CPCI-DIO1616

CompactPCI-digital I/O-Karte

Software-Handbuch

zu Artikel I.2309.02

esd electronic system design gmbh
Vahrenwalder Str. 207 • 30165 Hannover • Germany
www.esd-electronics.com • Fax: 0511/37 29 8-68
Phone: 0511/37 29 80 • International: +49-5 11-37 29 80

Der Inhalt dieses Handbuches wurde mit größter Sorgfalt erarbeitet und geprüft. **esd** übernimmt jedoch keine Verantwortung für Schäden, die aus Fehlern in der Dokumentation resultieren könnten. Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

esd hat das Recht, Änderungen am beschriebenen Produkt oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

esd electronic system design gmbh
Vahrenwalder Str. 207
30165 Hannover

Tel.: 0511/372 98-0
FAX : 0511/372 98-68
E-Mail: info@esd.electronics.com
Internet: www.esd-electronics.com

Handbuch-Datei:	I:\Texte\Doku\MANUALS\CPCI\DIO1616\CPCI-DIO1616_12S.ma9
Datum der Druck-vorlagenerstellung:	2008-04-17

Beschriebene Software:	**Beschriebene Software-Revision:**
Treiber für QNX6 (esd-Bestellnr.: I.2309.32)	Rev.: 1.0.5
Treiber für: Windows 2000 Windows XP Windows Vista (32-Bit) (esd-Bestellnr.: I.2309.12)	Zur Zeit nur Beta-Version (0.1.0) Nur folgende Dienste werden unterstützt: - `DIOCTL_SEND` - `DIOCTL_RECEIVE` - `DIOCTL_RECEIVE_OUTPUT`

Änderungen in der Software und/oder der Dokumentation

Kapitel	Änderung gegenüber der Vorversion
2.1	Beschreibung der Installation des Gerätetreibers unter QNX6. Änderung der Eingabe
2.2	Beschreibung der Installation des Gerätetreibers unter Windows eingefügt
4.1	`typdef uint64_t DIOTICK` eingefügt
4.3	`DIOCTL_RECEIVE_OUTPUT` und `DIOCTL_RECIRQ_FLUSH` in Kommandoübersicht aufgenommen
4.9	Beschreibung des Kommandos `DIOCTL_RECEIVE_OUTPUT`
4.10	Beschreibung des Kommandos `DIOCTL_RECIRQ_FLUSH`

Diese Seite ist bewusst unbedruckt.

Inhalt

Seite

Diese Seite ist bewusst unbedruckt.

1. Einleitung

Dieses Handbuch beschreibt die Treibersoftware des CompactPCI-Moduls CPCI-DIO1616 für die Betriebssysteme QNX6 und Windows 2000/XP/Vista(32-Bit).

2. Starten des Treibers

2.1 Installation des Gerätetreibers unter QNX6

Zum Starten des Treibers unter QNX6 ist

```
devio-pcidio &
```

einzugeben.

Hinweis:	Der Treiber kann nur mit “root”-Rechten gestartet werden! Der Treiber sollte über eines der Start-Scripte unter `/etc/system` oder `/etc/rc.d` gestartet werden oder über den PCI-Enumerator.

2.2 Installation der Gerätetreiber unter Windows

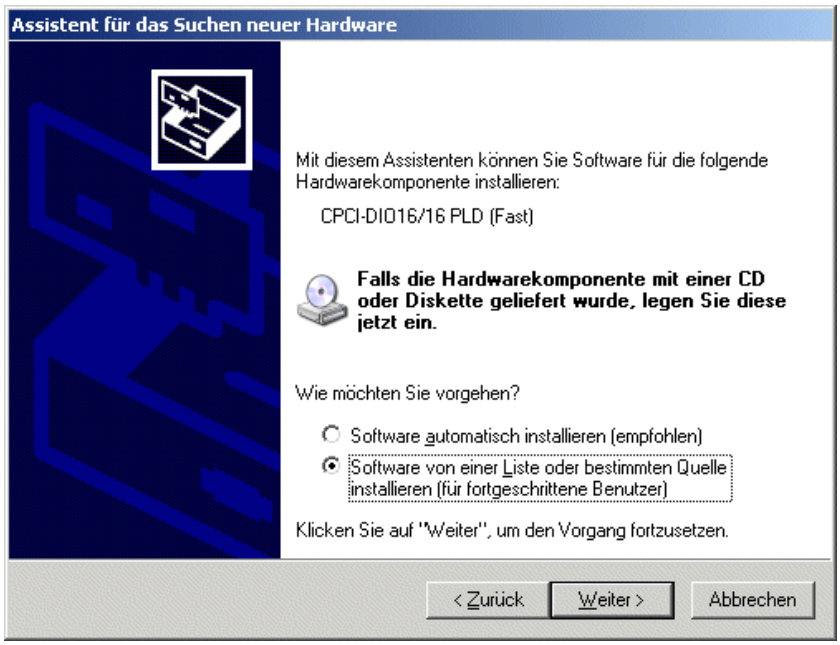
Hinweis:	Für die Installation der Gerätetreiber unter Windows 2000/XP/Vista(32-Bit) benötigen Sie Administrator-Rechte.

