

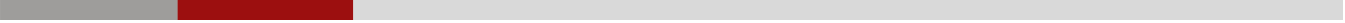


USB-OPTOIN-X-RE LAIS-X

Hardware-Beschreibung

1 Einleitung	5
1.1 Vorwort	5
1.2 Kundenzufriedenheit	5
1.3 Kundenresonanz	5
2 Hardware Beschreibung	5
2.1 Einführung	5
2.2 Kurzanleitung Installation	7
2.2.1 Schritt 1 - Installation der Software und Treiber	7
2.2.2 Schritt 2 - Anschluss des Moduls	7
2.3 Technische Daten	8
2.4 Übersichtsbilder	9
2.4.1 Übersichtsbild USB-OPTOIN-8-RELAIS-8	9
2.4.2 Übersichtsbild USB-OPTOIN-16-RELAIS-16	10
2.4.3 Übersichtsbild USB-OPTOIN-32-RELAIS-32	11
2.4.4 Übersichtsbild USB-OPTOIN-32	12
2.4.5 Übersichtsbild USB-RELAIS-32	13
2.4.6 Übersichtsbild USB-OPTOIN-64	14
2.4.7 Übersichtsbild USB-RELAIS-64	15
2.5 Pinbelegung	16
2.5.1 Pinbelegung Ausgänge	16
2.5.2 Pinbelegung Eingänge	17
2.6 J2 - Konfiguration der Spannungsversorgung	18
2.7 Ausgänge	19
2.7.1 Relais Ausgänge	19
2.7.2 Timeout-Schutz	19
2.7.3 Visuelle Kontrolle der Ausgänge	19
2.8 Eingänge	20
2.8.1 Erfassen von schnellen Eingangsimpulsen	20
2.8.2 Galvanische Trennung durch Optokoppler	20
2.8.3 Visuelle Kontrolle der Eingänge	20
3 Software	21
3.1 Benutzung unserer Produkte	21
3.1.1 Ansteuerung über grafische Anwendungen	21
3.1.2 Ansteuerung über unsere DELIB Treiberbibliothek	21
3.1.3 Ansteuerung auf Protokollebene	21
3.1.4 Ansteuerung über mitgelieferte Testprogramme	22

3.2 DELIB Treiberbibliothek	23
3.2.1 Übersicht	23
3.2.1.1 Programmieren unter diversen Betriebssystemen	23
3.2.1.2 Programmieren mit diversen Programmiersprachen	24
3.2.1.3 Schnittstellenunabhängiges programmieren	24
3.2.1.4 SDK-Kit für Programmierer	24
3.2.2 Unterstützte Betriebssysteme	25
3.2.3 Unterstützte Programmiersprachen	25
3.2.4 Installation DELIB-Treiberbibliothek	26
4 DELIB API Referenz	29
4.1 Verwaltungsfunktionen	29
4.1.1 DapiOpenModule	29
4.1.2 DapiCloseModule	30
4.1.3 DapiGetDELIBVersion	31
4.1.4 DapiSpecialCMDGetModuleConfig	32
4.2 Fehlerbehandlung	34
4.2.1 DapiGetLastError	34
4.2.2 DapiGetLastErrorText	35
4.3 Digitale Eingänge lesen	36
4.3.1 DapiDIGet1	36
4.3.2 DapiDIGet8	37
4.3.3 DapiDIGet16	38
4.3.4 DapiDIGet32	39
4.3.5 DapiDIGet64	40
4.3.6 DapiDIGetFF32	41
4.3.7 DapiDIGetCounter	42
4.4 Digitale Ausgänge verwalten	43
4.4.1 DapiDOSet1	43
4.4.2 DapiDOSet8	44
4.4.3 DapiDOSet16	45
4.4.4 DapiDOSet32	46
4.4.5 DapiDOSet64	47
4.4.6 DapiDOReadback32	48
4.4.7 DapiDOReadback64	49
4.5 Ausgabe-Timeout verwalten	50
4.5.1 DapiSpecialCMDTimeout	50
4.5.2 DapiSpecialCMDTimeoutGetStatus	51
5 Programmier-Beispiel	52
6 Anhang	55



6.1 Revisionen	55
6.2 Urheberrechte und Marken	56

1 Einleitung

1.1 Vorwort

Wir beglückwünschen Sie zum Kauf eines hochwertigen 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!

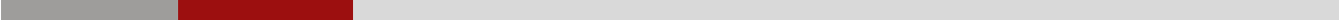
1.3 Kundenresonanz

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

2 Hardware Beschreibung

2.1 Einführung

Die USB-OPTOIN-X-RELAIS-X Module verfügen über Relais mit einer maximalen Schaltspannung von 36V DC (max. 1A, 15 Watt) sowie über Opto-In Eingänge, die sich für den industriellen Einsatz zur Erfassung von Zuständen oder auch zum Zählen von Zustandsänderungen der Eingänge eignen.



Unsere USB Module sind für den industriellen Einsatz zur Messung, Steuerung und Regelung entwickelt worden. Die Module verfügen alle über ein USB-Interface und können daher an PC-Systeme mit USB-Bus angeschlossen werden. Der USB-Bus hat sich seit vielen Jahren im Einsatz bewährt und zeichnet sich durch seine hohe Flexibilität aus.

Als Anschlussklemmen kommen servicefreundliche Steckleisten mit Verriegelungsschutz und Auswerfmechanik zum Einsatz. Diese ermöglichen ein schnelles, nachträgliches Umstecken der angeschlossenen Anlagen. Der Leitungsanschluß selbst erfolgt über ein schraubenloses Stecksystem.



2.2 Kurzanleitung Installation

2.2.1 Schritt 1 - Installation der Software und Treiber

Installieren Sie nun die DELIB-Treiberbibliothek mit der Datei "delib_install.exe" von der im Lieferumfang enthaltenen Treiber CD.

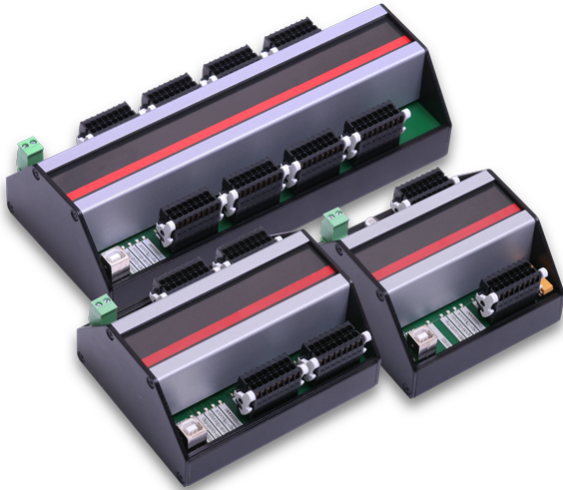
Diese finden Sie im Verzeichnis "\zip\delib\delib_install.exe" der Treiber CD.

2.2.2 Schritt 2 - Anschluss des Moduls

Verbinden Sie ihren PC per USB-Kabel mit dem USB-Anschluss des Moduls.

Nach etwa 20 Sekunden wird das Modul vom Treiber erkannt und kann nun getestet und betrieben werden.

