



# USB-TTL-IN8-OUT8

Hardware-Description

2010 Oktober

# INDEX

<b><u>1. Introduction</u></b>	<b>5</b>
<u>1.1. General remarks</u>	5
<u>1.2. Customer satisfaction</u>	5
<u>1.3. Customer response</u>	5
<b><u>2. Hardware description</u></b>	<b>7</b>
<u>2.1. Technical data</u>	7
<u>2.2. Pin assignment</u>	8
<b><u>3. Software</u></b>	<b>10</b>
<u>3.1. Using our products</u>	10
<u>3.1.1. Access via graphical applications</u>	10
<u>3.1.2. Access via the DELIB driver library</u>	10
<u>3.1.3. Access via protocol</u>	10
<u>3.1.4. Access via provided test programs</u>	11
<u>3.2. DELIB driver library</u>	12
<u>3.2.1. Overview</u>	12
<u>3.2.2. Supported operating systems</u>	14
<u>3.2.3. Supported programming languages</u>	14
<u>3.2.4. Installation DELIB driver library</u>	15
<u>3.2.5. DELIB Configuration Utility</u>	17
<u>3.3. Test programs</u>	18
<u>3.3.1. Digital Input-Output Demo</u>	18
<u>3.3.2. Analog Input-Output Demo</u>	19
<b><u>4. DELIB API reference</u></b>	<b>21</b>
<u>4.1. Management functions</u>	21
<u>4.1.1. DapiOpenModule</u>	21
<u>4.1.2. DapiCloseModule</u>	22
<u>4.2. Reading Digital inputs</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>4.2.7. DapiDIGetCounter</u>	29
<u>4.3. Setting Digital outputs</u>	30
<u>4.3.1. DapiDOSet1</u>	30
<u>4.3.2. DapiDOSet8</u>	31
<u>4.3.3. DapiDOSet16</u>	32
<u>4.3.4. DapiDOSet32</u>	33
<u>4.3.5. DapiDOSet64</u>	34
<u>4.3.6. DapiDOReadback32</u>	35
<u>4.3.7. DapiDOReadback64</u>	36
<u>4.4. Example program</u>	37
<u>5. Appendix</u>	40
<u>5.1. Revisions</u>	40
<u>5.2. Copyrights and trademarks</u>	41



# 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. Therefore 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. Therefore 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.

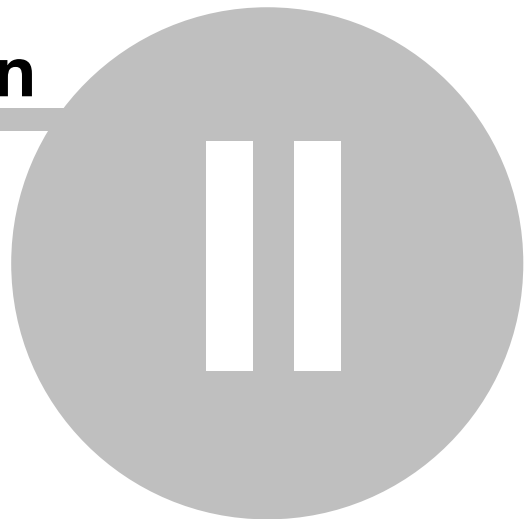
We take care of the problem.

