



USB-TTL-IN8-OUT8

Hardware-Beschreibung

2010 Oktober

INDEX

<u>1. Einleitung</u>	5
<u>1.1. Vorwort</u>	5
<u>1.2. Kundenzufriedenheit</u>	5
<u>1.3. Kundenresonanz</u>	5
<u>2. Hardware Beschreibung</u>	7
<u>2.1. Technische Daten</u>	7
<u>2.2. Steckerbelegung</u>	8
<u>3. Software</u>	10
<u>3.1. Benutzung unserer Produkte</u>	10
<u>3.1.1. Ansteuerung über grafische Anwendungen</u>	10
<u>3.1.2. Ansteuerung über unsere DELIB Treiberbibliothek</u>	10
<u>3.1.3. Ansteuerung auf Protokollebene</u>	10
<u>3.1.4. Ansteuerung über mitgelieferte Testprogramme</u>	11
<u>3.2. DELIB Treiberbibliothek</u>	12
<u>3.2.1. Übersicht</u>	12
<u>3.2.2. Unterstützte Betriebssysteme</u>	14
<u>3.2.3. Unterstützte Programmiersprachen</u>	14
<u>3.2.4. Installation DELIB-Treiberbibliothek</u>	15
<u>3.2.5. DELIB Configuration Utility</u>	17
<u>3.3. Testprogramme</u>	18
<u>3.3.1. Digital Input-Output Demo</u>	18
<u>3.3.2. Analog Input-Output Demo</u>	19
<u>4. DELIB API Referenz</u>	21
<u>4.1. Verwaltungsfunktionen</u>	21
<u>4.1.1. DapiOpenModule</u>	21
<u>4.1.2. DapiCloseModule</u>	22
<u>4.2. Digitale Eingänge lesen</u>	23
<u>4.2.1. DapiDIGet1</u>	23

INDEX

<u>4.2.2. DapiDIGet8</u>	24
<u>4.2.3. DapiDIGet16</u>	25
<u>4.2.4. DapiDIGet32</u>	26
<u>4.2.5. DapiDIGet64</u>	27
<u>4.2.6. DapiDIGetFF32</u>	28
<u>4.3. Digitale Ausgänge verwalten</u>	29
<u>4.3.1. DapiDOSet1</u>	29
<u>4.3.2. DapiDOSet8</u>	30
<u>4.3.3. DapiDOSet16</u>	31
<u>4.3.4. DapiDOSet32</u>	32
<u>4.3.5. DapiDOSet64</u>	33
<u>4.3.6. DapiDOReadback32</u>	34
<u>4.3.7. DapiDOReadback64</u>	35
<u>4.4. Programmier-Beispiel</u>	36
<u>5. Anhang</u>	39
<u>5.1. Revisionen</u>	39
<u>5.2. Urheberrechte und Marken</u>	40



Einleitung



1. Einleitung

1.1. Vorwort

Wir beglückwünschen Sie zum Kauf eines hochwertigen DEDITEC Produktes!

Unsere Produkte werden von unseren Ingenieuren nach den heutigen geforderten Qualitätsanforderungen entwickelt. Wir achten bereits bei der Entwicklung auf flexible Erweiterbarkeit und lange Verfügbarkeit.

Wir entwickeln modular!

Durch eine modulare Entwicklung verkürzt sich bei uns die Entwicklungszeit und - was natürlich dem Kunden zu Gute kommt - ein fairer Preis!

Wir sorgen für eine lange Lieferverfügbarkeit!

Sollten verwendete Halbleiter nicht mehr verfügbar sein, so können wir schneller reagieren. Bei uns müssen meistens nur Module redesigned werden und nicht das gesamte Produkt. Dies erhöht die Lieferverfügbarkeit.

1.2. Kundenzufriedenheit

Ein zufriedener Kunde steht bei uns an erster Stelle!

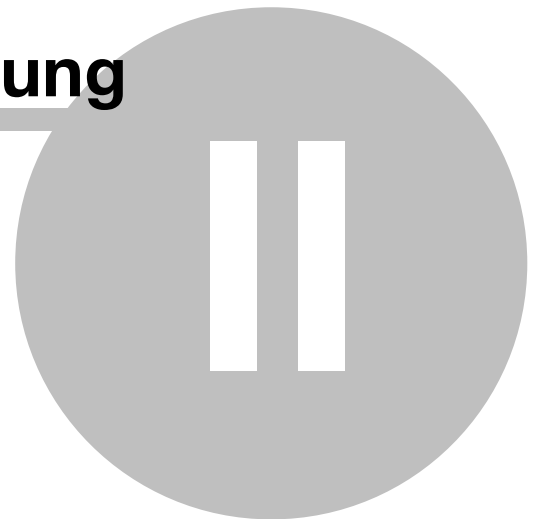
Sollte mal etwas nicht zu Ihrer Zufriedenheit sein, wenden Sie sich einfach per Telefon oder mail an uns.

