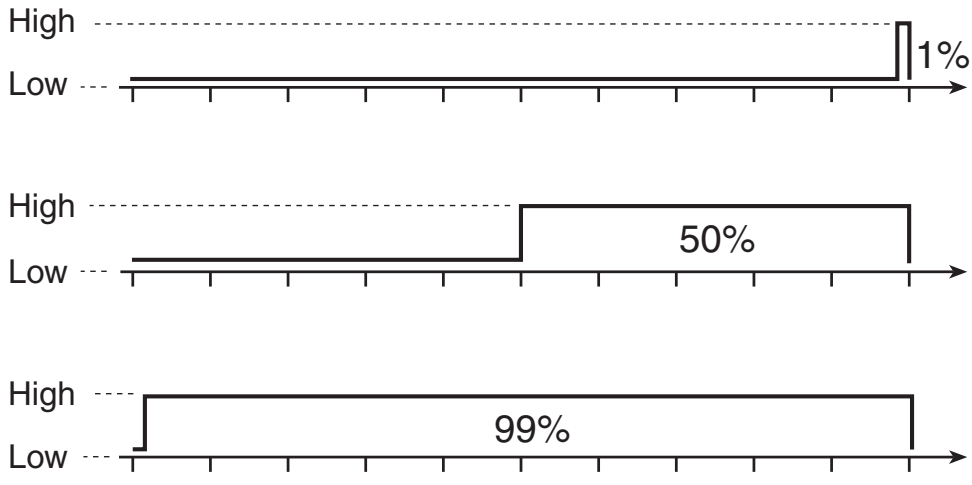


Abb. 16: Tastverhältnis PWM-Signal

Der Basistakt kann entweder über einen ext. Frequenzgenerator (max. 10MHz) oder vom internen Quarzoszillator (1MHz oder 10MHz) zugeführt werden (Parameter `<ClockSource>`). Mit dem Vorteiler (Parameter `<Prescaler>`) können Sie damit die Frequenz zwischen Basistakt/2 und Basistakt/65535 variieren. Mit dem Parameter `<DutyCycle>` kann das Tastverhältnis zwischen 1…99% in Schritten von 1% eingestellt werden. Das Ausgangssignal wird stets am Ausgang von Zähler 2 des jeweiligen Zählerbausteins ausgegeben (OUT_2, OUT_5, usw.). Die Frequenz des Ausgangssignals kann max. 50kHz betragen.

Bei Verwendung der in Abb. 17 gezeigten Beschaltung können Sie mit den Funktionen *me1400CntPWMStart/Stop* die Programmierung stark vereinfachen (siehe Seite 61ff).

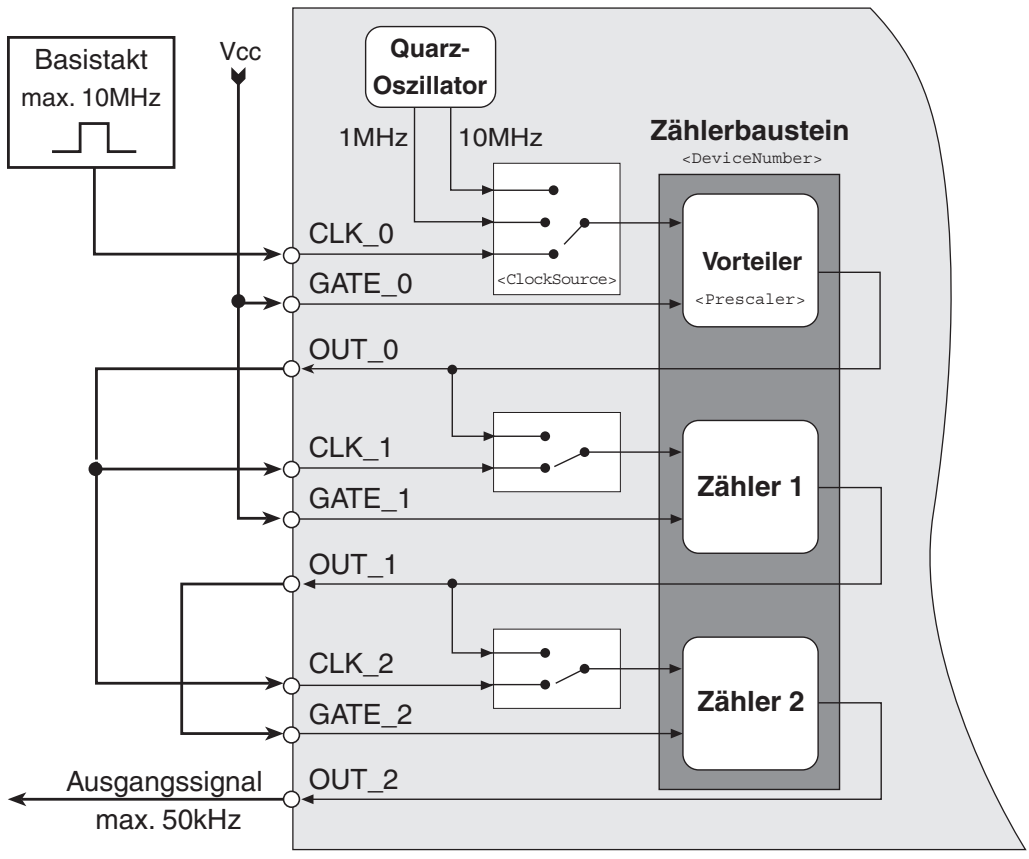


Abb. 17: Beschaltung Pulsweiten-Modulation

4.5 Registerprogrammierung

Die ME-14 ISA-Modelle können, z. B. unter DOS, auch auf Registerebene programmiert werden. Informieren Sie sich bitte gegebenenfalls über die passende Befehlssyntax zur Port-Programmierung im Handbuch der Hochsprache Ihrer Wahl. Wir empfehlen jedoch grundsätzlich die Verwendung der mitgelieferten Treibersoftware. Die Register der Karten sind im Folgenden kurz beschrieben:

4.5.1 Registerbeschreibung

Die Karten belegen, mit der über DIP-Schalter eingestellten Basisadresse beginnend, 16 Bytes im I/O-Adreßraum des PC´s. „BA“ steht für die eingestellte Basisadresse der Karte. In der Spalte „Funktion“ gibt die Klammer an, in welchen Modellen das betreffende Register zur Verfügung steht.

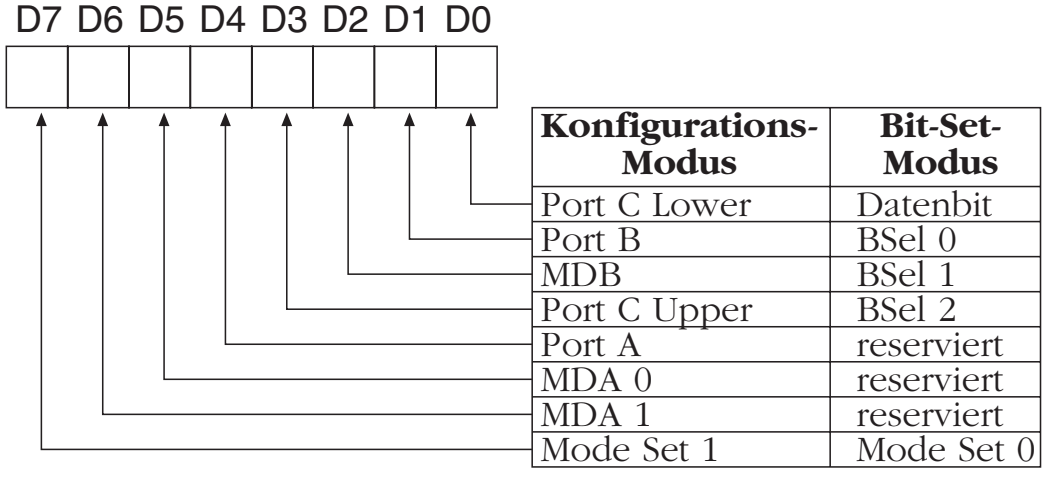
Adr.	Funktion	Adr.	Funktion
BA+0H	Port A des 8255 (alle)	**BA+8H**	Port A des 8255 (B)
BA+1H	Port B des 8255 (alle)	**BA+9H**	Port B des 8255 (B)
BA+2H	Port C des 8255 (alle)	**BA+AH**	Port C des 8255 (B)
BA+3H	Kontrollwort für 8255 (alle)	**BA+BH**	Kontrollwort für 8255 (B)
BA+4H	Daten Zähler 0 (A+B)	**BA+CH**	Daten Zähler 0 (B)
BA+5H	Daten Zähler 1 (A+B)	**BA+DH**	Daten Zähler 1 (B)
BA+6H	Daten Zähler 2 (A+B)	**BA+EH**	Daten Zähler 2 (B)
BA+7H	Kontrollwort für 8254 (A+B)	**BA+FH**	Kontrollwort für 8254 (B)

Tabelle 12: Adreßraum der ME-14

4.5.1.1 Die Register des 82C55

Die Register des 8255 sind nur in den ISA-Modellen unter DOS zugänglich. In allen anderen Fällen verwenden Sie bitte die mitgelieferte Treibersoftware, die die volle Funktionalität gewährleistet.

Für den Anwender sind vier Register des 82C55 zugänglich. Die Daten werden über drei 8 Bit Register (BA+0H … BA+2H, zusätzlich bei B-Modellen: BA+8H … BA+AH) ausgetauscht. Die Konfiguration der Betriebsmodi erfolgt über ein 8 Bit breites Kontrollregister (BA+3H bzw. BA+BH); siehe folgende Abbildung.



Konfigurations-Modus	Bit-Set-Modus
Port C Lower	Datenbit
Port B	BSel 0
MDB	BSel 1
Port C Upper	BSel 2
Port A	reserviert
MDA 0	reserviert
MDA 1	reserviert
Mode Set 1	Mode Set 0

Abb. 18: Kontrollwort des 82C55

Der 82C55 kennt drei grundlegende Betriebsmodi, von denen jedoch nur Modus 0 für die ME-14/1400 relevant ist. Weitergehende Informationen zur Programmierung des 82C55 finden Sie im Datenblatt des Bausteinherstellers.