2.3 Technische Daten



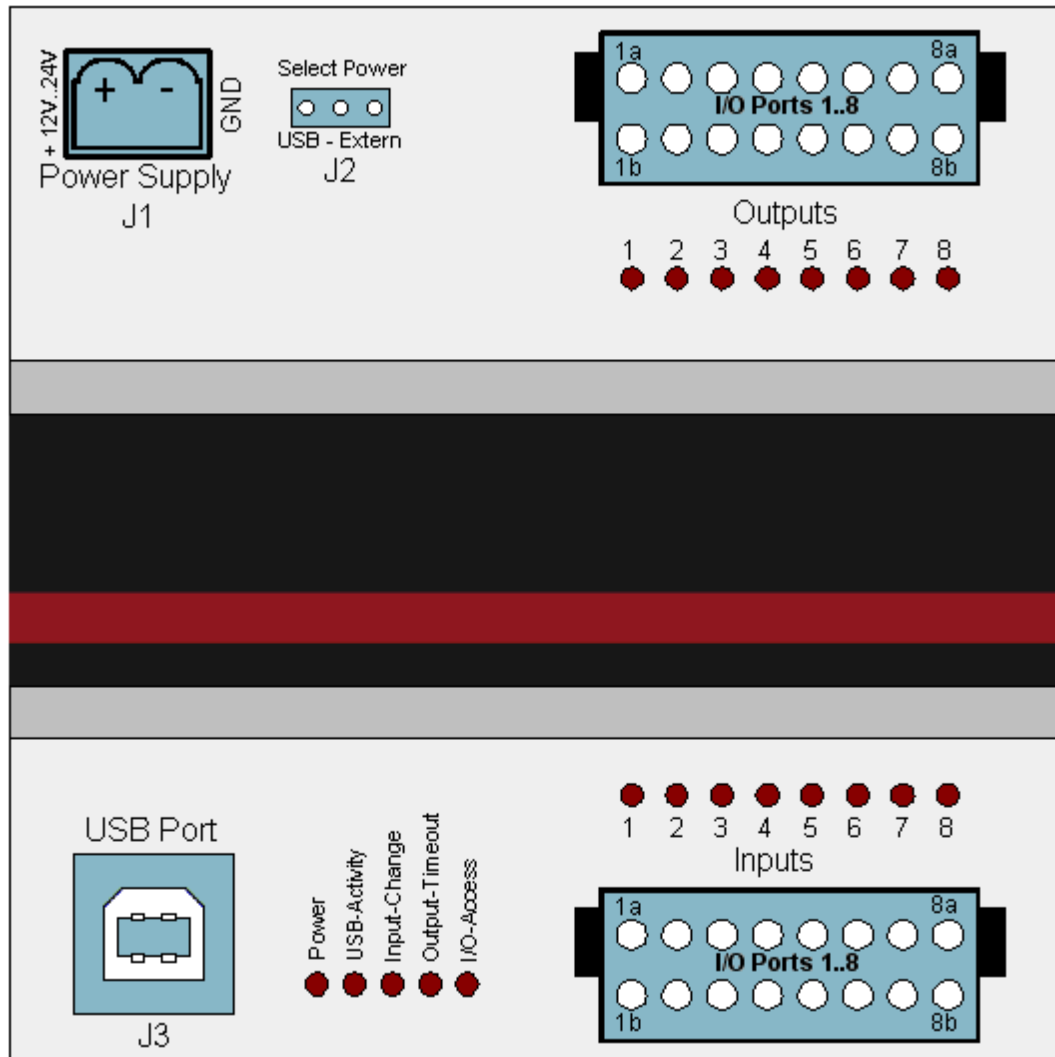
- USB-Interface (USB 1.1 / USB 2.0)
- Spannungsversorgung: +5V (wird über USB-Bus versorgt)
- 8/16/32/64 Optokoppler Eingänge je nach Modul
- Variabler Eingangsspannungsbereich min. 5V, max. 30V AC (Standard: 15-30V)
- Galvanisch getrennt durch Optokoppler
- Erfassung von Impulsen zwischen 2 Auslesetakten
- 8/16/32/64 Relais Ausgänge je nach Modul (36V, 1A, 15W, Schließer-Relais)
- Timeout Schutz
- Galvanisch getrennt durch Relais
- Betriebstemperatur: 10°C .. 50°C

Produktspezifische Daten:

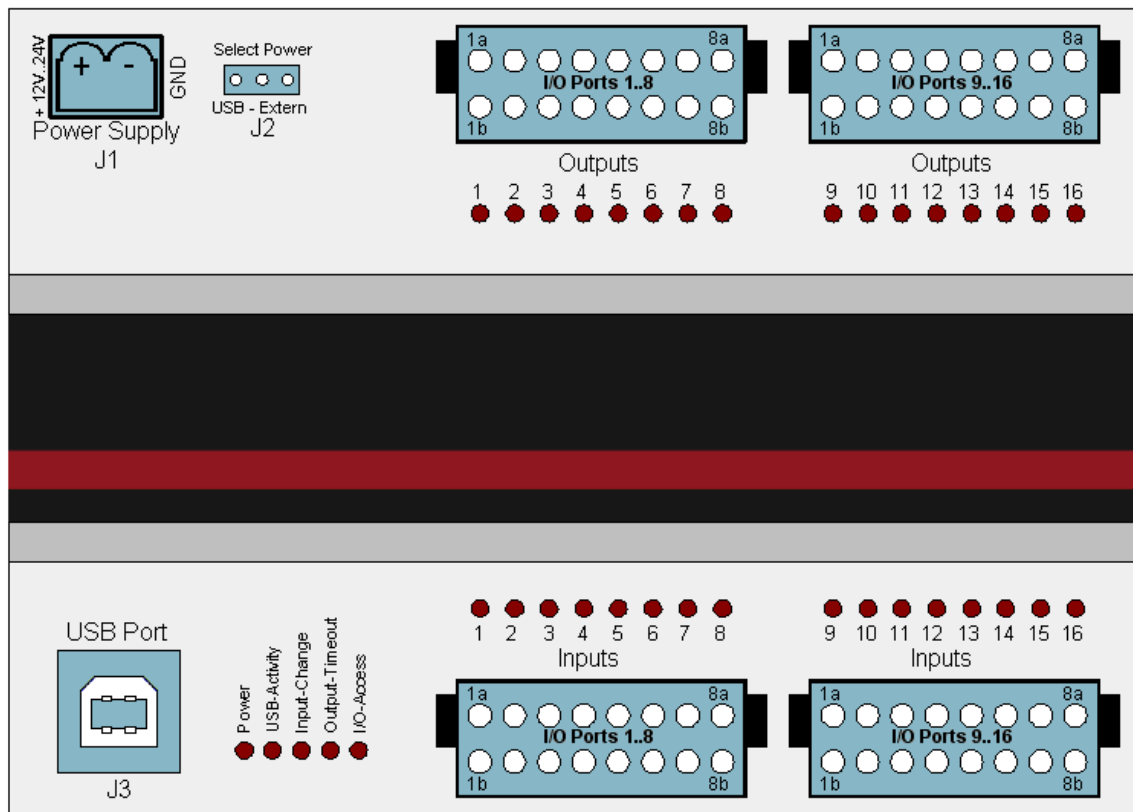
Produkt	Eingänge	Ausgänge	Länge x Breite x Höhe in mm
USB-OPTOIN-8-RELAIS-8	8	8	102,5 x 105 x 74,5
USB-OPTOIN-16-RELAIS-16	16	16	125 x 105 x 74,5
USB-OPTOIN-32	32	-	
USB-RELAIS-32	-	32	230 x 105 x 74,5
USB-OPTOIN-32-RELAIS-32	32	32	
USB-OPTOIN-64	64	-	
USB-RELAIS-64	-	64	

