DEDITEC GMBH

USB-RELAIS-8 / USB-OPTOIN-8

Hardware-Description

2024 December

INDEX

1. Introduction 11

1.1. Foreword 12

1.2. Customer satisfaction 12

1.3. Customer response 12

1.4. Brief description 13

1.5. Scope of delivery 14

2. Commissioning 15

2.1. Step 1 - Safety instructions 16

2.2. Step 2 - Installing the software and drivers 17

2.3. Step 3 - Connecting the power supply 27

2.4. Step 4 - Connect to the PC or network 27

2.4.1. Connection via USB 27

2.5. Step 5 - Function test 27

3. Hardware description 28

3.1. General technical data 29

3.1.1. USB-RELAIS-8 29

3.1.2. USB-OPTOIN-8 30

3.2. Overview image of a USB-RELAIS-8 / USB-OPTOIN-8 31

3.3. Interfaces 32

3.3.1. USB 32

3.4. LEDs 32

3.4.1. Definition of the LEDs 32

3.5. Digital outputs 33

3.5.1. Technical data 33

3.5.2. Pin assignment J1 (top) 34

3.5.3. Pin assignment J2 (bottom) 34

3.5.4. Connection example for relay 35

3.5.5. Timeout Function 36

INDEX

3.6. Digital inputs 37
3.6.1. Technical data 37
3.6.2. Pin assignment J1 (top) 38
3.6.3. Pin assignment J2 (bottom) 38
3.6.4. Connection example of an optocoupler 39
3.6.5. Input filter 40
3.6.6. Monitor changes of state 41
4. Software description 42
4.1. Using our products 43
4.1.1. Control via our DELIB driver library 43
4.1.2. Control via supplied test programs 43
4.1.3. Control at protocol level 44
4.1.4. DELIB CLI (command-line interface) für Windows 45
4.1.4.1. Configuration of the DELIB CLI 47
4.1.4.2. DELIB CLI Examples 48
4.1.5. Control via graphical applications 52
4.1.5.1. LabVIEW 52
4.1.5.2. ProfiLab 52
4.1.5.3. Licht24 Pro 53
4.1.6. Integration of the DELIB in programming languages 54
4.1.6.1. Embedding the DELIB in Visual-C/C++ 54
4.1.6.2. Embedding the DELIB in Visual-C/C++ (Visual Studio 2015) 56
4.1.6.3. Embedding the DELIB in Visual-C# 59
4.1.6.4. Embedding the DELIB in Delphi 60
4.1.6.5. Embedding the DELIB in Visual-Basic (VB) 61
4.1.6.6. Embedding the DELIB in Visual-Basic.NET (VB.NET) 62
4.1.6.7. Embedding the DELIB in MS-Office (VBA) 63
4.1.6.8. Embedding the DELIB in LabVIEW 65
4.1.6.8.1. Embedding the DELIB in LabVIEW 65
4.1.6.8.2. Using the VIs in LabVIEW 74
4.1.6.8.3. Setting the module ID in LabVIEW 76

INDEX

4.1.6.9. Embedding the DELIB in Java 78

4.2. DELIB driver library 79

4.2.1. Overview 80

4.2.1.1. Supported programming languages 81

4.2.1.2. Supported operating systems 83

4.2.1.3. SDK kit for programmers 83

4.2.2. DELIB Setup 84

4.2.3. ICT-Tool 89

4.3. DELIB Sample Sources (Windows program examples) 89

4.3.1. Installation DELIB Sample Sources 90

4.3.2. Use of the DELIB Sample Sources 93

4.3.2.1. Step 1 - Product selection 93

4.3.2.2. Step 2 - Category selection 95

4.3.2.3. Step 3 - Programming language selection 96

4.3.2.4. Step 4 - Source code 97

4.4. DELIB for Linux 100

4.4.1. Using the DELIB driver library for Linux 102

4.4.1.1. Delib USB sample in Linux 102

4.4.1.2. Delib ETH sample in Linux 105

4.4.2. DELIB CLI (command-line interface) for Linux 108

4.4.2.1. Configuration of the DELIB CLI 111

4.4.2.2. DELIB CLI Examples 114

4.5. Web interface 115

4.5.1. Configuration 117

4.5.1.1. General 117

4.5.1.2. Network configuration 118

4.5.1.3. User Manager 119

4.5.1.4. Status 122

4.5.1.5. Security 123

4.5.2. Inputs/outputs 125

4.5.2.1. General 125

4.5.2.2. Digital inputs 126

INDEX

4.5.2.3. Digital inputs Counter 128
4.5.2.4. Digital outputs 130
4.5.2.5. Analog inputs 132
4.5.2.6. Analog outputs 133
4.5.2.7. Configuration 134

5. ICT-Tool 137

5.1. General descriptions 138

5.2. Add module 143

5.2.1. USB Module 145
5.2.2. ETH Module 146
5.2.2.1. Ethernet standard method 147
5.2.2.2. Ethernet Direct method 149
5.2.2.3. Search via Broadcast 151
5.2.3. CAN Module 152
5.2.4. SER Module 153

5.3. Configure module 155

5.3.1. Module info 156
5.3.2. Module identification 157
5.3.3. Module restart 158
5.3.4. USB configuration 159
5.3.5. LAN network information 160
5.3.6. LAN network settings 162
5.3.7. WiFi network information 164
5.3.8. WiFi network settings 166
5.3.9. WiFi WPS connection 168
5.3.10. NTP configuration 169
5.3.11. TCP encryption 171
5.3.12. WEB-Login 176
5.3.13. Serial configuration 177
5.3.14. I/O channel names 180
5.3.15. D/A default values 181

INDEX

5.3.16. CAN configuration 182
5.3.16.1. CAN status interface 182
5.3.16.2. CAN statistics TX/RX 184
5.3.16.3. CAN Main Interface 185
5.3.16.4. CAN Main I/O Init 186
5.3.16.5. CAN TX Mode 188
5.3.16.6. CAN RX Mode 190
5.3.17. Watchdog configuration 191
5.4. M2M configuration 192
5.4.1. Statistics 193
5.4.2. Receiver module 194
5.4.3. Jobs 196
5.5. Event control 198
5.5.1. Statistics 199
5.5.2. Event configuration 200
5.5.3. Actions 202
5.6. Test module 203
5.6.1. Timeout test function 203
5.6.2. Digital Out 204
5.6.3. Digital TTL 206
5.6.4. Digital In 208
5.6.5. Digital In Counter 209
5.6.6. Analog In 211
5.6.7. Analog Out 212
5.6.8. Temp 214
5.6.9. Graphical representation 215
5.6.10. CAN Runtime Parameter 217
5.6.11. Puls Generator 218
5.6.12. PWM Out 219
5.6.13. Watchdog switching behavior 221
5.7. Diagnose module 222

INDEX

5.7.1. Digital cable return test 222
5.7.2. Analog cable return test 225
5.7.3. Access time tests 227
5.7.4. A/D noise graph 229
5.7.5. A/D read with ripple 230
5.7.6. FIFO In/Out 231
5.7.7. Main Loop 233
5.7.8. Module-Config-Memory 234
5.8. Firmware update 235
5.8.1. About DEDITEC firmware 235
5.8.2. Update flash files via ICT tool 236
5.8.3. Perform firmware update 237
5.9. Settings 239
5.9.1. General 239
5.9.2. Updates 240
5.9.3. Error log 241
5.9.4. DELIB-DebugView-Global 242
5.9.5. DELIB-DebugView-Details 243
6. DELIB API Reference 245
6.1. Available DEDITEC module IDs 246
6.2. DELIB directory structure 250
6.2.1. Include directory 252
6.2.2. Library directory 252
6.2.3. Library directory for Borland 252
6.2.4. Environment variables 252
6.3. Administrative functions 252
6.3.1. DapiOpenModule 252
6.3.2. DapiCloseModule 254
6.3.3. DapiGetDELIBVersion 254
6.3.4. DapiSpecialCMDGetModuleConfig 255

INDEX

6.3.5. DapiOpenModuleEx 258
6.4. Error handling 260
6.4.1. DapiGetLastError 260
6.4.2. DapiGetLastErrorText 261
6.4.3. DapiClearLastError 262
6.4.4. DapiGetLastErrorByHandle 263
6.4.5. DapiClearLastErrorByHandle 264
6.5. Read digital inputs 265
6.5.1. DapiDIGet1 265
6.5.2. DapiDIGet8 265
6.5.3. DapiDIGet16 266
6.5.4. DapiDIGet32 267
6.5.5. DapiDIGet64 268
6.5.6. DapiDIGetFF32 269
6.5.7. DapiDIGetCounter 270
6.5.8. DapiSpecialCounterLatchAll 271
6.5.9. DapiSpecialCounterLatchAllWithReset 272
6.5.10. DapiSpecialDIFilterValueSet 273
6.5.11. DapiSpecialDIFilterValueGet 274
6.5.12. Dapi_Special_DI_FF_Filter_Value_Get 275
6.5.13. Dapi_Special_DI_FF_Filter_Value_Set 276
6.6. Manage digital outputs 277
6.6.1. DapiDOSet1 277
6.6.2. DapiDOSet8 277
6.6.3. DapiDOSet16 278
6.6.4. DapiDOSet32 279
6.6.5. DapiDOSet64 280
6.6.6. DapiDOSet1_WithTimer 281
6.6.7. DapiDOReadback32 282
6.6.8. DapiDOReadback64 282
6.6.9. DapiDOSetBit32 283