4.5.1.1.1 Modus 0: Einfache Ein-/Ausgabe

Im Modus 0 können einfache Ein-/Ausgabevorgänge auf allen drei Ports vorgenommen werden. Dabei werden die Daten einfach vom jeweiligen Port gelesen bzw. auf diesen geschrieben; Handshakes sind nicht erforderlich. Durch die Konfiguration des 82C55 für den Modus 0 stehen drei 8 Bit Ports zur Verfügung, die jeweils als Eingabe- (nicht gelatcht) oder Ausgabeport (gelatcht) konfiguriert werden können.

Im Modus 0 sind insgesamt 8 verschiedene Ein-/Ausgabe-Konfigurationen auf der ME-14/1400 möglich (die Bits 7, 6, 5 u. 2 bleiben gleich):

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	x	x	0	x	x

Tabelle 13: Kontrollwort des 8255

Kontrollwort D7…D0	Hex	Port A	Port B	Port C
1 0 0 0 0 0 0 0	$80	Ausgang	Ausgang	Ausgang
1 0 0 0 1 0 0 1	$89	Ausgang	Ausgang	Eingang
1 0 0 0 0 0 1 0	$82	Ausgang	Eingang	Ausgang
1 0 0 0 1 0 1 1	$8B	Ausgang	Eingang	Eingang
1 0 0 1 0 0 0 0	$90	Eingang	Ausgang	Ausgang
1 0 0 1 1 0 0 1	$99	Eingang	Ausgang	Eingang
1 0 0 1 0 0 1 0	$92	Eingang	Eingang	Ausgang
1 0 0 1 1 0 1 1	$9B	Eingang	Eingang	Eingang

Tabelle 14: Mögliche Konfigurationen des 82C55 im Modus 0

4.5.1.2 Die Register des 82C54

Die Register des 8254 sind nur in den ISA-Modellen unter DOS zugänglich. In allen anderen Fällen verwenden Sie bitte die mitgelieferte Treibersoftware, die die volle Funktionalität gewährleistet.

Das Kontrollwort (BA+7H bzw. BA+FH) legt den Betriebsmodus und das Zählsystem (binär oder BCD-Code) des Zählerbausteins fest und kontrolliert das Laden der Zähler-Register.

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Tabelle 15: Format des Kontrollwortes

Mit den Bits SC1/0 wird der Zähler ausgewählt:

SC1	SC0	Funktion
0	0	Zähler 0
0	1	Zähler 1
1	0	Zähler 2
1	1	nicht erlaubt

Tabelle 16: Auswahl des Zählers

Mit den Bits RL1/0 wird der Read-/Write-Modus ausgewählt:

RL1	RL0	Funktion
0	0	Zähler Latching Operation
1	0	nur MSB
0	1	nur LSB
1	1	erst LSB, dann MSB (2 x 8 Bit)

Tabelle 17: Auswahl des Read-/Write-Modus

Mit den Bits M0…M2 wird der Betriebsmodus für die einzelnen Zähler bestimmt:

M2	M1	M0	Funktion
0	0	0	Modus 0
0	0	1	Modus 1
X	1	0	Modus 2
X	1	1	Modus 3
1	0	0	Modus 4
1	0	1	Modus 5

Tabelle 18: Auswahl des Betriebs-Modus

Mit dem Bit BCD wird das Zählsystem jedes einzelnen Zählers festgelegt:

BCD	Funktion
0	16 Bit Binär-Zähler
1	BCD-Zähler (4 Dekaden)

Tabelle 19: Auswahl der Zählart

Zunächst wird der Zähler initialisiert, indem man das Kontrollwort mit Zählernummer, Read/Write Modus, Zählsystem und der Modusnummer ins betreffende Kontrollregister (BA+7H bzw. BA+FH) schreibt. Danach lädt man den Startwert des Zählers in das entsprechende Zählregister (BA+4H…BA+6H, zusätzlich für B-Modelle: BA+CH…BA+EH). Der Zählerstand wird mit jeder negativen Flanke des Clk-Signals um 1 dekrementiert. Im folgenden sind die 6 zur Verfügung stehenden Modi kurz erklärt.

4.5.1.2.1 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 sobald der Zähler initialisiert wird oder ein neuer Startwert in den Zähler geladen wird. Zur Freigabe des Zählers muß am Gate-Eingang High-Pegel liegen. Sobald der Zähler geladen und freigegeben wurde beginnt er abwärts zu zählen während der Ausgang im Low-Zustand bleibt.

Bei Erreichen des Nulldurchganges geht der Ausgang in den High-Zustand 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.

4.5.1.2.2 Modus 1: Retriggerbarer „One-Shot“

Der Zähler-Ausgang (Out 0…2) geht in den High-Zustand sobald der Zähler initialisiert wird. Nachdem ein Startwert in den Zähler geladen wurde geht der Ausgang mit dem auf den ersten Triggerimpuls (positive Flanke des Gate-Signals) folgenden Takt in den Low-Zustand. Nach Ablauf des Zählers geht der Ausgang wieder in den High-Zustand.

Mit einer positiven Flanke am Gate-Eingang kann der Zähler jederzeit auf den Startwert zurückgesetzt („retriggered“) werden. Der Ausgang bleibt solange im Low-Zustand bis der Zähler den Nulldurchgang erreicht.

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

4.5.1.2.3 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. Nach Freigabe des Zählers durch einen High-Pegel am

Gate-Eingang 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. Dieser Ablauf wiederholt sich periodisch solange das Gate-Signal im High-Zustand ist, ansonsten geht der Ausgang sofort in den High-Zustand.

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.

4.5.1.2.4 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. Nach Freigabe des Zählers durch einen High-Pegel am Gate-Eingang wird in 2er-Schritten abwärts gezählt. Nun wechselt der Ausgang, beginnend mit High-Pegel alle Startwert/2 Perioden des Eingangstaktes seinen Zustand. Solange am Gate-Eingang ein High-Pegel anliegt, wiederholt sich dieser Ablauf periodisch, ansonsten geht der Ausgang sofort in den High-Zustand.

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.

4.5.1.2.5 Modus 4: Zählerstart durch Softwaretrigger

Der Zähler-Ausgang (Out 0…2) geht in den High-Zustand sobald der Zähler initialisiert wird. Zur Freigabe des Zählers muß am Gate-Eingang High-Pegel liegen. Sobald der Zähler geladen (Softwartrigger) und freigegeben wurde, beginnt er abwärts zu zählen, während der Ausgang noch im High-Zustand bleibt.

Bei Erreichen des Nulldurchganges geht der Ausgang für die Dauer einer Takt-Periode in den Low-Zustand. Danach geht der Ausgang wieder in den High-Zustand und bleibt dort bis der Zähler neu initialisiert wird und ein neuer Startwert geladen wird.

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

4.5.1.2.6 Modus 5: Zählerstart durch Hardwaretrigger

Der Zähler-Ausgang (Out 0…2) geht in den High-Zustand sobald der Zähler initialisiert wird. Nachdem ein Startwert in den Zähler geladen wurde beginnt der Zählvorgang mit dem auf den ersten Triggerimpuls (positive Flanke des Gate-Signals) folgenden Takt. Bei Erreichen des Nulldurchganges geht der Ausgang für die Dauer einer Takt-Periode in den Low-Zustand. Danach geht der Ausgang wieder in den High-Zustand 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 Flanke am Gate-Eingang kann der Zähler jederzeit auf den Startwert zurückgesetzt („retriggered“) werden. Der Ausgang bleibt solange im High-Zustand bis der Zähler den Nulldurchgang erreicht.

5 Funktionsreferenz

5.1 Allgemeines

Das Treiberkonzept für die ME-14/1400 Familie sieht jeweils eigenständige Treiber für ISA- und PCI/cPCI-Karten vor. Aufgrund des Treiberkonzepts können ISA-Karten grundsätzlich nur mit dem ME-14 Treibersystem angesprochen werden und PCI-Karten nur mit dem ME-1400 Treibersystem. Um bei Bedarf eine Kompatibilität von PCI-Karten zu bereits vorhandener Applikationssoftware zu erreichen sind im PCI-Treiber zusätzlich alle Funktionen des ISA-Treibers (*_me14...*) deklariert, die dann auf analog aufgebaute Funktionen des PCI-Treibers zugreifen (*me1400...*). Wenn Sie von dieser Möglichkeit Gebrauch machen möchten, beachten Sie bitte die genaue Vorgehensweise in den entsprechenden README-Dateien.

Die Funktionen der API-DLL (`ME14_32.DLL`) für die ME-14 werden von folgenden 32 Bit-Treibern unterstützt:

- VxD-Treiber (`ME14_32.VXD`) für Windows 95/98/Me
- Kernel-Treiber (`ME14_32.SYS`) für Windows NT4.0/2000/XP

  desweiteren benötigt: Dialog-DLL (`MEDLG32.DLL`).

Die Funktionen der API-DLL (`ME1400.DLL`) für die ME-1400 werden von folgenden 32 Bit-Treibern unterstützt:

- VxD-Treiber (`ME1400.VXD`) für Windows 95
- Kernel-Treiber (`ME1400.SYS`) für Windows NT4.0
- WDM-Treiber (`ME1400.SYS`) für Windows 98/Me/2000/XP

Nachdem der Treiber erfolgreich geladen wurde, ermöglichen die API-Funktionen einen komfortablen Zugriff auf die Hardware. Jede Funktion, die auf eine Karte der ME-14/1400 Familie zugreifen soll, benötigt zur Identifizierung der Karte einen Integerwert. In der nun folgenden Beschreibung der Funktionen ist dieser Parameter mit `<BoardNumber>` bezeichnet. Er spezifiziert die anzusprechende Karte.

5.2 Nomenklatur

Die API-Funktionen sind kartenspezifisch gehalten. Das Präfix jeder Funktion bezeichnet unmittelbar die Karte(n), für die die Funktion zutrifft. Für die Funktionsnamen wurden weitgehend „selbstredende“ Bezeichner verwendet. Jeder Funktionsname besteht aus aus einem kartentypspezifischen Präfix und mehreren Bestandteilen für die entsprechende Funktionsgruppe (z. B.“DI“ für digitale Eingabe).