## 1.3. Customer response

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

# Hardware description

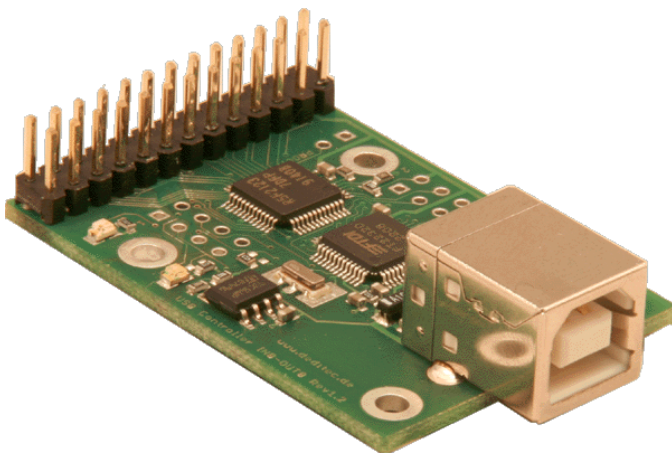
---



## 2. Hardware description

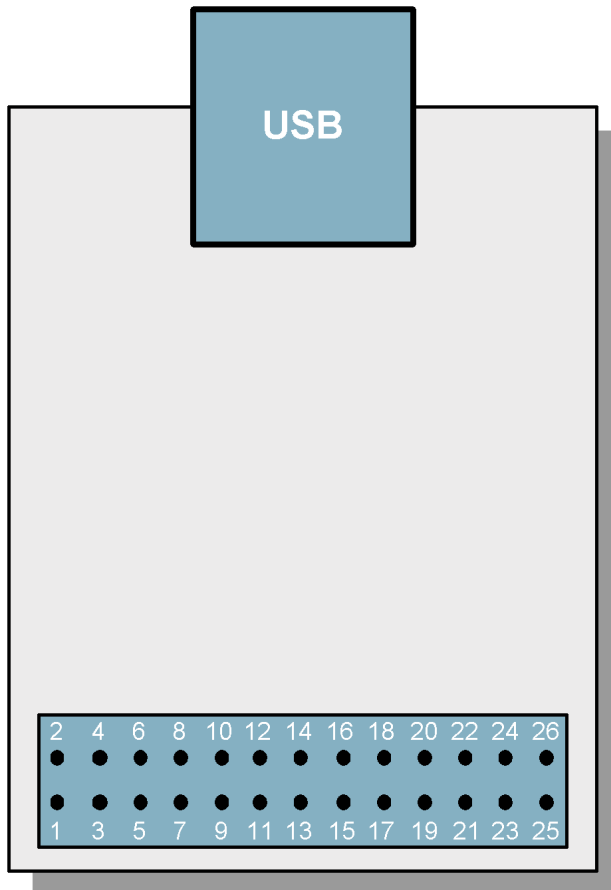
The USB - I/O Interface conduces to a simplification of the well complex data of the USB-Bussystem. Commands such as "8-bit data write" and "8-bit data read" will be transmitted to the 16 in-/outputs by the provided driver library.

### 2.1. Technical data



- Input current: ~53 mA
- USB 2.0 / 1.1
- 8\*TTL-Input
- 8\*TTL-Output
- Simple response under Windows

## 2.2. Pin assignment



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. Using our products

#### 3.1.1. Access via graphical applications

We provide driver interfaces 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**

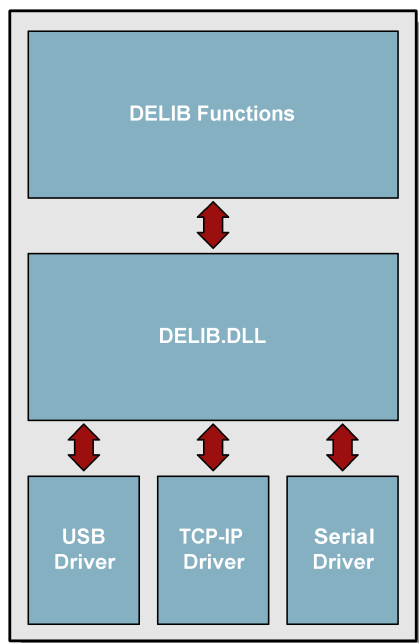
We provide simple handling test programs for the most important functions of our products. These will be installed automatically by the installation of the DELIB driver library.

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

#### **Program under diverse operating systems**

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

We has made sure, that all of our products can be responded by a few commands.

Whatever which operating system you use. - Therefore the DELIB cares!

### **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® .

### **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 !

### **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 2000
- Windows XP
- Windows Vista
- Windows 7
- Linux

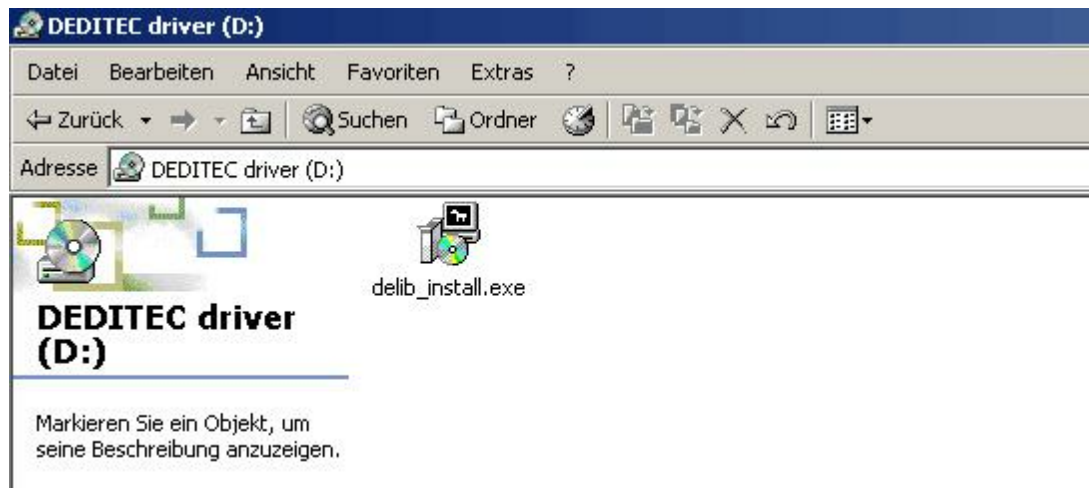
### **3.2.3. Supported programming languages**

