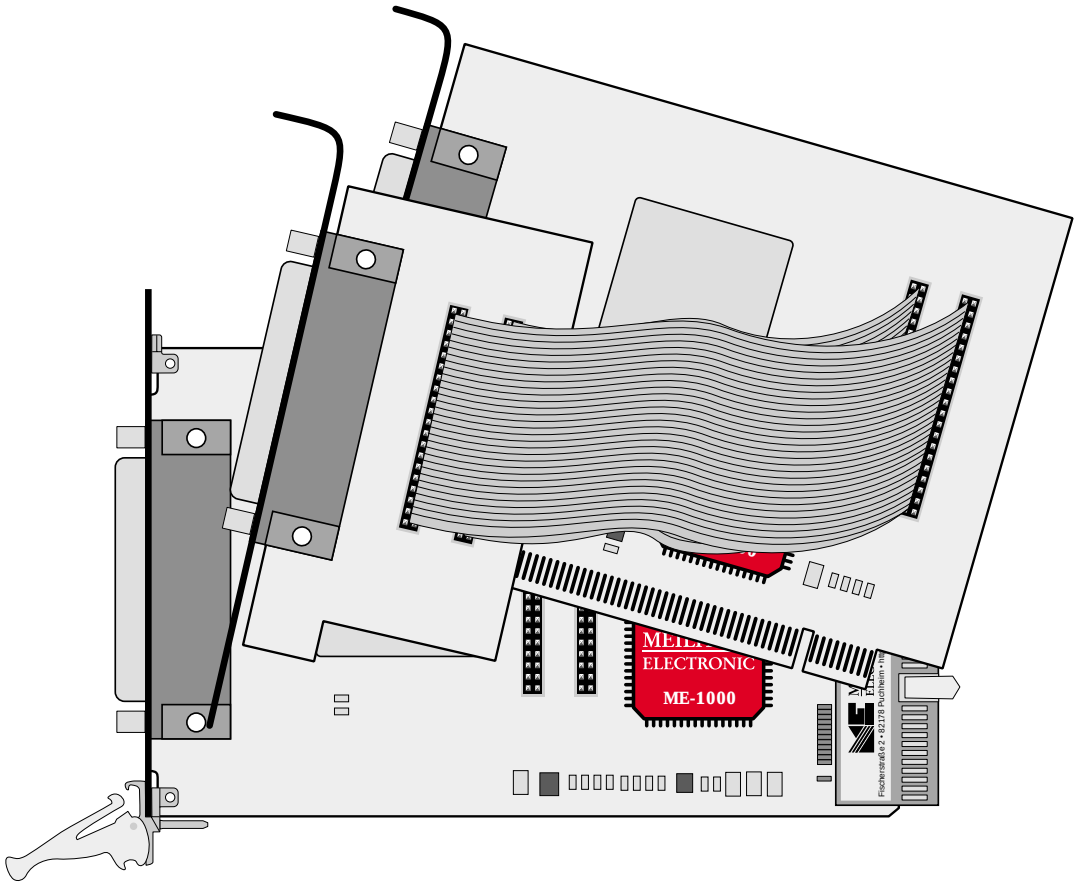
Meilhaus Electronic Handbuch

ME-1000 1.4D

PCI- und CompactPCI-Varianten



64/128-Kanal TTL Digital I/O-Karte

Impressum

Handbuch ME-1000 PCI/cPCI

Revision 1.4D
Ausgabedatum: 23. April 2002

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 dem Kauf dieser Karte 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. Die ME-1000 besitzt volle Plug&Play-Funktionalität sodaß Sie auf der Karte keinerlei Jumper- oder DIP-Schalter-Einstellungen vornehmen müssen.

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:

- 64/128-Kanal TTL Digital I/O-Karte (je nach Version) für PCI- oder CompactPCI-Bus
- Handbuch im PDF-Format auf CDROM (optional in gedruckter Form)
- Treiber-Software auf CD-ROM
- Ein bzw. zwei 78polige(r) Sub-D Gegenstecker
- Version mit 128 Kanälen: Extender-Karte ME-1001 mit 78poliger Sub-D Buchse (incl. 2 Flachband-Kabel)

1.2 Leistungsmerkmale

Modell-Übersicht

Modell	Bus	TTL Digital I/Os
ME-1000/64 PCI	Standard-PCI	2 x 32 Bit Ports, voneinander unabhängig als Ein- oder Ausgangs-Port konfigurierbar (Ausgänge rücklesbar)
ME-1000/64 cPCI	CompactPCI	
ME-1000/128 PCI	Standard-PCI	4 x 32 Bit Ports, voneinander unabhängig als Ein- oder Ausgangs-Port konfigurierbar (Ausgänge rücklesbar)
ME-1000/128 cPCI	CompactPCI	

Tabelle 1: Modell-Übersicht ME-1000 Familie

Die ME-1000 ist eine hochintegrierte Digital-I/O-Karte für den PCI- bzw. CompactPCI-Bus. Es stehen Versionen mit 64 bzw. 128 I/O-Leitungen im TTL-Pegel zur Verfügung. Optional können auf der Karte Pull-Up Widerstandsarrays für alle I/O-Leitungen bestückt werden.

Die **ME-1000/64** besitzt zwei 32 Bit Ports, die unabhängig voneinander als Ein- oder Ausgangsport konfiguriert werden können. Ein als Ausgang konfigurierter Port ist rücklesbar.

Die 128 I/O-Leitungen der **ME-1000/128** sind in insgesamt vier 32 Bit Ports aufgeteilt, die unabhängig voneinander als Ein- oder Ausgangsport konfiguriert werden können. Ein als Ausgang konfigurierter Port ist rücklesbar.

Die Version mit 64 Kanälen kann mit Hilfe der Extender-Karte **ME-1001** jederzeit auf 128 Kanäle erweitert werden.

1.3 Systemanforderungen

Die ME-1000 kann in jedem Rechner mit Intel® Pentium® Prozessor oder Kompatiblen eingesetzt werden, der über einen freien Standard-PCI bzw. CompactPCI Steckplatz (je nach Version) verfügt.