Die Installation wird durch den *Assistenten für das Suchen neuer Hardware* des Windows-Betriebssystems ausgeführt. Die Installation wird im Folgenden am Beispiel von Windows XP beschrieben. Das Aussehen der Fenster kann daher für andere Windows-Versionen abweichen.

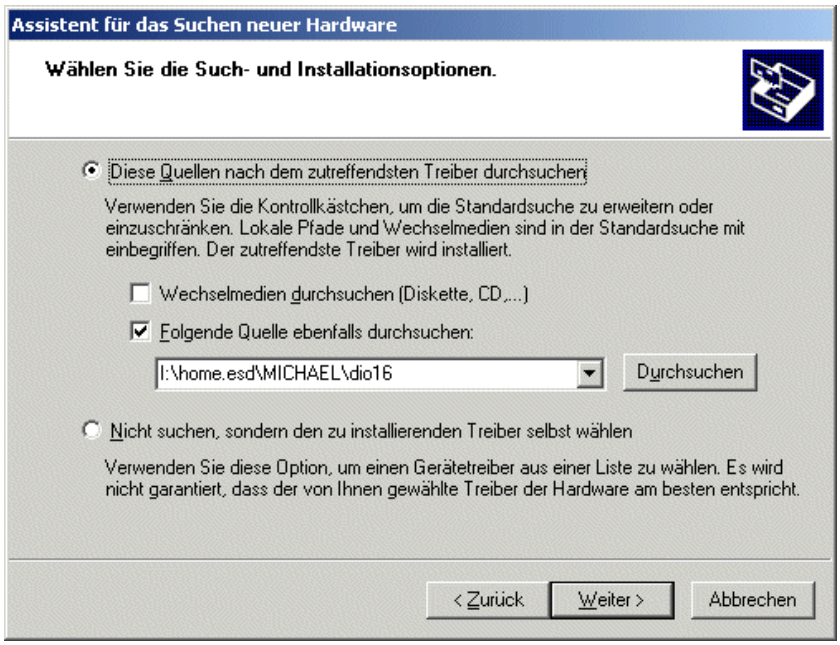
Die Hardware-Installation des Moduls muss vor dem Start von Windows XP erfolgen. Findet Windows XP die neue Hardware, wird das Fenster *Assistent für das Suchen neuer Hardware* angezeigt.
Folgen Sie weiter den Anweisungen des Assistenten.

Wählen Sie *Nein, diesmal nicht* und klicken auf *Weiter*, um fortzufahren.

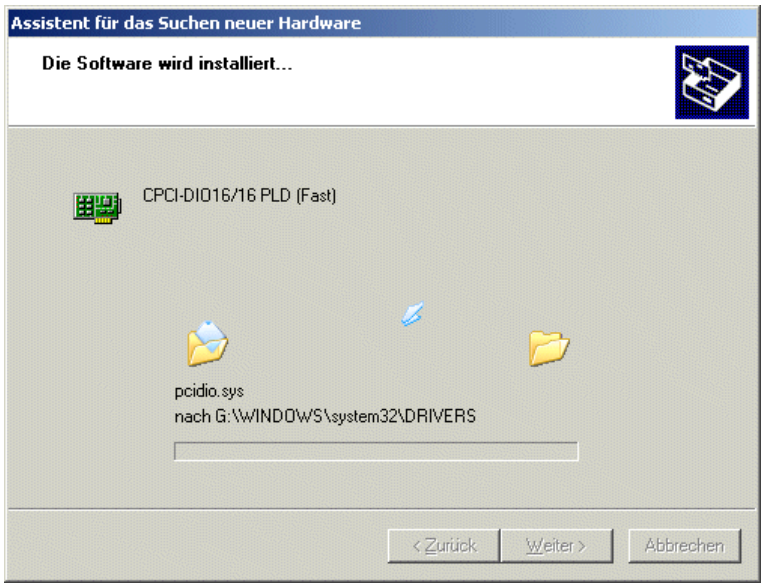
In der folgenden Dialog-Box wählen Sie *Software von einer Liste oder bestimmten Quelle installieren (für fortgeschrittene Benutzer)* und bestätigen mit *Weiter*.