2.4 Übersichtsbilder

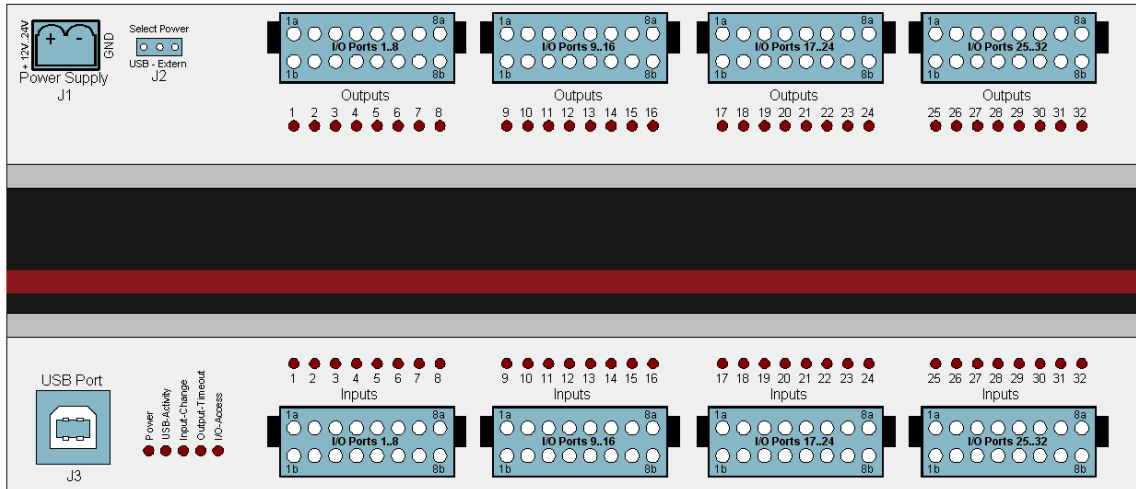
2.4.1 Übersichtsbild USB-OPTOIN-8-RELAIS-8



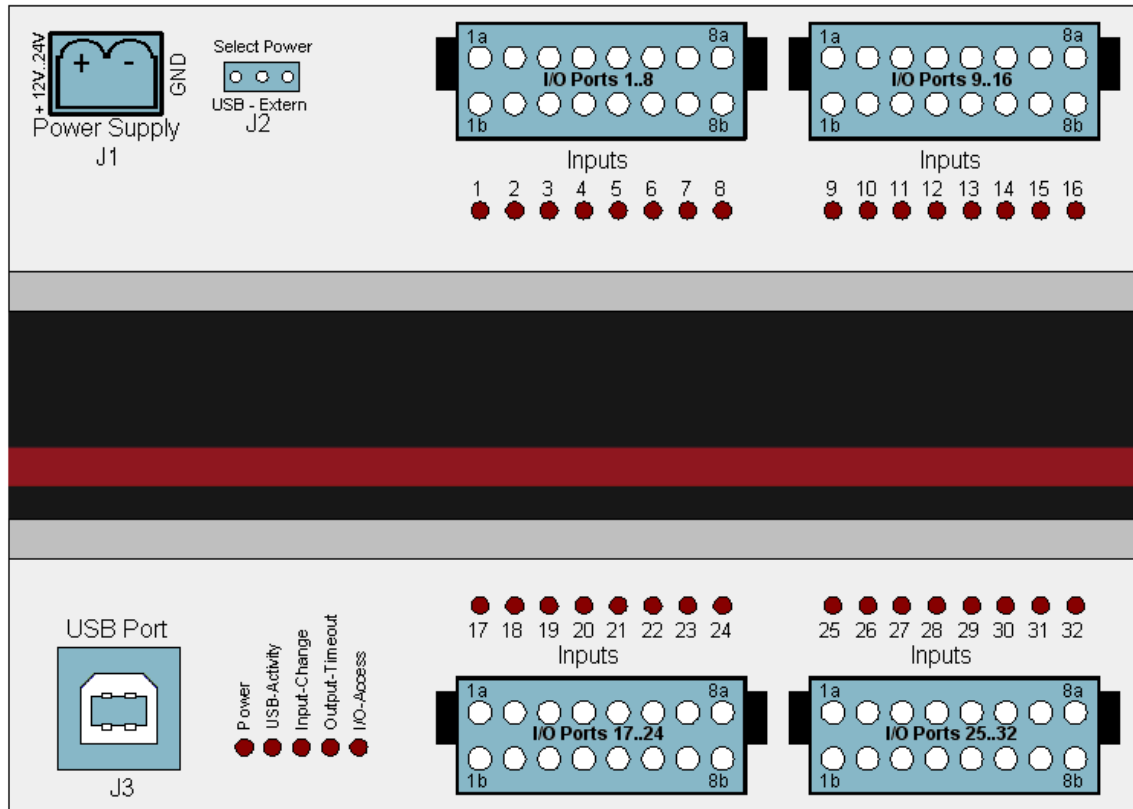
2.4.2 Übersichtsbild USB-OPTOIN-16-RELAIS-16



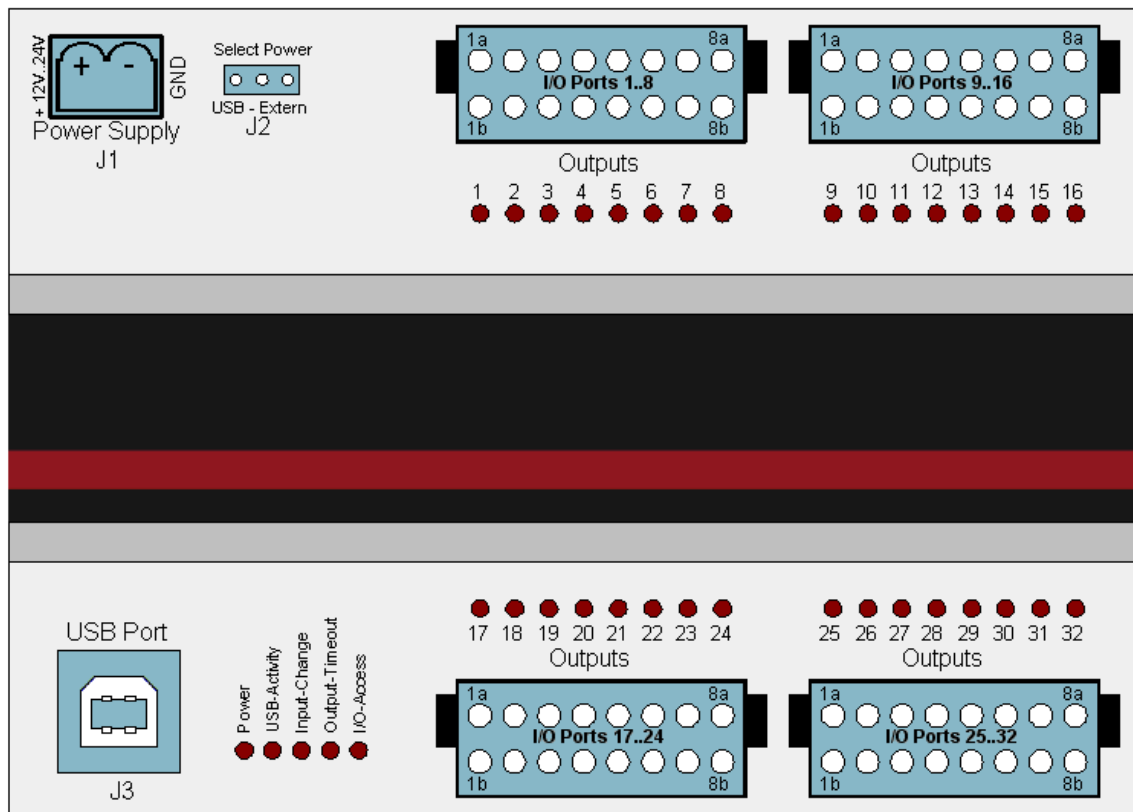
2.4.3 Übersichtsbild USB-OPTOIN-32-RELAIS-32