Wir kümmern uns darum!

1.3. Kundenresonanz

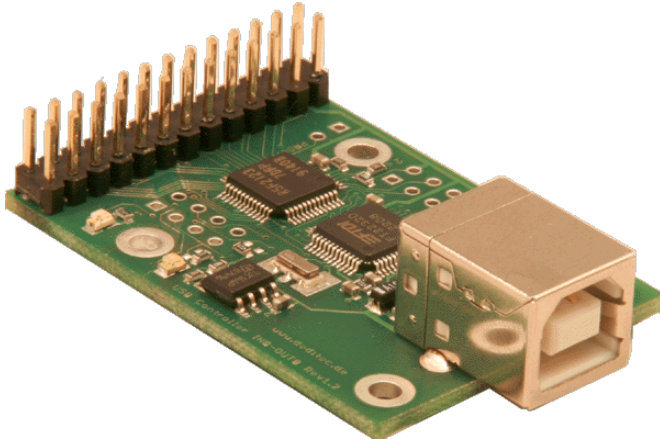
Die besten Produkte wachsen mit unseren Kunden. Für Anregungen oder Vorschläge sind wir jederzeit dankbar.

Hardware Beschreibung



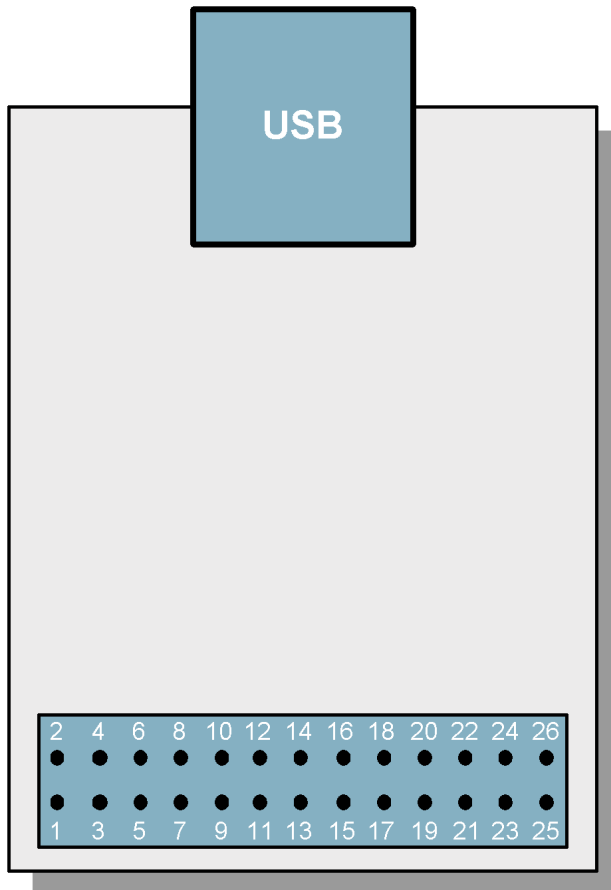
2. Hardware Beschreibung

2.1. Technische Daten



- Eingangsstrom: ~53 mA
- USB 2.0 / 1.1
- 8*TTL-Eingänge
- 8*TTL-Ausgänge
- Einfaches Ansprechen unter Windows

2.2. Steckerbelegung



Pin		Pin	
1	VCC	2	LED
3	GND	4	GND
5	IN6	6	IN 7
7	IN 4	8	IN 5
9	IN 2	10	IN 3
11	IN 0	12	IN 1
13	OUT 0	14	OUT1
15	OUT 2	16	OUT3
17	OUT4	18	OUT5
19	OUT6	20	OUT7
21	AD3	22	AD2
23	AD1	24	AD0
25	GND	26	VREF

Software



3. Software

3.1. Benutzung unserer Produkte

3.1.1. Ansteuerung über grafische Anwendungen

Wir stellen Treiberinterfaces z.B. für LabVIEW und ProfiLab zur Verfügung. Als Basis dient die DELIB Treiberbibliothek, die von ProfiLab direkt angesteuert werden kann.

Für LabVIEW bieten wir eine einfache Treiberanbindung mit Beispielen an!

3.1.2. Ansteuerung über unsere DELIB Treiberbibliothek

Im Anhang befindet sich die komplette Funktionsreferenz für das Integrieren unserer API-Funktionen in Ihre Software. Des Weiteren bieten wir passende Beispiele für folgende Programmiersprachen:

- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

3.1.3. Ansteuerung auf Protokollebene

Das Protokoll für die Ansteuerung unserer Produkte legen wir komplett offen. So können Sie auch auf Systemen ohne Windows oder Linux unsere Produkte einsetzen!