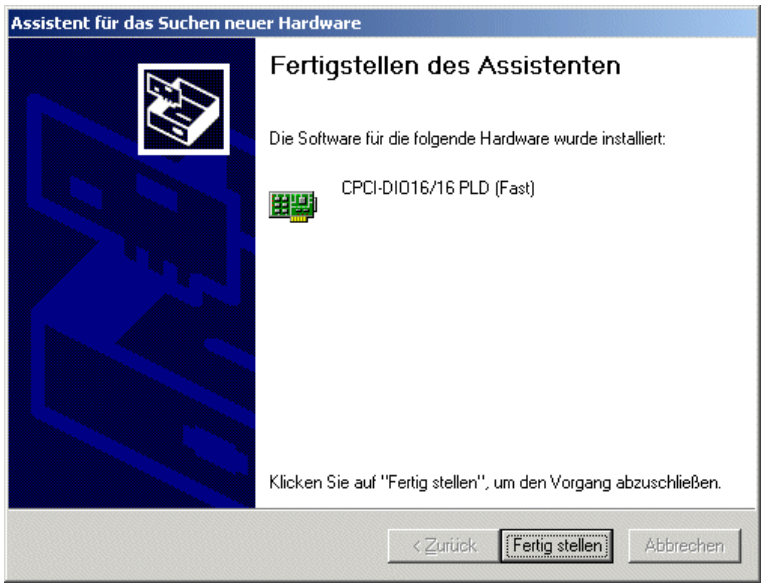
Selektieren Sie im folgenden Fenster *Diese Quelle nach dem zutreffendsten Treiber durchsuchen* und weiter *Wechselmedien durchsuchen*, wenn Sie von einer-CD installieren oder wählen Sie *Folgende Quelle ebenfalls durchsuchen* und suchen Sie das Verzeichnis der Treiberdateien. Mit dem Anklicken von *Weiter* startet der Assistent die Installation der Gerätetreiber.



Das Treiberpaket ist nicht von WHQL (Windows Hardware Quality Labs) signiert. Je nach Konfiguration Ihres Rechners kann es sein, dass eine Warnung bezüglich des Treibers ausgegeben wird, wenn der Treiber (non-WHQL certified) installiert werden soll.
Klicken Sie auf die Schaltfläche *Ja*, um mit der Treiberinstallation fortzufahren.
Der Treiber wird nun installiert.



Nach dem Kopieren der benötigten Treiberdateien wird der Erfolg der Installation in einem Fenster anzeigt. Zum Beenden der Installation klicken Sie auf die Schaltfläche *Fertig stellen*.



Nach der Treiberinstallation können Sie den Gerätemanager aufrufen, um die erfolgreiche Installation zu überprüfen und verschiedene Treiberparameter zu konfigurieren.

3. Verwendung des Treibers

Der Zugriff auf den Treiber erfolgt über die folgenden Aufrufe:

QNX6: `devctl()`

Windows XP: `DeviceIoControl()`

Die Kommandos und Parameter sind im Header `esdio.h` beschrieben.

Im Lieferumfang ist außerdem das Beispielprogramm `diotest.c` als Quell-Code enthalten.

