



# **USB-RELAIS-8 / OPTOIN-8**

## **Hardware-Description**

2011 November

# INDEX

<b><u>1. Introduction</u></b>	<b>6</b>
<b><u>1.1. General remarks</u></b>	<b>6</b>
<b><u>1.2. Customer satisfaction</u></b>	<b>6</b>
<b><u>1.3. Customer response</u></b>	<b>6</b>
<b><u>2. Hardware description</u></b>	<b>8</b>
<b><u>2.1. Quick installation</u></b>	<b>8</b>
<b><u>2.1.1. Step 1 - Installation of the software and driver</u></b>	<b>8</b>
<b><u>2.1.2. Step 2 - Connecting of the module</u></b>	<b>8</b>
<b><u>2.1.3. Step 3 - Testing the connection and the module</u></b>	<b>8</b>
<b><u>2.2. USB-RELAIS-8</u></b>	<b>9</b>
<b><u>2.2.1. Technical data</u></b>	<b>9</b>
<b><u>2.2.2. Product pictures</u></b>	<b>10</b>
<b><u>2.2.3. Overview screen</u></b>	<b>11</b>
<b><u>2.2.4. Pin assignment</u></b>	<b>12</b>
<b><u>2.2.4.1. Pin assignment J1</u></b>	<b>12</b>
<b><u>2.2.4.2. Pin assignment J2</u></b>	<b>12</b>
<b><u>2.2.5. Outputs</u></b>	<b>13</b>
<b><u>2.2.5.1. Relay outputs</u></b>	<b>13</b>
<b><u>2.2.5.2. Timeout protection</u></b>	<b>13</b>
<b><u>2.2.5.3. Visual control of the outputs (depends of module)</u></b>	<b>13</b>
<b><u>2.3. USB-OPTOIN-8</u></b>	<b>14</b>
<b><u>2.3.1. Technical data</u></b>	<b>14</b>
<b><u>2.3.2. Product pictures</u></b>	<b>15</b>
<b><u>2.3.3. Overview screen</u></b>	<b>16</b>
<b><u>2.3.4. Pin assignment</u></b>	<b>17</b>
<b><u>2.3.4.1. Pin assignment J1</u></b>	<b>17</b>
<b><u>2.3.4.2. Pin assignment J2</u></b>	<b>17</b>
<b><u>2.3.5. Inputs</u></b>	<b>18</b>
<b><u>2.3.5.1. Registering short input pulses</u></b>	<b>18</b>
<b><u>2.3.5.2. Galvanically decoupled through optocouplers</u></b>	<b>18</b>
<b><u>2.3.5.3. Visual control of the inputs (depends of module)</u></b>	<b>18</b>

# INDEX

<b><u>3. Software</u></b>	<b>21</b>
<b><u>3.1. Using our products</u></b>	<b>21</b>
<b><u>3.1.1. Access via graphical applications</u></b>	<b>21</b>
<b><u>3.1.2. Access via the DELIB driver library</u></b>	<b>21</b>
<b><u>3.1.3. Access via protocol</u></b>	<b>21</b>
<b><u>3.1.4. Access via provided test programs</u></b>	<b>22</b>
<b><u>3.2. DELIB driver library</u></b>	<b>23</b>
<b><u>3.2.1. Overview</u></b>	<b>23</b>
<b><u>3.2.1.1. Program under diverse operating systems</u></b>	<b>23</b>
<b><u>3.2.1.2. Program with diverse programming languages</u></b>	<b>24</b>
<b><u>3.2.1.3. Program independent of the interface</u></b>	<b>24</b>
<b><u>3.2.1.4. SDK-Kit for Programmer</u></b>	<b>24</b>
<b><u>3.2.2. Supported operating systems</u></b>	<b>25</b>
<b><u>3.2.3. Supported programming languages</u></b>	<b>25</b>
<b><u>3.2.4. Installation DELIB driver library</u></b>	<b>26</b>
<b><u>3.2.5. DELIB Configuration Utility</u></b>	<b>30</b>
<b><u>3.3. Test programs</u></b>	<b>31</b>
<b><u>3.3.1. Digital Input-Output Demo</u></b>	<b>31</b>
<b><u>4. DELIB API reference</u></b>	<b>33</b>
<b><u>4.1. Management functions</u></b>	<b>33</b>
<b><u>4.1.1. DapiOpenModule</u></b>	<b>33</b>
<b><u>4.1.2. DapiCloseModule</u></b>	<b>34</b>
<b><u>4.1.3. DapiGetDELIBVersion</u></b>	<b>35</b>
<b><u>4.2. Error handling</u></b>	<b>36</b>
<b><u>4.2.1. DapiGetLastError</u></b>	<b>36</b>
<b><u>4.2.2. DapiGetLastErrorText</u></b>	<b>37</b>
<b><u>4.3. Reading Digital inputs</u></b>	<b>38</b>
<b><u>4.3.1. DapiDIGet1</u></b>	<b>38</b>
<b><u>4.3.2. DapiDIGet8</u></b>	<b>39</b>
<b><u>4.3.3. DapiDIGet16</u></b>	<b>40</b>
<b><u>4.3.4. DapiDIGet32</u></b>	<b>41</b>
<b><u>4.3.5. DapiDIGet64</u></b>	<b>42</b>
<b><u>4.3.6. DapiDIGetFF32</u></b>	<b>43</b>