3.1.4. Ansteuerung über mitgelieferte Testprogramme

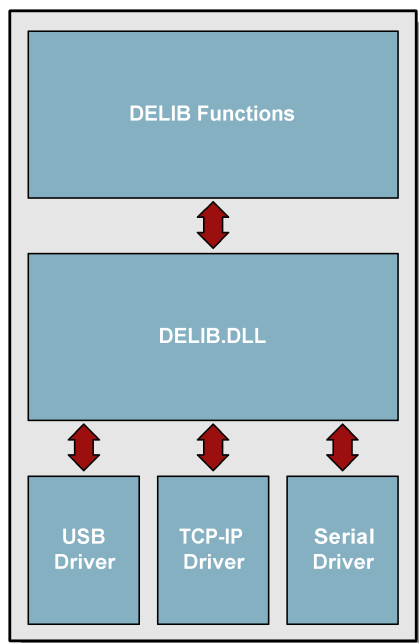
Für die wichtigsten Funktionen unserer Produkte stellen wir einfach zu bedienende Testprogramme zur Verfügung,. Diese werden bei der Installation der DELIB Treiberbibliothek direkt mit installiert.

So können z.B. Relais direkt getestet werden oder Spannungen am A/D Wandler direkt überprüft werden.

3.2. DELIB Treiberbibliothek

3.2.1. Übersicht

Die folgende Abbildung erläutert den Aufbau der DELIB Treiberbibliothek



Die DELIB Treiberbibliothek ermöglicht ein einheitliches Ansprechen von DEDITEC Hardware, mit der besonderen Berücksichtigung folgender Gesichtspunkte:

- Betriebssystem unabhängig
- Programmiersprachen unabhängig
- Produkt unabhängig

Programmieren unter diversen Betriebssystemen

Die DELIB Treiberbibliothek ermöglicht ein einheitliches Ansprechen unserer Produkte auf diversen Betriebssystemen.

Wir haben dafür gesorgt, dass mit wenigen Befehlen alle unsere Produkte angesprochen werden können.

Dabei spielt es keine Rolle, welches Betriebssystem Sie verwenden. - Dafür sorgt die DELIB !

Programmieren mit diversen Programmiersprachen

Für das Erstellen eigener Anwendungen stellen wir Ihnen einheitliche Befehle zur Verfügung. Dies wird über die DELIB Treiberbibliothek gelöst.

Sie wählen die Programmiersprache !

So können leicht Anwendung unter C++, C, Visual Basic, Delphi oder LabVIEW® entwickelt werden.

Schnittstellenunabhängiges programmieren

Schreiben Sie Ihre Anwendung schnittstellenunabhängig !

Programmieren Sie eine Anwendung für ein USB-Produkt von uns. - Es wird auch mit einem Ethernet oder RS-232 Produkt von uns laufen !

SDK-Kit für Programmierer

Integrieren Sie die DELIB in Ihre Anwendung. Auf Anfrage erhalten Sie von uns kostenlos Installationskripte, die es ermöglichen, die DELIB Installation in Ihre Anwendung mit einzubinden.

3.2.2. Unterstützte Betriebssysteme

Unsere Produkte unterstützen folgende Betriebssysteme:

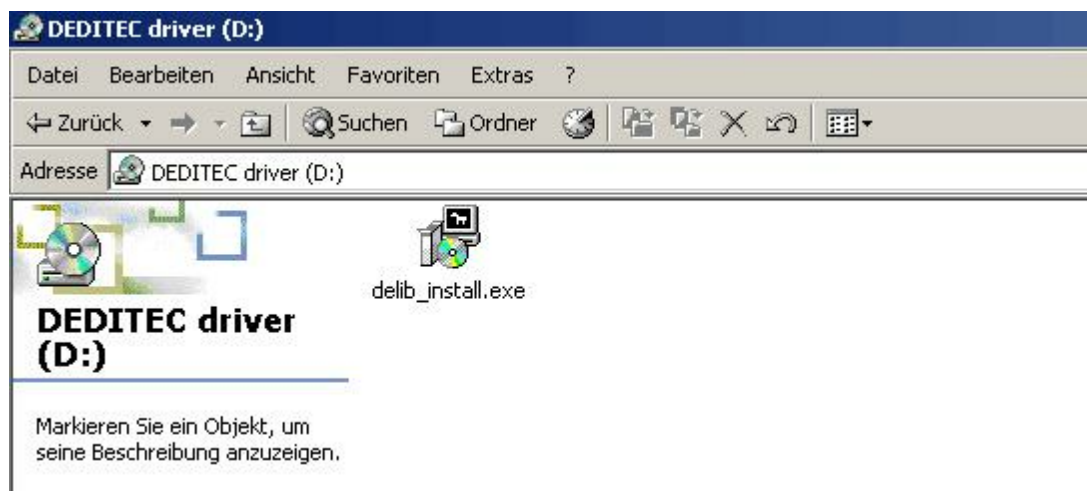
- Windows 2000
- Windows XP
- Windows Vista
- Windows 7
- Linux