INDEX

6.6.10. DapiDOClrBit32 284
6.7. Manage output timeout 285
6.7.1. DapiSpecialCMDTimeout 285
6.7.1.1. DapiSpecialTimeoutSetValueSec 288
6.7.1.2. DapiSpecialTimeoutActivate 289
6.7.1.3. DapiSpecialTimeoutActivateAutoReactivate 290
6.7.1.4. DapiSpecialTimeoutActivateSecureOutputs 291
6.7.1.5. DapiSpecialTimeoutDeactivate 292
6.7.1.6. DapiSpecialTimeoutGetStatus 293
6.7.1.7. DapiSpecialTimeoutDoValueMaskWRSet32 294
6.7.1.8. DapiSpecialTimeoutDoValueMaskRDSet32 295
6.7.1.9. DapiSpecialTimeoutDoValueMaskWRClr32 296
6.7.1.10. DapiSpecialTimeoutDoValueMaskRDClr32 297
6.7.1.11. DapiSpecialTimeoutDoValueLoadDefault 298
6.8. Test functions 299
6.8.1. DapiPing 299
6.9. Register write commands 300
6.9.1. DapiWriteByte 300
6.9.2. DapiWriteWord 301
6.9.3. DapiWriteLong 302
6.9.4. DapiWriteLongLong 303
6.10. Register read commands 304
6.10.1. DapiReadByte 304
6.10.2. DapiReadWord 305
6.10.3. DapiReadLong 306
6.10.4. DapiReadLongLong 307
6.11. Programming example 308
6.12. Delib overview table 310
7. Security measures 314
7.1. Write protection 315

INDEX

7.2. TCP encryption 315

7.3. Web interface 315

8. Appendix 317

8.1. Contact / Support 318

8.2. Environment and disposal 318

8.3. Revisions 319

8.4. Copyrights and trademarks 320

Introduction



1. Introduction

1.1. Foreword

Congratulations on purchasing a high quality DEDITEC product!

Our products are developed by our engineers according to today's required quality standards. We pay attention already during the development to flexible expandability and long availability.

We develop modular!

Due to a modular development we shorten the development time and - what of course benefits the customer - we sell at a fair price!

We ensure a long delivery availability!

If used semiconductors are no longer available, we can react faster. With us mostly only modules have to be redesigned and not the whole product. This increases the delivery availability.

1.2. Customer satisfaction

A satisfied customer is our first priority!

If something is not to your satisfaction, just contact us by phone or mail.

We will take care of it!

1.3. Customer response

The best products grow with our customers. We are always grateful for any suggestions or proposals.

1.4. Brief description

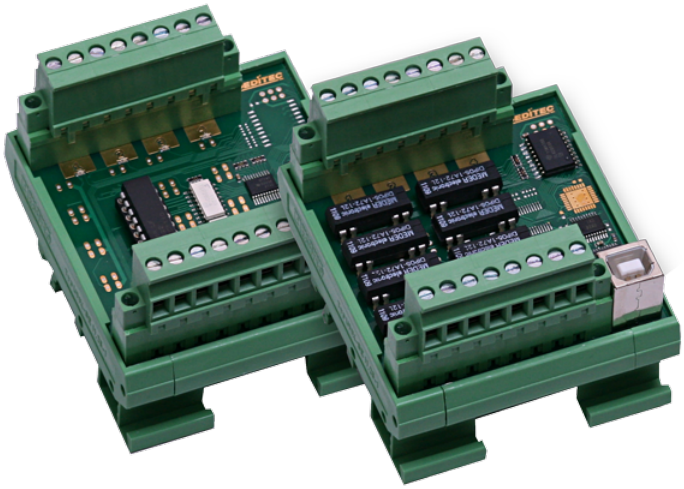
Our USB modules of the starter series with pure digital outputs or inputs are a cost-effective variant for the entry into the mobile world of digital I/Os.

The starter modules with 8 optocoupler inputs (USB-OPTOIN-8) or 8 relay outputs (USB-RELAIS-8) are compact stand-alone USB modules.

Digital data signals are acquired or output here via a USB interface. These signals are provided and processed by the customer application on the control PC. Our DELIB API, for example, or direct communication via our Ethernet protocol can be used as a simple programming interface.

The "ICT-Tool" configuration tool included in the scope of delivery also makes commissioning quick and easy.

The housing consists of a compact plastic profile and is suitable for mounting on top-hat rails, as typically used in control cabinets.



1.5. Scope of delivery

The following items are included:

- USB-RELAIS-8 or USB-OPTOIN-8 module
- 2x 8 pin connectors
- USB cable 1,5m
- Installation CD with manuals and drivers

Commissioning



2. Commissioning

2.1. Step 1 - Safety instructions

Before commissioning your DEDITEC product, please familiarize yourself with this manual and read the following points carefully:

- Damage caused by non-observance of these operating instructions will void the warranty or guarantee of this product. We do not accept any liability for consequential damage!
- We do not accept any liability for damage to property or personal injury that could result from improper handling or non-observance of the safety instructions!
- Avoid touching electronic components on the circuit board directly. This could lead to electrostatic discharges and destroy sensitive components. As a precaution, always discharge before touching an electrically grounded object.
- Unauthorized conversions or technical modifications to this product are not permitted for safety and approval reasons (CE) and will void the warranty or guarantee.
- Do not operate the module outside the maximum permissible technical data.
- The product is not suitable for operation in damp or wet environments.

2.2. Step 2 - Installing the software and drivers

The DELIB setup guides you through the installation of our DELIB driver library.

You will then be guided through the configuration process and the function test for our various products.

You can download our current version of the DELIB setup for Linux and Windows in the 32-bit or 64-bit version from our homepage.

Link: https://www.deditec.de/d/ndelib

The DELIB setup guides you step by step through the installation of the DELIB driver library.

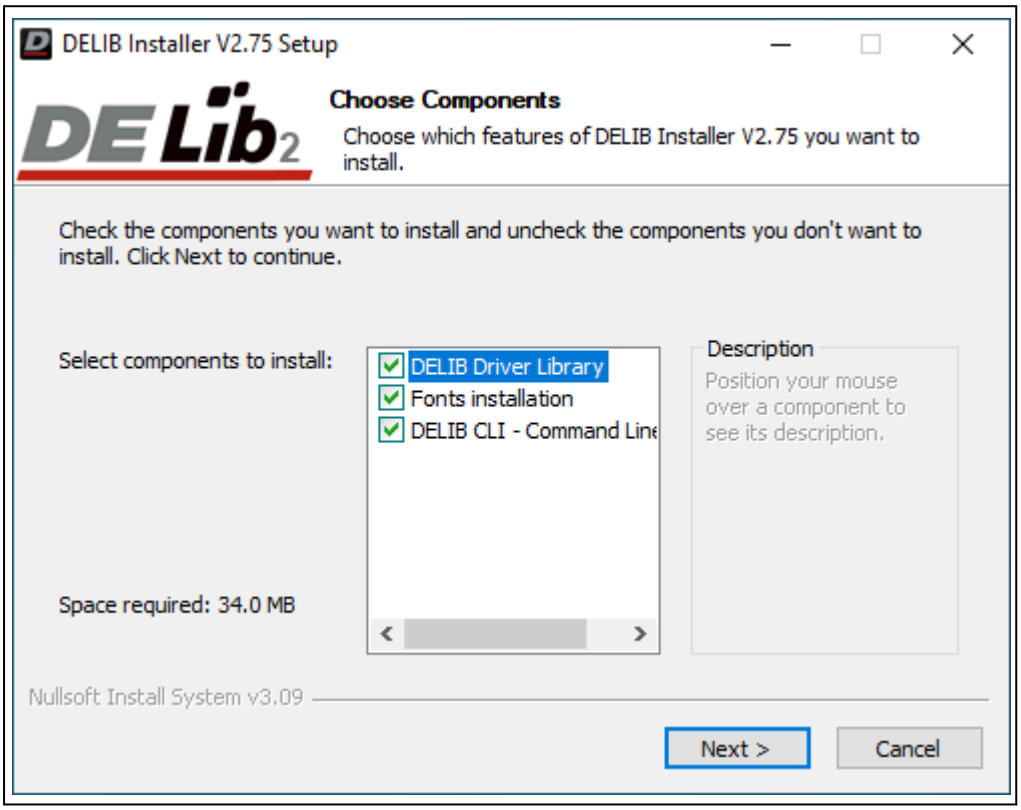


In addition to the DELIB driver library, you also have the option of installing our “DELIB CLI - Command Line” software.