# INDEX

<b><u>4.4. Setting Digital outputs</u></b>	<b>44</b>
<b><u>4.4.1. DapiDOSet1</u></b>	<b>44</b>
<b><u>4.4.2. DapiDOSet8</u></b>	<b>45</b>
<b><u>4.4.3. DapiDOSet16</u></b>	<b>46</b>
<b><u>4.4.4. DapiDOSet32</u></b>	<b>47</b>
<b><u>4.4.5. DapiDOSet64</u></b>	<b>48</b>
<b><u>4.4.6. DapiDOReadback32</u></b>	<b>49</b>
<b><u>4.4.7. DapiDOReadback64</u></b>	<b>50</b>
<b><u>4.5. Example program</u></b>	<b>51</b>
<b><u>5. Appendix</u></b>	<b>56</b>
<b><u>5.1. Revisions</u></b>	<b>56</b>
<b><u>5.2. Copyrights and trademarks</u></b>	<b>57</b>



# Introduction



# 1. Introduction

## 1.1. General remarks

First of all, we would like to congratulate you to the purchase of a high quality DEDITEC product.

Our products are being developed by our engineers according to quality requirements of high standard. Already during design and development we take care that our products have -besides quality- a long availability and an optimal flexibility.

### **Modular design**

The modular design of our products reduces the time and the cost of development. Therefor we can offer you high quality products at a competitive price.

### **Availability**

Because of the modular design of our products, we have to redesign only a module instead of the whole product, in case a specific component is no longer available.

## 1.2. Customer satisfaction

Our philosophy: a content customer will come again. Therefor customer satisfaction is in first place for us.

If by any chance, you are not content with the performance of our product, please contact us by phone or mail immediately.

We take care of the problem.

## 1.3. Customer response

Our best products are co-developments together with our customers. Therefor we are thankful for comments and suggestions.

# Hardware description

---



## 2. Hardware description

### 2.1. Quick installation

#### 2.1.1. Step 1 - Installation of the software and driver

Now install the driver DELIB library with the file "delib\_install.exe" from the supplied DEDITEC-Driver CD.

These can be found in the "\zip\DELIB\delib\_install.exe" on the DEDITEC-Driver CD.

Note: On our website [www.deditec.de](http://www.deditec.de) you can always find the latest DELIB driver version.

#### 2.1.2. Step 2 - Connecting of the module

Connect your PC via USB cable to the USB connector of the module.

After about 20 seconds, the module is detected by the driver and can now be tested and operated.

#### 2.1.3. Step 3 - Testing the connection and the module

In the Start menu, see "Start -> All Programs -> DEDITEC -> DELIB -> Sample Programs" you will find some example programs to test your module.



## 2.2. USB-RELAIS-8

### 2.2.1. Technical data

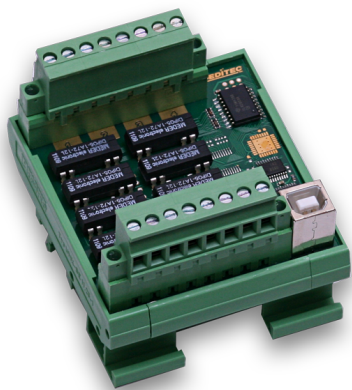
- Interface USB 2.0 / USB 1.1
- Power Supply +5V (power supply occurs over USB bus)
- 8x Relais outputs (36V, 1A, 15W)
  - max. switching voltage: 36V DC
  - max. switching current: 1A
  - max. switching power: 15W
  - max. transport current: 1,25A
  - Isolation (coil / contact): 1500V DC
  - contact resistance: 150mW
  - switching time: 0,5 ms
  - fall time: 0,1 ms
- Outputs Selectable timeout-protection, if the module is not longer adressed
- Galvanically isolated using relays
- Control-LED LED for 5V power supply
- Dimensions 77 mm x 67,5 mm x 55 mm (L x W x H)
- Operating temperature 10°C .. 50°C

#### Produktspezifische Daten:

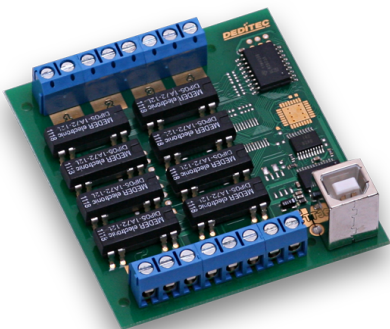
Product	Connection	Activity-LED	Case
USB-RELAIS-8	screwable terminal block	1 for each output	Yes
USB-RELAIS-8_A	screwable	-	-
USB-RELAIS-8_B	screwable terminal block	1 for each output	-