3.2.3. Unterstützte Programmiersprachen

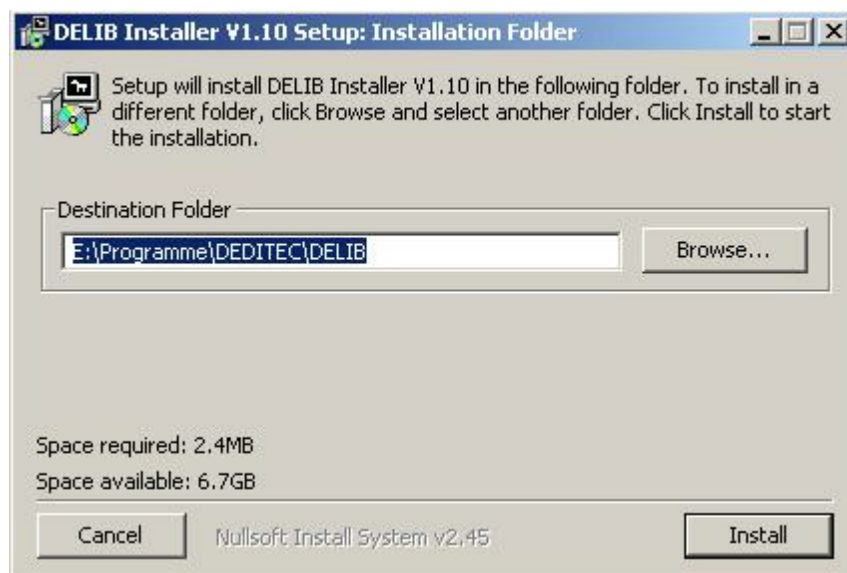
Unsere Produkte sind über folgende Programmiersprachen ansprechbar:

- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

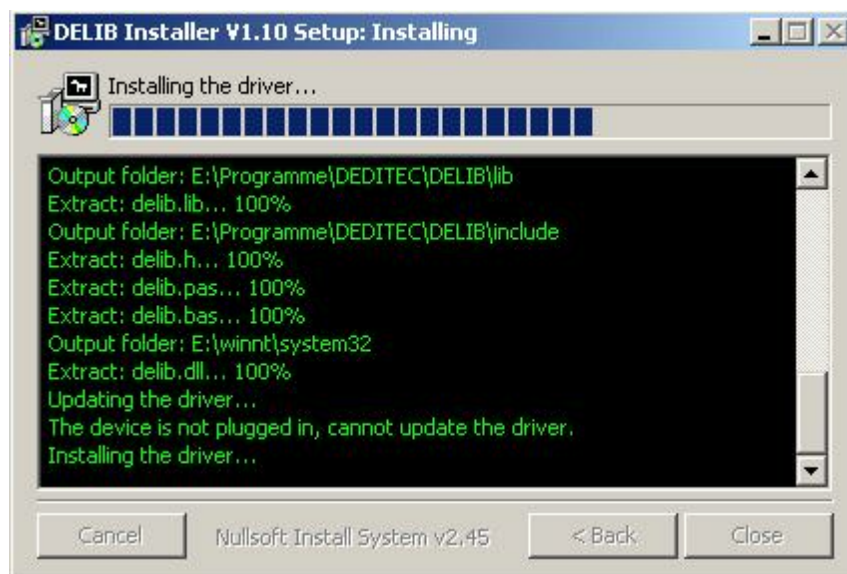
3.2.4. Installation DELIB-Treiberbibliothek