_me14... Funktionen nur gültig für die ME-14 ISA

me1400... Funktionen nur gültig für die ME-1400 PCI/cPCI

Für die Funktionsbeschreibung gelten folgende Vereinbarungen:

Funktionsnamen	werden im Fließtext kursiv geschrieben z. B. *me1400GetBoardVersion*
`<Parameter>`	werden in spitzen Klammern in der Schriftart `Courier` geschrieben
<*Variablen*>	als Platzhalter für vordefinierte Konstanten werden kursiv geschrieben und in spitze Klammern gesetzt
[eckige Klammern]	werden für optional verwendbare Variablen verwendet
`DATEINAMEN`	oder `PFADE` werden in Großbuchstaben in der Schriftart `Courier` geschrieben
`me1400...()`	Programmausschnitte sind in der Schriftart `Courier` geschrieben

Zur Kennzeichnung des Datentyps werden folgende Kennbuchstaben verwendet:

i... oder dw...	32-Bit Integer-Wert
s... oder w...	16-Bit Short-Wert
c... oder b...	8-Bit Character-Wert
p...	Zeiger auf Datentyp (i, s oder c)

5.3 Beschreibung der API-Funktionen

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

Funktion	Kurzbeschreibung	Seite
Allgemeine Funktionen		
_me14GetBoardVersion me1400GetBoardVersion	Kartenversion ermitteln	49
_me14GetDLLVersion me1400GetDLLVersion	DLL-Versionsnummer ermitteln	50
_me14GetDriverVersion me1400GetDriverVersion	Treiber-Versionsnummer ermitteln	50
me1400SetMultifunctionPin	Konfiguration von Pin 39 (OSC/ IR_IN) als IRQ-Eingang oder Taktausgang	51
Digitale Ein-/Ausgabe		
_me14DIOSetPortDirection me1400DIOSetPortDirection	Konfiguriert einen Port als Ein- oder Ausgang	52
_me14DIGetBit me1400DIGetBit	Bit einlesen	53
_me14DIGetByte me1400DIGetByte	Byte einlesen	55
_me14DOSetBit me1400DOSetBit	Bit ausgeben	56
_me14DOSetByte me1400DOSetByte	Byte ausgeben	57

Tabelle 20: Übersicht der Bibliotheksfunktionen

Funktion	Kurzbeschreibung	Seite
Zählerfunktionen		
me1400CntInitSrc	Konfiguriert die Zähler 0…29	59
me1400CntPWMStart	PWM-Ausgabe starten	61
me1400CntPWMStop	PWM-Ausgabe stoppen	63
_me14CntRead me1400CntRead	Aktuellen Zählerstand einlesen	64
_me14CntWrite me1400CntWrite	Zähler konfigurieren und starten	65
me1400InitModeTimerA	(nicht für neue Projekte)	66
me1400InitModeTimerB	(nicht für neue Projekte)	68
Interrupt-Handling		
_me14DisableInt me1400DisableInt	Interruptsteuerung deaktivieren	70
_me14EnableInt me1400EnableInt	Interruptsteuerung aktivieren	71
me1400GetIrqCnt	Anzahl der Interrupts ermitteln	72
Fehler-Behandlung		
_me14GetDrvErrMess me1400GetDrvErrMess	Fehlerstring gemäß Fehlercode	73

Tabelle 20: Übersicht der Bibliotheksfunktionen

Hinweis: Stellvertretend für die beiden Präfixe *_me14* (für ISA-Karten) und *me1400* (für PCI-Karten) wird in der folgenden Funktionsbeschreibung das Präfix *me14xx* verwendet.

5.3.1 Allgemeine Funktionen

_me14GetBoardVersion
me1400GetBoardVersion

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Es wird die Kartenversion für eine installierte Karte der Kartenfamilie ME-14/1400 ermittelt.

● Definitionen

C: int me14xxGetBoardVersion (int iBoardNumber, int *piDevices)

Delphi: Function me14xxGetBoardVersion (iBoardNumber: integer; Var iDevices: integer): integer;

Basic: Declare Function me14xxGetBoardVersion Lib "me14xx" Alias "_VBme14xxGetBoardVersion@8" (ByVal iBoardNumber As Long, iDevices As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<Version>

Zeiger auf Integer-Variable, in der die Device-ID zurückgegeben wird. Mögliche Werte sind:

014AHex: ME-14A

014BHex: ME-14B

1400Hex: ME-1400 oder ME-1400E

140AHex: ME-1400A oder ME-1400EA

140BHex: ME-1400B oder ME-1400EB

140CHex: ME-1400C

140DHex: ME-1400D

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

_me14GetDLLVersion
me1400GetDLLVersion

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Ermittelt die Versionsnummer der Karten-DLL für die Karten der ME-14/1400 Kartenfamilie

● Definitionen

C: int me14xxGetDLLVersion();

Delphi: Function me14xxGetDLLVersion: integer;

Basic: Declare Function me14xxGetDLLVersion Lib "me14xx" Alias "_VBme14xxGetDLLVersion@0" () As Long

➔ Parameter

keine

‹ Rückgabewert

Versionsnummer; der 32-Bit-Wert enthält in den höherwertigen 16 Bit die Hauptversion und in den niederwertigen 16 Bit die Unterversion. Beispiel: 00020001Hex ergibt die Version 2.01 (muß immer hexadezimal interpretiert werden)

me14GetDriverVersion
me1400GetDriverVersion

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Ermittelt die Versionsnummer des Treibers für die ME-14/1400.

● Definitionen

C: int me14xxGetDriverVersion(char *pBuffer);

Delphi: Function me14xxGetDriverVersion (Var pBuffer: string): integer;

Basic: Declare Function me14xxGetDriverVersion Lib „me14xx“ Alias "_VBme14xxGetDriverVersion@4" (ByVal pBuffer As String) As Long

➔ Parameter

<Buffer>

Zeiger auf den String der Treiberversion.

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann über *me14xxGetDrvErrMess* ermittelt werden.

me1400SetMultifunctionPin

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	–	✔ (nicht ME-1400/E)	–

Diese Funktion konfiguriert den Eingang „OSC/IR_IN“ (Pin 39) wahlweise als Interrupt-Eingang oder als Taktausgang.

● Definitionen

C: int me1400SetMultifunctionPin (int iBoardNumber, int iMultiPin)

Delphi: Function me1400SetMultifunctionPin (iBoardNumber: integer; MultiPin: integer): integer;

Basic: Declare Function me1400SetMultifunctionPin Lib "me1400" Alias "_VBme1400SetMultifunctionPin@8" (ByVal iBoardNumber As Long, ByVal MultiPin As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-1400 (0…31)

<MultiPin>

Multifunktions-Pin (OSC/IR_IN) konfigurieren:

- ME1400_MULTIPIN_IRQ (00Hex)
  Leitung wird als Interrupt-Eingang verwendet (IR_IN) (Default)
- ME1400_MULTIPIN_INTERNALCLOCK (01Hex)
  Leitung dient zur Ausgabe des internen Oszillatortaktes (OSC)

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

5.3.2 Digital-I/O-Funktionen

_me14DIOSetPortDirection
me1400DIOSetPortDirection

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Portweise Konfigurierung der Digital-Ports als Ein- oder Ausgang.

☞ Wichtiger Hinweis:

Diese Funktion muß vor allen Bit-/Byte-Lese-/Schreib-Operationen für jeden betreffenden Port vorher einmalig aufgerufen werden.

● Definitionen

C: int me14xxDIOSetPortDirection (int iBoardNumber, int iPortNo, int iDir);

Delphi: Function me14xxDIOSetPortDirection (iBoardNumber, iPortNo, iDir: integer): integer;

Basic: Declare Function me14xxDIOSetPortDirection Lib "me14xx" Alias "_VBme14xxDIOSetPortDirection@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iDir As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<PortNo>

Port-Name; möglich sind:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (nur ME-14A/B)
- PORTCH (04Hex) Port C High (nur ME-14A/B)
- PORTD (08Hex) Port D (nur B/D-Vers.)
- PORTE (09Hex) Port E (nur B/D-Vers.)
- PORTF (0AHex) Port F (nur B/D-Vers.)
- PORTFL (0BHex) Port F Low (nur ME-14B)
- PORTFH (0CHex) Port F High (nur ME-14B)

<Dir>

Richtung des Ports; möglich sind:

- MEINPUT (00Hex) Eingangs-Port
- MEOUTPUT (01Hex) Ausgangs-Port

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

_me14DIGetBit
me1400DIGetBit

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Liefert den Zustand einer einzelnen Eingangsleitung zurück.

☞ Wichtiger Hinweis:

Zur Richtungsbestimmung muß vorher für den betreffenden Port die Funktion *…DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int _e14xxDIGetBit (int iBoardNumber, int iPortNo, int iBitNo, int *piBitValue);

Delphi: Function _e14xxDIGetBit (iBoardNumber, iPortNo, iBitNo: integer; Var iBitValue: integer): integer;

Basic: Declare Function me14xxDIGetBit Lib "me14xx" Alias "_VBme14xxDIGetBit@16" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iBitNo As Long, ByRef iBitValue As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<PortNo>

Port-Name; möglich sind:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (nur ME-14A/B)
- PORTCH (04Hex) Port C High (nur ME-14A/B)
- PORTD (08Hex) Port D (nur B/D-Vers.)
- PORTE (09Hex) Port E (nur B/D-Vers.)
- PORTF (0AHex) Port F (nur B/D-Vers.)
- PORTFL (0BHex) Port F Low (nur ME-14B)
- PORTFH (0CHex) Port F High (nur ME-14B)

<BitNo>

Nummer der Eingangsleitung, die abgefragt werden soll; mögliche Werte sind: 0…7 entsprechend den Bits 0…7 eines Ports

<BitValue>

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

0: Leitung auf logisch '0' gesetzt

1: Leitung auf logisch '1' gesetzt

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

_me14DIGetByte
me1400DIGetByte

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Liest ein Byte (8 Bits) von einem Eingang.

☞ Wichtiger Hinweis:

Zur Richtungsbestimmung muß vorher für den betreffenden Port die Funktion *…DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me14xxDIGetByte (int iBoardNumber, int iPortNo, int *piValue);

Delphi: Function me14xxDIGetByte (iBoardNumber, iPortNo: integer; Var iValue: integer): integer;

Basic: Declare Function me14xxDIGetByte Lib "me14xx" Alias "_VBme14xxDIGetByte@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, iValue As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<PortNo>

Port-Name; möglich sind:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (nur ME-14A/B)
- PORTCH (04Hex) Port C High (nur ME-14A/B)
- PORTD (08Hex) Port D (nur B/D-Vers.)
- PORTE (09Hex) Port E (nur B/D-Vers.)
- PORTF (0AHex) Port F (nur B/D-Vers.)
- PORTFL (0BHex) Port F Low (nur ME-14B)
- PORTFH (0CHex) Port F High (nur ME-14B)

<Value>

Zeiger auf einen Integerwert, der das gelesene Byte enthält; nur die niederwertigen 8 Bits signifikant.