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 ab Version 4.0, HP VEE Lab, Agilent VEE Pro und Agilent VEE OneLab
	Treibersystem für LabView™ ab Version 4.0

2 Installation

Bitte lesen Sie zuerst das Handbuch Ihres Rechners bzgl. der Installation von zusätzlichen Hardwarekomponenten. Eine Anleitung zur Installation der Treiber-Software finden Sie auf CD-ROM.

Grundsätzlich gilt für die Installation der Karte 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:\temp).

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)*

2.1 Testprogramm

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

3 Hardware

3.1 Blockschaltbild

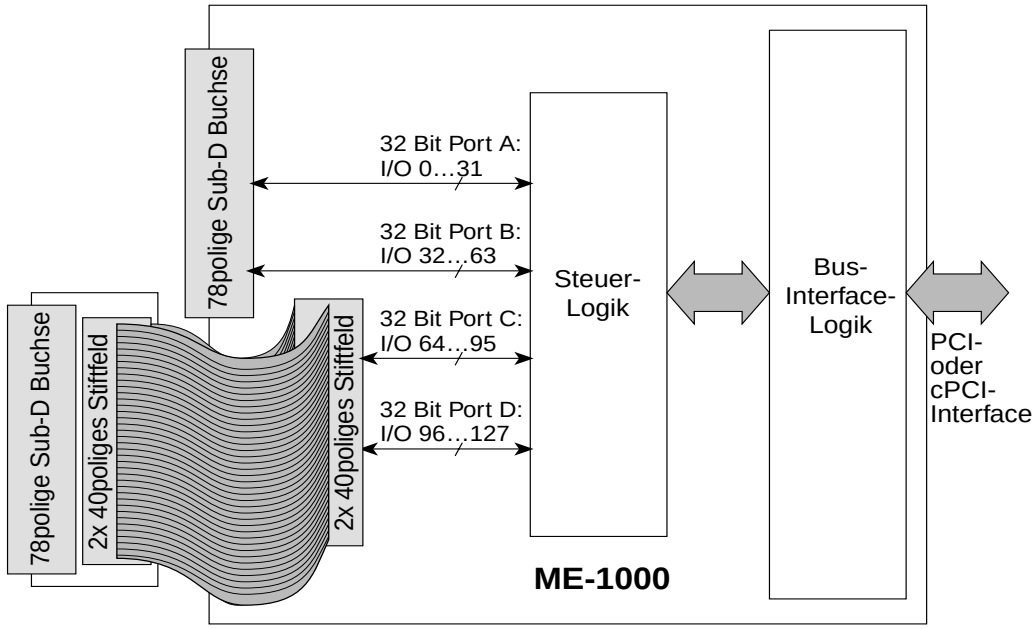


Abb. 1: Blockschaltbild der ME-1000

3.2 Generelle Hinweise

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 ist!

Alle unbenutzten Eingangskanäle sind grundsätzlich auf Masse zu legen, um ein Übersprechen zwischen den Eingangskanälen zu vermeiden.

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

Die Belegung der 78poligen Sub-D Buchse(n) finden Sie im Anhang (siehe „ME-1000 und ME-1001” auf Seite 45).

3.3 Betriebsarten

Je nach Version, verfügt die ME-1000 über 64 bzw. 128 I/O-Leitungen. Diese sind in 2 bzw. 4 Ports mit je 32 Bit breite organisiert. Per Software kann jeder Port als 32 Bit Eingangs- oder Ausgangs-Port konfiguriert werden. Verwenden Sie für Ein-/Ausgabeoperationen die Funktionen der ME-1000 Funktionsbibliothek. Ein als Ausgang konfigurierter Port ist rücklesbar.

Hinweis:
Beachten Sie, daß nach dem Einschalten der Versorgungsspannung alle Ports als Eingänge konfiguriert sind, d. h. alle I/O-Leitungen sind zunächst hochohmig (siehe „Pull-Up/Pull-Down Widerstände” auf Seite 13).

3.4 Beschaltung der Ein-/Ausgänge

Die Ports der ME-1000 sind folgendermaßen aufgeteilt:

- Port A (PA0…PA31) und Port B (PB0…PB31) stehen an der 78poligen Sub-D Buchse der Basis-Platine zur Verfügung.
- Zusätzlich für ME-1000/128: Port C (PC0…PC31) und Port D (PD0…PD31) stehen an der 78poligen Sub-D Buchse der Extender-Platine ME-1001 zur Verfügung.

Beachten Sie beim Anlegen der Signale, daß der TTL- bzw. CMOS-Pegel eingehalten wird und eine Verbindung zur PC-Masse (GND) vorhanden ist.

An den Pins 19, 20, 38, 39, 58, 59, 77 und 78 stehen +5 V vom PC zur Verfügung. Die Gesamtbelastung dieser 8 Pins sollte 500 mA nicht übersteigen. Die Anschlußbelegung der 78poligen Sub-D Buchse(n) finden sind im Anhang „B Anschlußbelegung“ auf Seite 45.

Achtung:
Beschalten Sie einen als Ausgang konfigurierten Port niemals mit einem Eingangssignal!

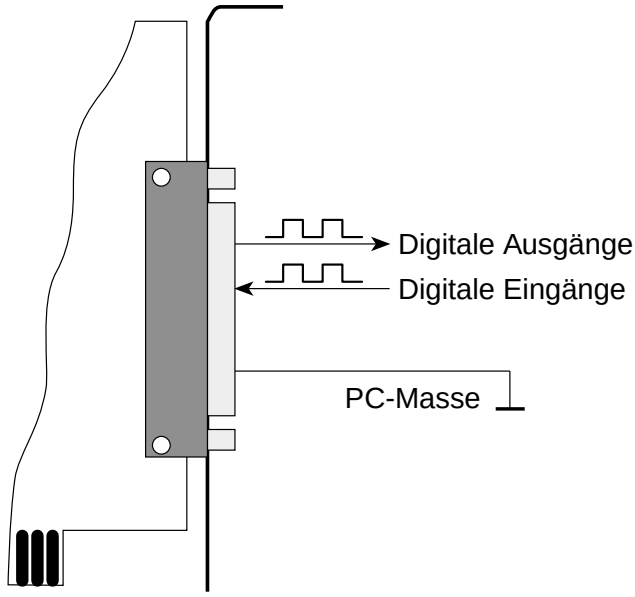


Abb. 2: Beschaltung der ME-1000

Die Strombelastbarkeit der Karte (ohne Luftzirkulation) läßt sich nach folgender Formel abschätzen:

$$T_J \geq T_u + (14°C/W \times P_G)$$

T_J= max. spezifizierte Betriebstemperatur des Bausteines von 70°C

P_G = Gesamtleistung der benutzten Ausgänge

T_u = Umgebungstemperatur der Karte (ohne Konvektion)

