



# USB-Controller-8

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
<u>2.3. Signaltiming D0-D7, A0-A4, RD, WR, CS</u>	9
<u>2.3.1. Timing read cycle</u>	9
<u>2.3.2. Timing write cycle</u>	10
<b><u>3. Software</u></b>	<b>12</b>
<u>3.1. Using our products</u>	12
<u>3.1.1. Access via graphical applications</u>	12
<u>3.1.2. Access via the DELIB driver library</u>	12
<u>3.1.3. Access via protocol</u>	12
<u>3.1.4. Access via provided test programs</u>	13
<u>3.2. DELIB driver library</u>	14
<u>3.2.1. Overview</u>	14
<u>3.2.2. Supported operating systems</u>	16
<u>3.2.3. Supported programming languages</u>	16
<u>3.2.4. Installation DELIB driver library</u>	17
<u>3.2.5. DELIB Configuration Utility</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. Register write commands</u>	23
<u>4.2.1. DapiWriteByte</u>	23

# INDEX

<u>4.2.2. DapiWriteWord</u>	24
<u>4.2.3. DapiWriteLong</u>	25
<u>4.2.4. DapiWriteLongLong</u>	26
<u>4.3. Register read commands</u>	27
<u>4.3.1. DapiReadByte</u>	27
<u>4.3.2. DapiReadWord</u>	28
<u>4.3.3. DapiReadLong</u>	29
<u>4.3.4. DapiReadLongLong</u>	30
<u>4.4. Example program</u>	31
<u>5. Appendix</u>	34
<u>5.1. Revisions</u>	34
<u>5.2. Copyrights and trademarks</u>	35



# 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.

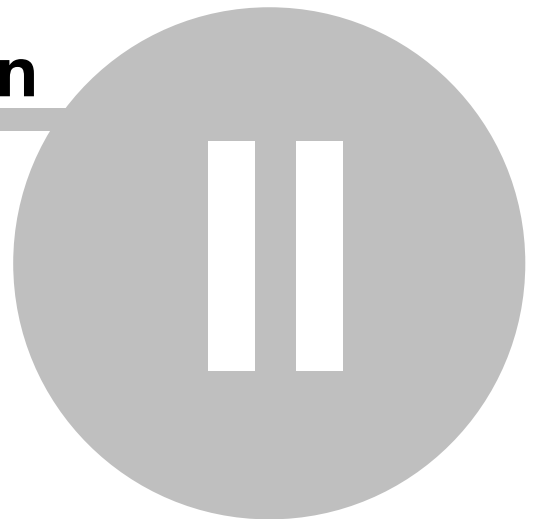
We take care of the problem.