Our products are responsive via the following programming languages:

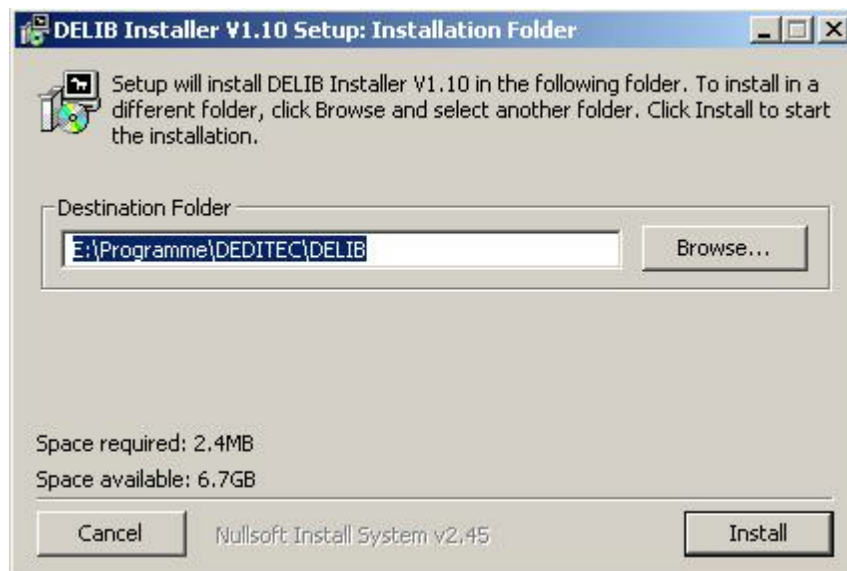
- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

### 3.2.4. Installation DELIB driver library

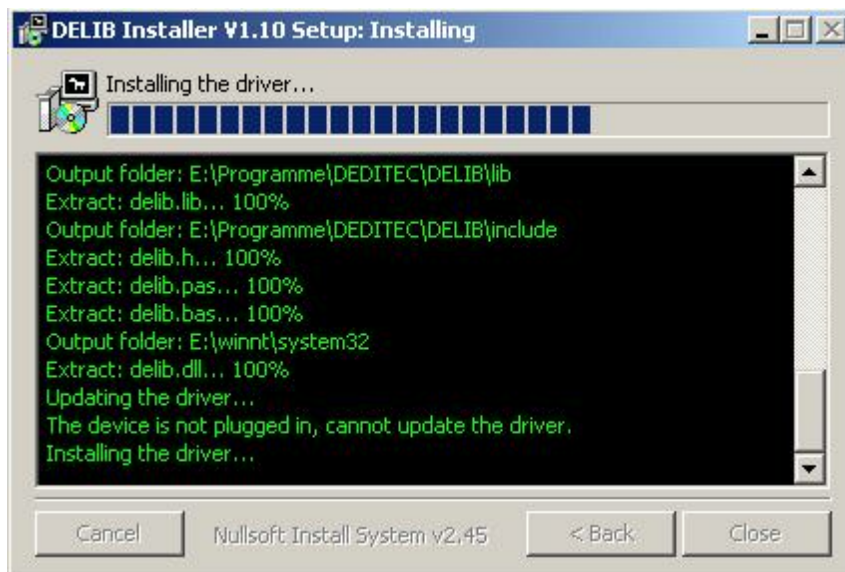
DELIB stands for DEDITEC Library and contains the necessary libraries for the modules in the programming languages C, Delphi and Visual Basic.