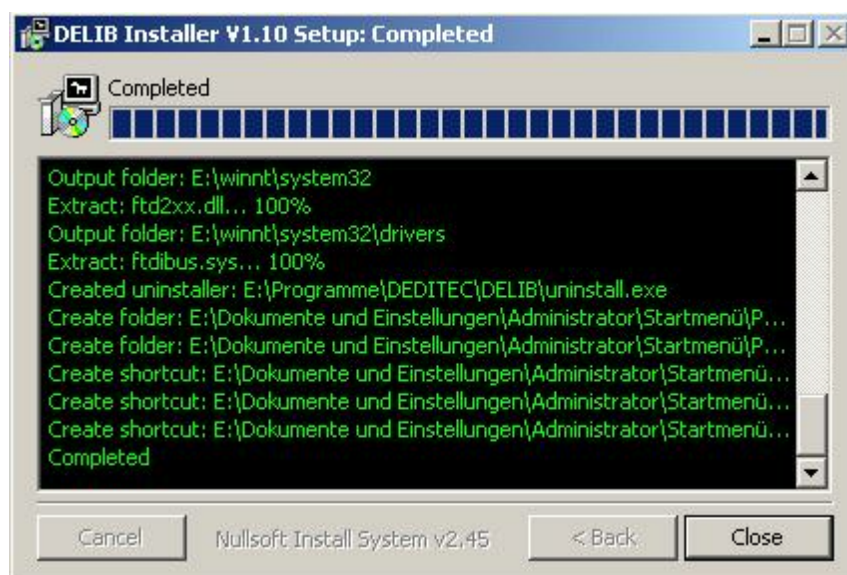
Legen Sie die DEDITEC driver CD in das Laufwerk und starten Sie "delib_install.exe". Die DELIB-Treiberbibliothek ist auch unter <http://www.deditec.de/delib> erhältlich.



Drücken Sie auf "Install".



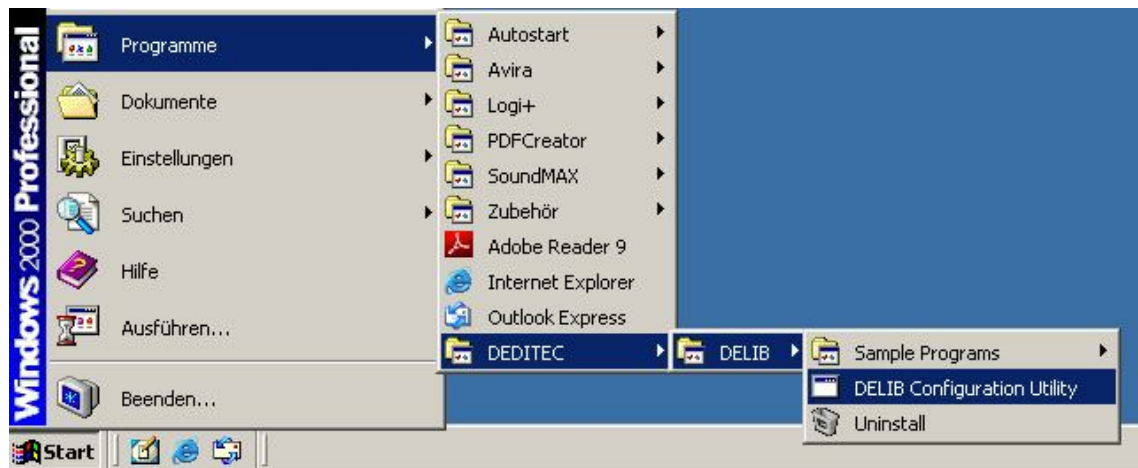
Die Treiber werden nun installiert.



Die DELIB Treiberbibliothek wurde nun installiert. Drücken sie auf **“Close”** um die Installation zu beenden.

Mit dem **“DELIB Configuration Utility”** (nächstes Kapitel) können Sie Ihr Modul konfigurieren (dies ist nur nötig, wenn Sie mehr als ein Modul ansprechen möchten).

3.2.5. DELIB Configuration Utility



“**DELIB Configuration Utility**” wird auf dem folgendem Weg gestartet:

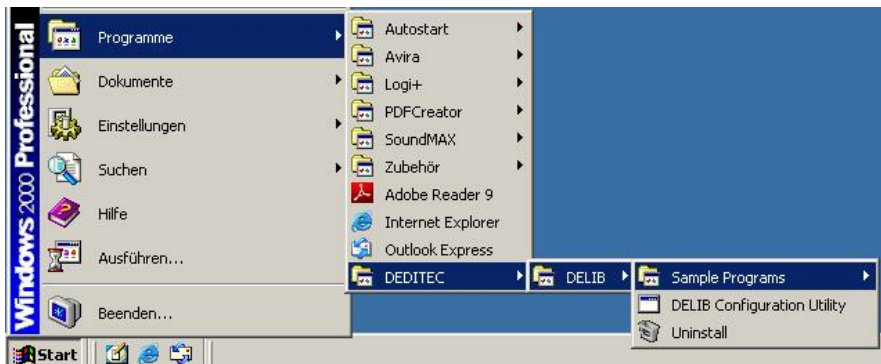
Start → Programme → DEDITEC → DELIB → DELIB Configuration Utility.

Das “**DELIB Configuration Utility**” ist ein Programm zur Konfiguration und Unterteilung Identischer USB-Module im System. Dies ist aber nicht nötig falls nur ein Modul vorhanden ist.