## 1.3. Customer response

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

# Hardware description

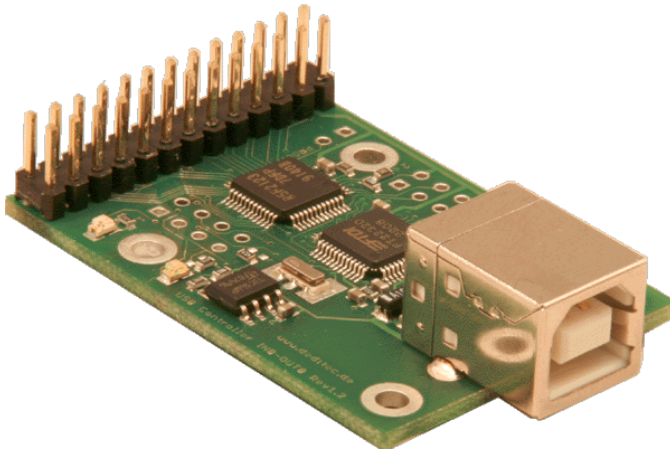
---



## 2. Hardware description

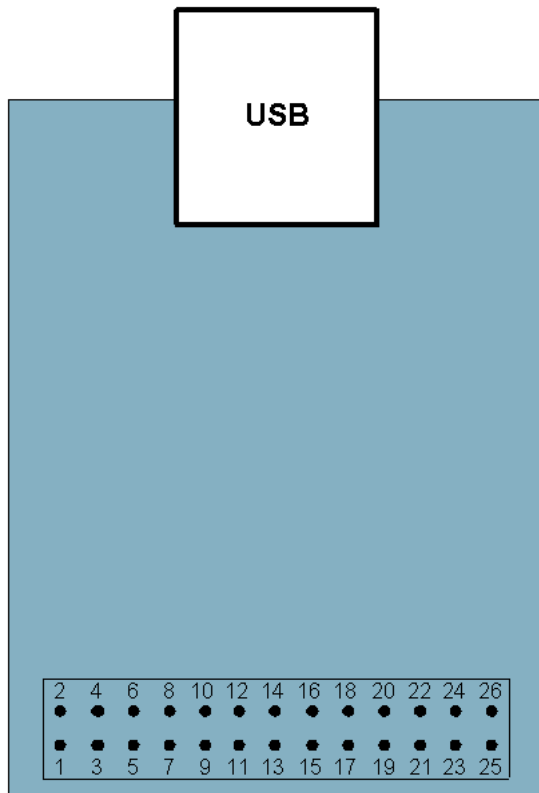
The "USB-CONTROLLER8" conduces to a simplification of the well complex data of the USB-Bussystem. It affords a simple triggering of a 8 bit data bus with low complexity. You do not need any previous USB knowledge because the read and write commands are available by an own software.

### 2.1. Technical data



- D0-7 (8x TTL I/O)
- A0-4 (5x TTL-Out)
- RD (TTL-Out)
- WR (TTL-Out)
- CS (TTL-Out)
- Header

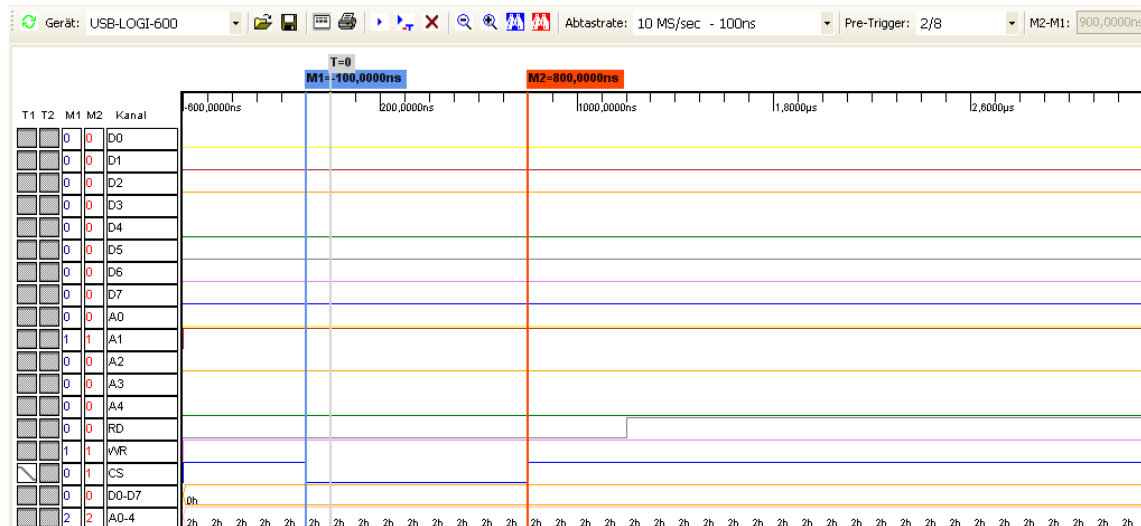
## 2.2. Pin Assignment



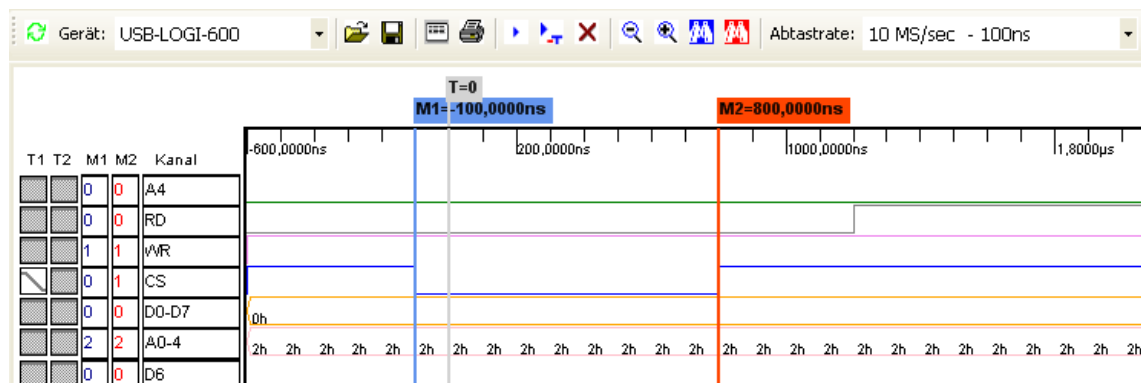
Pin	Name	Pin	Name
1	VCC	2	LED
3	GND	4	GND
5	D6	6	D7
7	D4	8	D5
9	D2	10	D3
11	D0	12	D1
13	A0	14	A1
15	A2	16	RD
17	WR	18	CS
19	A3	20	A4
21	AD3	22	AD2
23	AD1	24	AD0
25	GND	26	VREF