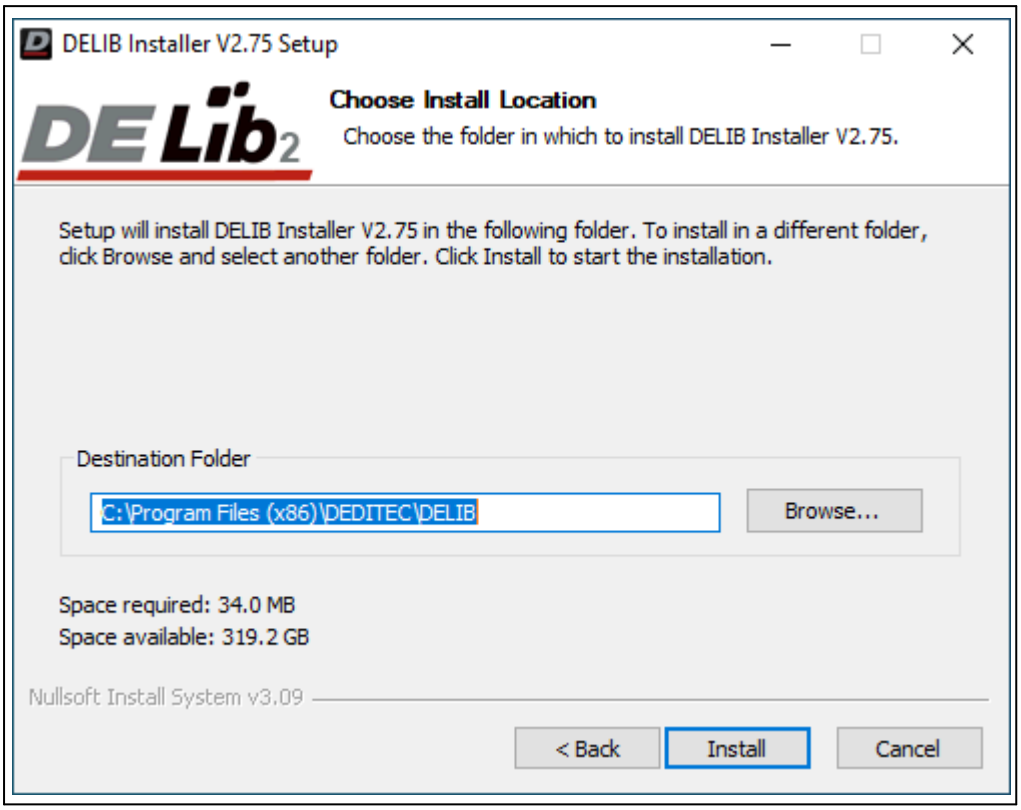
This program allows you to execute commands directly via the command line. You can find more information on this in the following chapter:

DELIB CLI (command-line interface) for Windows

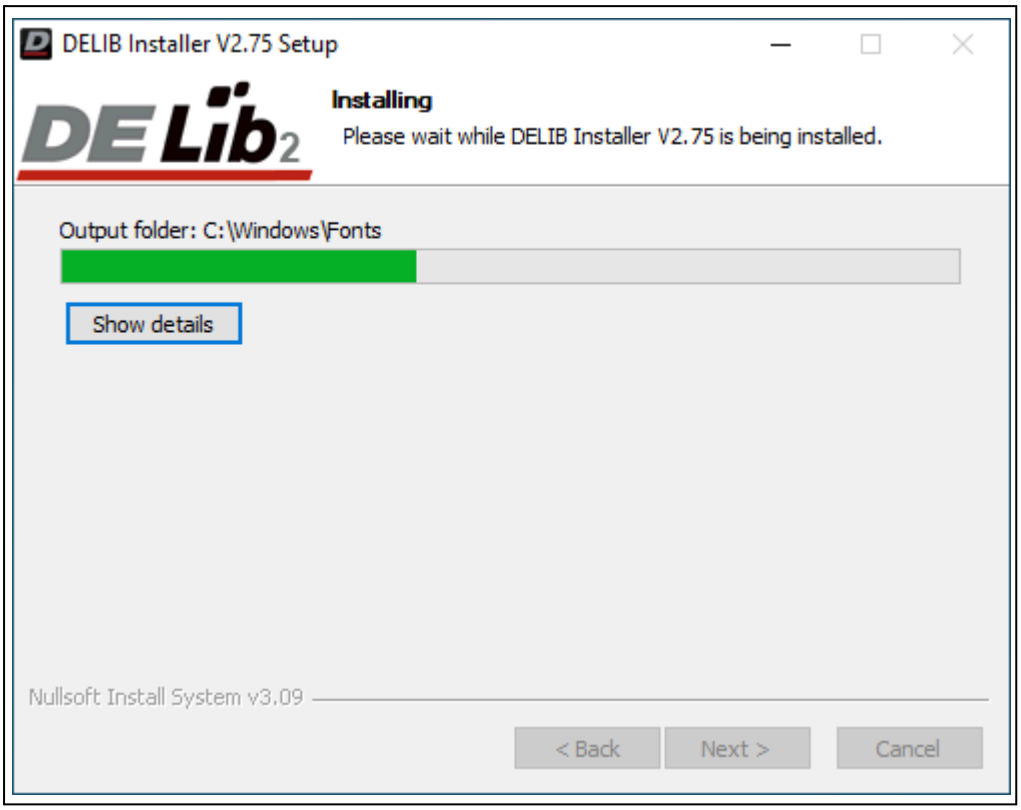
DELIB CLI (command-line interface) for Linux



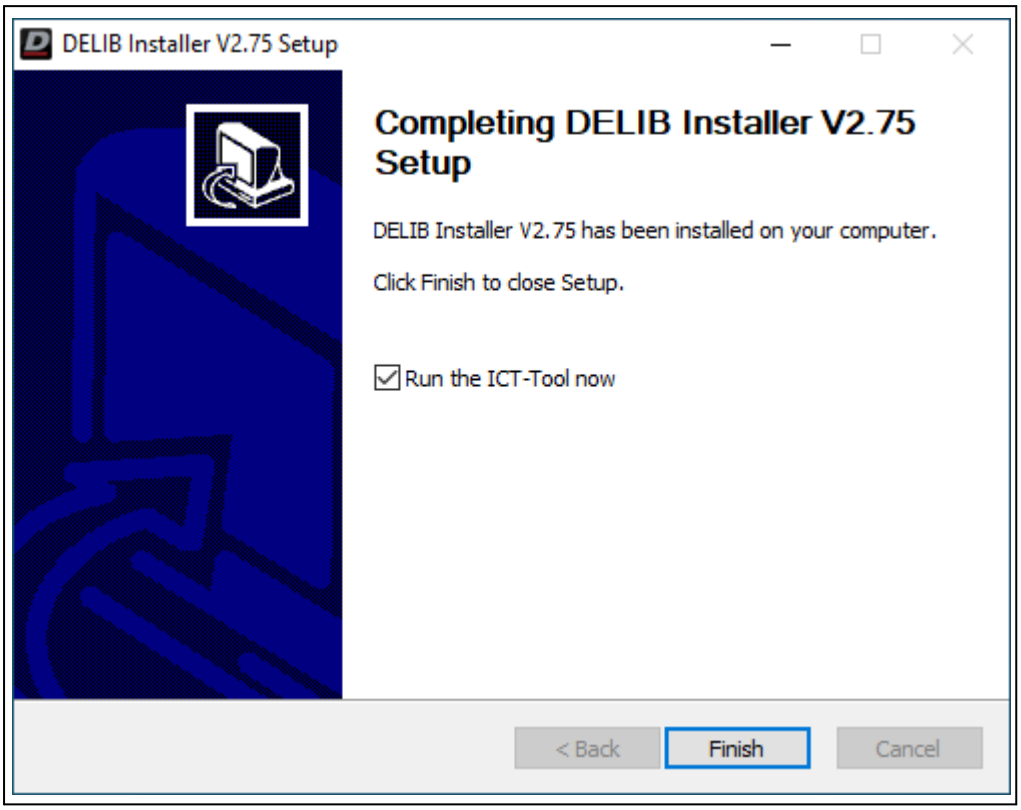
Select the desired installation path and confirm with the “Install” button.



Progress of the driver library installation.



Once the installation is complete, close the setup with the “Finish” button.



With “Run the ICT-Tool now” you have the option to start the ICT-Tool directly after exiting the DELIB setup.

In addition, you will now find the ICT-Tool and the DELIB-CLI in your Windows “Start” directory.

The DELIB Setup guides you through the installation of our DELIB driver library.
You will then be guided through the configuration process and function test for our various products.

Our current version of the DELIB setup for Linux and Windows in the 32-bit or 64-bit versions can be downloaded from our homepage.
Link: https://www.deditec.de/en/delib

The DELIB setup guides you step by step through the installation of the DELIB driver library.