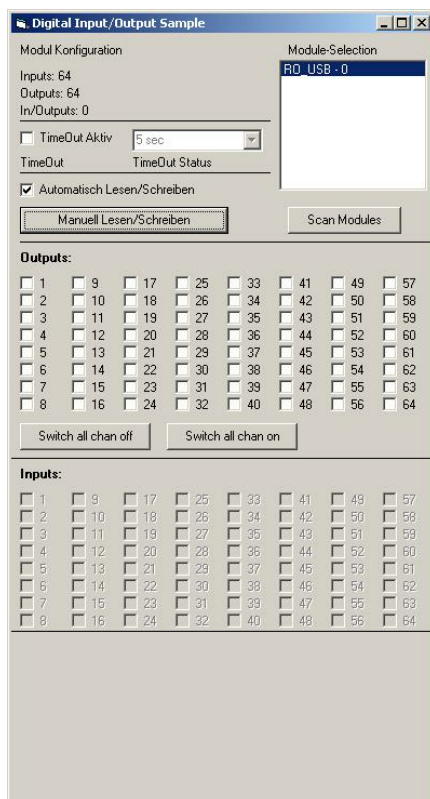
Weiteres zum Inhalt der “**DELIB Installation**”, siehe “**Manual für DELIB Treiberbibliothek**”

3.3. Testprogramme

3.3.1. Digital Input-Output Demo

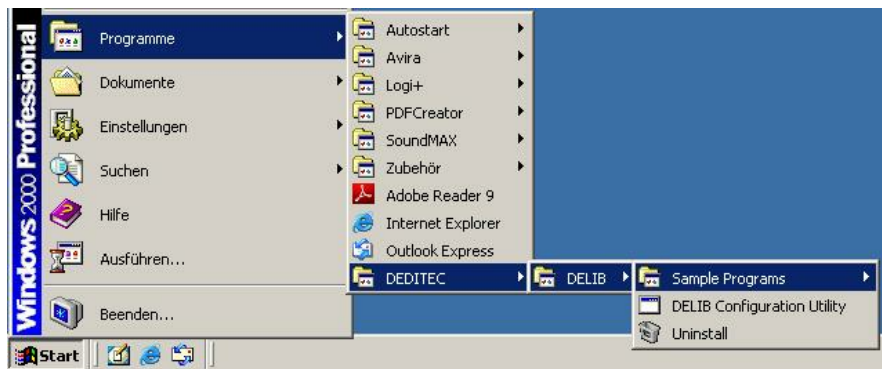


“Digital Input-Output Demo” wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → Digital Input-Output Demo.

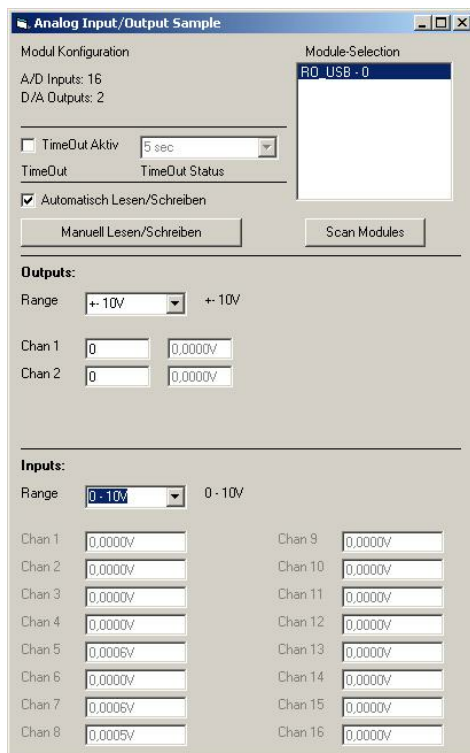


Diese Grafik zeigt einen Test des RO-USB-O64-R64. Oben links kann man die Konfiguration des Moduls ablesen (64 Eingänge und 64 Ausgänge).

3.3.2. Analog Input-Output Demo



“Analog Input-Output Demo” wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → Analog Input-Output Demo.



Diese Grafik zeigt einen Test des RO-USB-AD16-DA4. Oben links kann man die Konfiguration des Moduls ablesen (16 A/D-Eingänge und 4 D/A-Ausgänge).

DELIB API Referenz

IV

4. DELIB API Referenz

4.1. Verwaltungsfunktionen

4.1.1. DapiOpenModule

Beschreibung

Diese Funktion öffnet ein bestimmtes Modul.

Definition

ULONG DapiOpenModule(ULONG moduleID, ULONG nr);

Parameter

moduleID=Gibt das Modul an, welches geöffnet werden soll (siehe delib.h)

nr=Gibt an, welches (bei mehreren Modulen) geöffnet werden soll.

nr=0 -> 1. Modul

nr=1 -> 2. Modul

Return-Wert

handle=Entsprechender Handle für das Modul

handle=0 -> Modul wurde nicht gefunden

Bemerkung

Der von dieser Funktion zurückgegebene Handle wird zur Identifikation des Moduls für alle anderen Funktionen benötigt.