## 2.3. Signaltiming D0-D7, A0-A4, RD, WR, CS

### 2.3.1. Timing read cycle

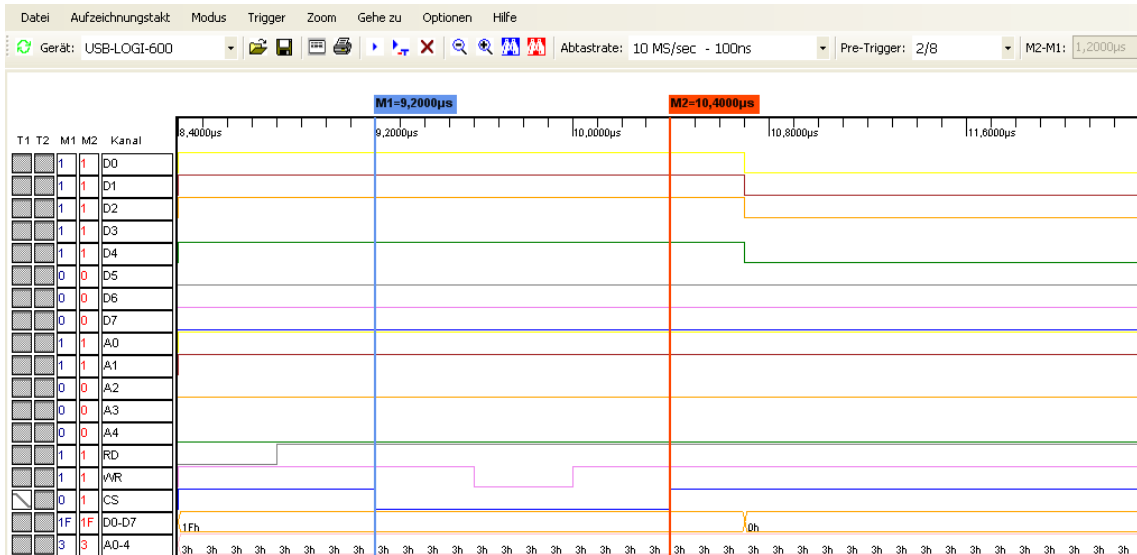


You can recognize by this graphic that a 8 bit read access proceeds. CS is active (=low) (see blue measuring line) and 1,2 us later inactive (red measuring line).

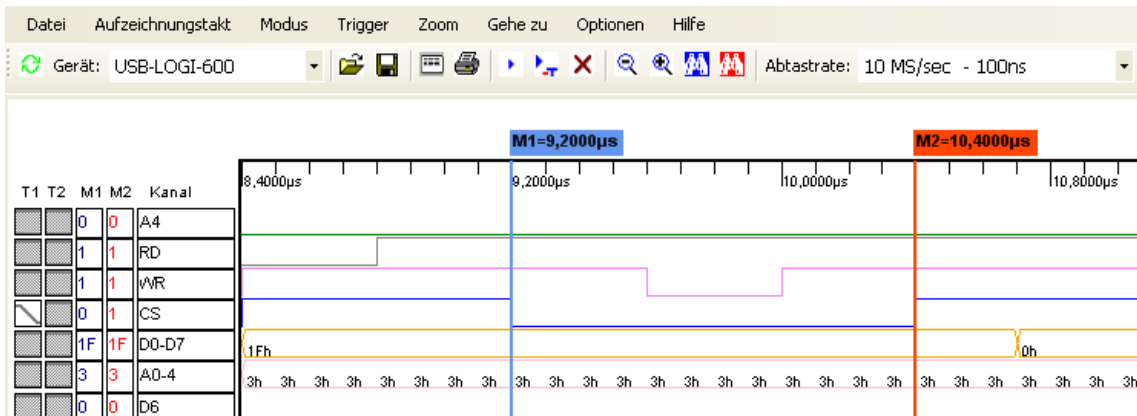


The section of measuring, which shows the read acces, is displayed by this picture.