## 2.2.2. Product pictures

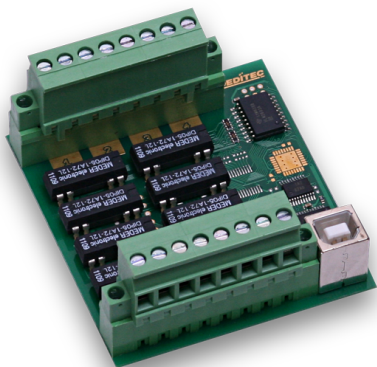
### USB-RELAIS-8



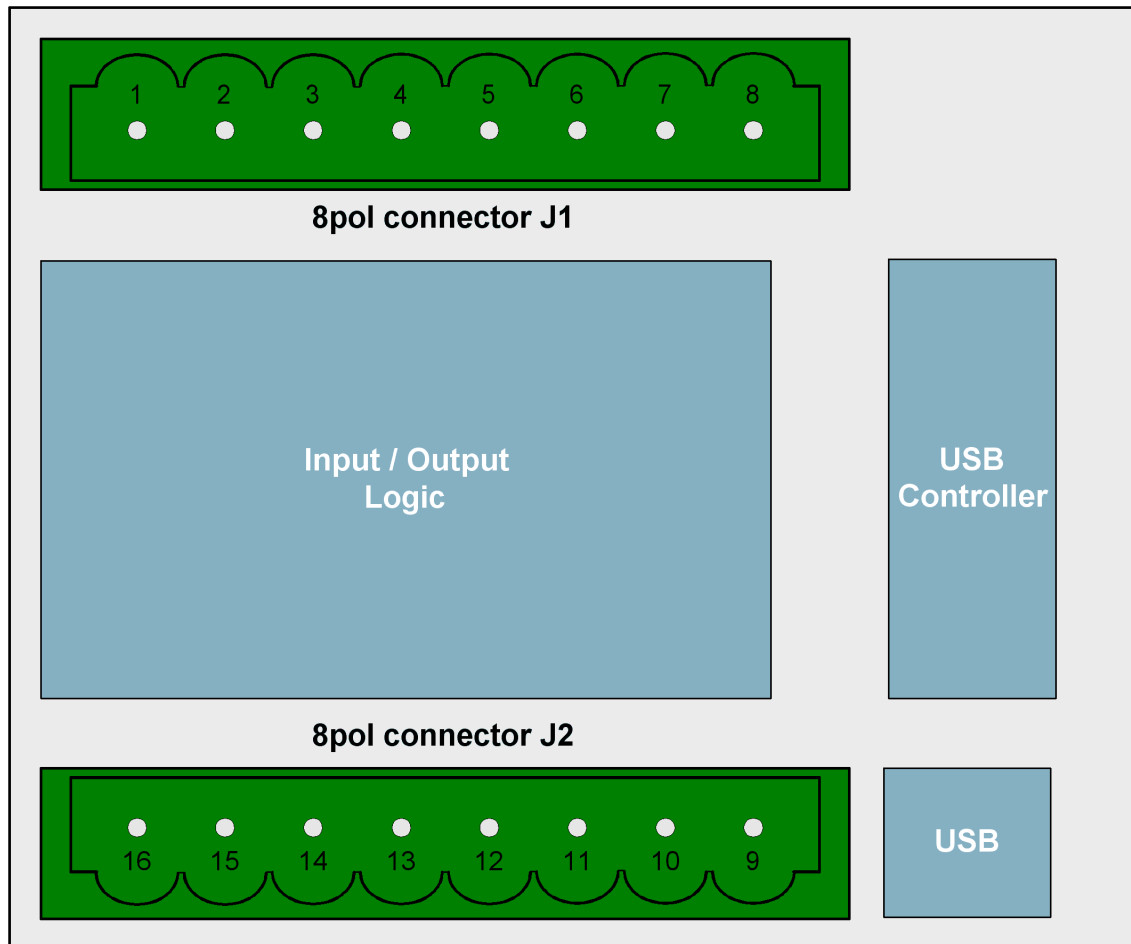
### USB-RELAIS-8\_A



### USB-RELAIS-8\_B

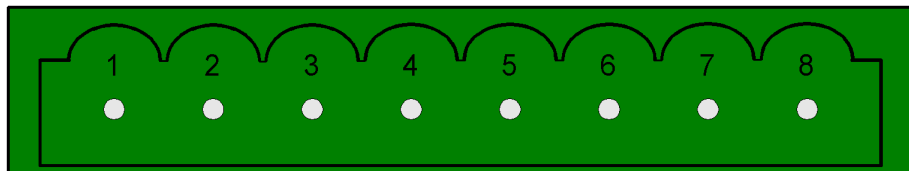


### 2.2.3. Overview screen



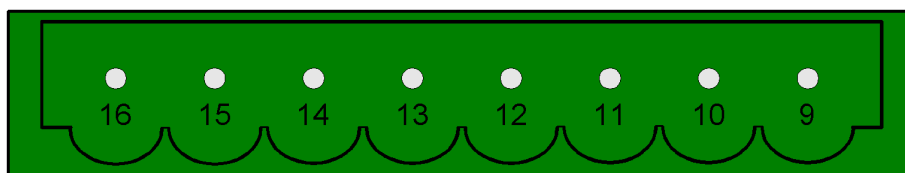
## 2.2.4. Pin assignment

### 2.2.4.1. Pin assignment J1



Pin	Description
1	Output Channel 1
2	Output Channel 1
3	Output Channel 2
4	Output Channel 2
5	Output Channel 3
6	Output Channel 3
7	Output Channel 4
8	Output Channel 4

### 2.2.4.2. Pin assignment J2



Pin	Description
9	Output Channel 5
10	Output Channel 5
11	Output Channel 6
12	Output Channel 6
13	Output Channel 7
14	Output Channel 7
15	Output Channel 8
16	Output Channel 8

## 2.2.5. Outputs

### 2.2.5.1. Relay outputs

The relays are able to switch voltages up to 36V. The max. current is 1A at a max. power of 15W.