Programmierbeispiel

```
// USB-Modul öffnen
handle = DapiOpenModule(RO_USB1, 0);
printf("handle = %x\n", handle);
if (handle==0)
{
// USB Modul wurde nicht gefunden
printf("Modul konnte nicht geöffnet werden\n");
return;
}
```

4.1.2. DapiCloseModule

Beschreibung

Dieser Befehl schliesst ein geöffnetes Modul.

Definition

ULONG DapiCloseModule(ULONG handle);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

Return-Wert

Keiner

Programmierbeispiel

```
// Modul schliessen  
DapiCloseModule (handle);
```

4.2. Digitale Eingänge lesen

4.2.1. DapiDIGet1

Beschreibung

Dieser Befehl liest einen einzelnen digitalen Eingang.

Definition

ULONG DapiDIGet1(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangs an, der gelesen werden soll (0, 1, 2, 3, ..)

Return-Wert

Zustand des Eingangs (0/1)

4.2.2. DapiDIGet8

Beschreibung

Dieser Befehl liest gleichzeitig 8 digitale Eingänge.

Definition

ULONG DapiDIGet8(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 8, 16, 24, ..)

Return-Wert

Zustand der gelesenen Eingänge

4.2.3. DapiDIGet16

Beschreibung

Dieser Befehl liest gleichzeitig 16 digitale Eingänge.

Definition

ULONG DapiDIGet16(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 16, 32, ...)

Return-Wert

Zustand der gelesenen Eingänge

4.2.4. DapiDIGet32

Beschreibung

Dieser Befehl liest gleichzeitig 32 digitale Eingänge.

Definition

ULONG DapiDIGet32(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 32, 64, ..)

Return-Wert

Zustand der gelesenen Eingänge

Programmierbeispiel

```
unsigned long data;
// -----
// Einen Wert von den Eingängen lesen (Eingang 1-31)
data = (unsigned long) DapiDIGet32(handle, 0);
// Chan Start = 0
printf("Eingang 0-31 : 0x%x\n", data);
printf("Taste für weiter\n");
getch();
// -----
// Einen Wert von den Eingängen lesen (Eingang 32-64)
data = (unsigned long) DapiDIGet32(handle, 32);
// Chan Start = 32
printf("Eingang 32-64 : 0x%x\n", data);
printf("Taste für weiter\n");
getch();
```

4.2.5. DapiDIGet64

Beschreibung

Dieser Befehl liest gleichzeitig 64 digitale Eingänge.

Definition

ULONGLONG DapiDIGet64(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 64, ..)

Return-Wert

Zustand der gelesenen Eingänge

4.2.6. DapiDIGetFF32

Beschreibung

Dieser Befehl liest die Flip-Flops der Eingänge aus und setzt diese zurück (Eingangszustands-Änderung).

Definition

ULONGLONG DapiDIGet64(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 32, 64, ..)

Return-Wert

Zustand von 32 Eingangszustandsänderungen

4.3. Digitale Ausgänge verwalten

4.3.1. DapiDOSet1

Beschreibung

Dieser Befehl setzt einen einzelnen Ausgang.

Definition

```
void DapiDOSet1(ULONG handle, ULONG ch, ULONG data);
```

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des zu setzenden Ausgangs an (0 ..)

data=Gibt den Datenwert an, der geschrieben wird (0 / 1)

Return-Wert

Keiner

4.3.2. DapiDOSet8

Beschreibung

Dieser Befehl setzt gleichzeitig 8 digitale Ausgänge.

Definition

void DapiDOSet8(ULONG handle, ULONG ch, ULONG data);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 8, 16, 24, 32, ..)

data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert

Keiner

4.3.3. DapiDOSet16

Beschreibung

Dieser Befehl setzt gleichzeitig 16 digitale Ausgänge.

Definition

void DapiDOSet16(ULONG handle, ULONG ch, ULONG data);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 16, 32, ..)

data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert

Keiner

4.3.4. DapiDOSet32

Beschreibung

Dieser Befehl setzt gleichzeitig 32 digitale Ausgänge.

Definition

void DapiDOSet32(ULONG handle, ULONG ch, ULONG data);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 32, 64, ..)

data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert

Keiner

Programmierbeispiel

```
// Einen Wert auf die Ausgänge schreiben
data = 0x0000ff00; // Ausgänge 9-16 werden auf 1 gesetzt
DapiDOSet32(handle, 0, data); // Chan Start = 0
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
// -----
// Einen Wert auf die Ausgänge schreiben
data = 0x80000000; // Ausgang 32 wird auf 1 gesetzt
DapiDOSet32(handle, 0, data); // Chan Start = 0
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
// -----
// Einen Wert auf die Ausgänge schreiben
data = 0x80000000; // Ausgang 64 wird auf 1 gesetzt
DapiDOSet32(handle, 32, data); // Chan Start = 32
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
```

4.3.5. DapiDOSet64

Beschreibung

Dieser Befehl setzt gleichzeitig 64 digitale Ausgänge.