3.5 Pull-Up/Pull-Down Widerstände

Da nach dem Einschalten der Versorgungsspannung alle Ports als Eingänge konfiguriert werden, sind alle I/O-Leitungen (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-1000 die Möglichkeit auf der Basis-Platine für alle 64 bzw. 128 I/O-Leitungen 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}=3,1 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. 3: und Abb. 4).

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

Port	Array-Nr.
Port A	RN9, 11,12, 15
Port B	RN2, 13, 14, 16
Port C	RN1, 3, 4, 6
Port D	RN5, 7, 8, 10

Tabelle 2: Zuordnung der Widerstandsarrays

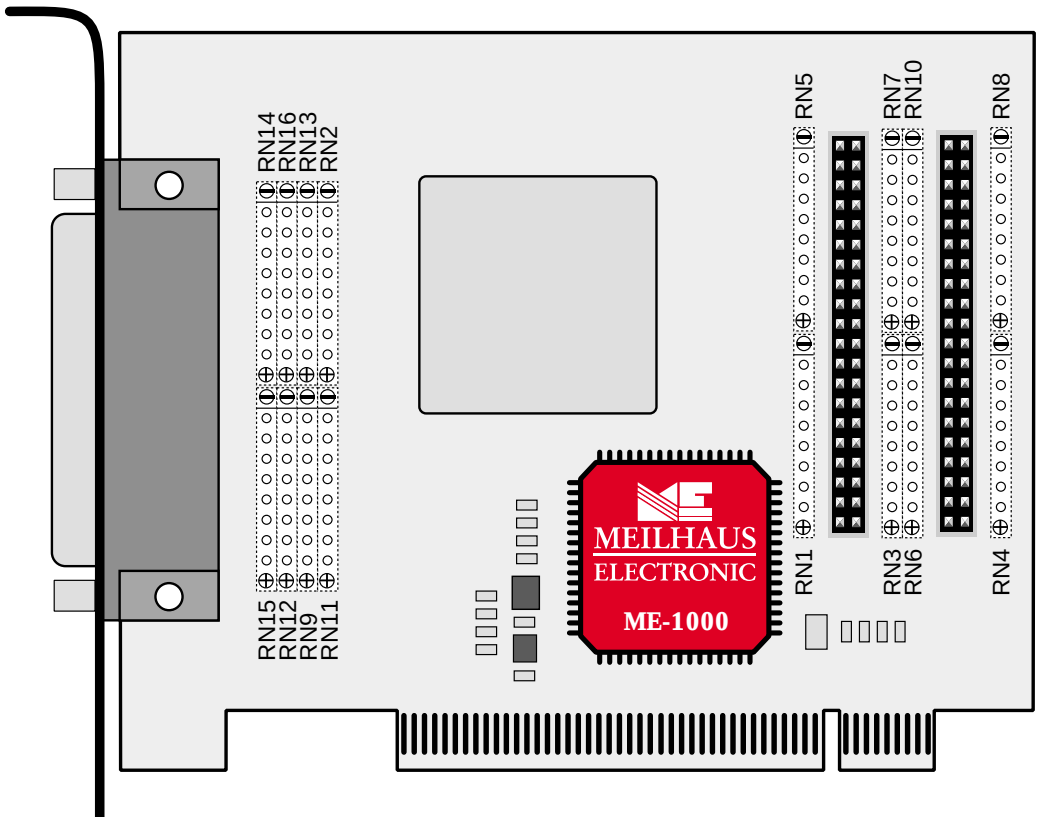


Abb. 3: Anordnung der Widerstandsarrays ME-1000 PCI

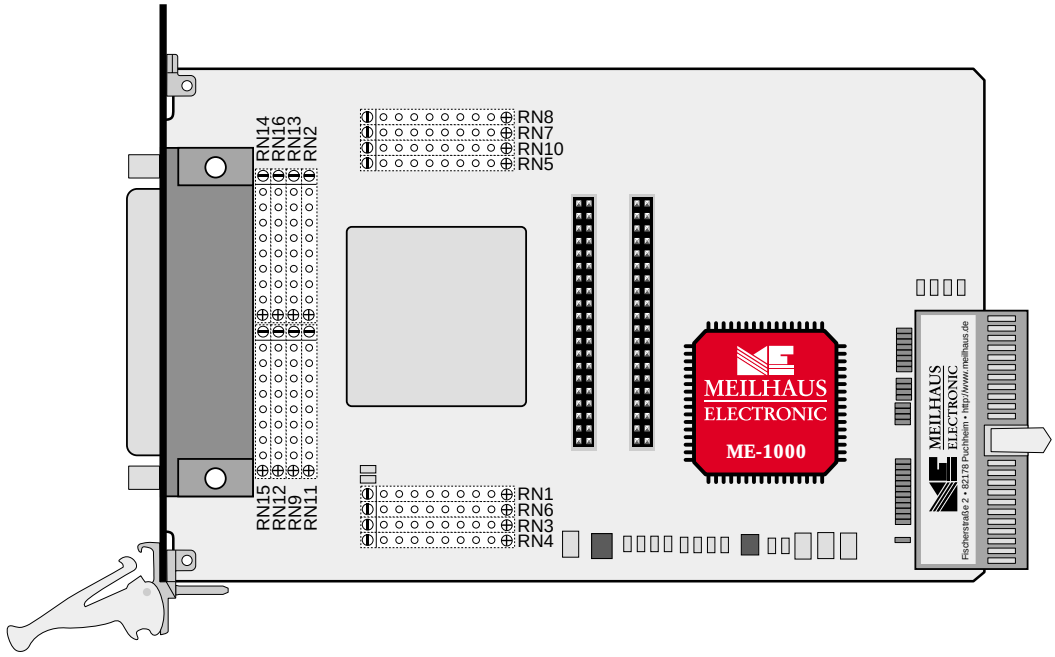


Abb. 4: Anordnung der Widerstandsarrays ME-1000 cPCI

4 Programmierung

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 Vorgehensweise

Nach erfolgreicher Konfiguration der Ein-/Ausgabe-Ports können alle Ein- und Ausgabeoperationen in beliebiger Reihenfolge verwendet werden. Nach jedem Funktionsaufruf sollte grundsätzlich eine Fehlerabfrage mit der Funktion *me1000GetDrvErrMess* durchgeführt werden