4. Funktionsbeschreibungen

4.1 Strukturen

Die in den folgenden Kapiteln beschriebenen Kommandos verwenden für ihre Parameter die folgenden Strukturen:

```
typedef struct
{
   uint32_t mask;
   uint32_t data;
   uint64_t time;
} DIODATA;

typedef struct
{
   uint32_t edgeRising;
   uint32_t edgeFalling;
} DIOIRQ;

typdef uint64_t DIOTICK;
```

4.2 Zuordnung der Ein- und Ausgänge zu den Parameter-Bits

Bit	Belegung	Pegelzuordnung
31 : 17	nicht ausgewertet : nicht ausgewertet	-
16	*User-LED-Bit*	‘0’ -> LED aus ‘1’ -> LED leuchtet
15 : 0	*OUT15* : *OUT0*	‘0’ -> Ausgang aus ‘1’ -> Ausgang ein

Tabelle 4.2.1: Belegung der Output-Register

Bit	Belegung	Pegelzuordnung
31 : 20	ohne Bedeutung : ohne Bedeutung	-
19 18 17 16	*Error_12-15* *Error_8-11* *Error_4-7* *Error_0-3*	‘0’ -> kein Fehler der Ausgänge ‘1’ -> Fehler der Ausgänge
15 : 0	*IN15* : *IN0*	‘0’ -> Eingang inaktiv ‘1’ -> Eingang aktiv

Tabelle 4.2.2: Belegung der Input-Register

4.3 Kommandoübersicht

Kommando	Funktion
DIOCTL_SEND	Setzen der digitalen Ausgänge und der User-LED
DIOCTL_RECEIVE	Lesen der digitalen Eingänge und der Fehlermeldungen der Ausgänge
DIOCTL_RECIRQ	Lesen der Interrupt-Quelle
DIOCTL_IRQSET	Initialisieren der Interrupts
DIOCTL_TIMERES	Ermitteln der Zeitbasis des zurückgegebenen Zeitstempels
DIOCTL_RECEIVE_OUTPUT	Lesen der digitalen Ausgänge
DIOCTL_RECIRQ_FLUSH	Löschen der Input-Event-Queue

Tabelle 4.3.1: Übersicht der implementierten Kommandos

Hinweis:	Der Windows-Treiber (Windows 2000/XP/Vista) unterstützt derzeit nur die folgenden Kommandos: DIOCTL_SEND DIOCTL_RECEIVE DIOCTL_RECEIVE_OUTPUT

4.4 DIOCTL_SEND - Setzen der digitalen Ausgänge und der User-LED

DIOCTL_SEND

Name: **DIOCTL_SEND** - Setzen der digitalen Ausgänge und der User-LED

Input-Parameter: *DIODATA.mask*: bit(x)=1 => output x to send
- maskiert die Ausgangsbits, die gesetzt werden sollen

DIODATA.data: output state for bits matched in mask
- setzt die Ausgänge auf ‘0’ oder ‘1’

Output-Parameter: keine

Beschreibung: Dieses Kommando dient zum Setzen der Ausgänge. Die Ausgänge, die verändert werden sollen, müssen zunächst maskiert werden.

Nach einem Reset oder Systemstart sind alle Ausgänge ausgeschaltet.

4.5 DIOCTL_RECEIVE - Lesen der digitalen Eingänge

DIOCTL_RECEIVE

Name: **DIOCTL_RECEIVE** - Lesen der digitalen Eingänge und der Fehlermeldungen

Input-Parameter: *DIODATA.mask*: bit(x)=1 => input x to receive
- maskiert die Eingangsbits, die gelesen werden sollen

Output-Parameter: *DIODATA.mask*: echo of input mask
- zurückgelesener Wert der Maskierung der Eingänge

DIODATA.data: input state for bits matched in mask; unmatched bits set to zero
- Zustand der gelesenen Eingangsbits, bzw. Fehlermeldungen der Ausgangstreiber

Beschreibung: Dieses Kommando dient zum Lesen der digitalen Eingänge und zur Auswertung der Fehlersignale der Ausgangstreiberbausteine. Über eine Maske werden die Bits ausgewählt, deren Pegel gelesen werden soll. Der Wert der Maske kann zurückgelesen werden.
Der aktuelle Zustand der Eingangsbits kann in *DIODATA.data* gelesen werden. Der Zustand der Bits, die ausmaskiert sind, wird immer als ‘0’ zurückgelesen.