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

_me14DOSetBit
me1400DOSetBit

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Setzt eine einzelne Ausgangsleitung in den gewünschten Zustand.

☞ Wichtiger Hinweis:

Zur Richtungsbestimmung muß vorher für den betreffenden Port die Funktion *…DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me14xxDOSetBit (int iBoardNumber, int iPortNo, int iBitNo, int iBitValue);

Delphi: function me14xxDOSetBit (iBoardNumber, iPortNo, iBitNo, iBitValue: integer): integer;

Basic: Declare Function me14xxDOSetBit Lib "me14xx" Alias "_VBme14xxDOSetBit@16" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iBitNo As Long, ByVal iBitValue As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<PortNo>

Port-Name; möglich sind:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (nur ME-14A/B)
- PORTCH (04Hex) Port C High (nur ME-14A/B)
- PORTD (08Hex) Port D (nur B/D-Vers.)
- PORTE (09Hex) Port E (nur B/D-Vers.)
- PORTF (0AHex) Port F (nur B/D-Vers.)
- PORTFL (0BHex) Port F Low (nur ME-14B)
- PORTFH (0CHex) Port F High (nur ME-14B)

<BitNo>

Nummer der Ausgangsleitung, die gesetzt werden soll; mögliche Werte sind: 0…7 entsprechend den Bitnummern des Ports

<BitValue>

Mögliche Werte sind:

0: Bit wird auf logisch '0' gesetzt

1: Bit wird auf logisch '1' gesetzt

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

_me14DOSetByte
me1400DOSetByte

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Schreibt ein Byte an den spezifizierten Ausgangs-Port.

☞ Wichtiger Hinweis:

Zur Richtungsbestimmung muß vorher für den betreffenden Port die Funktion *…DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me14xxDOSetByte (int iBoardNumber, int iPortNo, int iValue);

Delphi: function me14xxDOSetByte (iBoardNumber, iPortNo, iValue: integer): integer;

Basic: Declare Function me14xxDOSetByte Lib "me14xx" Alias "_VBme14xxDOSetByte@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iValue As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<PortNo>

Port-Name; möglich sind:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (nur ME-14A/B)
- PORTCH (04Hex) Port C High (nur ME-14A/B)
- PORTD (08Hex) Port D (nur B/D-Vers.)
- PORTE (09Hex) Port E (nur B/D-Vers.)
- PORTF (0AHex) Port F (nur B/D-Vers.)
- PORTFL (0BHex) Port F Low (nur ME-14B)
- PORTFH (0CHex) Port F High (nur ME-14B)

<Value>

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

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

5.3.3 Zählerfunktionen

me1400CntInitSrc

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	–	✔ (nicht ME-1400/E)	✔

Diese Funktion bestimmt die Taktquelle für den spezifizierten Zähler. Grundsätzlich stehen 4 Möglichkeiten zur Verfügung: externer Takt von der Sub-D-Buchse, interner Oszillator mit 1 MHz bzw. 10 MHz oder Ausgangstakt vom vorhergehenden Zähler. Es gelten dabei folgende Einschränkungen: Zähler 0 und Zähler 15 können nicht vom vorhergehenden Zähler gespeist werden und der interne Takt kann nur jeweils dem ersten Zähler eines Bausteins zugeführt werden.

● Definitionen

C: int me1400CntInitSrc (int iBoardNumber, int iCounter, int iCounterSource);

Delphi: Function me1400CntInitSrc (iBoardNumber: integer; Counter: integer; CounterSource: integer): integer;

Basic: Declare Function me1400CntInitSrc Lib "me1400" Alias "_VBme1400CntInitSrc@12" (ByVal iBoardNumber As Long, ByVal Counter As Long, ByVal CounterSource As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-1400 (0…31)

<Counter>

Zähler, dessen Zählerstand eingelesen werden soll, mögliche Werte sind:

- ME-1400A/EA (3 Zähler): 0…2
- ME-1400B/EB (6 Zähler): 0…5
- ME-1400C (15 Zähler): 0…14
- ME-1400D (30 Zähler): 0…29

<CounterSource>

Quelle des Taktsignals für spezifizierten Zähler:

- COUNTER_SOURCE_SUBD (00Hex)
  Externer Takt von Sub-D-Buchse (Default)
  (für alle Zähler möglich)
- COUNTER_SOURCE_1MHZ (01Hex)
  Interner 1 MHz Takt (kombinierbar mit Zähler 0, 3, 6, 9, 12, 15, 18, 21, 24 oder 27)
- COUNTER_SOURCE_10MHZ (02Hex)
  Interner 10 MHz Takt (kombinierbar mit Zähler 0, 3, 6, 9, 12, 15, 18, 21, 24 oder 27)
- COUNTER_SOURCE_PREV (03Hex)
  Takt vom vorhergehenden Zähler (nicht kombinierbar mit Zähler 0 aller Modelle, mit Zähler 3 der ME-1400B/EB und Zähler 15 der ME-1400D)

☞ Beispiel

Zähler 0 soll intern mit 1 MHz gespeist werden und mit Zähler 1 kaskadiert werden. Zähler 2 soll von der Sub-D-Buchse gespeist werden und unabhängig zählen.

```
iErrorCode =
   me1400CntInitSrc(BoardNumber, 0, COUNTER_SOURCE_1MHZ);

iErrorCode =
   me1400CntInitSrc(BoardNumber, 1, COUNTER_SOURCE_PREV);

iErrorCode =
   me1400CntInitSrc(BoardNumber, 2, COUNTER_SOURCE_SUBD);

...
```

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1400GetDrvErrMess* ermittelt werden.

me1400CntPWMStart

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	–	✔ (nicht ME-1400/E)	✔

Diese Funktion konfiguriert die 3 Zähler eines Zählerbausteins (8254) für die Betriebsart „Pulsweiten-Modulation“ (PWM) und startet die Ausgabe. Eine vorausgehende Programmierung dieser Zähler wird überschrieben. Das Ausgangssignal wird stets am Ausgang von Zähler 2 des jeweiligen Zählerbausteins ausgegeben (OUT_2, OUT_5, usw.). Ein Basistakt (max. 10 MHz) kann von außen zugeführt werden. Alternativ können Sie auch den internen Quarz-Oszillator mit 1 MHz oder 10 MHz verwenden. Zähler 0 kann als Vorteiler für das PWM-Signal verwendet werden (siehe Abb. 17 auf Seite 35). Die Frequenz des Ausgangssignals kann max. 50 kHz betragen. Das Tastverhältnis kann zwischen 1…99% in Schritten von 1% eingestellt werden (siehe Abb. 16 auf Seite 34). Die Ausgabe wird unmittelbar durch Aufruf der Funktion *…CntPWMStart* gestartet und mit *…CntPWMStop* beendet. Es ist keine weitere Programmierung der Zähler erforderlich.

☞ Hinweis!

Die Verwendung dieser Funktion ist nur sinnvoll in Verbindung mit der in Abb. 17 auf Seite 35 gezeigten externen Beschaltung.

● Definitionen

VC: me1400CntPWMStart(int iBoardNumber, int iDeviceNumber, int iClockSource, int iPrescaler, int iDutyCycle);

Delphi: Function me1400CntPWMStart(iBoardNumber: integer; iDeviceNumber: integer; iClockSource: integer; iPrescaler: integer; iDutyCycle: integer): integer;

Basic: Declare Function me1400CntPWMStart Lib "me1400" Alias "_VBme1400CntPWMStart@20" (ByVal iBoardNumber As Long, ByVal DeviceNumber As Long, ByVal ClockSource As Long, ByVal Prescaler As Long, ByVal DutyCycle As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-1400 (0…31)

<DeviceNumber>

Zählerbaustein, dessen Zähler für PWM-Betrieb konfiguriert werden soll, mögliche Werte je nach Kartentyp:

- COUNTER_DEVICE_A Zählerbaustein A
- COUNTER_DEVICE_B Zählerbaustein B
- COUNTER_DEVICE_C Zählerbaustein C
- COUNTER_DEVICE_D Zählerbaustein D
- COUNTER_DEVICE_E Zählerbaustein E
- COUNTER_DEVICE_F Zählerbaustein F
- COUNTER_DEVICE_G Zählerbaustein G
- COUNTER_DEVICE_H Zählerbaustein H
- COUNTER_DEVICE_I Zählerbaustein I
- COUNTER_DEVICE_J Zählerbaustein J

<ClockSource>

Quelle des Taktsignals für spezifizierten Zählerbaustein:

- COUNTER_SOURCE_SUBD (00Hex)
  Externer Takt von Sub-D-Buchse
  (für alle Zähler möglich)
- COUNTER_SOURCE_1MHZ (01Hex)
  Interner 1 MHz Takt
- COUNTER_SOURCE_10MHZ (02Hex)
  Interner 10 MHz Takt

<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 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1400GetDrvErrMess* ermittelt werden.

me1400CntPWMStop

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	–	✔ (nicht ME-1400/E)	✔

Mit dieser Funktion beenden Sie den PWM-Betrieb.

● Definitionen

VC: me1400CntPWMStop(int iBoardNumber, int iDeviceNumber);

Delphi: Function me1400CntPWMStop(iBoardNumber: integer; iDeviceNumber: integer): integer;

Basic: Declare Function me1400CntPWMStop Lib "me1400" Alias "_VBme1400CntPWMStop@8" (ByVal iBoardNumber As Long, ByVal DeviceNumber As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14xx (0…31)

<DeviceNumber>

Zählerbaustein, dessen PWM-Betrieb gestoppt werden soll:

- COUNTER_DEVICE_A Zählerbaustein A
- COUNTER_DEVICE_B Zählerbaustein B
- COUNTER_DEVICE_C Zählerbaustein C
- COUNTER_DEVICE_D Zählerbaustein D
- COUNTER_DEVICE_E Zählerbaustein E
- COUNTER_DEVICE_F Zählerbaustein F
- COUNTER_DEVICE_G Zählerbaustein G
- COUNTER_DEVICE_H Zählerbaustein H
- COUNTER_DEVICE_I Zählerbaustein I
- COUNTER_DEVICE_J Zählerbaustein J

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1400GetDrvErrMess* ermittelt werden.