Additionally, the relays provide a safe electrical isolation of the module to the connected equipment.

### 2.2.5.2. Timeout protection

The timeout-protection gives the possibility to switch-off automatically the outputs to prevent damage. This takes place, if in a predefined time frame any communication with the module was possible. Reasons could be cable disruption, PC-crash and so on. This way damage control, surcharge of connected equipment and risk of accidents can be avoided.

### 2.2.5.3. Visual control of the outputs (depends of module)

The state of each output is directly signalized by a separate LED. This simplifies to detect and rectify wiring errors, because the signals on the cables are directly observable.

**Note:** Only available for the USB-RELAIS-8 and USB-RELAIS-8\_B module

## 2.3. USB-OPTOIN-8

### 2.3.1. Technical data

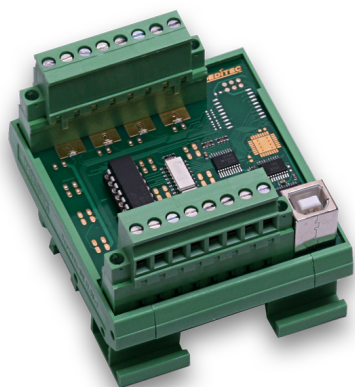
- USB-Interface (USB 1.1 / USB 2.0)
- Power supply: +5V (supply occurs over USB bus)
- 8 opto-coupler inputs
  - 24V AC switching voltage (optional 15V, 12V and 5V are available)
  - 16 Bit-Counter for each input
  - Detection of pulses between two selection cycles
- Galvanically isolated using opto-couplers
- Variable input voltage range min 5V, max 30V AC (standard: 15-30V)
- Logging of impulses between 2 read out cycles
- Control-LED LED for 5V power supply
- Dimensions: 77 x 67,5 x 55 mm (L x W x H)
- Operating temperature: 10°C .. 50°C

#### Produktspezifische Daten:

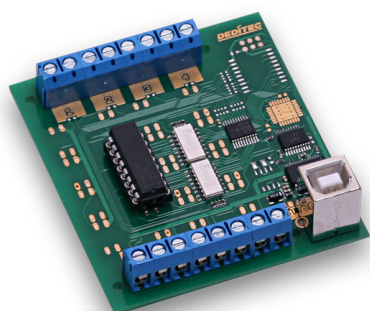
Product	Connection	Activity-LED	Case
USB-OPTOIN-8	screwable terminal block	1 for each input	Yes
USB-OPTOIN-8_A	screwable	-	-
USB-OPTOIN-8_B	screwable terminal block	1 for each input	-