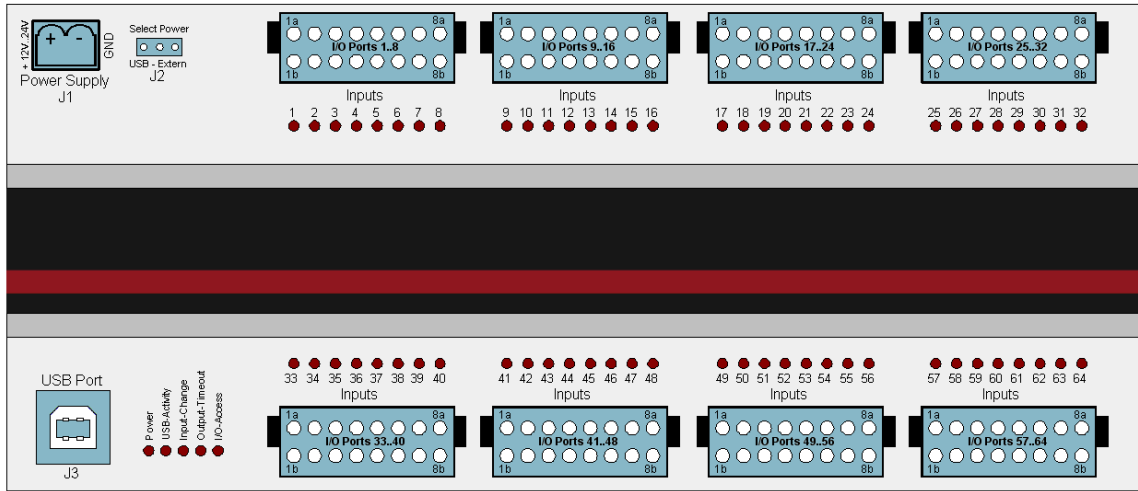
2.4.4 Übersichtsbild USB-OPTOIN-32



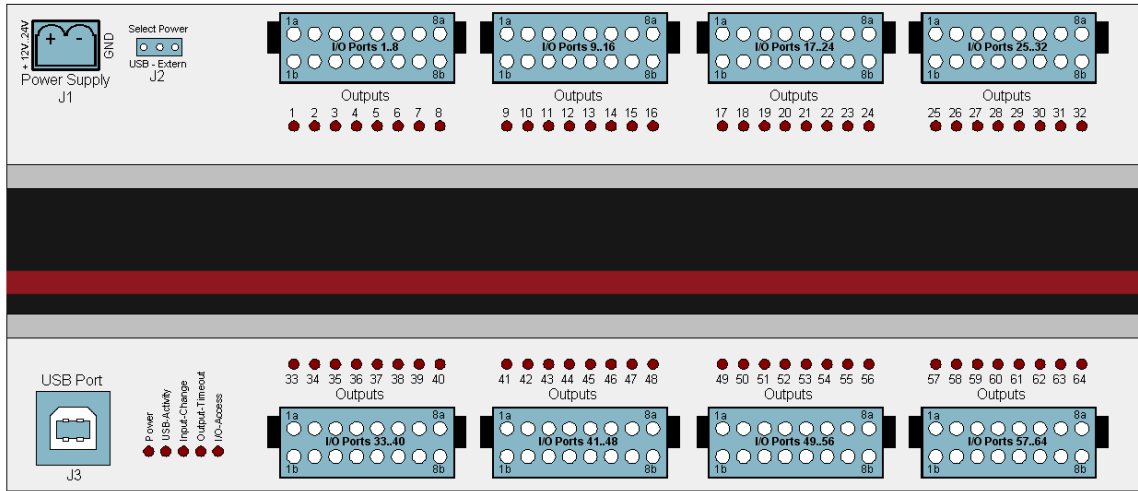
2.4.5 Übersichtsbild USB-RELAIS-32



2.4.6 Übersichtsbild USB-OPTOIN-64

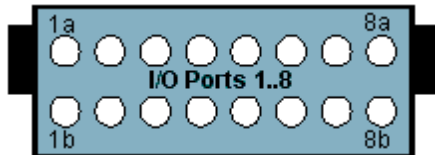


2.4.7 Übersichtsbild USB-RELAIS-64



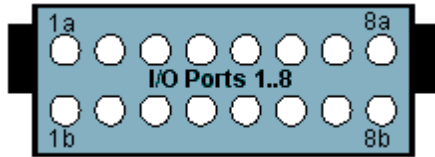
2.5 Pinbelegung

2.5.1 Pinbelegung Ausgänge



Pin	Belegung
1a	Output Channel 1
1b	Output Channel 1
2a	Output Channel 2
2b	Output Channel 2
3a	Output Channel 3
3b	Output Channel 3
4a	Output Channel 4
4b	Output Channel 4
5a	Output Channel 5
5b	Output Channel 5
6a	Output Channel 6
6b	Output Channel 6
7a	Output Channel 7
7b	Output Channel 7
8a	Output Channel 8
8b	Output Channel 8

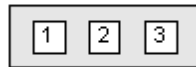
2.5.2 Pinbelegung Eingänge



Pin	Belegung
1a	Input Channel 1
1b	Input Channel 1
2a	Input Channel 2
2b	Input Channel 2
3a	Input Channel 3
3b	Input Channel 3
4a	Input Channel 4
4b	Input Channel 4
5a	Input Channel 5
5b	Input Channel 5
6a	Input Channel 6
6b	Input Channel 6
7a	Input Channel 7
7b	Input Channel 7
8a	Input Channel 8
8b	Input Channel 8

2.6 J2 - Konfiguration der Spannungsversorgung

Select Power J2



USB - Extern

Die Konfiguration der Spannungsversorgung erfolgt über einen Jumper. (J2)

Versorgung über den USB-Bus (J3) des Modules

Jumperstellung Pin 1 & 2

(+5V DC)

Anmerkung: Bei zuvielen Relais kann eine ausreichende Spannungsversorgung über die USB-Schnittstelle nicht immer gewährleistet werden.

Versorgung durch eine externe Spannungszufuhr (J1)

Jumperstellung Pin 2 & 3

(+7-24V DC)

2.7 Ausgänge

2.7.1 Relais Ausgänge

Durch den Einsatz von Relais lassen sich Spannungen von bis zu 36V schalten. Die maximale Strombelastbarkeit beträgt 1A bei einer maximalen Schaltleistung von 15W.