_me14CntRead
me1400CntRead

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔ (nicht ME-1400/E)	✔

Puffert bei Funktionsaufruf den aktuellen Zählerstand eines Zählers und liest den Wert ein.

● Definitionen

C: int me14xxCntRead(int iBoardNumber, int iCounterNo, int *piValue);

Delphi: Function me14xxCntRead(iBoardNumber, iCounterNo: integer; Var iValue: integer): integer;

Basic: Declare Function me14xxCntRead Lib "me14xx" Alias "_VBme14xxCntRead@12" (ByVal iBoardNumber As Long, ByVal iCounterNo As Long, iValue As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<Counter>

Zähler, dessen Zählerstand eingelesen werden soll:

- ME-14/1400A/EA (3 Zähler): 0…2
- ME-14/1400B/EB (6 Zähler): 0…5
- ME-1400C (15 Zähler): 0…14
- ME-1400D (30 Zähler): 0…29

<Value>

16-Bit Wert vom spezifizierten Zähler; es sind nur die niederwertigen 16 Bits signifikant.

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

_me14CntWrite
me1400CntWrite

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔ (nicht ME-1400/E)	✔

Konfiguriert den spezifizierten Zähler in der gewünschten Betriebsart, lädt den Startwert und startet den Zählvorgang unmittelbar durch Aufruf dieser Funktion. Eine detaillierte Beschreibung der Betriebsarten finden Sie auf Seite 41ff.

● Definitionen

C: int me14xxCntWrite (int iBoardNumber, int iCounterNo, int iMode, int iValue);

Delphi: Function me14xxCntWrite(iBoardNumber, iCounterNo, iMode, iValue: integer): integer;

Basic: Declare Function me14xxCntWrite Lib "me14xx" Alias "_VBme14xxCntWrite@16" (ByVal iBoardNumber As Long, ByVal iCounterNo As Long, ByVal iMode As Long, ByVal iValue As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<CounterNo>

Zähler, der konfiguriert und beschrieben werden soll, mögliche Werte sind:

- ME-14/1400A/EA (3 Zähler): 0…2
- ME-14/1400B/EB (6 Zähler): 0…5
- ME-1400C (15 Zähler): 0…14
- ME-1400D (30 Zähler): 0…29

<Mode>

Betriebsart des Zählers, mögliche Werte sind:

- Modus 0 (00Hex), "Zustandsänderung bei Nulldurchgang"
- Modus 1 (01Hex), "Retriggerbarer One-Shot"
- Modus 2 (02Hex), "Asymmetrischer Teiler"
- Modus 3 (03Hex), "Symmetrischer Teiler"
- Modus 4 (04Hex), "Start durch Softwaretrigger"
- Modus 5 (05Hex), "Start durch Hardwaretrigger"

<Value>

16-Bit Ladewert für spezifizierten Zähler; mögliche Werte sind: 0…65535 (0000Hex…FFFFHex)

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden

me1400InitModeTimerA

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	–	(nur ME-1400A/B)	–

Diese Funktion sollte für Neuentwicklungen nicht mehr verwendet werden und wird nur aus Gründen der Abwärtskompatibilität erwähnt (ersetzt durch *me1400CntInitSrc*).

Funktion bestimmt die Taktquelle für die Zähler 0…2 der ME-1400A und ME-1400B.

● Definitionen

C: int me1400InitModeTimerA (int iBoardNumber, int iCtrlWordA);

Delphi: Function me1400InitModeTimerA (iBoardNumber: integer; CtrlWordA: integer): integer;

Basic: Declare Function me1400InitModeTimerA Lib "me1400" Alias "_VBme1400InitModeTimerA@8" (ByVal iBoardNumber As Long, ByVal CtrlWordA As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-1400 (0…31)

<CtrlWordA>

Steuerwort für Zähler A wird aus vier Konstanten gebildet, die durch die Bitoperation „ODER“ verknüpft werden müssen:

Frequenz des internen Oszillators

- ME1400_TIMERINTERNCLOCK_1MHZ (00Hex)
  Interner Oszillator mit 1 MHz (Default)
- ME1400_TIMERINTERNCLOCK_10MHZ (08Hex)
  Interner Oszillator mit 10 MHz

„ODER“

Taktquelle Zähler 0:

- ME1400_TIMERCLOCKSOURCE0_SUBD (00Hex)
  Ext. Takt von Sub-D Buchse (Default)
- ME1400_TIMERCLOCKSOURCE0_INTERN (04Hex)
  Takt von int. Oszillator

„ODER“

Taktquelle Zähler 1:

- ME1400_TIMERCLOCKSOURCE1_SUBD (00Hex)
  Ext. Takt von Sub-D Buchse (Default)
- ME1400_TIMERCLOCKSOURCE1_OUT0 (02Hex)
  Takt von Ausgang Zähler 0

„ODER“

Taktquelle Zähler 2

- ME1400_TIMERCLOCKSOURCE2_SUBD (00Hex)
  Ext. Takt von Sub-D Buchse (Default)
- ME1400_TIMERCLOCKSOURCE2_OUT1 (01Hex)
  Takt von Ausgang Zähler 1

☞ Beispiel

Zähler 0 soll von internem 1 MHz Takt gespeist werden und alle drei Zähler von Baustein A sollen kaskadiert also „in Reihe“ geschaltet werden:

```
iErrorCode = me1400InitModeTimerA(iBoardNumber,
    ME1400_TIMERINTERNCLOCK_1MHZ |
    ME1400_TIMERCLOCKSOURCE0_INTERN |
    ME1400_TIMERCLOCKSOURCE1_OUT0 |
    ME1400_TIMERCLOCKSOURCE2_OUT1);
```

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

me1400InitModeTimerB

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	–	(nur ME-1400B)	–

Diese Funktion sollte für Neuentwicklungen nicht mehr verwendet werden und wird nur aus Gründen der Abwärtskompatibilität erwähnt (ersetzt durch *me1400CntInitSrc*).

Funktion bestimmt die Taktquelle für die Zähler 3…5 der ME-1400B.

● Definitionen

C: int me1400InitModeTimerB (int iBoardNumber, int iCtrlWordB);

Delphi: Function me1400InitModeTimerB (iBoardNumber: integer; CtrlWordB: integer): integer;

Basic: Declare Function me1400InitModeTimerB Lib "me1400" Alias "_VBme1400InitModeTimerB@8" (ByVal iBoardNumber As Long, ByVal CtrlWordB As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-1400 (0…31)

<CtrlWordB>

Steuerwort für Zähler B wird aus vier Konstanten gebildet, die durch die Bitoperation „ODER“ verknüpft werden müssen:

Frequenz des internen Oszillators

- ME1400_TIMERINTERNCLOCK_1MHZ (00Hex)
  Interner Oszillator mit 1 MHz (Default)
- ME1400_TIMERINTERNCLOCK_10MHZ (08Hex)
  Interner Oszillator mit 10 MHz

„ODER“

Taktquelle Zähler 3:

- ME1400_TIMERCLOCKSOURCE0_SUBD (00Hex)
  Ext. Takt von Sub-D Buchse (Default)
- ME1400_TIMERCLOCKSOURCE0_INTERN (04Hex)
  Takt von int. Oszillator

„ODER“

Taktquelle Zähler 4:

- ME1400_TIMERCLOCKSOURCE1_SUBD (00Hex)
  Ext. Takt von Sub-D Buchse (Default)
- ME1400_TIMERCLOCKSOURCE1_OUT0 (02Hex)
  Takt von Ausgang Zähler 3

„ODER“

Taktquelle Zähler 5

- ME1400_TIMERCLOCKSOURCE2_SUBD (00Hex)
  Ext. Takt von Sub-D Buchse (Default)
- ME1400_TIMERCLOCKSOURCE2_OUT1 (01Hex)
  Takt von Ausgang Zähler 4

☞ Beispiel

Zähler 3 soll von internem 1 MHz Takt gespeist werden und alle drei Zähler von Baustein B sollen kaskadiert also „in Reihe“ geschaltet werden:

```
iErrorCode = _me1400InitModeTimerB(iBoardNumber,
   ME1400_TIMERINTERNCLOCK_1MHZ |
   ME1400_TIMERCLOCKSOURCE0_INTERN |
   ME1400_TIMERCLOCKSOURCE1_OUT0 |
   ME1400_TIMERCLOCKSOURCE2_OUT1);
```

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

5.3.4 Interrupt-Handling

_me14DisableInt
me1400DisableInt

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔ (nicht ME-1400/E)	✔

Diese Funktion deaktiviert den externen Interrupteingang (OSC/ IR_IN). Aufruf macht nur Sinn wenn zuvor …*EnableInt* aufgerufen wurde.

☞ Wichtiger Hinweis:

Verwendung dieser Funktion in Agilent VEE nur in Verbindung mit *me1400GetIrqCnt* möglich!

● Definitionen

C: int me14xxDisableInt (int iBoardNumber, int iServiceNo);

Delphi: Function me14xxDisableInt (iBoardNumber, iServiceNo: integer): integer;

Basic: nicht realisiert

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<ServiceNo>

Interruptkanal; hier muß „1“ übergeben werden.

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

_me14EnableInt
me1400EnableInt

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔ (nicht ME-1400/E)	✔

Diese Funktion aktiviert den externen Interrupteingang (OSC/ IR_IN). Bei auftretendem Interrupt wird automatisch eine vom Anwender definierte Interruptroutine ausgeführt. Zu jedem Aufruf von *…EnableInt* muß am Programmende ein Aufruf von *…DisableInt* folgen. Wird diese Funktion in Verbindung mit *me1400GetIrqCnt* benutzt, so wird anstatt der Interruptfunktion ein Null-Pointer übergeben.

☞ Wichtiger Hinweis:

Verwendung dieser Funktion in Agilent VEE nur in Verbindung mit *me1400GetIrqCnt* möglich!

● Definitionen

C: int me14xxEnableInt (int iBoardNumber, pSERVICE_PROC IrqFunc, int ServiceNo)

Delphi: Function me14xxEnableInt (iBoardNumber: integer; IrqFunc: Pointer; iServiceNo: integer): integer;

Basic: nicht realisiert

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-14 (0…3) bzw. ME-1400 (0…31)

<IrqFunc>

Adresse einer anwenderdefinierten Funktion vom Typ (void SERVICE_PROC (void)) für C bzw. vom Typ Pointer für Delphi, die bei einem aufgetretenen Interrupt ausgeführt wird.