## 2.3.2. Product pictures

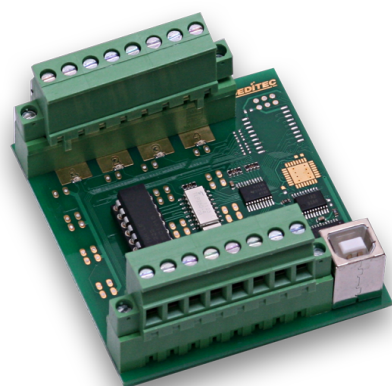
### USB-OPTOIN-8



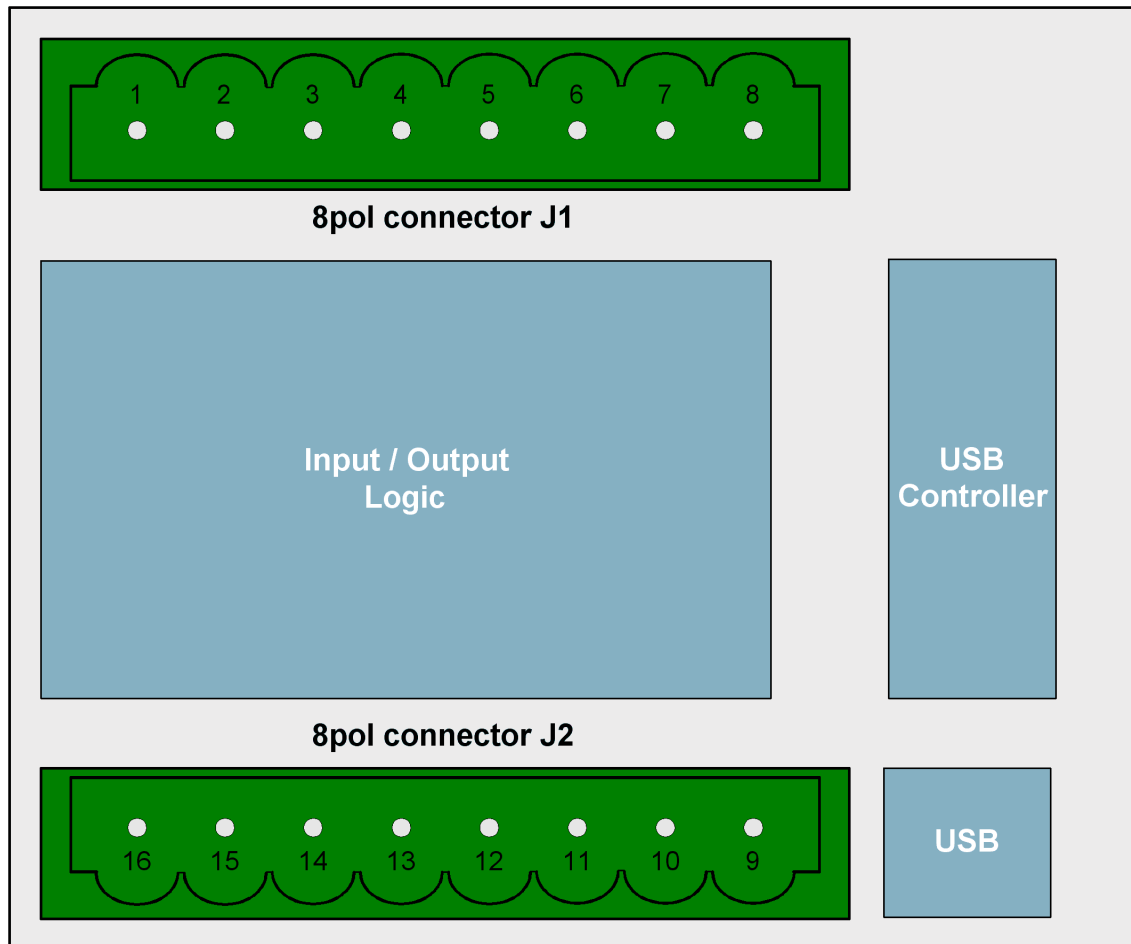
### USB-OPTOIN-8\_A



### USB-OPTOIN-8\_B



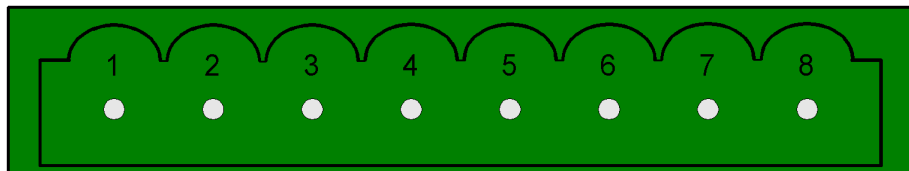
### 2.3.3. Overview screen





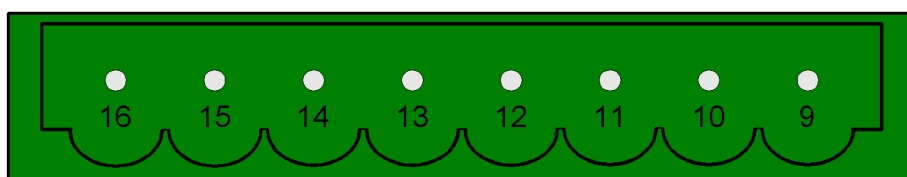
## 2.3.4. Pin assignment

### 2.3.4.1. Pin assignment J1



Pin	Description
1	Input Channel 1 +
2	Input Channel 1 -
3	Input Channel 2 +
4	Input Channel 2 -
5	Input Channel 3 +
6	Input Channel 3 -
7	Input Channel 4 +
8	Input Channel 4 -

### 2.3.4.2. Pin assignment J2



Pin	Description
9	Input Channel 5 +
10	Input Channel 5 -
11	Input Channel 6 +
12	Input Channel 6 -
13	Input Channel 7 +
14	Input Channel 7 -
15	Input Channel 8 +
16	Input Channel 8 -

## **2.3.5. Inputs**

### **2.3.5.1. Registering short input pulses**

Short input pulses between to read-out cycles are registered through an additional logic and can be separately read-out.

### **2.3.5.2. Galvanically decoupled through optocouplers**

AC input opto-couplers provide a galvanic isolation of the module towards the connected equipment. They also provide a safe connection to the module for reverse currents and high voltage peaks.

### **2.3.5.3. Visual control of the inputs (depends of module)**

The state of each input is directly signalized by a separate LED. This simplifies to detect and rectify wiring errors, because the signals on the cables are directly observable.

**Note:** Only available for the USB-RELAIS-8 and USB-RELAIS-8\_B module



# Software



## 3. Software

### 3.1. Using our products

#### 3.1.1. Access via graphical applications

We provide driverinterfaces e.g. for LabVIEW and ProfiLab. The DELIB driver library is the basis, which can be directly activated by ProfiLAB.

For LabVIEW, we provide a simple driver connection with examples!

#### 3.1.2. Access via the DELIB driver library

In the appendix, you can find the complete function reference for the integration of our API-functions in your software. In addition we provide examples for the following programming languages:

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

#### 3.1.3. Access via protocol

The protocol for the activation of our products is open source. So you are able to use our products on systems without Windows or Linux.

### **3.1.4. Access via provided test programs**

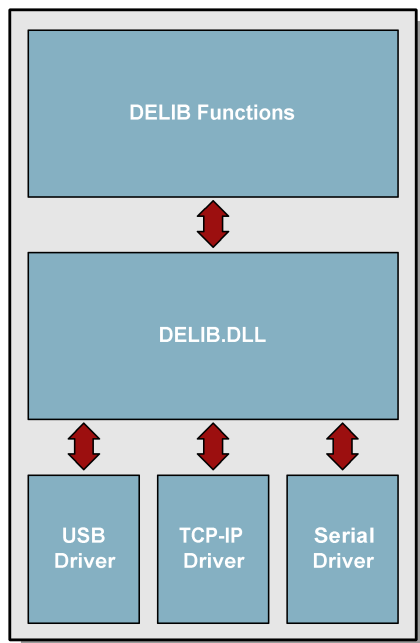
We provide simple handling test programs for the most important functions of our products. These will be installed automatically by the installation of the DELIB driver library.