Außerdem sorgen die Relais für eine sichere galvanische Trennung des Moduls von den angeschlossenen Anlagen.

2.7.2 Timeout-Schutz

Der Timeout-Schutz bietet die Möglichkeit die Ausgänge selbstständig abzuschalten. Dies geschieht immer dann, wenn in einem vorher definierten Zeitfenster keine Nachrichten mehr vom Modul empfangen werden. Gründe können sein: Leitungsunterbrechung, PC / Serverabsturz usw. Dadurch können Steuerungsschäden, Überlastung der angeschlossenen Anlagen und Unfallgefahren verhindert werden.

2.7.3 Visuelle Kontrolle der Ausgänge

Über eine LED wird der Zustand jedes Ausgangs direkt angezeigt. Signale an den Ausgängen sind somit einfacher zu erkennen und Fehler in der Verdrahtung lassen sich dadurch schneller beheben.

2.8 Eingänge

2.8.1 Erfassen von schnellen Eingangsimpulsen

Schnelle Zustandswechsel an den Eingängen, die innerhalb von größeren Auslesezyklen auftreten, werden durch eine zusätzliche Logik erfasst und können separat per Software ausgelesen werden.

2.8.2 Galvanische Trennung durch Optokoppler

Wechselspannungs geeignete Eingangs-Optokoppler sorgen zum einem für eine galvanische Trennung des Moduls zu den angeschlossenen Anlagen und zum anderen verhindert man eine Beschädigung des Moduls bei verpoltem Anschluss oder auftretenden Spannungspitzen o.ä. im Steuerstromkreis.

2.8.3 Visuelle Kontrolle der Eingänge

Über eine LED wird der Zustand jedes Eingangs direkt angezeigt. Signale an den Eingängen sind somit einfacher zu erkennen und Fehler in der Verdrahtung lassen sich dadurch schneller beheben.

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

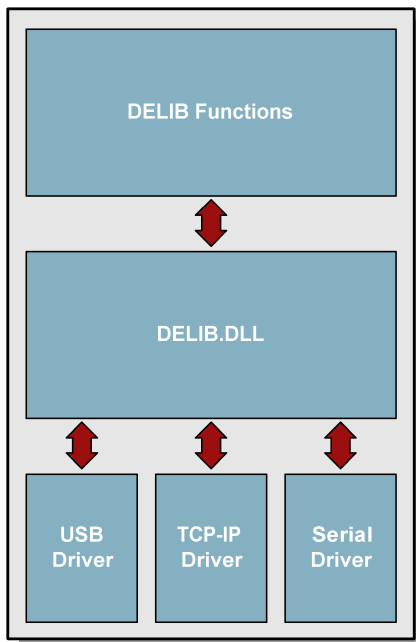
Für die wichtigsten Funktionen unserer Produkte stellen wir einfach zu bedienende Testprogramme zur Verfügung,. Diese werden bei der Installation der DELIB Treiberbibliothek direkt mit installiert.

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 der Module, mit der besonderen Berücksichtigung folgender Gesichtspunkte:

- Betriebssystem unabhängig
- Programmiersprachen unabhängig
- Produkt unabhängig

3.2.1.1 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 !

3.2.1.2 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.

3.2.1.3 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 !

3.2.1.4 SDK-Kit für Programmierer

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

3.2.2 Unterstützte Betriebssysteme

Unsere Produkte unterstützen folgende Betriebssysteme:

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

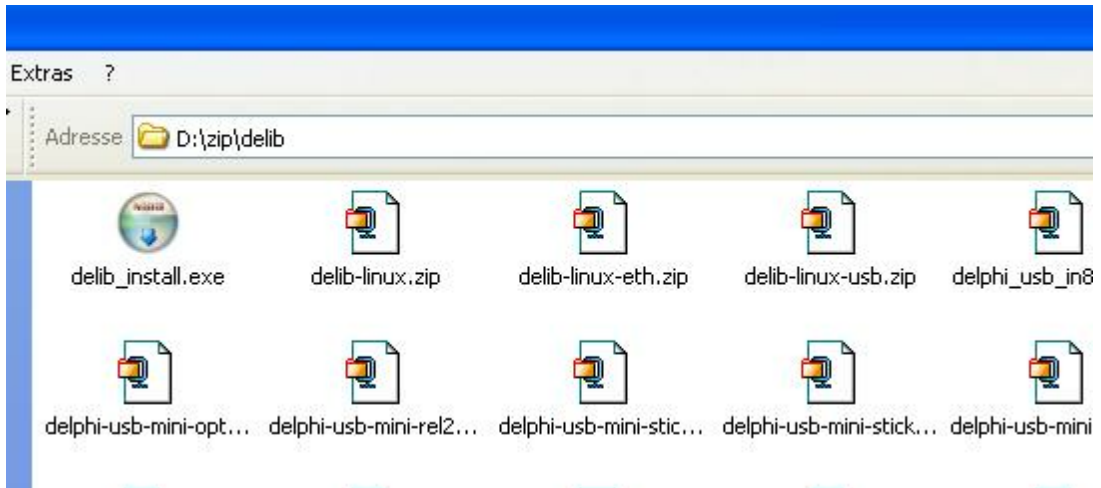
3.2.3 Unterstützte Programmiersprachen

Unsere Produkte sind über folgende Programmiersprachen ansprechbar:

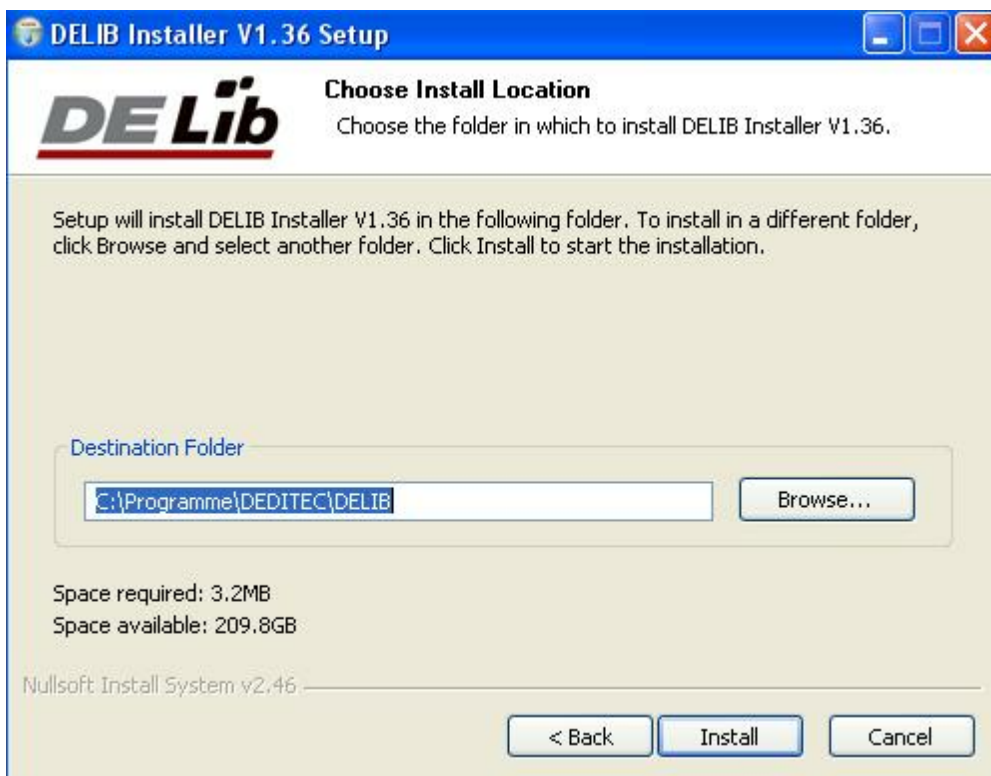
- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