<ServiceNo>

Interruptkanal; hier muß „1“ übergeben werden.

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden.

me1400GetIrqCnt

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	–	✔ (nicht ME-1400/E)	✔

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

● Definitionen

C: int me1400GetIrqCnt (int iBoardNumber, int* piIrqCnt);

Delphi: Function me1400GetIrqCnt (iBoardNumber: integer; Var iIrqCnt: integer): integer;

Basic: Declare Function me1400GetIrqCnt Lib "me1400" Alias "_VBme1400GetIrqCnt@8" (ByVal iBoardNumber As Long, ByRef iIrqCnt As Long) As Long

➔ Parameter

<BoardNumber>

Nummer der anzusprechenden Karte vom Typ ME-1400 (0…31)

<IrqCnt>

Anzahl der seit dem Gerätestart eingetroffenen Interrupts.

☞ Beispiel

```
iErrorCode = me1400SetMultifunctionPin(0, ME1400_MULTIPIN_IRQ);
iErrorCode = me1400EnableInt(0, 0, 1);
iErrorCode = me1400GetIrqCnt(0, &iIrqCntBefore);

Sleep(1000);           //auf Interrupts warten

iErrorCode = me1400GetIrqCnt(0, &iIrqCntAfter);
iErrorCode = me1400DisableInt(0, 1);
IrqCnt = (iIrqCntAfter-iIrqCntBefore);
```

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me14xxGetDrvErrMess* ermittelt werden

5.3.5 Fehlerbehandlung

_me14GetDrvErrMess
me1400GetDrvErrMess

✎ Beschreibung

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
verfügbar:	✔	✔	✔

Falls bei der unmittelbar vorher aufgerufenen API-Funktion des ME-14/1400 Treibers ein Fehler aufgetreten ist, liefert diese Funktion den entsprechenden Fehlercode mit Fehlertext zurück.

☞ Wichtiger Hinweis!

Diese Funktion darf nur aufgerufen werden, wenn die unmittelbar vorher aufgerufene API-Funktion der `ME14_32.DLL` oder `ME1400.DLL` fehlerhaft (d. h. Funktionswert 0) ausgeführt wurde!

● Definitionen

C: int me14xxGetDrvErrMess (char *pcErrortext);

Delphi: Function me14xxGetDrvErrMess (Var errortext: errorstring): integer;

Basic: Declare Function me14xxGetDrvErrMess Lib "me14xx" Alias "_VBme14xxGetDrvErrMess@4" (ByVal errortext As String) As Long

➔ Parameter

`<Errortext>`

Zeiger auf Fehlertext; der Fehlercode wird als Funktionswert zurückgegeben.

❮ Rückgabewert

0, falls kein Fehler aufgetreten ist oder Fehlercode entsprechend aufgetretenem Fehler

Anhang

A Spezifikationen

PCI/cPCI Interface (ME-1400/A/B/C/D/E/EA/EB)

Bus-System	Standard PCI (32 Bit, 33 MHz);
(je nach Modell)	CompactPCI (32 Bit, 33 MHz)
Plug&Play-Funktionalität	Automatische Ressourcen-Zuweisung

ISA Interface (ME-14A/B)

Bus-System	ISA-Bus (8 Bit)
Basisadresse	0Hex…3F0Hex in Schritten von 10Hex einstellbar (DIP-Schalter)
Adreßraum	8 Byte (ME-14A), 16 Byte (ME-14B)
Interruptkanal	IRQ2…7 wählbar (per Jumper)
Wait-States	Ein/Aus (per Jumper)

Digitale Ein-/Ausgänge

Anzahl	ME-14A, ME-1400/A/C/E/EA: 24, TTL-kompatibel; ME-14B, ME-1400B/D/EB: 48, TTL-kompatibel
Typ	82(C)55 im Modus 0
Eingangsspannung	Low: -0,5 V…+0,8 V (I_{ILmax} =±10 µA) High: +2,0 V…+5,5 V (I_{IHmax} =±10 µA)
Ausgangsspannung	Low: max. +0,45 V (I_{OL}=+2,5 mA) High: min. +2,4 V (I_{OH}=-2,5 mA)

Zähler

Anzahl	ME-14A, ME-1400A/EA: 3 unabhängig ME-14B, ME-1400B/EB: 6 unabhängig ME-1400C: 15 unabhängig ME-1400D: 15 zusätzlich zur ME-1400C
Typ	82(C)54
Auflösung	16 Bit
Eingangsspannung	Low: -0,5 V…+0,8 V (I_{ILmax}=±10 µA) High: +2,2 V…+6 V (I_{IHmax}=±10 µA)
Ausgangsspannung	Low: max. +0,45 V (I_{OL}=+2,5 mA) High: min. +2,4 V (I_{OH}=-2,5 mA)

Quarzoszillator

Frequenz	1 MHz oder 10 MHz wählbar (ISA: per Jumper; PCI/cPCI: per Software)
Genauigkeit	±100 ppm (±0,01%)
Ausgangspegel	LS-TTL

Allgemeine Daten

Stromverbrauch bei +5 V (ohne Last)	ME-14A: typ. 400 mA ME-14B: typ. 450 mA ME-1400: typ. 200 mA ME-1400A: typ. 220 mA ME-1400B: typ. 400 mA ME-1400C: typ. 1 A ME-1400D: typ. 800 mA ME-1400E: typ. 200 mA ME-1400EA: typ. 220 mA ME-1400EB: typ. 400 mA
VCC-Belastbarkeit an der Sub-D-Buchse	ME-14 ISA: 50 mA ME-1400 PCI/cPCI: 200 mA
Kartenabmessungen (ohne Slotblech und Stecker)	ME-14A: 140 x 106 mm ME-14B: 166 x 106 mm ME-1400/A/B: 132 x 99 mm ME-1400C/D: 129 x 99 mm ME-1400E/EA/EB: 175 x 99 mm cPCI Modelle: 100 x 160 mm
Anschlüsse	**ME-14A/B, ME-1400E/EA/EB:** 37polige Sub-D-Buchse am Slotblech der Karte **zusätzlich ME-14B, ME-1400EB:** 40poliger Stiftstecker für Adapter auf 37polige Sub-D-Buchse an zusätzlichem Slotblech montiert (Belegung wie Buchse am Slotblech der Karte) **ME-1400/A/B/C/D:** 78polige Sub-D-Buchse am Slotblech der Karte
Betriebstemperatur	0…70˚C
Lagertemperatur	0…50˚C
Luftfeuchtigkeit	20…55% (nicht kondensierend)

CE-Zertifizierung

EMV-Direktive	89/336/EMC
Emission	EN 55022
Störfestigkeit	EN 50082-2

B Anschlußbelegungen

B1 ME-1400/A/B

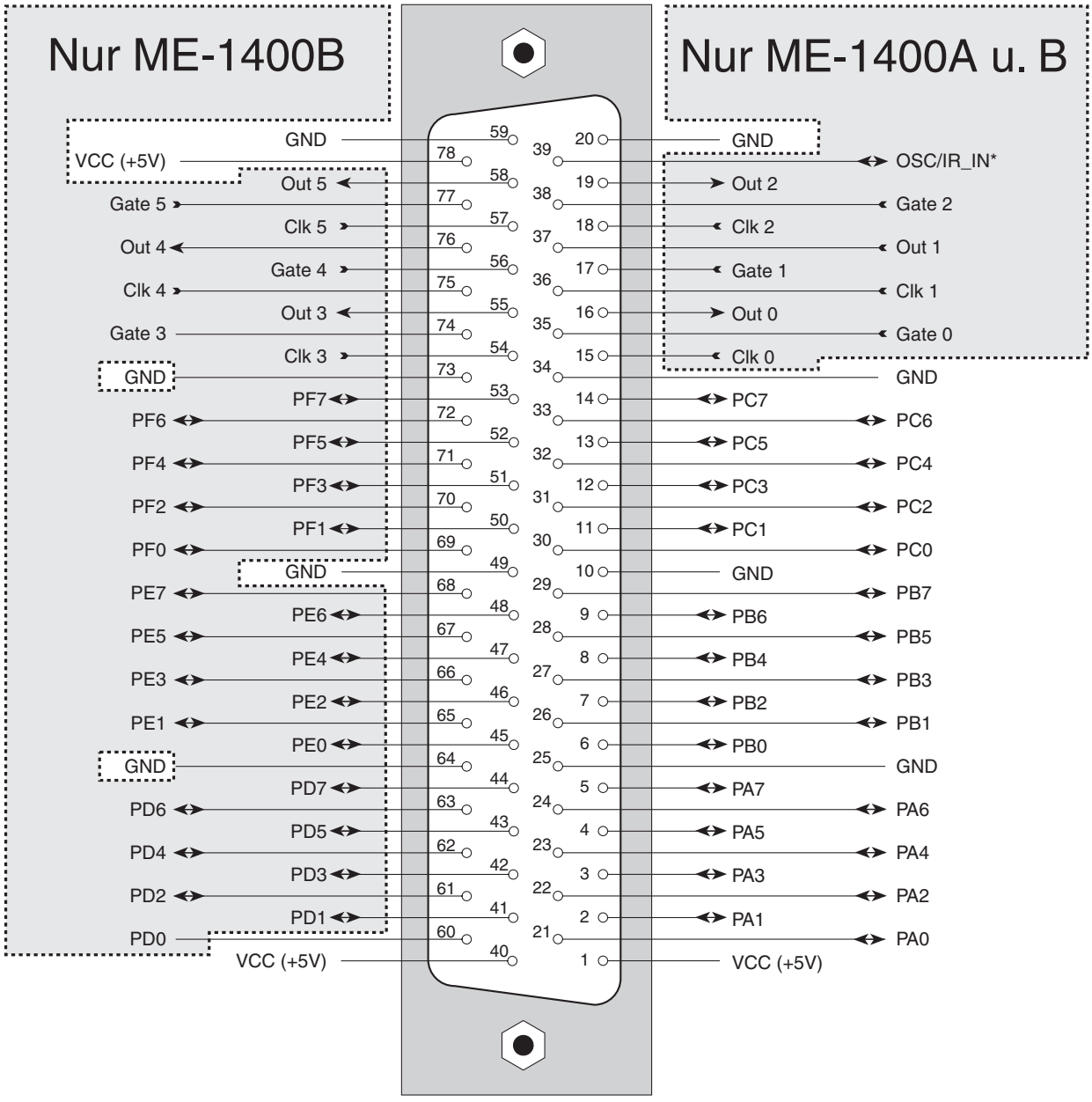


Abb. 19: 78polige Sub-D-Buchse ME-1400/A/B

* Funktionalität steht nur auf der ME-1400A/B zur Verfügung.