So you can test directly e.g. relays or you can check the voltage of an A/D converter.

## 3.2. DELIB driver library

### 3.2.1. Overview

The following figure explains the structure of the DELIB driver library



The DELIB driver library allows an uniform response of DEDITEC hardware with particular consideration of the following viewpoints:

- Independent of operating system
- Independent of programming language
- Independent of the product

#### 3.2.1.1. Program under diverse operating systems

The DELIB driver library allows an uniform response of our products on diverse operating systems.

We have made sure, that all of our products can be responded by a few commands. Whatever which operating system you use. - Therefore the DELIB cares!

### **3.2.1.2. Program with diverse programming languages**

We provide uniform commands to create own applications. This will be solved by the DELIB driver library.

**You choose the programming language!**

It can be simply developed applications under C++, C, Visual Basic, Delphi or LabVIEW®.

### **3.2.1.3. Program independent of the interface**

Write your application independent of the interface !

Program an application for an USB product of us.- Also, it will work with an ethernet or RS-232 product of us !

### **3.2.1.4. SDK-Kit for Programmer**

Integrate the DELIB in your application. On demand you receive an installation script for free, which allows you, to integrate the DELIB installation in your application.



### **3.2.2. Supported operating systems**

Our products support the following operating systems:

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

### **3.2.3. Supported programming languages**

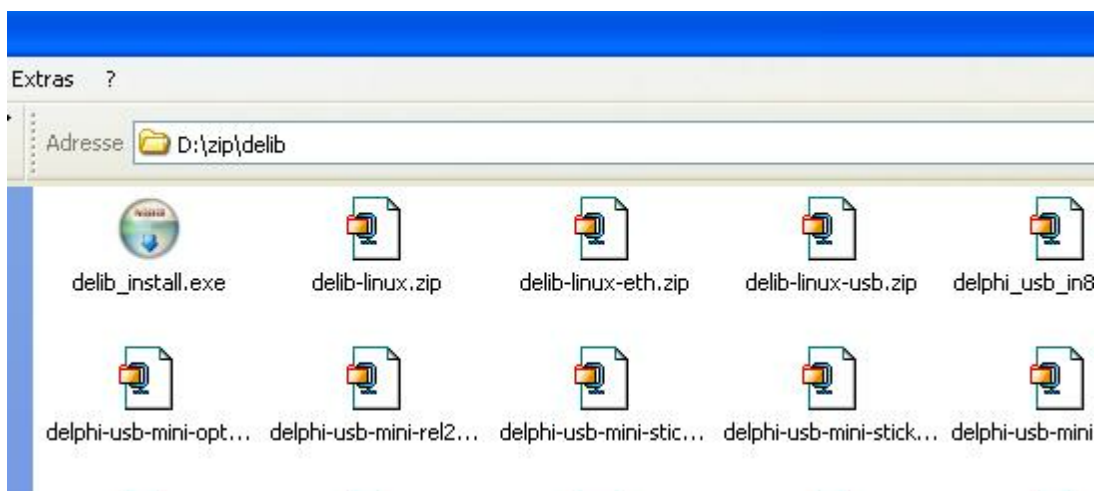
Our products are responsive via the following programming languages:

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

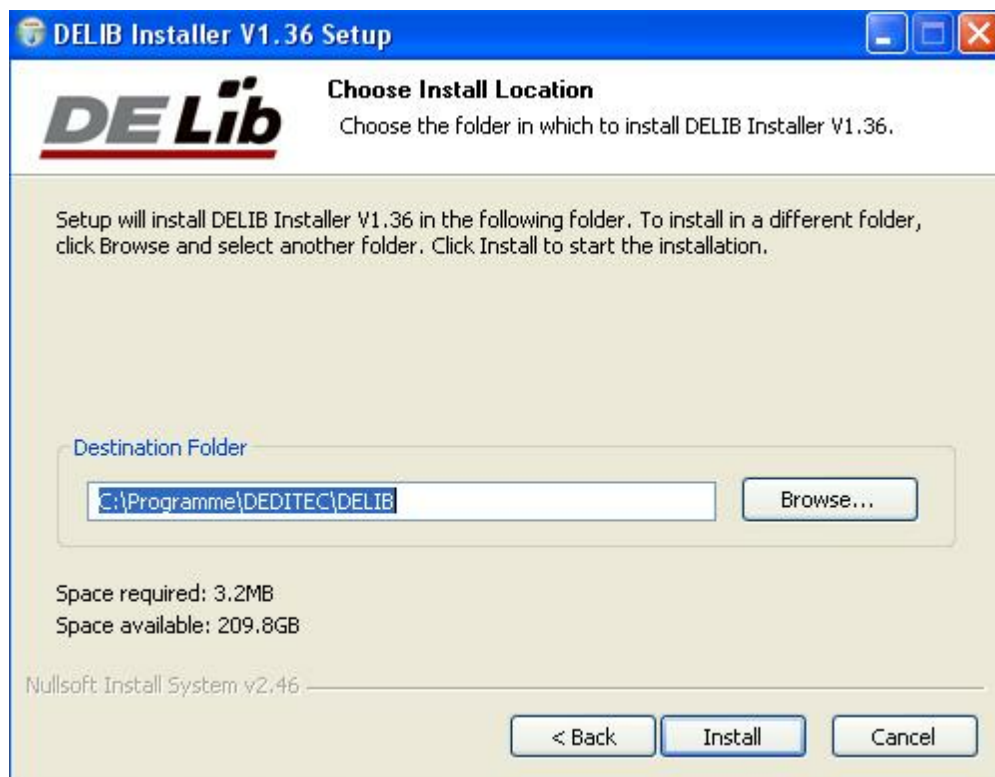
### 3.2.4. Installation DELIB driver library