Definition

void DapiDOSet64(ULONG handle, ULONG ch, ULONG data);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 64, ..)

data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert

Keiner

4.3.6. DapiDOReadback32

Beschreibung

Dieser Befehl liest die 32 digitalen Ausgänge zurück.

Definition

ULONG DapiDOReadback32(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem zurückgelesen werden soll (0, 32, 64, ..)

Return-Wert

Zustand von 32 Ausgängen.

4.3.7. DapiDOReadback64

Beschreibung

Dieser Befehl liest die 64 digitalen Ausgänge zurück.

Definition

ULONGLONG DapiDOReadback64(ULONG handle, ULONG ch);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem zurückgelesen werden soll (0, 64, ..)

Return-Wert

Zustand von 64 Ausgängen.

4.4. Programmier-Beispiel

```
// *****  
// *****  
// *****  
// *****  
// *****  
//  
// (c) DEDITEC GmbH, 2009  
//  
// web: http://www.deditec.de  
//  
// mail: vertrieb@deditec.de  
//  
//  
// dtapi_prog_beispiel_input_output.cpp  
//  
//  
// *****  
// *****  
// *****  
// *****  
// *****  
//  
//  
// Folgende Bibliotheken beim Linken mit einbinden: delib.lib  
// Dies bitte in den Projekteinstellungen (Projekt/Einstellungen/Linker (Objekt-  
// Bibliothek-Module) .. letzter Eintrag konfigurieren  
#include <windows.h>  
#include <stdio.h>  
#include "conio.h"  
#include "delib.h"  
// -----  
// -----  
// -----  
// -----  
// -----  
  
void main(void)  
{  
    unsigned long handle;  
    unsigned long data;  
    unsigned long anz;  
    unsigned long i;  
    unsigned long chan;  
    // -----  
    // USB-Modul öffnen  
    handle = DapiOpenModule(USB_Interface8,0);  
    printf("USB_Interface8 handle = %x\n", handle);  
    if (handle==0)  
    {  
        // USB Modul wurde nicht gefunden  
        printf("Modul konnte nicht geöffnet werden\n");  
        printf("TASTE für weiter\n");  
        getch();  
    }
```

```

return;
}
// Zum Testen - ein Ping senden
// -----
printf("PING\n");
anz=10;
for(i=0;i!=anz;++i)
{
data=DapiPing(handle, i);
if(i==data)
{
// OK
printf(".");
}
else
{
// No answer
printf("E");
}
}
printf("\n");

// -----
// Einen Wert auf die Ausgänge schreiben
data = 255;
DapiWriteByte(handle, 0, data);
printf("Schreibe auf Adresse=0 daten=0x%x\n", data);
// -----
// Einen Wert auf die Ausgänge schreiben
data = 255;
DapiWriteByte(handle, 1, data);
printf("Schreibe auf Adresse=1 daten=0x%x\n", data);
// -----
// Einen Wert auf die Ausgänge schreiben
data = 255;
DapiWriteByte(handle, 2, data);
printf("Schreibe auf Adresse=2 daten=0x%x\n", data);
// -----
// Einen Wert von den Eingängen lesen
data = (unsigned long) DapiReadByte(handle, 0);
printf("Gelesene Daten = 0x%x\n", data);
// -----
// Einen A/D Wert lesen
chan=11; // read chan. 11
data = DapiReadWord(handle, 0xff010000 + chan*2);
printf("Adress=%x, ret=%x volt=%f\n", chan, data, ((float) data) / 1024*5); //
Bei 5 Volt Ref
// -----
// Modul wieder schliessen
DapiCloseModule(handle);
printf("TASTE für weiter\n");
getch();
return ;
}

```

Anhang



5. Anhang

5.1. Revisionen

Rev 1.00	Erste DEDITEC Anleitung
Rev 2.00	Designänderung
Rev 2.01	Ergänzung Technische Daten

5.2. Urheberrechte und Marken

Linux ist eine registrierte Marke von Linus Torvalds.

Windows CE ist eine registrierte Marke von Microsoft Corporation.

USB ist eine registrierte Marke von USB Implementers Forum Inc.

LabVIEW ist eine registrierte Marke von National Instruments.

Intel ist eine registrierte Marke von Intel Corporation

AMD ist eine registrierte Marke von Advanced Micro Devices, Inc.