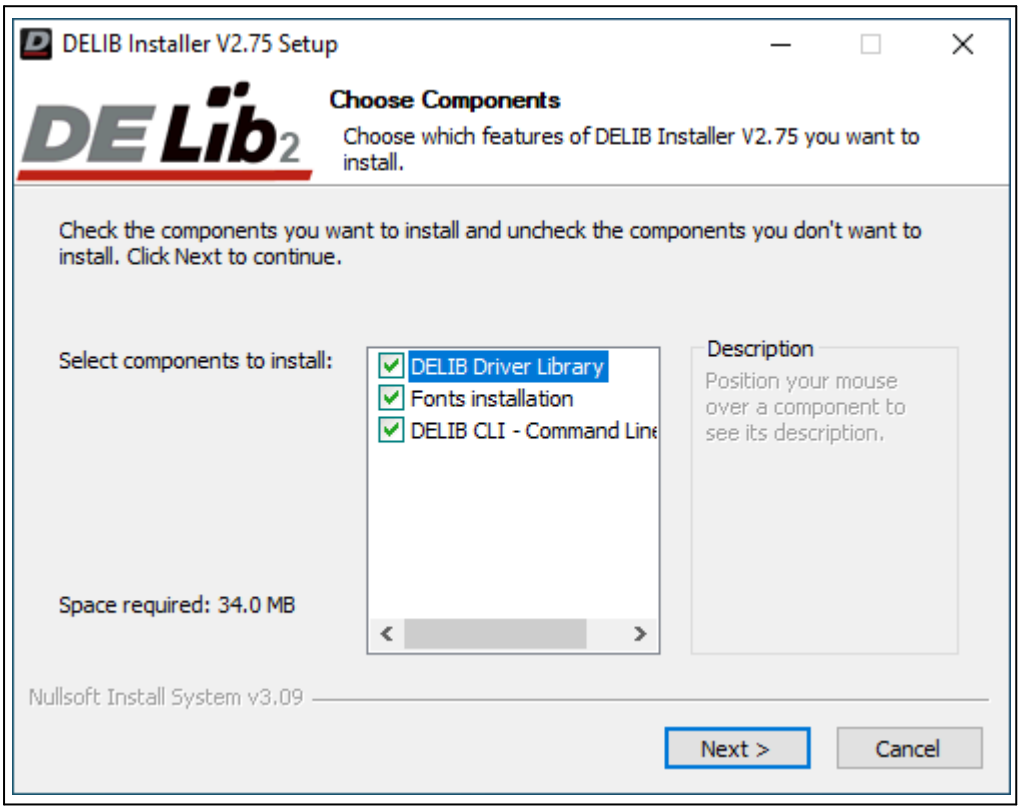


In addition to the DELIB driver library, you also have the option of installing our "DELIB CLI - Command Line" software.

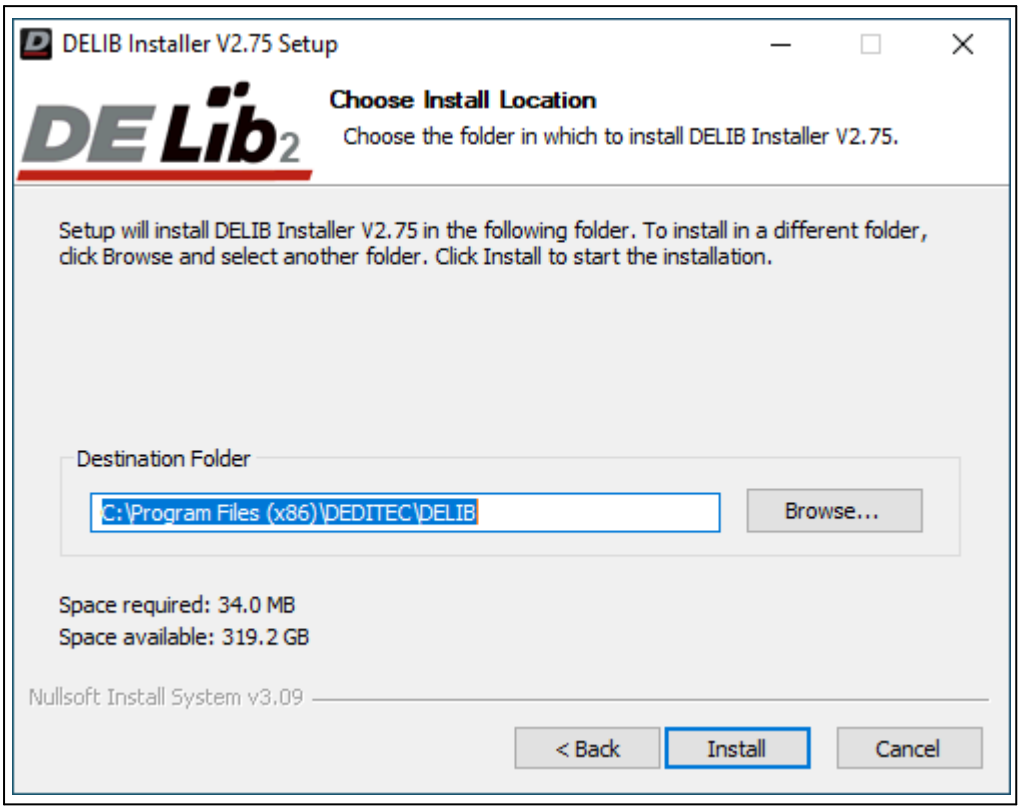
You can use this program to execute commands directly from the command line. You can find more information on this in the following chapter:

DELIB CLI (command-line interface) for Windows

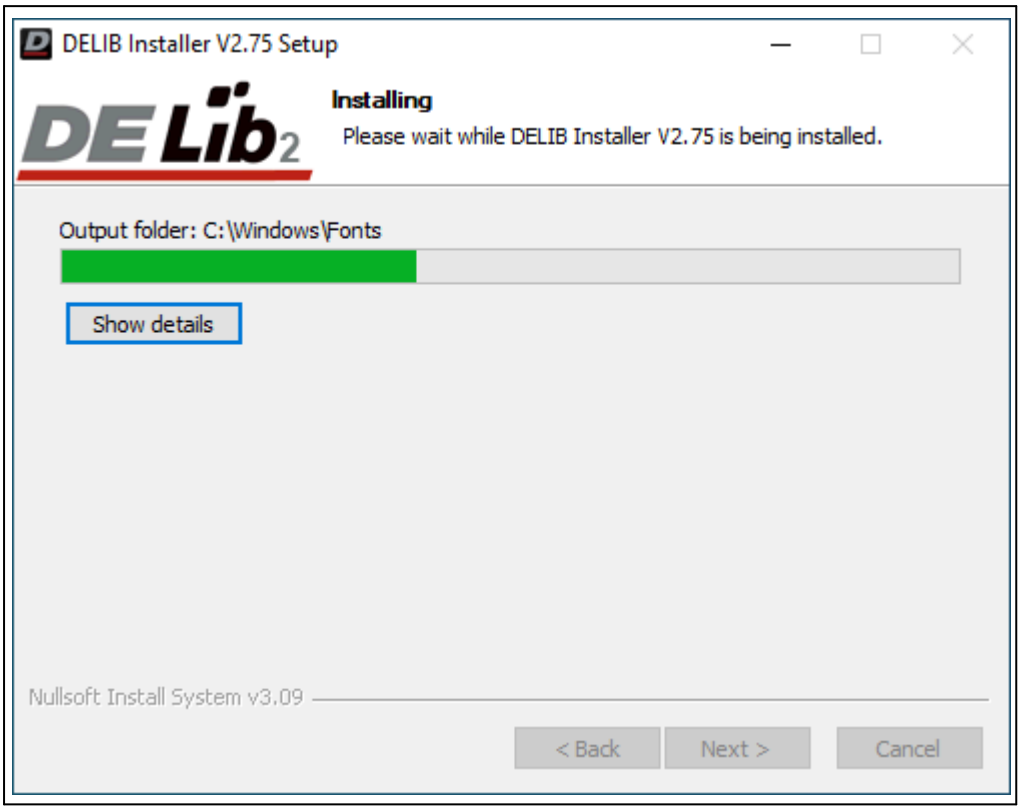
DELIB CLI (command-line interface) for Linux