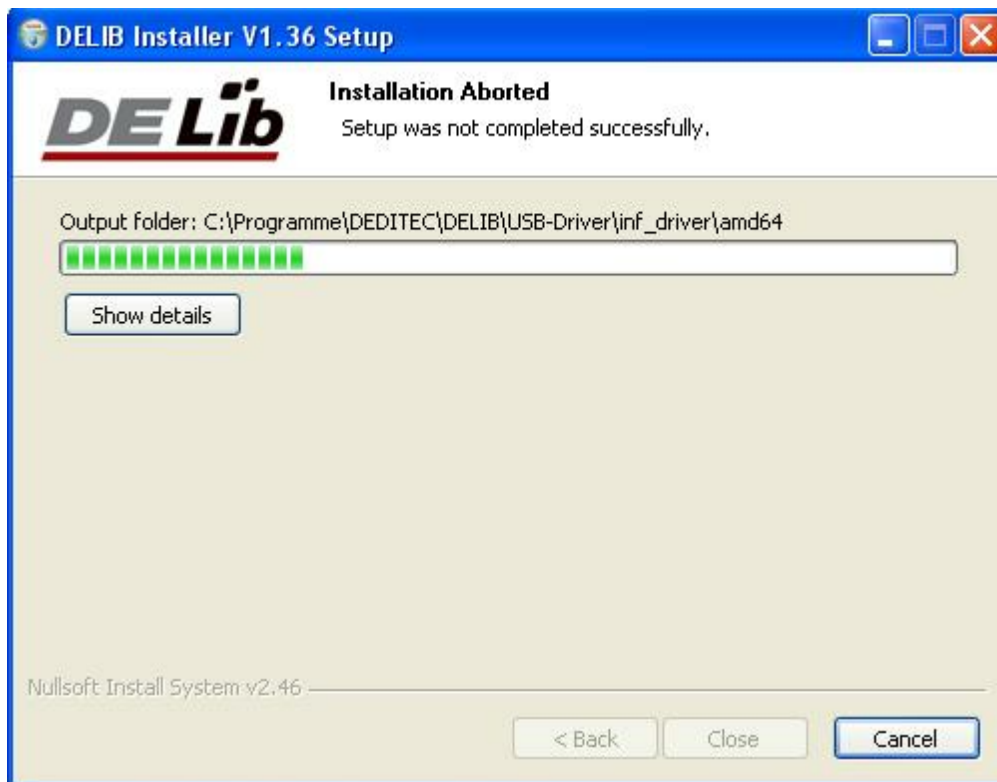
Our DELIB installer start screen.



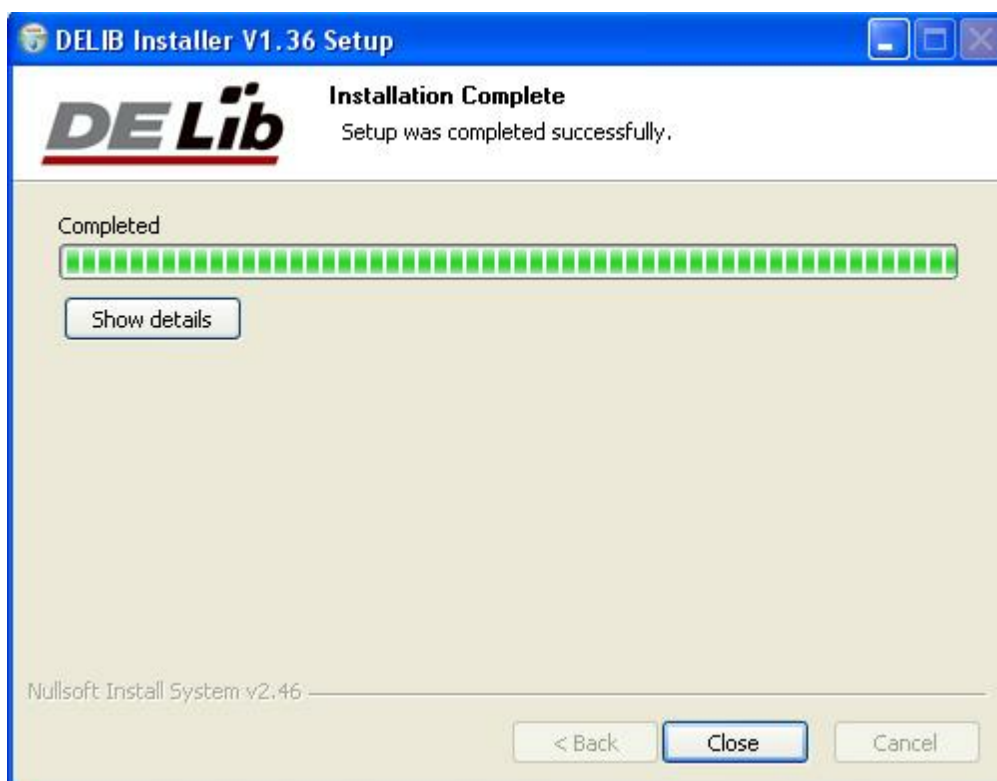
Insert the DEDITEC driver CD into the drive and start „**delib\_install.exe**“. The DELIB driver library is also available on <http://www.deditec.eu/delib>