Aus Gründen der Laufzeitoptimierung wird keine Plausibilitätsprüfung in den Funktionsaufrufen durchgeführt.

… zum Beispiel:

```
//Im folgenden Beispiel sprechen alle Funktionen die Karte mit der
//logischen <BoardNumber> „0“ an

iBoardNumber = 0;

//Port B als Eingang konfigurieren
return = me1000DIOSetPortDirection(iBoardNumber, PORTB, MEINPUT);

if (return == 0) then
   me1000GetDrvErrMess("ME1000-Test");
endif;
```

```
//Das niederwertige Byte von Port B einlesen
return = me1000DIGetByte(iBoardNumber, PORTB, BYTE_0,
ipByteValue);

if (return == 0) then
   me1000GetDrvErrMess("ME1000-Test");
endif;

...

//Port A als Ausgang konfigurieren
return = me1000DIOSetPortDirection(iBoardNumber, PORTA, MEOUTPUT);

if (return == 0) then
   me1000GetDrvErrMess("ME1000-Test");
endif;

//Bit Nr. 8 von Port A auf High-Pegel setzen
return = me1000DISetBit(iBoardNumber, PORTA, 8, 1);

if (return == 0) then
   me1000GetDrvErrMess("ME1000-Test");
endif;

...
```

4.1.2 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.

Die Meilhaus LabVIEW™-Treiber unterstützen die LabVIEW™ Vollversionen 4.x oder höher. 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.

5 Funktionsreferenz

5.1 Allgemeines

Die Funktionen der API-DLL (`ME1000.DLL`) für die ME-1000 werden von folgenden 32 Bit-Treibern unterstützt:

- VxD-Treiber (`ME1000.VXD`) für Windows 95
- Kernel-Treiber (`ME1000.SYS`) für Windows NT4.0
- WDM-Treiber (`ME1000.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-1000 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, wobei folgendes gilt:

- Wertebereich: 0…31
- Wert für die erste Karte: 0
- Wert für die zweite Karte: 1
- Wert für die x-te Karte: x-1

5.2 Nomenklatur

Die API-Funktionen sind kartenspezifisch gehalten. Jede API-Funktion für die ME-1000 beginnt mit dem Präfix "*me1000…*". 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).

Für die Funktionsbeschreibung gelten folgende Vereinbarungen:

Funktionsnamen	werden im Fließtext kursiv geschrieben z. B. *me1000GetBoardVersion*
<`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
`me1000…()`	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:

„5.3.1 Allgemeine Funktionen“ auf Seite 26

„5.3.2 Digitale Ein-/Ausgabe“ auf Seite 30

„5.3.3 Fehler-Behandlung“ auf Seite 41

Funktion	Kurzbeschreibung	Seite
Allgemeine Funktionen		
me1000GetBoardVersion	Kartenversion ermitteln	26
me1000GetDevInfo	Detail-Infos der Karte ermitteln	26
me1000GetDLLVersion	DLL-Versionsnummer ermitteln	28
me1000GetDriverVersion	Treiberversion ermitteln	29
me1000GetSerialNumber	Seriennummer ermitteln	29
Digitale Ein-/Ausgabe		
me1000DIOSetPortDirection	Port-Richtung definieren	30
me1000DIGetBit	Einzelnes Bit einlesen	31
me1000DIGetByte	Byte (8 Bit) einlesen	32
me1000DIGetWord	Word (16 Bit) einlesen	33
me1000DIGetLong	Longword (32 Bit) einlesen	35
me1000DIOReset	Alle Digital-Ports rücksetzen	36
me1000DOSetBit	Einzelnes Bit ausgeben	36
me1000DOSetByte	Byte (8 Bit) ausgeben	37
me1000DOSetWord	Word (16 Bit) ausgeben	39
me1000DOSetLong	Longword (32 Bit) ausgeben	40
Fehler-Behandlung		
me1000GetDrvErrMess	Fehlerstring gemäß Fehlercode	41

Tabelle 3: Übersicht der Bibliotheksfunktionen

5.3.1 Allgemeine Funktionen

me1000GetBoardVersion

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128.

Es wird die Kartenversion für eine installierte Karte der Kartenfamilie ME-1000 ermittelt.

● Definitionen

C:	int me1000GetBoardVersion (int iBoardNumber, int *piVersion)
Delphi:	Function me1000GetBoardVersion (iBoardNumber: integer; Var iVersion: integer): integer;
Basic:	Declare Function me1000GetBoardVersion Lib "me1000" Alias "_VBme1000GetBoardVersion@8" (ByVal iBoardNumber As Long, iVersion As Long) As Long

➔ Parameter

`<BoardNumber>`	Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23
`<Version>`	Zeiger auf Integer-Variable, in der die Device-ID zurückgegeben wird. Mögliche Werte sind: 100AHex: ME-1000/64 100BHex: ME-1000/128

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgege-- ben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über me1000GetDrvErrMess ermittelt werden.)

me1000GetDevInfo

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128.

Mit Hilfe dieser Funktion kann der erfahrene Programmierer tiefergehende Informationen über die spezifizierte Karte ermitteln.