B2 ME-1400C/D

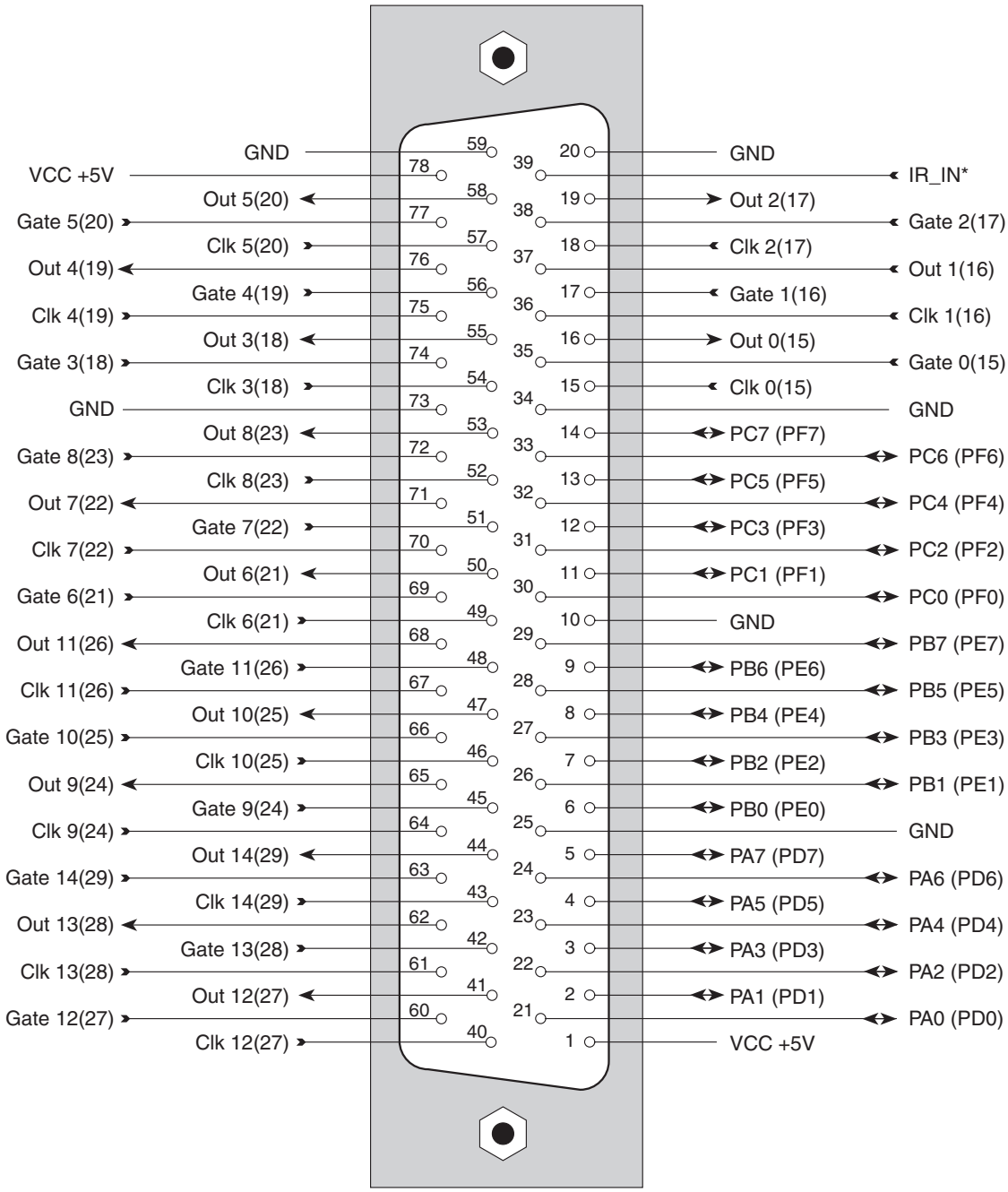


Abb. 20: 78polige Sub-D-Buchse ME-1400C/D

Hinweis:
Die Digital-Ports D, E und F (in Klammer) sind nur in Verbindung mit der Erweiterungskarte ME-1400D verfügbar.

* Der Interrupt-Eingang „IR_IN“ ist nur auf der ME-1400C verfügbar. Auf der ME-1400D **bitte nicht beschalten!**

B3 Spezial-Anschlußkabel ME-1400C/D

Bezeichnung	Farbe	Signal	Sub-D Pin	Bezeichnung	Farbe	Signal	Sub-D Pin
Zähler 0 (T0)	schwarz	Clk 0	15	Zähler 9 (T9)	schwarz	Clk 9	64
	braun	Out 0	16		braun	Out 9	65
	rot	Gate 0	35		rot	Gate 9	45
Zähler 1 (T1)	schwarz	Clk 1	36	Zähler 10 (T10)	schwarz	Clk 10	46
	braun	Out 1	37		braun	Out 10	47
	rot	Gate 1	17		rot	Gate 10	66
Zähler 2 (T2)	schwarz	Clk 2	18	Zähler 11 (T11)	schwarz	Clk 11	67
	braun	Out 2	19		braun	Out 11	68
	rot	Gate 2	38		rot	Gate 11	48
Zähler 3 (T3)	schwarz	Clk 3	54	Zähler 12 (T12)	schwarz	Clk 12	40
	braun	Out 3	55		braun	Out 12	41
	rot	Gate 3	74		rot	Gate 12	60
Zähler 4 (T4)	schwarz	Clk 4	75	Zähler 13 (T13)	schwarz	Clk 13	61
	braun	Out 4	76		braun	Out 13	62
	rot	Gate 4	56		rot	Gate 13	42
	orange	GND	73	Zähler 14 (T14)	schwarz	Clk 14	43
Zähler 5 (T5)	schwarz	Clk 5	57		braun	Out 14	44
	braun	Out 5	58		rot	Gate 14	63
	rot	Gate 5	77	IRQ	schwarz	Vcc	78
Zähler 6 (T6)	schwarz	Clk 6	49		braun	GND	59
	braun	Out 6	50		rot	GND	20
	rot	Gate 6	69		orange	IR_IN	39
Zähler 7 (T7)	schwarz	Clk 7	70				
	braun	Out 7	71				
	rot	Gate 7	51				
Zähler 8 (T8)	schwarz	Clk 8	52				
	braun	Out 8	53				
	rot	Gate 8	72				

(Digital-Ports siehe nächste Seite)

Tabelle 21: Spezialkabel ME-1400C/D

Spezialkabel ME-1400C/D (Fortsetzung)

Bezeichnung	Farbe	Signal	Sub-D Pin	Bezeichnung	Farbe	Signal	Sub-D Pin
DIO	weiß	PA0	21	DIO (Fortsetzung)	orange	PC0	30
	weiß/schwarz	PA1	2		orange/schwarz	PC1	11
	schwarz	PA2	22		orange/weiß	PC2	31
	schwarz/weiß	PA3	3		orange/braun	PC3	12
	braun	PA4	23		gelb	PC4	32
	braun/weiß	PA5	4		gelb/schwarz	PC5	13
	lila	PA6	24		gelb/weiß	PC6	33
	lila/weiß	PA7	5		gelb/braun	PC7	14
	rot	PB0	6		grün	Vcc	1
	rot/schwarz	PB1	26		grün/schwarz	GND	25
	rot/weiß	PB2	7		grün/weiß	GND	10
	rot/braun	PB3	27		grün/braun	GND	34
	pink	PB4	8				
	pink/schwarz	PB5	28				
	pink/weiß	PB6	9				
	pink/braun	PB7	29				

Tabelle 22: Spezialkabel ME-1400C/D (Fortsetzung)