### 2.3.2. Timing write cycle



You can recognize by this graphic that a 8 bit write access proceeds. CS is active (=low) (see blue measuring line) and 1,2 us later inactive (red measuring line).



The section of measuring, which shows the write acces, is displayed by this picture.

# 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**

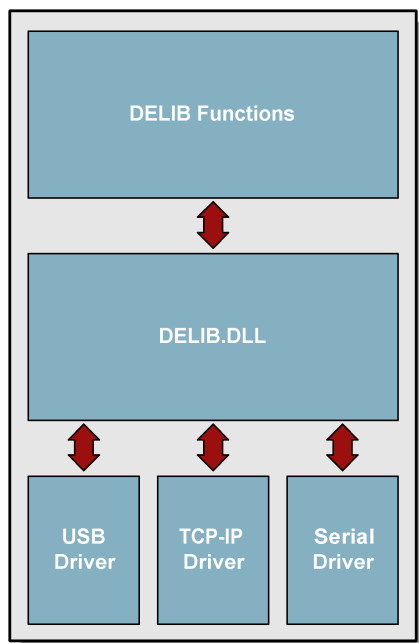
We provide simple handling test programs for the most important functions of our products. These will be installed automatically by the installation of the DELIB driver library.

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

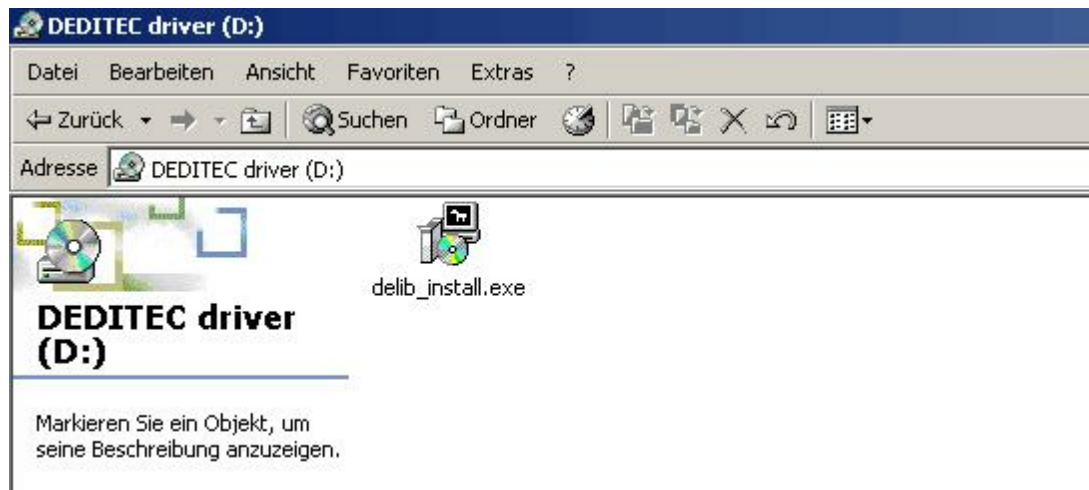
### **3.2.3. Supported programming languages**

Our products are responsive via the following programming languages:

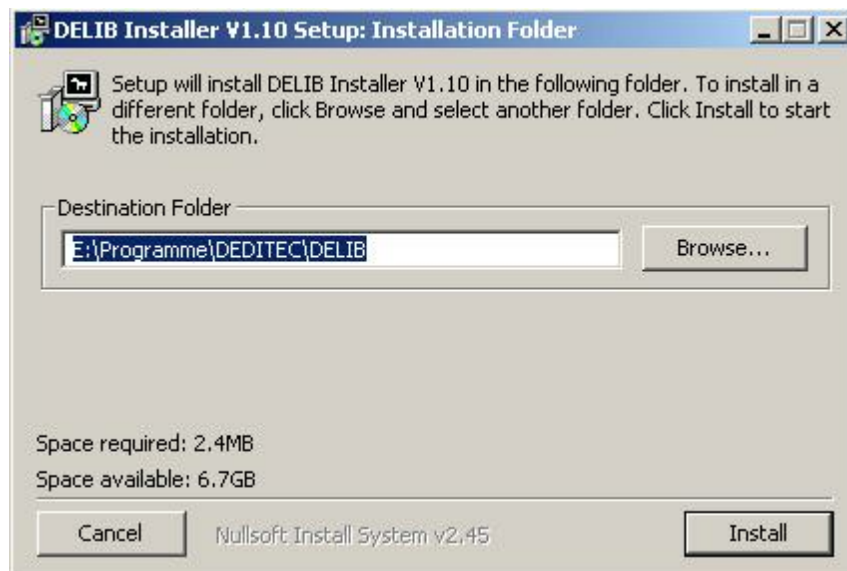
- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