● Definitionen

C: int me1000GetDevInfo (int iBoardNumber, DEVICEINFOSTRUCT *pDevInfo)

Delphi: Function me1000GetDevInfo (iBoardNumber: integer; Var DevInfo: DEVICEINFOSTRUCT): integer;

Basic: Declare Function me1000GetDevInfo Lib "me1000" Alias "_VBme1000GetDevInfo@8" (ByVal iBoardNumber As Long, ByVal DevInfo As DEVICEINFOSTRUCT) As Long

➔ Parameter

`<BoardNumber>` Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

`<pDevInfo>` Struktur die verschiedene Informationen zur Karte zurückgibt:

<dwBoardNumber>
Kartennummer der angesprochenen Karte

<dwVendorID>
PCI-Hersteller-ID (1402Hex für ME-Karten)

<dwDeviceID>
PCI-Geräte-ID

<dwSystemSlotNumber>
PCI-Slot-Nr. in dem die angesprochene Karte steckt

<dwPortBase>
Basisadresse des ME-Registersatzes

<dwPortCount>
Anzahl der durch den ME-Registersatz belegten Adressen

<dwPortBasePLX>
Port-Adresse des PCI-Controllers

<dwPortCountPLX>
Anzahl der Port-Adressen des PCI-Controllers

<dwSerialNumber>
Seriennummer der Karte

<dwBusNumber>
Bus-Nummer des PCI-Buses (meist 0)

<dwHWRevision>
Hardware-Version der Karte

<dwVersion>
Kartentyp

☞ Beispiel

Das folgende Beispiel gibt einige Elemente der oben beschriebenen Struktur aus:

```
iErrorCode = me1000GetDeviceInfo(iBoardNumber, &DevInfo);
if (iErrorCode
{
   printf("\nTechnical Values:\n");
   printf("Vendor ID:  0x%04x\n, DevInfo.dwVendorID);
   printf("Device ID:  0x%04x\n, DevInfo.dwDeviceID);
   printf("IOPortBase: 0x%04x\n, DevInfo.dwPortBase);
   printf("IOPortPLX:  0x%04x\n, DevInfo.dwPortBasePLX);
   printf("SerialNo:   0x%04x\n, DevInfo.dwSerialNumber);
};
```

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000GetDrvErrMess* ermittelt werden.

me1000GetDLLVersion

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128.

Ermittelt die Versionsnummer der API-DLL für die ME-1000.

● Definitionen

C: int me1000GetDLLVersion(void);

Delphi: Function me1000GetDLLVersion: integer;

Basic: Declare Function me1000GetDLLVersion Lib „me1000“ Alias "_VBme1000GetDLLVersion@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: Rückgabewert 00010003Hex ergibt die Version 1.03

me1000GetDriverVersion

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128.

Ermittelt die Versionsnummer des Treibers für die ME-1000. Es muß mind. eine Karte vom Typ ME-1000 im Rechner vorhanden sein.

● Definitionen

C: int me1000GetDriverVersion(int *piDriverVersion);

Delphi: Function me1000GetDriverVersion (Var piDriverVersion: integer): integer;

Basic: Declare Function me1000GetDriverVersion Lib „me1000“ Alias "_VBme1000GetDriverVersion@4" (ByRef lDriverVersion As Long) As Long

→ Parameter

<DriverVersion> Zeiger auf den Integerwert, der Treiberversion enthält

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann über *me1000ErrorMessage* ermittelt werden.

me1000GetSerialNumber

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128.

Ermittelt die Seriennummer der ausgewählten ME-1000.

● Definitionen

C: int me1000GetSerialNumber (int iBoardNumber, int *piSerialNumber;)

Delphi: Function me1000GetSerialNumber (iBoardNumber: integer; Var piSerialNumber: integer): integer;

Basic: Declare Function me1000GetSerialNumber Lib „me1000“ Alias "_VBme1000GetSerialNumber@8" (ByVal lBoardNumber As Long, ByRef lSerialNumber As Long) As Long

➔ Parameter

<BoardNumber> Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

<SerialNumber> Zeiger auf Integerwert, der die Seriennummer enthält.

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann über *me1000ErrorMessage* ermittelt werden.

5.3.2 Digitale Ein-/Ausgabe

me1000DIOSetPortDirection

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Konfiguriert einen digitalen Port als Eingang oder Ausgang.

☞ Wichtiger Hinweis!

Diese Funktion muß vor allen Digital-I/O Operationen für jeden Port getrennt, einmalig aufgerufen werden.

● Definitionen

C: int me1000DIOSetPortDirection (int iBoardNumber, int iPortNo, int iDir);

Delphi: Function me1000DIOSetPortDirection (iBoardNumber, iPortNo, iDir: integer): integer;

Basic: Declare Function me1000DIOSetPortDirection Lib „me1000“ Alias "_VBme1000DIOSetPortDirection@12" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByVal lDir As Long) As Long

➔ Parameter

<BoardNumber> Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

<PortNo>	Port für Ein- oder Ausgabe wählen:	
	<Port>	Port
	PORTA (00Hex)	32 Bit Port A
	PORTB (01Hex)	32 Bit Port B
	PORTC (02Hex)	32 Bit Port C
	PORTD (03Hex)	32 Bit Port D
<Dir>	Port-Richtung bestimmen:	
	<Dir>	Ein-/Ausgang
	MEINPUT (00Hex)	Port als Eingang
	MEOUTPUT (01Hex)	Port als Ausgang

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DIGetBit

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Liefert den Zustand der selektierten Eingangsleitung zurück.

☞ Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher für den betreffenden Port die Funktion *me1000DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C:	int me1000DIGetBit (int iBoardNumber, int iPortNo, int iBitNo, int *piBitValue);
Delphi:	Function me1000DIGetBit (iBoardNumber, iPortNo, iBitNo: integer; Var piBitValue: integer): integer;
Basic:	Declare Function me1000DIGetBit Lib „me1000“ Alias "_VBme1000DIGetBit@16" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByVal lBitNo As Long, ByRef lBitValue As Long) As Long

→ Parameter

<BoardNumber>	Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

<PortNo>	Port wählen:	
	<Port>	Eingabe-Port
	PORTA (00Hex)	32 Bit Port A
	PORTB (01Hex)	32 Bit Port B
	PORTC (02Hex)	32 Bit Port C
	PORTD (03Hex)	32 Bit Port D
<BitNo>	Nummer der Eingangsleitung, die abgefragt werden soll; möglich sind:	
	<BitNo>	Bit-Nr.
	BIT_0…31 (00…1FHex	0…31
<BitValue>	Zeiger auf einen Integerwert, der dem Leitungszustand entsprechend gelesen wird:	
	Mögliche Rückgabewerte:	
	0: Leitung hat Low-Pegel	
	1: Leitung hat High-Pegel	

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DIGetByte

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Liest ein Byte von einem als Eingang definierten Port.