B4 ME-14A/B, ME-1400E/EA/EB

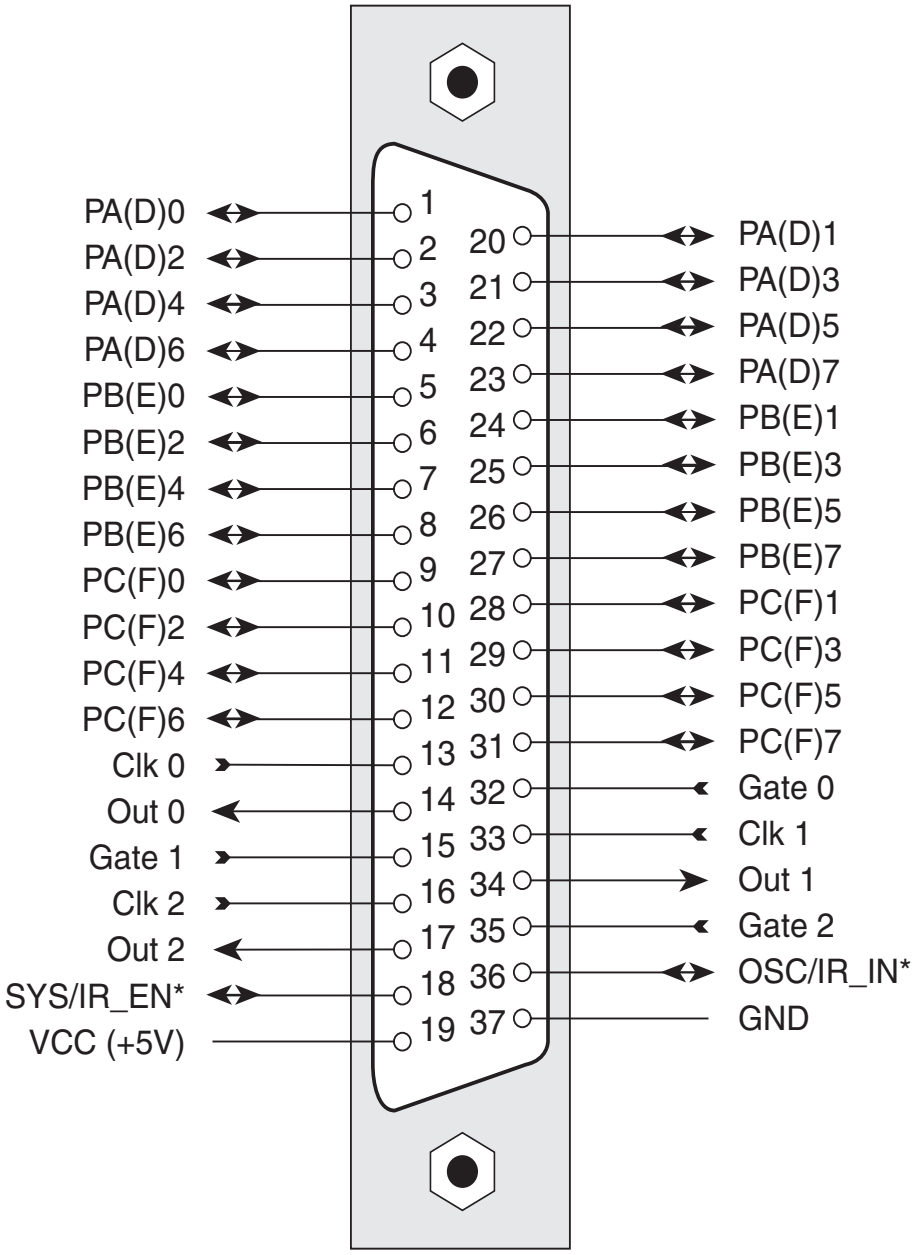


Abb. 21: 37polige Sub-D-Buchse

Hinweis:
Die Digital-Ports D, E und F (in Klammer) sind nur auf den B-Versionen in Verbindung mit einem zusätzlichen Slotblech (im Lieferumfang) verfügbar (siehe auch Anhang B5 und B6).

* Funktionsumfang siehe Tabelle auf nächster Seite.

B5 Stiftstecker B-Versionen (ST2)

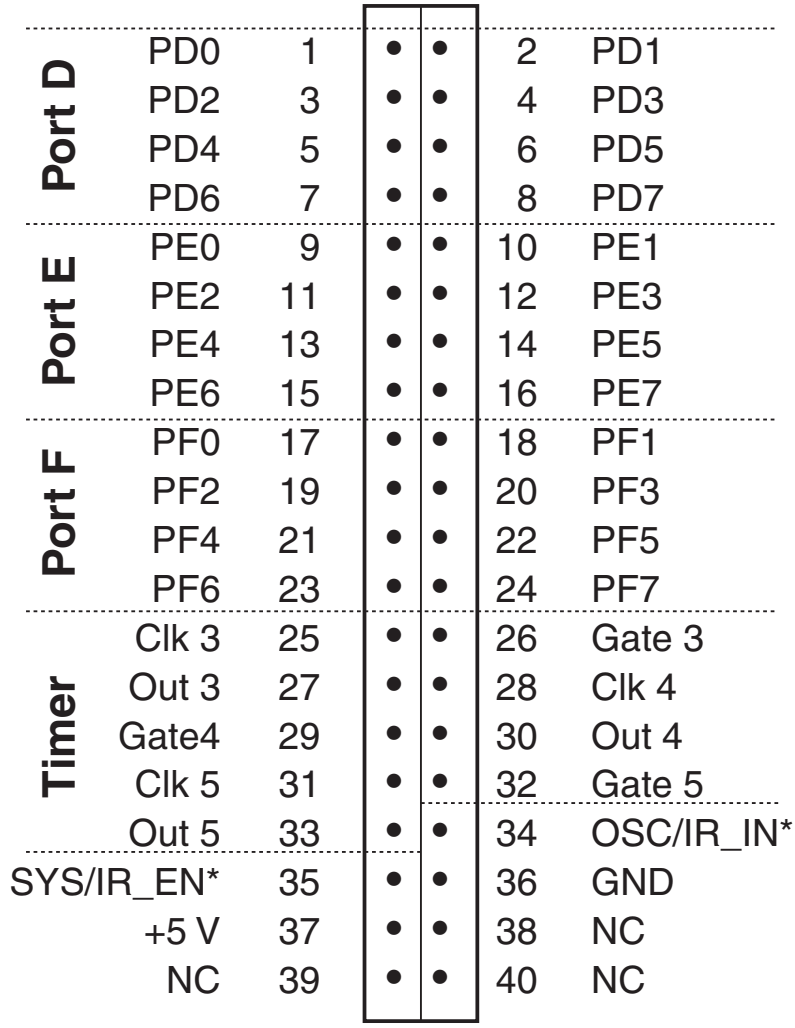


Abb. 22: Belegung des 40poligen Stiftsteckers

	37pol. Sub-D		40pol. Stiftstecker (ST2)	
	SYS/IR_EN (Pin 18)	OSC/IR_IN (Pin 36)	SYS/IR_EN (Pin 35)	OSC/IR_IN (Pin 34)
ME-14A	✔/✔	✔/✔	–	–
ME-14B	✔/✔	✔/✔	✔/❍	✔/❍
ME-1400E	n.c.	–	–	–
ME-1400EA	n.c.	✔/✔	–	–
ME-1400EB	n.c.	✔/✔	n.c.	n.c.

❍ Wird vom Windows-Treiber nicht unterstützt.

B6 Zusatz-Slotblech

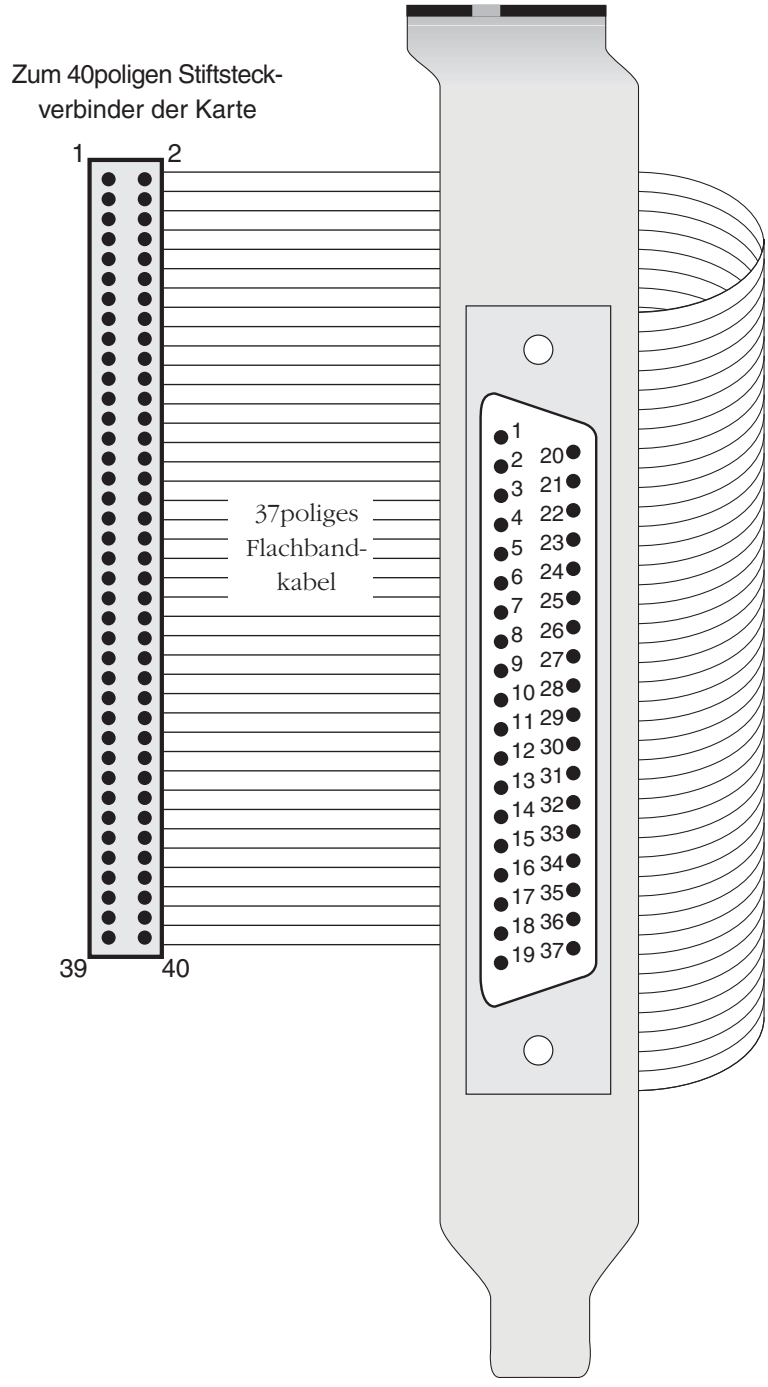


Abb. 23: Slot-Blech mit Sub-D Buchse für ME-14B und ME-1400EB (Schematische Darstellung, Steckerbelegung siehe Abb. 21)

C Zubehör

Als Optionen sind folgende Produkte erhältlich (weitere Informationen über Zusatzprodukte entnehmen Sie bitte dem Meilhaus Electronic Gesamtkatalog)

ME-63Xtend Serie

Externe Relais- und Digital-I/O-Karten zum direkten Anschluß an ME-1400/A/B mittels 1:1 Verbindungskabel z. B. ME AK-D78.

ME-UB Serie

Externe Anschluss- sowie Relais- und Digital-I/O-Boxen zum direkten Anschluss an ME-1400/A/B mittels Spezial-Anschlusskabel ME AK-D7815/1400.

ME AB-D37M

37poliger Sub-D Anschluß-Block (Stecker) für ME-14A/B, ME-1400E

ME AK-D37

37poliges Sub-D Anschluß-Kabel (Stecker-Buchse), 2 m, für ME-14A/B, ME-1400E

ME AB-D78M

78poliger Sub-D Anschluß-Block (Stecker) für ME-1400, ME-1400A, ME-1400B, ME-1400C, ME-1400D

ME AK-D78 (/1)

78poliges Sub-D Anschluß-Kabel (Stecker-Buchse) für ME-1400, ME-1400A, ME-1400B, ME-1400C, ME-1400D; Länge: 2 bzw. 1 m

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 Index