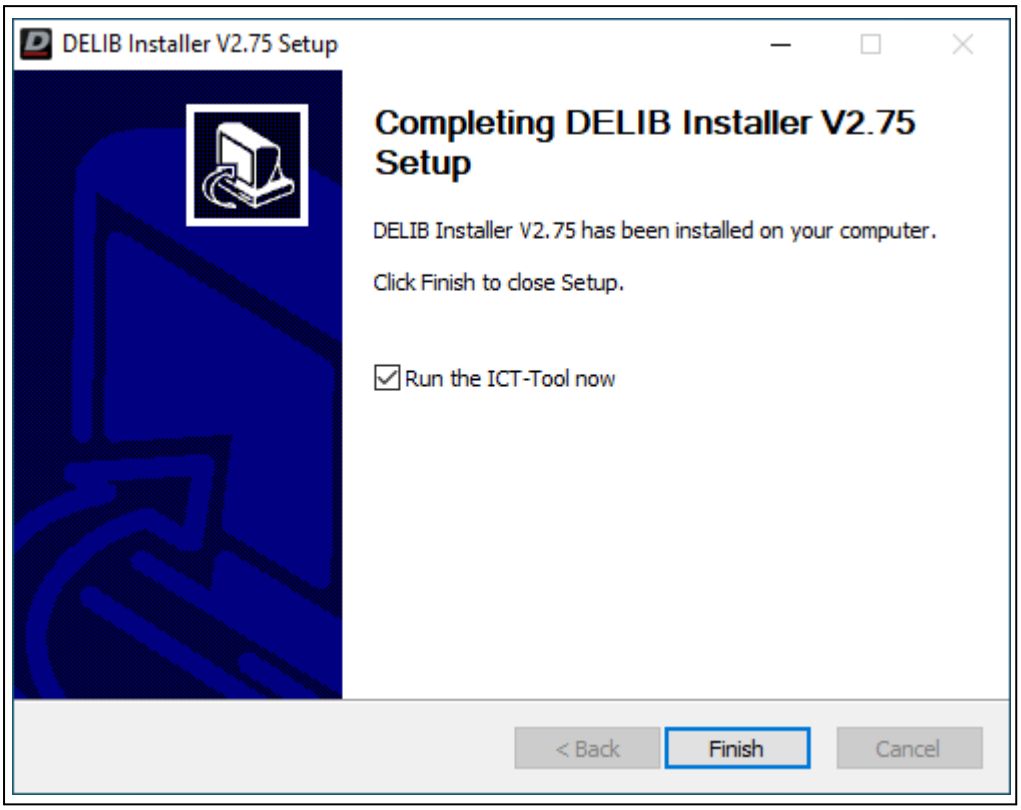
Select your desired installation path and confirm with the "Install" button



Installation progress of the driver library.



After completing the installation, close the setup with the "Finish" button



With "Run the ICT-Tool now" you have the option of starting the ICT-Tool directly after exiting the DELIB setup.

In addition, you will now find the ICT tool and the DELIB-CLI in your Windows "Start" directory.

2.3. Step 3 - Connecting the power supply

With our pure USB modules of the starter series, the module is supplied with the necessary supply voltage of +5V via the USB interface.

An external supply is not required.

2.4. Step 4 - Connect to the PC or network

Once you have successfully installed the driver library, start the "ICT-Tool" program in the installation path you have selected or in the Windows start menu.

2.4.1. Connection via USB

Connect the module to your PC or USB hub using the USB cable supplied.

See chapter → **USB Configuration**

2.5. Step 5 - Function test

With our tool "ICT-Tool" you can start up the module relatively fast and easy and without programming knowledge and check its functionality.

To do this, follow the instructions in the **"ICT-Tool"** chapter.

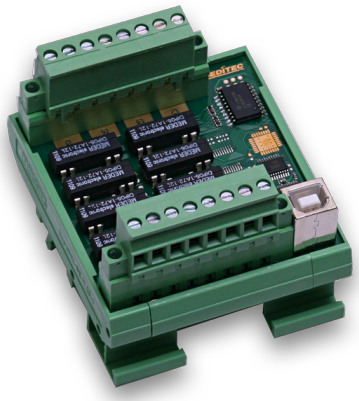
Hardware description



3. Hardware description

3.1. General technical data

3.1.1. USB-RELAIS-8



Electrical data:

Supply voltage	+5V (via USB bus)
Power consumption:	max. 5W

Environment:

Ambient temperature:	+10..+50 °C
Humidity:	90 %
Dewing:	Not allowed

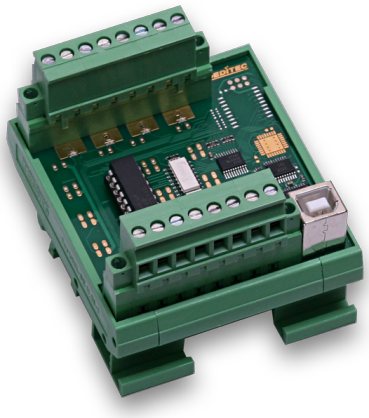
Mechanics:

Dimensions in mm (LxWxH):	67,5 x 77 x 55
Mounting:	Top-hat rail TS 35 x 7.5 mm

Possible variants:

Produkt	Connection wiring	Act.-LED	Housing
USB-RELAIS-8	Plug-in screw terminal	1 per output	Yes
USB-RELAIS-8_A	Screw terminal not pluggable	-	-
USB-RELAIS-8_B	Plug-in screw terminal	1 per output	-

3.1.2. USB-OPTOIN-8



Electrical data:

Supply voltage	+5V (via USB bus)
Power consumption:	max. 5W

Surroundings:

Ambient temperature:	+10..+50 °C
Humidity:	90 %
Condensation:	Not allowed

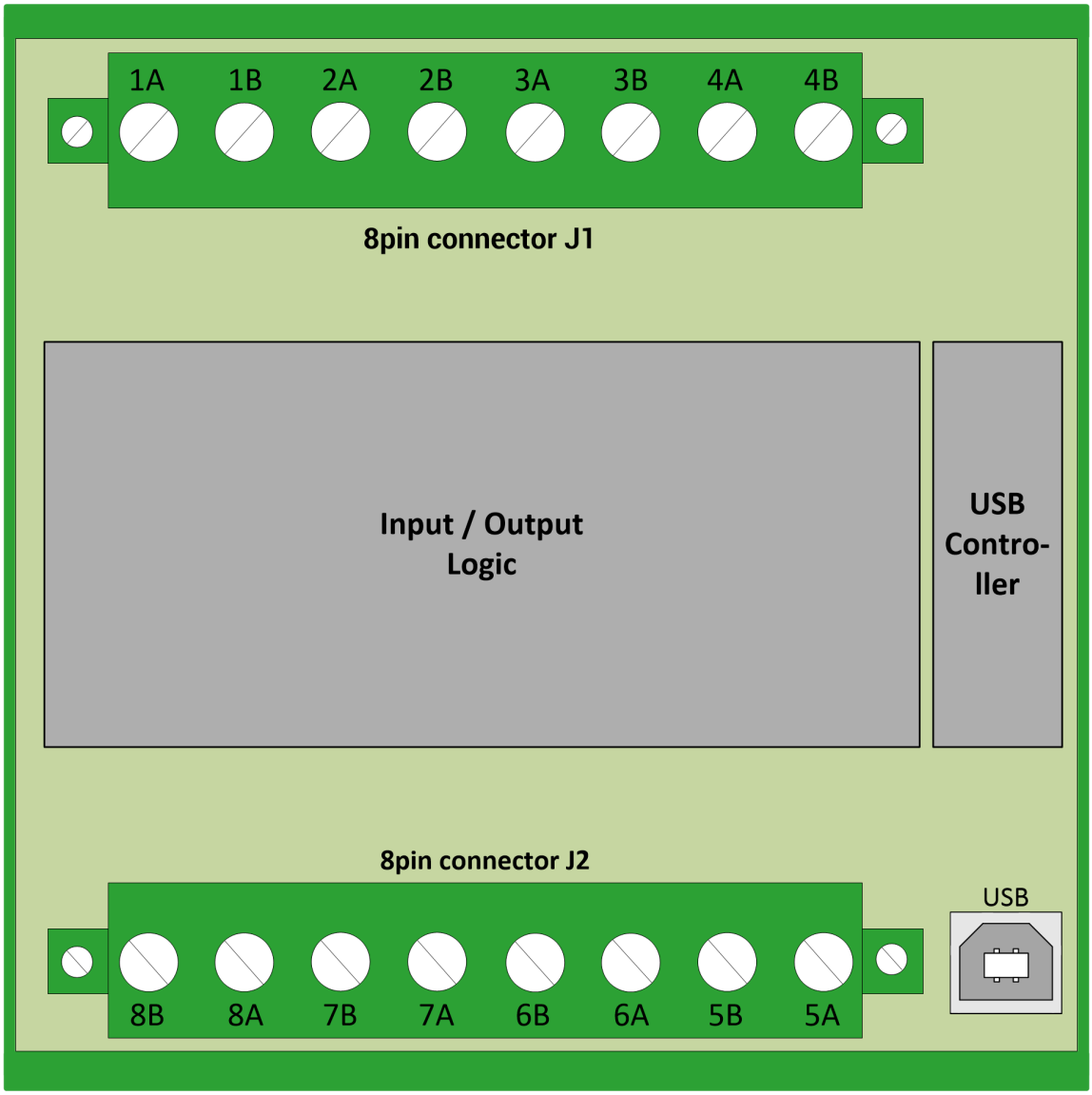
Mechanics:

Dimensions in mm (LxWxH):	67,5 x 77 x 55
Mounting:	Top-hat rail TS 35 x 7,5 mm

Possible variants:

Product	Connection wiring	Act.-LED	Housing
USB-OPTOIN-8	Plug-in screw terminal	1 per output	Yes
USB-OPTOIN-8_A	Screw terminal not pluggable	-	-
USB-OPTOIN-8_B	Plug-in screw terminal	1 per output	-

3.2. Overview image of a USB-RELAIS-8 / USB-OPTOIN-8



3.3. Interfaces

3.3.1. USB

Technical data:

Standard:	USB 1.1 / USB 2.0
Connection establishment:	USB type B
Access time PC to module*:	4,06 ms**

* Calculated with 1000 accesses to the module via the DELIB driver library with the command DapiDoSet32

** average time for 32-bit accesses

3.4. LEDs

3.4.1. Definition of the LEDs

LED ACT:

Lights up during USB activity.