☞ Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher für den betreffenden Port die Funktion *me1000DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me1000DIGetByte (int iBoardNumber, int iPortNo, int iByteNo, int *piByteValue);

Delphi: Function me1000DIGetByte (iBoardNumber, iPortNo, iByteNo: integer; Var piByteValue: integer): integer;

Basic: Declare Function me1000DIGetByte Lib „me1000“ Alias "_VBme1000DIGetByte@16" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByVal lByteNo As Long, ByRef lByteValue As Long) As Long

→ Parameter

`<BoardNumber>` Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

`<PortNo>` Port wählen:

<Port>	Eingabe-Port
PORTA (00Hex)	32 Bit Port A
PORTB (01Hex)	32 Bit Port B
PORTC (02Hex)	32 Bit Port C
PORTD (03Hex)	32 Bit Port D

`<ByteNo>` Auswahl des Bytes innerhalb eines 32 Bit Wortes; möglich sind:

<ByteNo>	8 Bit Wort
BYTE_0 (00Hex)	P*x*0…P*x*7
BYTE_1 (01Hex)	P*x*8…P*x*15
BYTE_2 (02Hex)	P*x*16…P*x*23
BYTE_3 (03Hex)	P*x*24…P*x*31

`<ByteValue>` Zeiger auf einen Integerwert, der das gelesene Byte aufnimmt; nur die niederwertigsten 8 Bits des Wertes sind signifikant.

← Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DIGetWord

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Liest ein 16 Bit Wort von einem als Eingang definierten Port.

☞ Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher für den betreffenden Port die Funktion *me1000DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me1000DIGetWord (int iBoardNumber, int iPortNo, int iWordNo, int *piWordValue);

Delphi: Function me1000DIGetWord (iBoardNumber, iPortNo, iWordNo: integer; Var piWordValue: integer): integer;

Basic: Declare Function me1000DIGetWord Lib „me1000“ Alias "_VBme1000DIGetWord@12" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByVal lWordNo As Long, ByRef lWordValue As Long) As Long

→ Parameter

`<BoardNumber>` Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

`<PortNo>` Port wählen:

<Port>	Eingabe-Port
PORTA (00Hex)	32 Bit Port A
PORTB (01Hex)	32 Bit Port B
PORTC (02Hex)	32 Bit Port C
PORTD (03Hex)	32 Bit Port D

`<WordNo>` Auswahl des 16 Bit Wortes innerhalb eines 32 Bit Wortes; möglich sind:

<WordNo>	16 Bit Wort
WORD_0 (00Hex)	P*x*0…P*x*15
WORD_1 (01Hex)	P*x*16…P*x*31

`<WordValue>` Zeiger auf einen Integerwert, der das gelesene 16 Bit Wort aufnimmt. Nur die niederwertigen 16 Bits sind signifikant.

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DIGetLong

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Liest ein 32 Bit Wort von einem als Eingang definierten Port.

☞ Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher für den betreffenden Port die Funktion *me1000DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me1000DIGetLong (int iBoardNumber, int iPortNo, int *piValue);

Delphi: Function me1000DIGetLong (iBoardNumber, iPortNo: integer; Var piValue: integer): integer;

Basic: Declare Function me1000DIGetLong Lib „me1000“ Alias "_VBme1000DIGetLong@12" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByRef lValue As Long) As Long

➔ Parameter

`<BoardNumber>` Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

`<PortNo>` Port wählen:

<Port>	Eingabe-Port
PORTA (00Hex)	32 Bit Port A
PORTB (01Hex)	32 Bit Port B
PORTC (02Hex)	32 Bit Port C
PORTD (03Hex)	32 Bit Port D

`<Value>` Zeiger auf einen Integerwert, der das gelesene 32 Bit Wort aufnimmt.

❮ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DIOReset

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Diese Funktion setzt alle Ports in den Grundzustand (Eingänge).

● Definitionen

C: int me1000DIOReset (int iBoardNumber);

Delphi: function me1000DIOReset (iBoardNumber: integer): integer;

Basic: Declare Function me1000DIOReset Lib „me1000“ Alias "_VBme1000DIOReset@4" (ByVal lBoardNumber As Long) As Long

➔ Parameter

`<BoardNumber>` Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DOSetBit

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Setzt eine digitale Ausgangsleitung in den gewünschten Zustand.

☞ Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher für den betreffenden Port die Funktion *me1000DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me1000DOSetBit (int iBoardNumber, int iPortNo, int iBitNo, int iBitValue);

Delphi: Function me1000DOSetBit (iBoardNumber, iPortNo, iBitNo, iBitValue: integer): integer;

Basic: Declare Function me1000DOSetBit Lib „me1000“ Alias "_VBme1000DOSetBit@16" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByVal lBitNo As Long, ByVal lBitValue As Long) As Long

➔ Parameter

`<BoardNumber>` Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

`<PortNo>` Port wählen:

<Port>	Ausgabe-Port
PORTA (00Hex)	32 Bit Port A
PORTB (01Hex)	32 Bit Port B
PORTC (02Hex)	32 Bit Port C
PORTD (03Hex)	32 Bit Port D

`<BitNo>` Nummer der Ausgangsleitung, die gesetzt werden soll; möglich sind:

<BitNo>	Bit-Nr.
BIT_0…31 (00…1FHex)	0…31

`<BitValue>` Mögliche Werte sind:

= 0: Bit wird auf Low-Pegel gesetzt

> 0: Bit wird auf High-Pegel gesetzt

➔ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DOSetByte

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Schreibt ein Byte an einen als Ausgang konfigurierten digitalen Port.