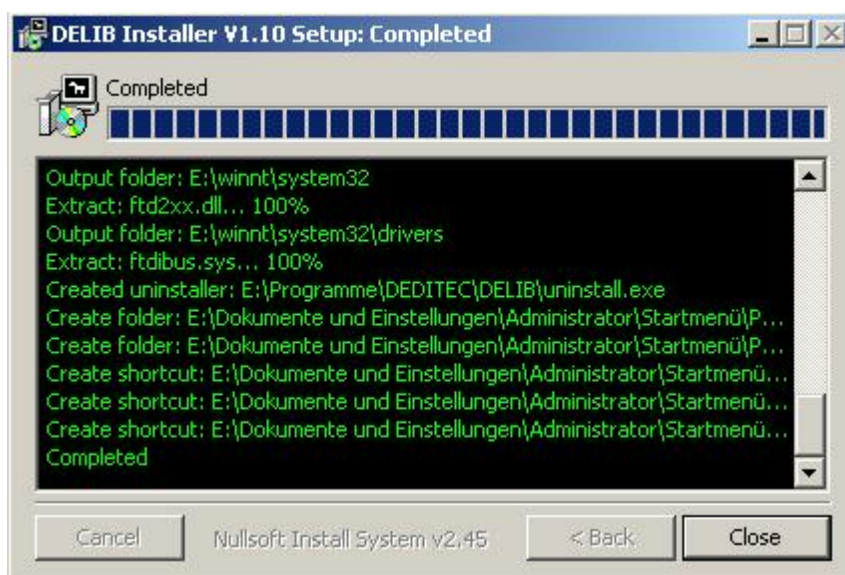
Insert the DEDITEC driver CD into the drive and start „**delib\_install.exe**“. The DELIB driver library is also available on <http://www.deditec.en/delib>



Click on „**Install**“.



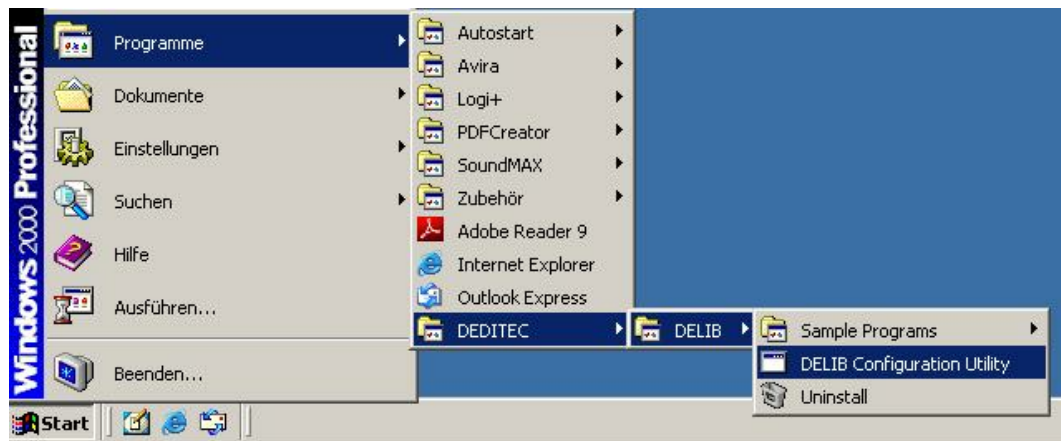
The drivers will be installed.



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



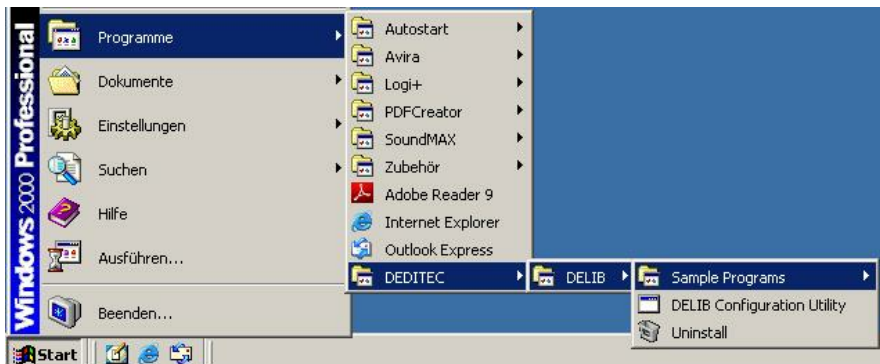
Start the “**DELIB Configuration Utility**” as follows:

**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.

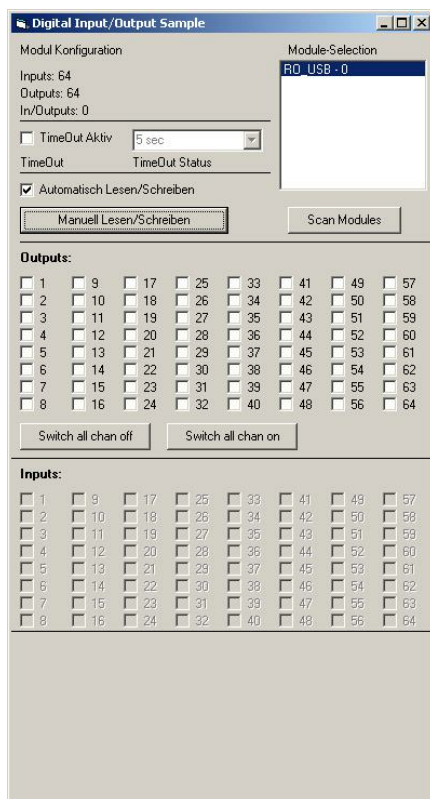
### 3.3. Test programs

#### 3.3.1. Digital Input-Output Demo



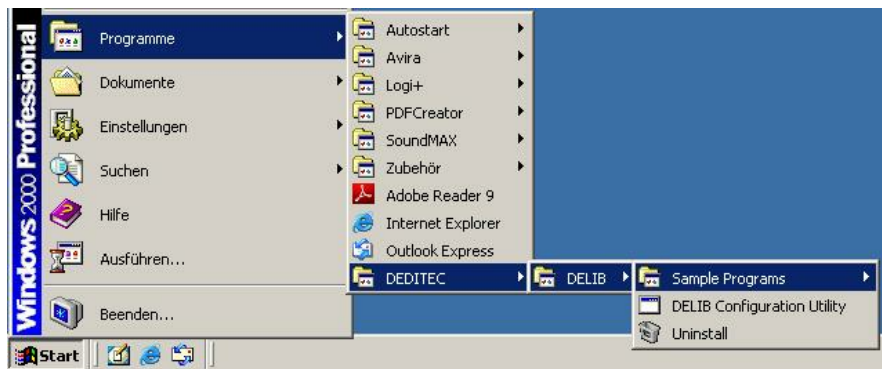
Start “Digital Input-Output Demo” as follows:

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



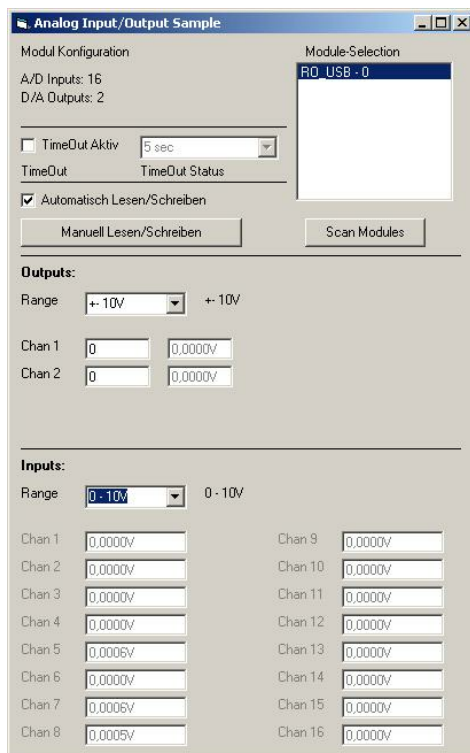
The screenshot shows a test of the RO-USB-O64-R64. The configuration of the module (64 inputs and 64 outputs) is shown on the upper left side.

### 3.3.2. Analog Input-Output Demo



Start “Analog Input-Output Demo” as follows:

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



The screenshot shows a test of the RO-USB-AD16-DA2\_ISO. The configuration of the module (16 A/D inputs and 2 D/A 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);
```

## 4.2. Reading Digital inputs

### 4.2.1. 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).

## 4.2.2. 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.

### 4.2.3. 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.

## 4.2.4. 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();
```

#### 4.2.5. 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.

## 4.2.6. DapiDIGetFF32

### Description

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

### 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, 32, ..)

### Return value

State of 32 input change states

## 4.2.7. DapiDIGetCounter

### Description

This command reads the counter of a digital input

### Definition

*ULONG DapiDIGetCounter(handle, ch, par1);*

### Parameters

handle=This is the handle of an opened module.

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

par1=0 (Normal counter function)

par1=DAPI\_CNT\_MODE\_READ\_WITH\_RESET (Reading and resetting the 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
```

## 4.3. Setting Digital outputs

### 4.3.1. 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

### 4.3.2. 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

### 4.3.3. 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

### 4.3.4. 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();
```

### 4.3.5. 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

### 4.3.6. 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.

### 4.3.7. 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.

## 4.4. Example program

```
// *****  
// *****  
// *****  
// *****  
// *****  
//  
// (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=0 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 ;
}

```

# Appendix



# 5. Appendix

## 5.1. Revisions

Rev 1.00	First issue
Rev 2.00	Design change
Rev 2.01	Supplement to technical data

## 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.