LED I/O:

Each digital input or output has a separate LED that lights up when the signal state is active.

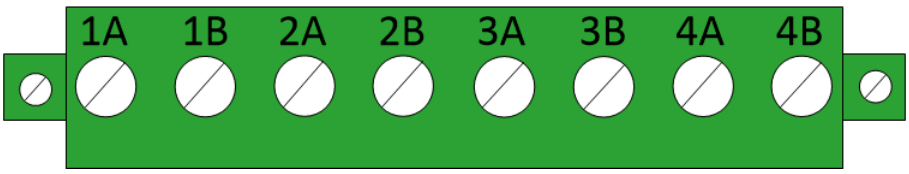
3.5. Digital outputs

3.5.1. Technical data

Technical data relay 1A

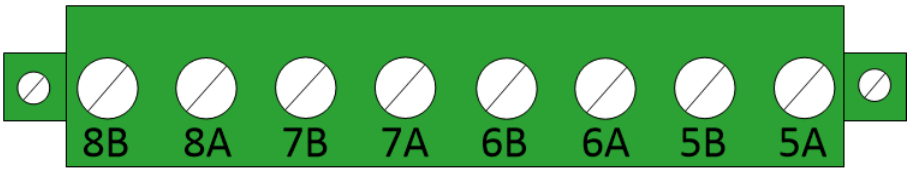
Number of channels per block:	8
Typ:	Closer (NO)
Max. Switching voltage:	36V AC / DC
Max. Switching current:	0.5A AC / DC
Max. Transport current:	1A AC / DC
Max. Switching power:	10W

3.5.2. Pin assignment J1 (top)



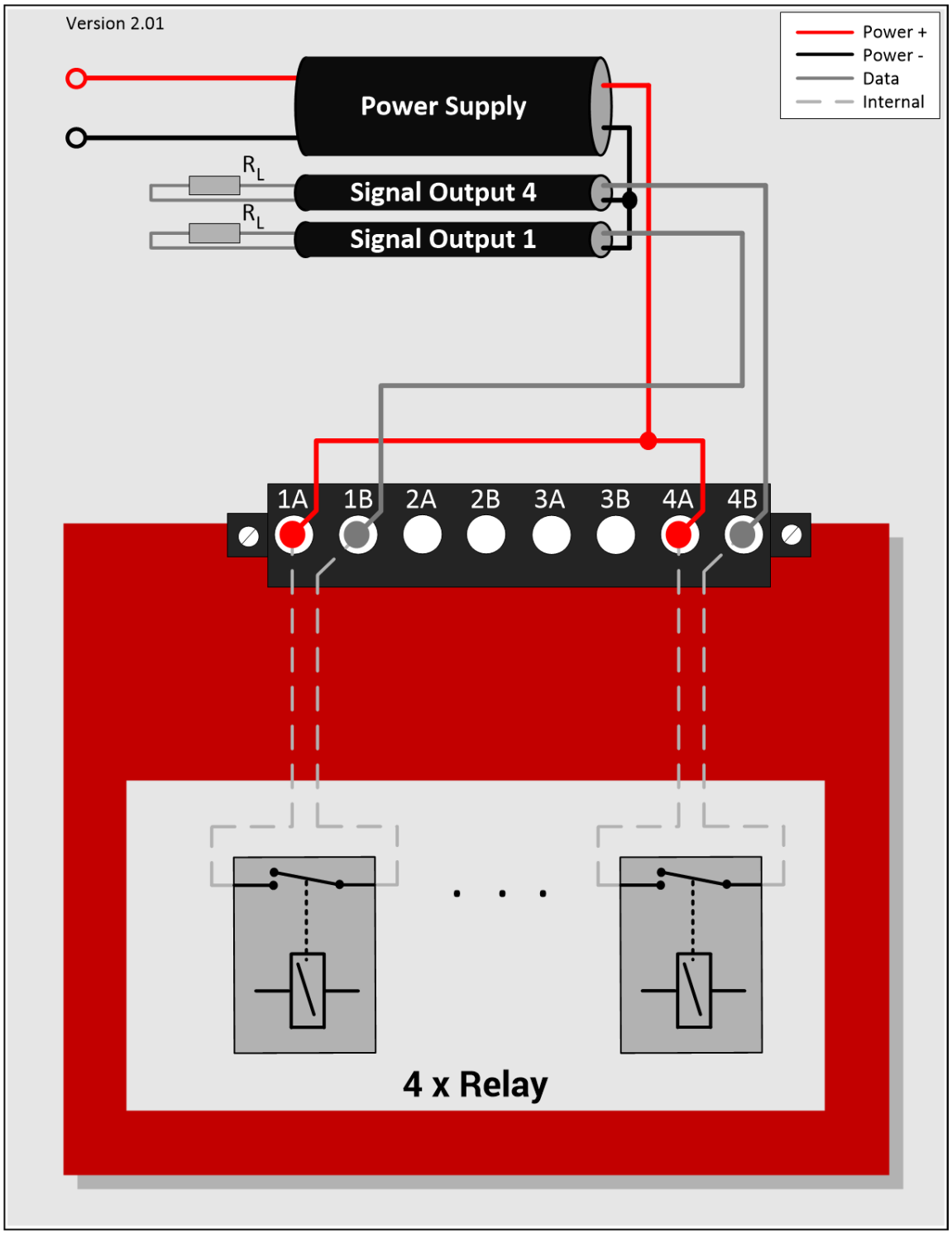
Digital Output Channel	Pin	
Output Channel 1	1A	1B
Output Channel 2	2A	2B
Output Channel 3	3A	3B
Output Channel 4	4A	4B

3.5.3. Pin assignment J2 (bottom)



Digital Output Channel	Pin	
Output Channel 5	5A	5B
Output Channel 6	6A	6B
Output Channel 7	7A	7B
Output Channel 8	8A	8B

3.5.4. Connection example for relay



3.5.5. Timeout Function

The timeout function offers the possibility to automatically switch the outputs on or off in case of a connection loss between the control PC and the DEDITEC module. This can be defined for each individual channel by software.

Functions:

- Time definable automatic activation of the timeout protection function in case of timeout (between 0.1 seconds and 6553 seconds).
- In timeout case digital outputs can be activated, deactivated or left unchanged.
- 3 different timeout modes: "normal", "auto reactive" and "secure output" for different procedures in case of timeout.

For more details see Chapter→ **Manage output timeout.**

3.6. Digital inputs

3.6.1. Technical data

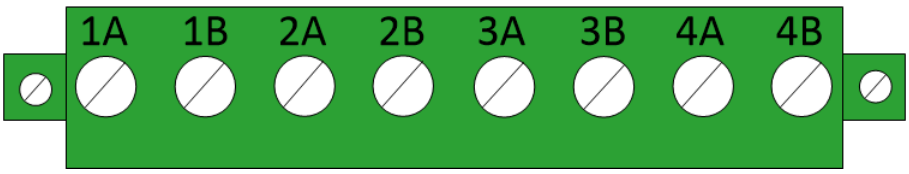
Technical data optocoupler

Number of channels per block:	8
Voltage ranges:	15V – 30V DC/AC (optional 5V – 15V or 30V – 50V DC/AC)
Input current:	max. 14mA
Galvanic isolation:	up to 2.5kV AC for 1 minute

Additional functions

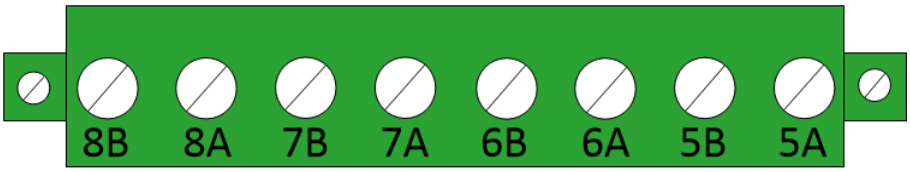
- 16 bit counter per channel.
- Maximum possible counts: 65535 / channel. Reset to zero after memory overflow.
- Internal counting logic up to 10kHz with latch function.
- Programmable filter for input channels (flip-flop and counter): Minimum low or high pulse duration: 5ms...255ms.
- Detects change from low to high and high to low level.
- Detection of input state change between two readouts.