☞ Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher für den betreffenden Port die Funktion *me1000DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me1000DOSetByte (int iBoardNumber, int iPortNo, int iByteNo, int iByteValue);

Delphi: function me1000DOSetByte (iBoardNumber, iPortNo, iByteNo, iByteValue: integer): integer;

Basic: Declare Function me1000DOSetByte Lib „me1000“ Alias "_VBme1000DOSetByte@16" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByVal lByteNo As Long, ByVal lByteValue As Long) As Long

→ Parameter

<BoardNumber> Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

<PortNo> Port wählen:

<Port>	Ausgabe-Port
PORTA (00Hex)	32 Bit Port A
PORTB (01Hex)	32 Bit Port B
PORTC (02Hex)	32 Bit Port C
PORTD (03Hex)	32 Bit Port D

<ByteNo> Auswahl des Bytes innerhalb eines 32 Bit Wortes; möglich sind:

<ByteNo>	8 Bit Wort
BYTE_0 (00Hex)	P*x*0…P*x*7
BYTE_1 (01Hex)	P*x*8…P*x*15
BYTE_2 (02Hex)	P*x*16…P*x*23
BYTE_3 (03Hex)	P*x*24…P*x*31

<ByteValue> Ausgabewert; mögliche Werte sind: 00Hex…FFHex (0…256).

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DOSetWord

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Schreibt ein 16 Bit Wort an einen als Ausgang konfigurierten digitalen Port.

☞ Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher für den betreffenden Port die Funktion *me1000DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me1000DOSetWord (int iBoardNumber, int iPortNo, int iWordNo, int iWordValue);

Delphi: function me1000DOSetWord (iBoardNumber, iPortNo, iWordNo, iWordValue: integer): integer;

Basic: Declare Function me1000DOSetWord Lib „me1000“ Alias "_VBme1000DOSetWord@12" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByVal lWordNo As Long, ByVal lWordValue As Long) As Long

➔ Parameter

`<BoardNumber>` Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

`<PortNo>` Port wählen:

<Port>	Ausgabe-Port
PORTA (00Hex)	32 Bit Port A
PORTB (01Hex)	32 Bit Port B
PORTC (02Hex)	32 Bit Port C
PORTD (03Hex)	32 Bit Port D

`<WordNo>` Auswahl des Bytes innerhalb eines 32 Bit Wortes; möglich sind:

<WordNo>	16 Bit Wort
WORD_0 (00Hex)	P*x*0…P*x*15
WORD_1 (01Hex)	P*x*16…P*x*31

`<WordValue>` Ausgabewert; mögliche Werte sind: 0000Hex…FFFFHex (0…65535).

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

me1000DOSetLong

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128. Port C und D nur bei ME-1000/128.

Schreibt ein 32 Bit Wort an einen als Ausgang konfigurierten digitalen Port.

☞ Wichtiger Hinweis!

Zur Konfiguration des Ports muß vorher für den betreffenden Port die Funktion *me1000DIOSetPortDirection* einmalig aufgerufen werden.

● Definitionen

C: int me1000DOSetLong (int iBoardNumber, int iPortNo, int iValue);

Delphi: function me1000DOSetLong (iBoardNumber, iPortNo, iValue: integer): integer;

Basic: Declare Function me1000DOSetLong Lib „me1000“ Alias "_VBme1000DOSetLong@12" (ByVal lBoardNumber As Long, ByVal lPortNo As Long, ByVal lValue As Long) As Long

➔ Parameter

`<BoardNumber>` Nummer der anzusprechenden ME-1000 (erste: 0, zweite: 1, x-te: x-1), siehe auch Seite 23

`<PortNo>` Port wählen:

<Port>	Ausgabe-Port
PORTA (00Hex)	32 Bit Port A
PORTB (01Hex)	32 Bit Port B
PORTC (02Hex)	32 Bit Port C
PORTD (03Hex)	32 Bit Port D

`<Value>` Ausgabewert; mögliche Werte sind: 00000000Hex…FFFFFFFFHex (0…4294967295).

‹ Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird 1 zurückgegeben. Im Fehlerfall wird 0 zurückgegeben. Die genaue Fehlerursache kann dann über *me1000ErrorMessage* ermittelt werden.

5.3.3 Fehler-Behandlung

me1000GetDrvErrMess

✎ Beschreibung

Funktion gilt für die Modelle: ME-1000/64 und ME-1000/128.

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

☞ Wichtiger Hinweis!

Dieser Funktionsaufruf ist nur dann sinnvoll, wenn die unmittelbar zuvor aufgerufene API-Funktion der `ME1000.DLL` fehlerhaft (d. h. Funktionswert 0) ausgeführt wurde!

● Definitionen

C:	int me1000GetDrvErrMess (char *pcErrortext, int iBufferSize);
Delphi:	Function me1000GetDrvErrMess (Var errortext: errorstring; iBufferSize: integer): integer;
Basic:	Declare Function me1000GetDrvErrMess Lib "me1000_32" Alias "_VBme1000GetDrvErrMess@4" (ByVal errortext As String, ByVal iBufferSize As Long) As Long

→ Parameter

<`Errortext`>	Zeiger auf Fehlertext; der Fehlercode wird als Funktionswert zurückgegeben.
<`BufferSize`>	Puffergröße in Anzahl der Zeichen für Fehlertext wird reserviert (empfohlen: max. 128 Zeichen).

‹ Rückgabewert

Funktion gibt immer die globale Fehlervariable <`DLLErrorCode`> zurück.

Anhang

A Spezifikationen

PC Interface

Bus-System	PCI- bzw. CompactPCI-Bus (32 Bit, 33 MHz)
Plug&Play-Funktionalität	Wird voll unterstützt (keine Jumper für Basisadresse, oder Interrupt).