### 3.2.4. Installation DELIB driver library

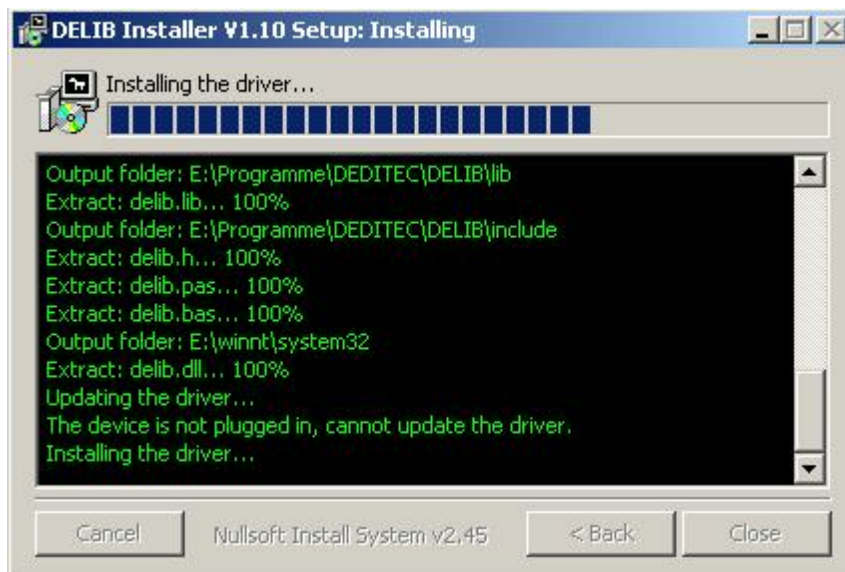
DELIB stands for DEDITEC Library and contains the necessary libraries for the modules in the programming languages C, Delphi and Visual Basic.



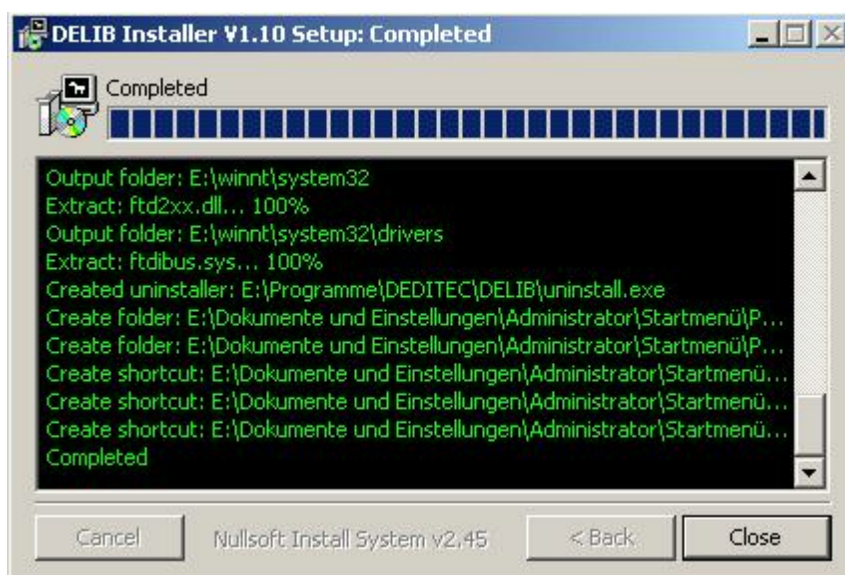
Insert the DEDITEC driver CD into the drive and start „**delib\_install.exe**“. The DELIB driver library is also available on <http://www.deditec.eu/delib>



Click on „**Install**“.