3.6.2. Pin assignment J1 (top)



Digital Input Channel	Pin	
Input Channel 1	1A	1B
Input Channel 2	2A	2B
Input Channel 3	3A	3B
Input Channel 4	4A	4B

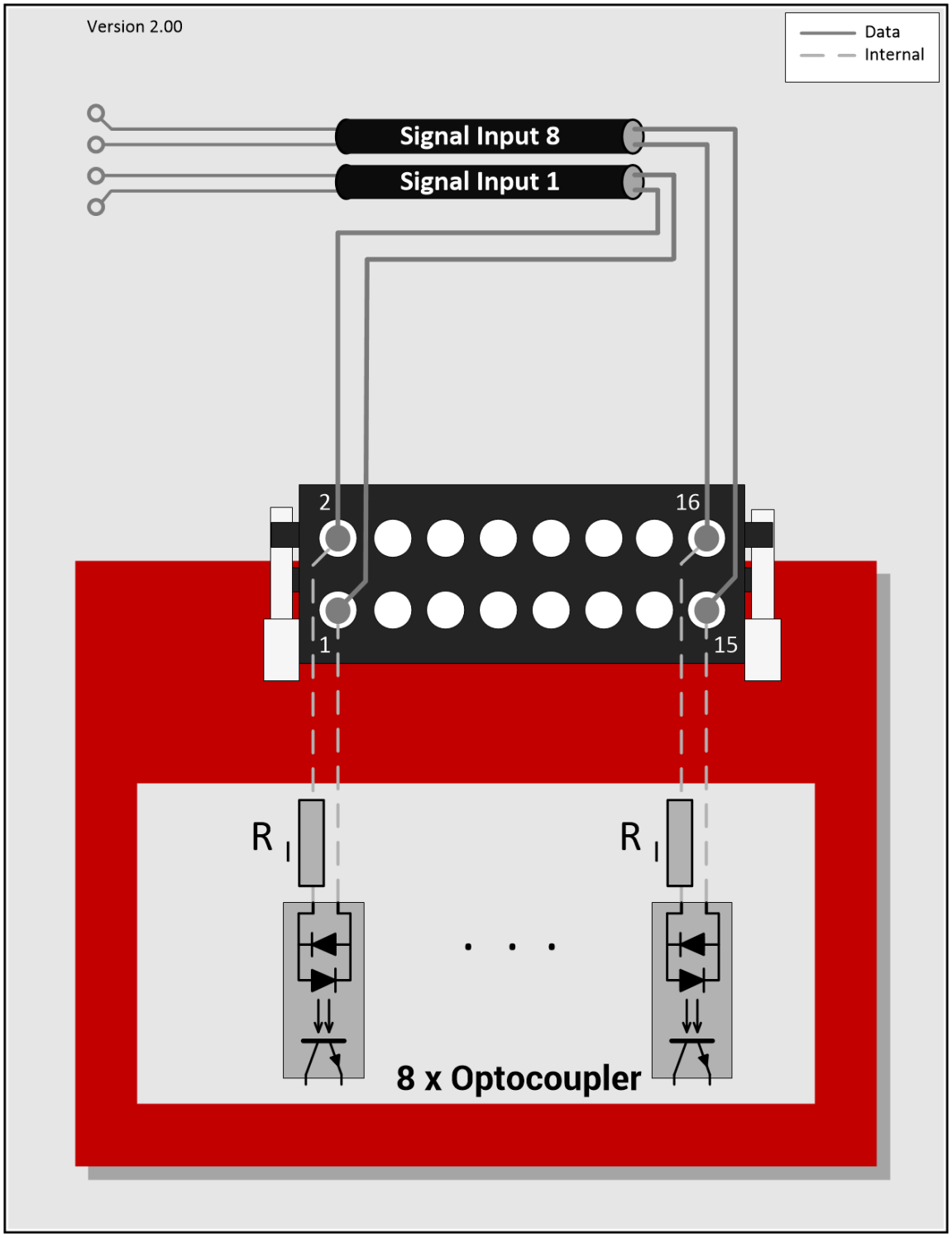
3.6.3. Pin assignment J2 (bottom)



Digital Input Channel	Pin	
Input Channel 5	5A	5B
Input Channel 6	6A	6B
Input Channel 7	7A	7B
Input Channel 8	8A	8B

3.6.4. Connection example of an optocoupler

When connecting a DC voltage source, the polarity can be neglected.

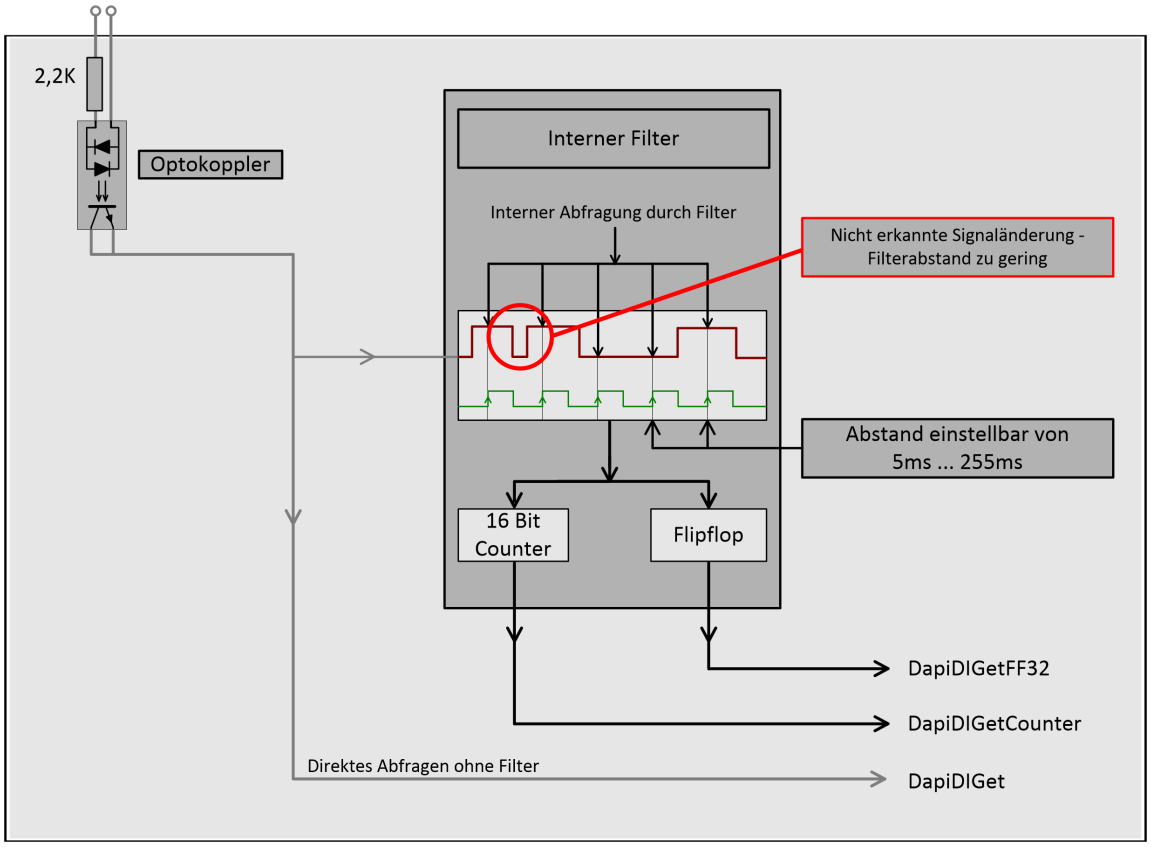


3.6.5. Input filter

Each input can be provided with a digital filter function. Unwanted pulses are thus ignored by the system. The valid pulse duration can be set on the software side between 0..255ms. A value of 0 ms means that the filter is not active.

See also chapter → **Digital inputs read**

Schematic view of the filter:



3.6.6. Monitor changes of state

This function makes it possible to monitor state changes at the inputs. An internal logic detects a change of state from High to Low or vice versa and writes this information into a register. The I-Change LED lights up. By reading the software registers, this information can be reset and the LED deactivated.

See chapter: → **Read digital inputs → DapiDiGetFF32**

Software description



4. Software description

4.1. Using our products

4.1.1. Control via our DELIB driver library

Included in the delivery of our DELIB driver library is the DELIB-API and various programs for the configuration test of our products.

The API gives you access to all functions you need to communicate with our products.

In the chapter **DELIB API Referenz** you will find all functions of our driver library explained and provided with application examples.

4.1.2. Control via supplied test programs

With our ICT-Tool you can test the functionality of our control & regulation products without much configuration effort.

For detailed information see chapter **ICT-Tool.**

4.1.3. Control at protocol level

For products with Ethernet, CAN or serial interface we offer our open protocols.

These protocols can be used without our DELIB driver library on devices with corresponding interface. The way over our protocols are operating system independent.

Our manual, protocols & register assignment can be found here:

Download PDF:

http://www.deditec.de/pdf/manual_d_deditec_communication_protocols.pdf

Online HTML manual:

http://manuals.deditec.de/de/manual_deditec_communication_protocols/index.html

This manual provides a complete overview of the required register addresses of our modules as well as the structure of the different communication protocols.

4.1.4. DELIB CLI (command-line interface) für Windows

Since in some programming languages (such as PHP) no DLLs can be included, there is an extra command line command for this, which can be called directly from the program (with the appropriate parameters).

The DELIB CLI command for Windows is located after the installation of the DELIB driver library in the directory C:\Programs\DEDITEC\DELIB\programs\cli\ .

Definition (Windows)

delib_cli command channel [value | unit ["nounit"]]

Note: The individual parameters are separated only by a space.

Upper and lower case are not considered here.