3.2.4 Installation DELIB-Treiberbibliothek

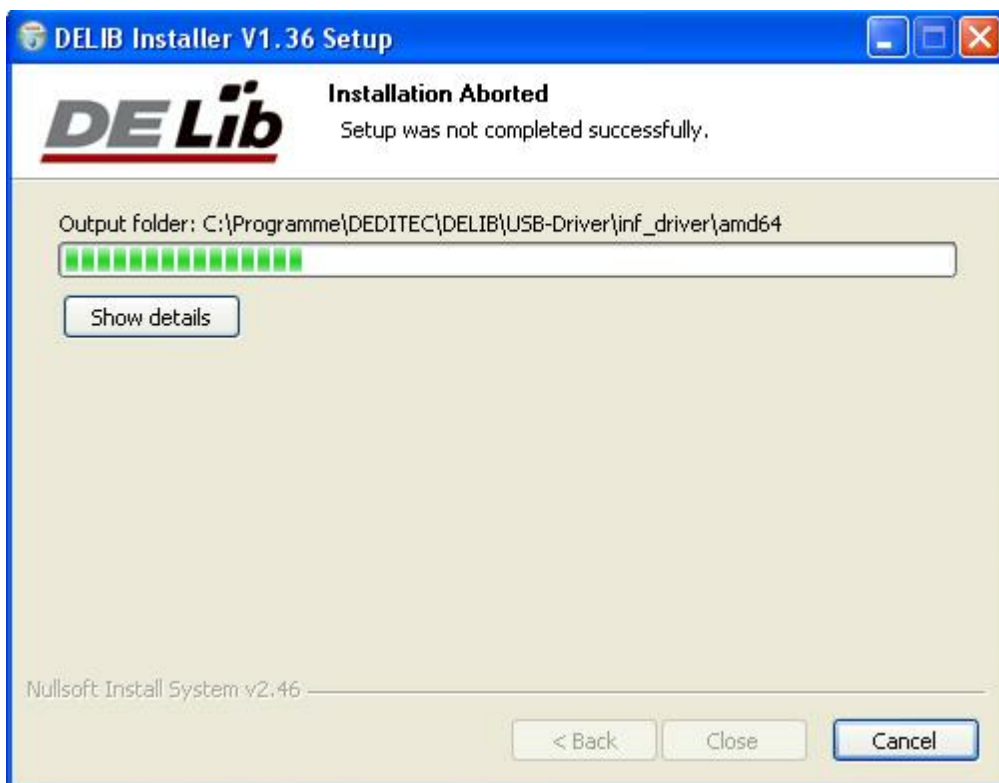
Startbild unseres DELIB Installers.



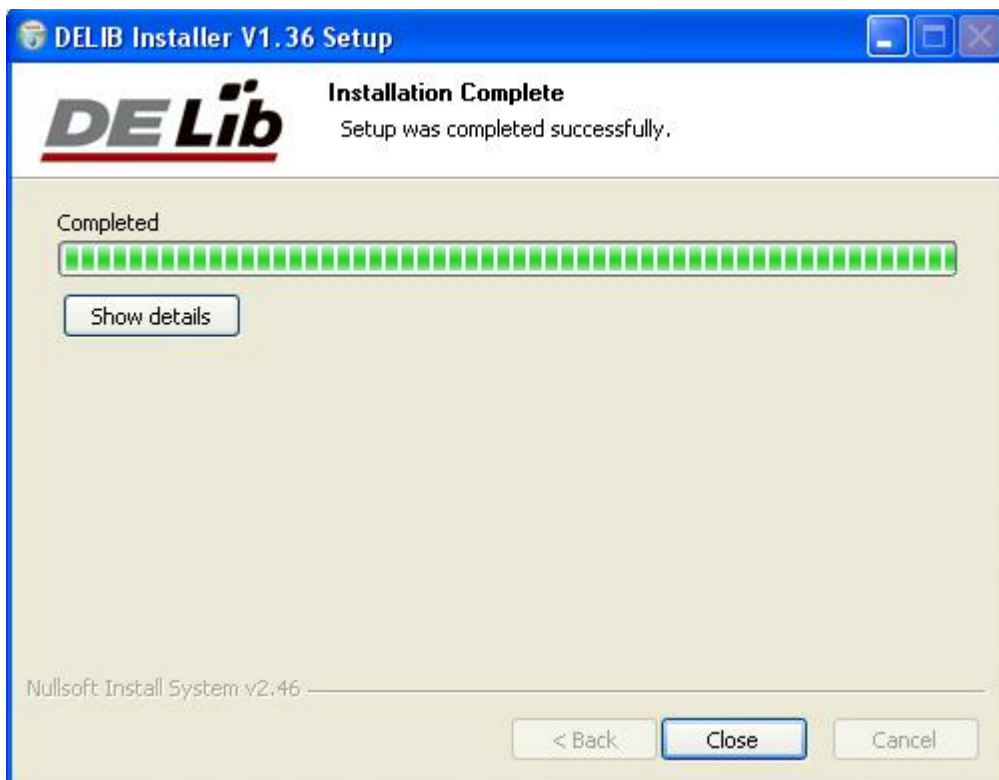
Legen Sie die Treiber-CD in das Laufwerk und starten Sie **"delib_install.exe"**.



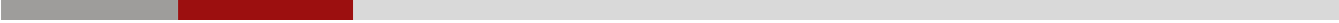
Drücken Sie auf **"Install"**.



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).

4 DELIB API Referenz

4.1 Verwaltungsfunktionen

4.1.1 DapiOpenModule

Description

This function opens a particular module.

Definition

ULONG DapiOpenModule(ULONG moduleID, ULONG nr);

Parameters

moduleID=Specifies the module, which is to be opened (see delib.h)

nr=Indicates No of module which is to be opened.

nr=0 -> 1. module

nr=1 -> 2. module

Return value

handle=handle to the corresponding module

handle=0 -> Module was not found

Remarks

The handle returned by this function is needed to identify the module for all other functions.