Digitale Ein-/Ausgänge

Anzahl	ME-1000/64: 2 x 32 Bit I/O-Ports (Ausgangsports rücklesbar) ME-1000/128: 4 x 32 Bit I/O-Ports (Ausgangsports rücklesbar)
Eingangsspannung	Low: 0 V…+0,8 V (I_{ILmax} =±10 µA) High: +2,0 V…+5,5 V (I_{IHmax} =±10 µA)
Ausgangsspannung	Low: 0…+0,8 V (I_{OL}=+20 mA) High: Min. +2,4 V (I_{OH}=-4 mA)
Ausgangsstrom pro Kanal	I_{OLmax} = 20 mA I_{OHmax} = 4 mA
Achtung:	Gesamtleistung darf nicht überschritten werden (siehe „Beschaltung der Ein-/Ausgänge” auf Seite 12

Allgemeine Daten

Strombelastbarkeit der +5V Pins (19, 20, 38, 39, 58, 59, 77, 78):	max. 500 mA bei +5 V
Stromverbrauch bei +5 V	typ. 1,2 A (ohne ext. Belastung)
Kartenabmessungen (ohne Slotblech und Stecker)	ME-1000 PCI: 136 x 107 mm ME-1000 cPCI: CompactPCI-Karte mit 3 HE ME-1001 PCI und cPCI: L: 55 mm, H: 100 mm
Anschlüsse	alle Modelle: 78polige Sub-D Buchse; zusätzlich ME-1000/128: weitere 78polige Sub-D Buchse auf Extender-Karte ME-1001
Betriebstemperatur	0…70 °C
Lagertemperatur	0…50 °C
Luftfeuchtigkeit	20…55% (nicht kondensierend)

CE-Zertifizierung

EG-Richtlinie	89/336/EMC
Emission	EN 55022
Störfestigkeit	EN 50082-2

B Anschlußbelegung

B1 ME-1000 und ME-1001

Die Anschlußbelegung der ME-1000 ist identisch mit der Extender-Platine ME-1001. Port A und B der ME-1000 korrespondieren dabei mit Port C und D der ME-1001:

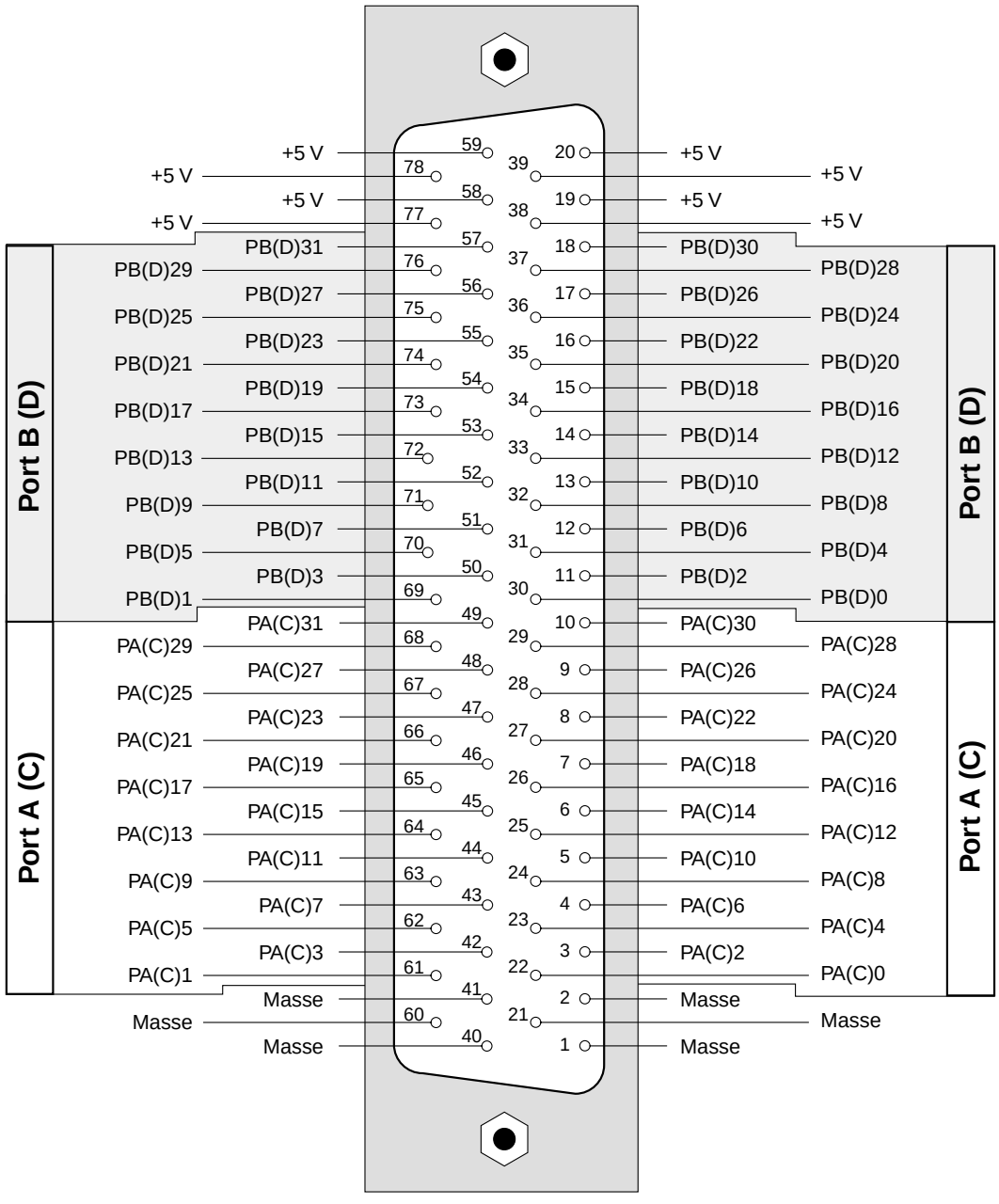


Abb. 5: Belegung der 78poligen Sub-D Buchse von ME-1000 und ME-1001

C Zubehör

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

ME-AB-D78M

78poliger Sub-D Anschluß-Block (Stecker) für ME-1000/ME-1001 (PCI- und cPCI-Version)

ME-AK-D78

78poliges Sub-D Anschluß-Kabel (Stecker-Buchse), 2 m, für ME-1000/ME-1001 (PCI- und cPCI-Version)

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