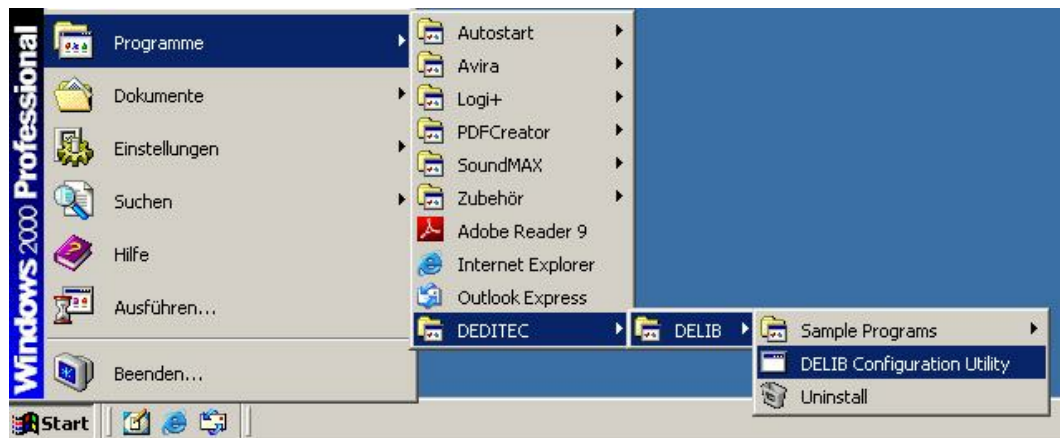
The drivers will be installed.



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.

## **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. Register write commands

### 4.2.1. DapiWriteByte

#### Description

This command performs a direct register write command to the module.

#### Definition

```
void DapiWriteByte(ULONG handle, ULONG adress, ULONG value);
```

#### Parameters

handle=This is the handle of an opened module

adress=Address to be accessed

value=Specifies the data value that is to be written (8 bits)

#### Return value

None

#### Remarks

This should only be used by experienced programmers. A directly access to every available register is possible.

## 4.2.2. DapiWriteWord

### Description

This command performs a direct register write command to the module.

### Definition

```
void DapiWriteWord(ULONG handle, ULONG adress, ULONG value);
```

### Parameters

handle=This is the handle of an opened module

adress=Address to be accessed

value=Specifies the data value to be written (16 bits)

### Return value

None

### Remarks

This should only be used by experienced programmers. A directly access to every available register is possible.

### 4.2.3. DapiWriteLong

#### Description

This command performs a direct register write command to the module.

#### Definition

```
void DapiWriteLong(ULONG handle, ULONG adress, ULONG value);
```

#### Parameters

handle=This is the handle of an opened module

adress=Address to be accessed

value=Specifies the data value to be written (32 bits)

#### Return value

None

#### Remarks

This should only be used by experienced programmers. A directly access to every available register is possible.

#### 4.2.4. DapiWriteLongLong

##### **Description**

This command performs a direct register write command to the module.

##### **Definition**

```
void DapiWriteLongLong(ULONG handle, ULONG adress, ULONGLONG value);
```

##### **Parameters**

handle=This is the handle of an opened module

adress=Address to be accessed

value=Specifies the data value that is to be written (64 bits)

##### **Return value**

None

##### **Remarks**

This should only be used by experienced programmers. A directly access to every available register is possible.

## 4.3. Register read commands

### 4.3.1. DapiReadByte

#### Description

This command performs a direct register read command to the module.

#### Definition

*ULONG DapiReadByte(ULONG handle, ULONG adress);*

#### Parameters

handle=This is the handle of an opened module

adress=Address to be accessed

#### Return value

Contents of register-to-read (8 bits)

#### Remarks

This should only be used by experienced programmers. A directly access to every available register is possible.

### 4.3.2. DapiReadWord

#### Description

This command performs a direct register read command to the module.

#### Definition

*ULONG DapiReadWord(ULONG handle, ULONG adress);*

#### Parameters

handle=This is the handle of an opened module

adress=Address to be accessed

#### Return value

Contents of register-to-read (16 bit)

#### Remarks

This should only be used by experienced programmers. A directly access to every available register is possible.

### 4.3.3. DapiReadLong

#### Description

This command performs a direct register read command to the module.

#### Definition

*ULONG DapiReadLong(ULONG handle, ULONG adress);*

#### Parameters

handle=This is the handle of an opened module

adress=Address to be accessed

#### Return value

Contents of register-to-read (32 bit)

#### Remarks

This should only be used by experienced programmers. A directly access to every available register is possible.

#### 4.3.4. DapiReadLongLong

##### **Description**

This command performs a direct register read command to the module.

##### **Definition**

*ULONGLONG DapiReadLongLong(ULONG handle, ULONG adress);*

##### **Parameters**

handle=This is the handle of an opened module

adress=Address to be accessed

##### **Return value**

Contents of register-to-read (64 bit)

##### **Remarks**

This should only be used by experienced programmers. A directly access to every available register is possible.

## 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=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 ;
}

```

# Appendix



# 5. Appendix

## 5.1. Revisions

Rev 1.00	First issue
Rev 2.00	Design change

## 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.