Example program

```
// 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

Description

This command closes an opened module.

Definition

ULONG DapiCloseModule(ULONG handle);

Parameters

handle=This is the handle of an opened module

Return value

none

Example program

```
// Close the module  
DapiCloseModule(handle);
```

4.1.3 DapiGetDELIBVersion

Description

This function returns the installed DELIB version.

Definition

ULONG DapiGetDELIBVersion(ULONG mode, ULONG par);

Parameters

mode=Mode, with which the version is readout (must be 0).

par=This parameter is not defined (must be 0).

Return value

version=Version number of the installed DELIB version [hex].

Example program

```
version = DapiGetDELIBVersion(0, 0);  
//Bei installierter Version 1.32 ist version = 132(hex)
```

4.1.4 DapiSpecialCMDGetModuleConfig

Beschreibung

Diese Funktion gibt die Hardwareaustattung (Anzahl der Ein- bzw. Ausgangskanäle) des Moduls zurück.

Definition

```
ULONG DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,  
par, 0, 0);
```

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

Anzahl der digitalen Eingangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI

Anzahl der digitalen Ausgangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO

Anzahl der digitalen Ein-/Ausgangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DX

Anzahl der analogen Eingangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_AD

Anzahl der analogen Ausgangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DA

Anzahl der Stepperkanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_STEPPER

Return-Wert

Anzahl der digitalen Eingangskanäle abfragen

return=Anzahl der digitalen Eingangskanäle

Anzahl der digitalen Ausgangskanäle abfragen

return=Anzahl der digitalen Ausgangskanäle

Anzahl der digitalen Ein-/Ausgangskanäle abfragen

return=Anzahl der digitalen Ein-/Ausgangskanäle

Anzahl der analogen Eingangskanäle abfragen

return=Anzahl der analogen Eingangskanäle

Anzahl der analogen Ausgangskanäle abfragen

return=Anzahl der analogen Ausgangskanäle

Anzahl der Stepperkanäle abfragen

return=Anzahl der Stepperkanäle

Programmierbeispiel

```
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,  
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI, 0, 0);  
//Gibt die Anzahl der digitalen Eingangskanäle zurück  
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,  
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO, 0, 0);  
//Gibt die Anzahl der digitalen Ausgangskanäle zurück  
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,  
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DX, 0, 0);  
//Gibt die Anzahl der digitalen Ein-/Ausgangskanäle zurück  
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,  
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_AD, 0, 0);  
//Gibt die Anzahl der analogen Eingangskanäle zurück  
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,  
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DA, 0, 0);  
//Gibt die Anzahl der analogen Ausgangskanäle zurück  
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,  
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_STEPPER, 0, 0);  
//Gibt die Anzahl der Stepperkanäle zurück
```

4.2 Fehlerbehandlung

4.2.1 DapiGetLastError

Description

This function returns the last registered error.

Definition

```
ULONG DapiGetLastError();
```

Parameters

None

Return value

Error code

0=no error. (see delib.h)

Example program

```
ULONG error;  
error=DapiGetLastError();  
if(error==0) return FALSE;  
printf("ERROR = %d", error);
```

4.2.2 DapiGetLastErrorText

Description

This function reads the text of the last registered error.

Definition

```
extern ULONG __stdcall DapiGetLastErrorText(unsigned char * msg, unsigned long msg_length);
```

Parameters

msg = text buffer

msg_length = length of the buffer

Example program

```
BOOL IsError ()
{
    if (DapiGetLastError () != DAPI_ERR_NONE)
    {
        unsigned char msg[500];

        DapiGetLastErrorText((unsigned char*) msg, sizeof(msg));
        printf ("Error Code = %x * Message = %s\n", 0, msg);
        return TRUE;
    }
    return FALSE;
}
```

4.3 Digitale Eingänge lesen

4.3.1 DapiDIGet1

Description

This command reads a single digit input.

Definition

ULONG DapiDIGet1(ULONG handle, ULONG ch);

Parameters

handle=This is the handle of an opened module.

ch=Specifies the number of input that is to be read (0 ..).

Return value

State of the input (0 / 1).

4.3.2 DapiDIGet8

Description

This command reads 8 digital inputs simultaneously.

Definition

ULONG DapiDIGet8(ULONG handle, ULONG ch);

Parameters

handle=This is the handle of an opened module.

ch=Specifies the number of the input, from which it begins to read from (0, 8, 16, 24, 32, ..)

Return value

State of the read inputs.

4.3.3 DapiDIGet16

Description

This command reads 16 digital inputs simultaneously.

Definition

ULONG DapiDIGet16(ULONG handle, ULONG ch);

Parameters

handle=This is the handle of an opened module.

ch=Specifies the number of the input, from which it begins to read from (0, 16, 32, ..)

Return value

State of the read inputs.

4.3.4 DapiDIGet32

Description

This command reads 32 digital inputs simultaneously.

Definition

ULONG DapiDIGet32(ULONG handle, ULONG ch);

Parameters

handle=This is the handle of an opened module.

ch=Specifies the number of the input, from which it begins to read from (0, 32, 64, ..)

Return value

State of the read inputs.

Example program

```
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();
```

4.3.5 DapiDIGet64

Description

This command reads 64 digital inputs simultaneously.

Definition

ULONGLONG DapiDIGet64(ULONG handle, ULONG ch);

Parameters

handle=This is the handle of an opened module.

ch=Specifies the number of the input,from which it begins to read from (0, 64, ..)

Return value

State of the read inputs.

4.3.6 DapiDIGetFF32

Description

This command reads the flip-flops from the inputs and resets them. (Input state change).

Definition

ULONG DapiDIGetFF32(ULONG handle, ULONG ch);

Parameters

handle=This is the handle of an opened module .

ch=Specifies the number of the input, from which it begins to read from (0, 32, ..)

Return value

State of 32 input change states

4.3.7 DapiDIGetCounter

Description

This command reads the counter of a digital input

Definition

ULONG DapiDIGetCounter(ULONG handle, ULONG ch, ULONG mode);

Parameters

handle=This is the handle of an opened module.

ch=Specifies the digital input,from which the counter will be read.

mode=0 (Normal counter function)

mode=DAPI_CNT_MODE_READ_WITH_RESET (Reading and resetting the counter)

mode=DAPI_CNT_MODE_READ_LATCHED (Reading the latched counter)

Return value

Value of the counter.

Example program

```
value = DapiDIGetCounter(handle, 0 ,0);
// Reading counter of DI Chan 0

value = DapiDIGetCounter(handle, 1 ,0);
// Reading counter of DI Chan 1

value = DapiDIGetCounter(handle, 8 ,0);
// Reading counter of DI Chan 8

value = DapiDIGetCounter(handle, 0 ,DAPI_CNT_MODE_READ_WITH_RESET);
// Reading AND resetting counter of DI Chan 0

value = DapiDIGetCounter(handle, 1, DAPI_CNT_MODE_READ_LATCHED);
// Reading the latched counter of DI Chan 1
```

4.4 Digitale Ausgänge verwalten

4.4.1 DapiDOSet1

Description

This is the command to set a single output.

Definition

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

Parameters

handle=This is the handle of an opened module

ch=Specifies the number of the output to be set to (0 ..)

data=Specifies the data value that is to be written (0 / 1)

Return value

None

4.4.2 DapiDOSet8

Description

This command sets 8 digital outputs simultaneously.

Definition

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

Parameters

handle=This is the handle of an opened module

ch=Specifies the number of the output, from which it begins to write to (0, 8, 16, 24, 32, ..)

data=Specifies the data values, to write to the outputs

Return value

None

4.4.3 DapiDOSet16

Description

This command sets 16 digital outputs simultaneously.

Definition

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

Parameters

handle=This is the handle of an opened module

ch=Specifies the number of the output, from which it begins to write to (0, 16, 32, ..)

data=Specifies the data values, to write to the outputs

Return value

None

4.4.4 DapiDOSet32

Description

This command sets 32 digital outputs simultaneously.

Definition

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

Parameters

handle=This is the handle of an opened module

ch=Specifies the number of the output, from which it begins to write to (0, 32, 64, ..)

data=Specifies the data values, to write to the outputs

Return value

None

Example program