Click on „Install“.



The drivers will be installed.



The DELIB driver library is now installed. Press „**Close**“ to finish the installation.

You can configure your module with the „**DELIB Configuration Utility**“ (see next chapter). This is only necessary, if more than one module is present.

### 3.2.5. DELIB Configuration Utility



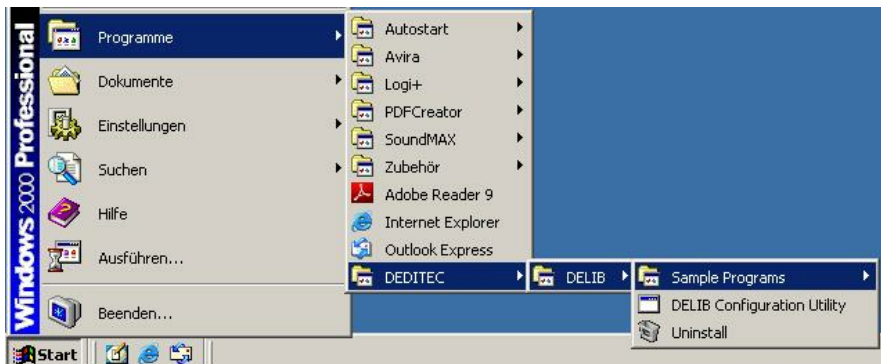
Start the “**DELIB Configuration Utility**” as follows:

**Start → Programs → DEDITEC → DELIB → DELIB Configuration Utility.**

The „**DELIB Configuration Utility**“ is a program to configure and subdivide identical USB-modules in the system. This is only necessary if more than one module is present.

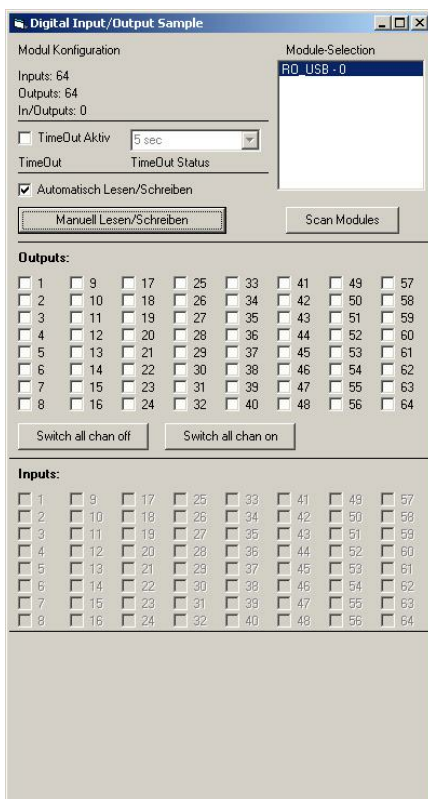
### 3.3. Test programs

#### 3.3.1. Digital Input-Output Demo



Start “Digital Input-Output Demo” as follows:

**Start → Programme → DEDITEC → DELIB → Digital Input-Output Demo.**



The screenshot shows a test of the RO-USB-O64-R64. The configuration of the module (64 inputs and 64 outputs) is shown on the upper left side.

## DELIB API reference

---

IV



## 4. DELIB API reference

### 4.1. Management functions

#### 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)
```

## 4.2. Error handling

### 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. Reading Digital inputs

### 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

## 4.4. Setting Digital outputs

### 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. Example program

```
//*****
//*****
//*****
//*****
//*****
//
//
// product: usb-optoin-8-relais-8 (ModuleID = USB_OPTOIN_8_RELAIS_8)
// configuration: digital-outputs
// programming language: vc
//
//
// (c) DEDITEC GmbH, 2011
// web: http://www.deditec.de/
// mail: vertrieb@deditec.de
//
//
//*****
//*****
//*****
//*****
//*****
//
//
// 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 ;
}

```

# Appendix



# 5. Appendix

## 5.1. Revisions

Rev 2.00

First DEDITEC issue



## **5.2. Copyrights and trademarks**

Linux is registered trade-mark of Linus Torvalds.

Windows CE is registered trade-mark of Microsoft Corporation.

USB is registered trade-mark of USB Implementers Forum Inc.

LabVIEW is registered trade-mark of National Instruments.

Intel is registered trade-mark of Intel Corporation

AMD is registered trade-mark of Advanced Micro Devices, Inc.