Für die auszuwertenden Bits gilt:
Eingang aktiv (Spannung liegt an): Bit = ‘1’
Fehlersignal aktiv Bit = ‘1’

Eingang inaktiv (keine oder zu geringe Eingangsspannung): Bit = ‘0’
Fehlersignal inaktiv Bit = ‘0’

4.6 DIOCTL_RECIRQ - Lesen der Interrupt-Quelle

DIOCTL_RECIRQ

Name: **DIOCTL_RECIRQ** - Lesen der Interrupt-Quelle

Input-Parameter: keine

Output-Parameter: *DIODATA.mask*: bit(x) = 1 => edge for input x detected
- Maskierung der Eingangs- und Fehler-Bits für die Interrupt-Auswertung

DIODATA.data: input state for bits matched in mask; unmatched bits set to zero
- Eingangs- oder Fehler-Bit, dass Interrupt ausgelöst hat

DIODATA.time: timestamp of detected edge(s)
- die Zeitbasis kann über das Kommando DIOCTL_TIMERES ermittelt werden

Beschreibung: Mit diesem Kommando wird der, bzw. werden die Eingangsbits oder Fehlerbits ermittelt, die einen Interrupt ausgelöst haben.
Der Zeitpunkt des Interrupts kann über *DIODATA.time* zurückgelesen werden.

Der Aufruf blockiert bis zum Auftreten der gewünschten Flanke(n).

4.7 DIOCTL_IRQSET - Initialisieren der Interrupts

DIOCTL_IRQSET

Name: **DIOCTL_IRQSET** - Initialisierung der Interrupts

Input-Parameter:

DIOIRQ.edgeRising:	bit(x) = 1 =>	interrupt enable for rising edge on input x
	bit(x) = 0 =>	interrupt disable for rising edge on input x
DIOIRQ.edgeFalling:	bit(x) = 1 =>	interrupt enable for falling edge on input x
	bit(x) = 0 =>	interrupt disabled for falling edge on input x

Output-Parameter: keine

Beschreibung: Mit diesem Kommando erfolgt die Freischaltung oder Unterdrückung der Interrupts bei steigender und/oder fallender Flanke. Nach dem Aufwachen sind alle Interrupts disabled.

4.8 DIOCTL_TIMERES - Ermitteln der Zeitbasis

DIOCTL_TIMERES

Name: **DIOCTL_TIMERES** - Ermitteln der Zeitbasis des Zeitstempels

Input-Parameter: keine

Output-Parameter: *DIOTICK resolution*: clock cycles per second
- Anzahl der Taktzyklen pro Sekunde

Beschreibung: Mit diesem Kommando kann die Zeitbasis des Taktzyklus des Controller-Taktes ermittelt werden, um den Wert des Zeitstempels sinnvoll auswerten zu können.

4.9 DIOCTL_RECEIVE_OUTPUT - Lesen der digitalen Ausgänge

DIOCTL_RECEIVE_OUTPUT

Name: **DIOCTL_RECEIVE_OUTPUT** - Lesen der digitalen Ausgänge

Input-Parameter:

DIODATA.mask: bit(x)=1=> output pin x to receive
- maskiert die Ausgangsbits, die gelesen werden sollen

Output-Parameter:

DIODATA.mask: echo of input mask
- Zurückgelesener Wert der Maskierung der Ausgänge

DIODATA.data: output state for pins matched in mask (unmatched bits are set to zero
- Liest den aktuellen Zustand der in der Maske festgelegten Ausgangsbits, ausmaskierte Bits werden auf ‘0’gesetzt.

Beschreibung: Dieses Kommando dient zum Lesen der digitalen Ausgänge. Über eine Maske werden die Bits ausgewählt, deren Pegel gelesen werden soll.
Der aktuelle Zustand der Ausgangsbits kann in *DIODATA.data* gelesen werden. Der Zustand der Bits, die ausmaskiert sind, wird immer als ‘0’ zurückgelesen.

4.10 DIOCTL_RECIRQ_FLUSH - Löschen der Input Event Queue

DIOCTL_RECIRQ_FLUSH

Name: **DIOCTL_RECIRQ_FLUSH** - Löschen der Input-Event-Queue

Input-Parameter: keine

Output-Parameter: keine

Beschreibung: Dieses Kommando dient zum Löschen der Interrupt-Events des aktuellen Handles. Ein geöffnetes Handle kann bis zu 256 Events zwischenspeichern.