```
// 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();
```

4.4.5 DapiDOSet64

Description

This command is to set 64 digital outputs.

Definition

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

Parameters

handle=This is the handle of an opened module

ch=Specifies the number of the output, from which it begins to write to (0, 64, ..)

data=Specifies the data values, to write to the outputs

Return value

None

4.4.6 DapiDOReadback32

Description

This command reads back the 32 digital outputs.

Definition

ULONG DapiDOReadback32(ULONG handle, ULONG ch);

Parameters

handle=This is the handle of an opened module

ch=Specifies the number of the input, from which it begins to read from (0, 32, ..)

Return value

Status of 32 outputs.

4.4.7 DapiDOReadback64

Description

This command reads back the 64 digital outputs.

Definition

ULONGLONG DapiDOReadback64(ULONG handle, ULONG ch);

Parameters

handle=This is the handle of an opened module

ch=Specifies the number of the input, from which it begins to read from (0, 64, ..)

Return value

Status of 64 outputs.

4.5 Ausgabe-Timeout verwalten

4.5.1 DapiSpecialCMDTimeout

Description

This command serves to set the timeout time

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, par1, par2);

Parameters

handle=This is the handle of an opened module

Set timeout time

cmd=DAPI_SPECIAL_CMD_TIMEOUT_SET_VALUE_SEC

par1=Seconds [s]

par2=Milliseconds [100ms] (value 6 stands for 600ms)

Activate timeout

cmd=DAPI_SPECIAL_CMD_TIMEOUT_ACTIVATE

Deactivate timeout

cmd=DAPI_SPECIAL_CMD_TIMEOUT_DEACTIVATE

Return value

None

Example program

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,  
DAPI_SPECIAL_TIMEOUT_SET_VALUE_SEC, 3, 7);  
//Die Zeit des Timeouts wird auf 3,7sek gesetzt.  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,  
DAPI_SPECIAL_TIMEOUT_ACTIVATE, 0, 0);  
//Der Timeout wird aktiviert.  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,  
DAPI_SPECIAL_TIMEOUT_DEACTIVATE, 0, 0);  
//Der Timeout wird deaktiviert.
```

4.5.2 DapiSpecialCMDTimeoutGetStatus

Description

This command reads the timeout status.

Definition

```
ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,  
DAPI_SPECIAL_TIMEOUT_GET_STATUS, 0, 0);
```

Parameters

handle=This is the handle of an opened module

Return value

Return=0 (timeout is deactivated)

Return=1 (timeout is activated)

Return=2 (timeout has occurred)

Example program

```
status = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,  
DAPI_SPECIAL_TIMEOUT_GET_STATUS, 0, 0); //Abfrage des Timeout-Status.
```

5 Programmier-Beispiel

```
//*****
//*****
//*****
//*****
//*****
//
//
// product: usb-optoin-8-relais-8 (ModuleID = USB_OPTOIN_8_RELAIS_8)
// configuration: digital-outputs
// programming language: vc
//
//
//*****
//*****
//*****
//*****
//*****
//
//
// Please include the following library on linking: delib.lib
//
// This can be done at the project settings (Project/Settings/Link ->
// Object/library modules) .. extend the existing line with the ending
// "$(DELIB_LIB)\delib.lib" (with quotation marks)
//
// Including the header file delib.h (Project/Settings/C/C++ -> select
// category
// "Preprocessor" -> Additional include directories) .. enter the line
// "$(DELIB_INCLUDE)" (with quotation marks)

#include <windows.h>
#include <stdio.h>
#include "conio.h"

#include "delib.h"

// -----
// GetLastError function

BOOL IsError()
{
    unsigned char msg[500];

    if (DapiGetLastError() != DAPI_ERR_NONE)
    {

        DapiGetLastErrorText((unsigned char*) msg, sizeof(msg));
        printf("Error Code = %x * Message = %s\n", 0, msg);

        DapiClearLastError();

        return TRUE;
    }

    return FALSE;
}
```

```

/*****
/*****
/*****
/*****
/*****

void main(void)
{
    unsigned long handle;
    unsigned long value;

    // -----
    // Open Module

    handle = DapiOpenModule(USB_OPTOIN_8_RELAIS_8,0);

    printf("Module handle = %x\n", handle);

    // -----
    // Module not found!

    if (handle==0)
    {
        printf("Could not open module!\n");
        printf("Press any key to exit\n");
        getch();
        return;
    }

    // -----
    // Module found!

    printf("Module has been opened\n");

    // -----
    // Show config of module

    value = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
    DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO, 0, 0);
    IsError();
    printf("Configuration of the module: no. of digital outputs %d\n",
    value);
    printf("Press any key to continue\n");
    getch();

    // -----
    // Write output channels

    DapiDOSet1(handle, 0, 1);
    IsError();
    printf("Output channel 0 has been switched on\n");
    printf("Press any key to continue\n");
    getch();

    DapiDOSet1(handle, 0, 0);
    IsError();
    printf("Output channel 0 has been switched off\n");
    printf("Press any key to continue\n");
    getch();
}

```

```

DapiDOSet1(handle, 1, 1);
IsError();
printf("Output channel 1 has been switched on\n");
printf("Press any key to continue\n");
getch();

DapiDOSet1(handle, 1, 0);
IsError();
printf("Output channel 1 has been switched off\n");
printf("Press any key to continue\n");
getch();

DapiDOSet8(handle, 0, 0xff); //hexadecimal
IsError();
printf("Output channel 0-7 have been switched on\n");
printf("Press any key to continue\n");
getch();

DapiDOSet8(handle, 0, 0);
IsError();
printf("Output channel 0-7 have been switched off\n");
printf("Press any key to continue\n");
getch();

// -----
// Write and readback output channels

DapiDOSet8(handle, 0, 31);
IsError();
printf("Output channel 0-7 have been switched on\n");
printf("Press any key to continue\n");
getch();

value = DapiDOReadback32(handle, 0);
IsError();
printf("Readback output channel 0-3\n");
printf("value = %d\n", value);
printf("Press any key to continue\n");
getch();

DapiDOSet8(handle, 0, 0);
IsError();
printf("Output channel 0-7 have been switched off\n");
printf("Press any key to continue\n");
getch();

value = DapiDOReadback32(handle, 0);
IsError();
printf("Readback output channel 0-3\n");
printf("value = %d\n", value);
printf("Press any key to continue\n");
getch();

// -----
// Set timeout of output channels to 5 seconds

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
                    DAPI_SPECIAL_TIMEOUT_SET_VALUE_SEC, 5, 0);
IsError();
printf("Timeout has been set to 5 seconds\n");

```

```

printf("Press any key to continue\n");
getch();

// -----
// Activate timeout and switch on output channels 0-3

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_ACTIVATE, 0, 0);
IsError();
DapiDOSet8(handle, 0, 15);
IsError();
printf("Timeout has been activated\n");
printf("Output channels 0-3 have been switched on and will be switched
off automatically after 5 seconds\n");
printf("Press any key to continue\n");
getch();

// -----
// Deactivate timeout

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_DEACTIVATE, 0, 0);
IsError();
printf("Timeout has been deactivated\n");
printf("Press any key to continue\n");
getch();

// -----
// Close Module

DapiCloseModule(handle);
printf("Module closed\n");
printf("End of program!\n");
printf("Press any key to exit\n");
getch();

return ;
}

```

6 Anhang

6.1 Revisionen

Rev 2.00 Erste Anleitung



6.2 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.