Parameter

<table>
<tr><th>Command</th><th>Channel</th><th colspan="2">Value</th><th>unit</th><th>nounit</th></tr>
<tr><td>di1</td><td>0, 1, 2, ...</td><td colspan="2" rowspan="4">-</td><td rowspan="4">hex</td><td rowspan="4">nounit</td></tr>
<tr><td>di8</td><td rowspan="3">0, 8, 16, ...</td></tr>
<tr><td>di16</td></tr>
<tr><td>di32</td></tr>
<tr><td>ff</td><td>0, 32, ...</td><td colspan="2">-</td><td>hex</td><td>nounit</td></tr>
<tr><td>do1</td><td>0, 1, 2, ...</td><td colspan="2">0/1 (1-Bit Command)</td><td rowspan="4">-</td><td rowspan="4">-</td></tr>
<tr><td>do8</td><td rowspan="3">0, 8, 16, ...</td><td>8-Bit Value</td><td rowspan="3">(Bit 0 for channel 1, Bit 1 for channel 2, ...)</td></tr>
<tr><td>do16</td><td>16-Bit Value</td></tr>
<tr><td>do32</td><td>32-Bit Value</td></tr>
</table>

Command	Channel	Value	unit	nounit
ai	0, 1, 2, ...	-	hex, volt, mA	nounit
ao	0, 1, 2, ...	Integer or hexadecimal number (starting with 0x).	-	-

Return-Value

State of the read digital inputs

In combination with parameter unit "hex" the state is read as hex

State of the FlipFlips of the digital inputs

In combination with parameter unit "hex" the state is read as hex

Status of the read analog inputs

In combination with parameter unit "hex" the state is read as hex

In combination with parameter unit "volt" the voltage is read

In combination with parameter unit "mA" the current is read

4.1.4.1. Configuration of the DELIB CLI

Before using the DELIB CLI for the first time, the "delib_cli.cfg" must be edited with a text editor.

Configuration under Windows

Under Windows the "delib_cli.cfg" is located in the directory "C:\Programs\DEDITEC\DELIB\programs\cli\" after the installation of the DELIB driver library.

Contents of the "delib_cli.cfg":

```
moduleID=14;
moduleNR=0;
RO-ETH_ipAddress=192.168.1.11;
```

moduleID

The corresponding number of the hardware used must be entered as moduleID.

This number can be taken from the "delib.h".

Under Windows you will find this in the directory C:\Programs\DEDITEC\DELIB\include\.

moduleNR

The moduleNR is assigned in the ICT-Tool.

This number is used to identify identical hardware.

The default value is 0.

RO-ETH_ipAddress

This entry is only required for the connection to our ETH modules.

The IP address of the ETH modules can be set via the ICT-Tool as well as via the web interface of the module.

4.1.4.2. DELIB CLI Examples

Digital outputs

```
delib_cli DO1 17 1
```
→ switches on the 18th digital relay

```
delib_cli DO1 3 0
```
→ switches off the 4th digital relay

```
delib_cli DO8 0 255
```
→ switches on the digital relays 1 to 8

```
delib_cli DO16 0 0
```
→ switches off the digital relays 1 to 16

```
delib_cli DO16 16 65535
```
→ switches on the digital relays 17 to 32

```
delib_cli DO32 0 4294967295
```
→ switches on the digital relays 1 to 32

Digital inputs

```
delib_cli DI1 3
```

Example of a return value: 1

→ read the state of the 4th digital input and return it

```
delib_cli DI8 0 hex
```

Example of a return value: **0xC8**

(a signal is present on channels 4, 7 and 8)

→ read the value of digital input 1-8 as hexadecimal number

```
delib_cli DI16 0 hex
```

Example of a return value: **0xE0C0**

(a signal is present on channels 7,8, 14 ,15 and 16)

→ read the value of digital input 1-16 as hexadecimal number

Alternatively, the "nounit" argument can be appended to all output requests to be formatted as follows:

```
delib_cli DI8 0 hex nounit
```

Example of a return value: **FF**

(a signal is present on channels 1-8)

→ read the value of digital input 1-8 as hexadecimal number

```
delib_cli FF  0
```

Example of a return value: **192**

(a change of state has been detected on channels 7 and 8).

→ read the value of the FlipFlops of the digital inputs 1-32

```
delib_cli FF  32
```

Example of a return value: **65535**

(a change of state has been detected on channels 33 to 64).

→ read the value of the FlipFlops of the digital inputs 33-64

```
delib_cli FF  0 hex
```

Example of a return value: **0xD00**

(a change of state was detected on channels 9, 11 and 12)

→ read the value of the FlipFlops of the digital inputs 1-32 as hexadecimal number

Analog outputs

```
delib_cli AO 7 4711
```

→ sets the decimal value 4711 to the 8th analog output

```
delib_cli AO 6 0x4711
```

→ sets the hexadecimal value 0x4AF1 to the 7th analog output

```
delib_cli AO 7 3.7V
```

→ sets the voltage of the 8th analog output to 3.7 volts
(both comma "," and dot "." can be used for comma separation)

```
delib_cli AO 7 13.3mA
```

→ sets the current of the 8th analog output to 13.3 milliamperes
(both comma "," and period "." can be used for comma separation)

Analog inputs

```
delib_cli AI 2
```

Example of a return value: **1234**

→ reads the value of the 3rd analog input as decimal number

```
delib_cli AI 2 hex
```

Example of a return value: **0x1FA**

→ reads the value of the 3rd analog input as hexadecimal number

```
delib_cli AI 2 V
```

Example of a return value: **12.500000V**

→ reads the voltage of the 3rd analog input as a comma number

```
delib_cli AI 2 mA
```

Example of a return value: **20.551600mA**

→ reads the current of the 3rd analog input as a comma number

Alternatively, the argument "nounit" can also be appended to all output requests to be formatted as follows:

```
delib_cli AI 3 hex nounit
```

Example of a return value: **1FA**

→ reads the value of the 4th analog input as hexadecimal number

```
delib_cli AI 3 V nounit
```

Example of a return value: **12.500000**

→ reads the voltage of the 4th analog input as a comma number

4.1.5. Control via graphical applications

4.1.5.1. LabVIEW

Our DELIB API can be imported and used in LabVIEW. All products that use our DELIB API are therefore compatible with LabVIEW.

The following chapter shows how to include the DELIB API in LabVIEW: Including the DELIB in LabVIEW

4.1.5.2. ProfiLab

The ProfiLab software of the company Abacom supports a large number of our control & regulation products.

Link to the manufacturer: http://www.abacom-online.de/html/profilab-expert.html

The following I/Os are supported:

Digital inputs/outputs

- Relais
- MOSFET
- Optokoppler
- Bistabile-Relais

Analog inputs/outputs

- Analog to digital converter
- Digital to analog converter

TTL-I/Os

- 8/32/64 TTL channels

4.1.5.3. Licht24 Pro

The Licht24 Pro software from the company bksoft also supports a high number of our products.

You can find more information at: http://www.bksoft.de/licht24pro.htm

4.1.6. Integration of the DELIB in programming languages

4.1.6.1. Embedding the DELIB in Visual-C/C++

To make it easier to reference the DELIB include and the DELIB lib directory, environment variables are defined when the DELIB is installed.

DELIB_LIB = C:\Programs\DEDITEC\DELIB\lib

DELIB_INCLUDE = C:\Programs\DEDITEC\DELIB\include

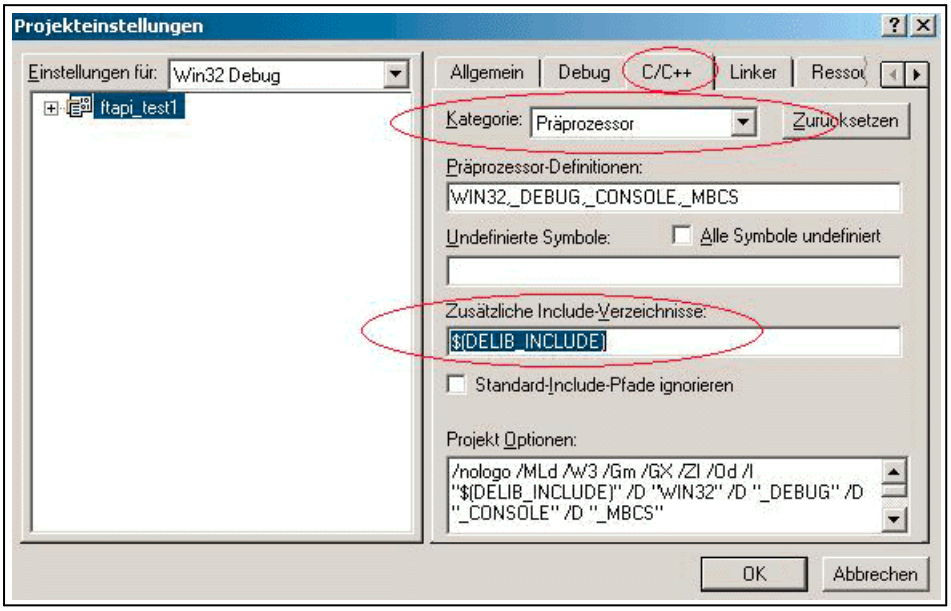
These are entered below in the project settings of the compiler.

Start Visual-C/C++ and open the menu "Project → Settings".



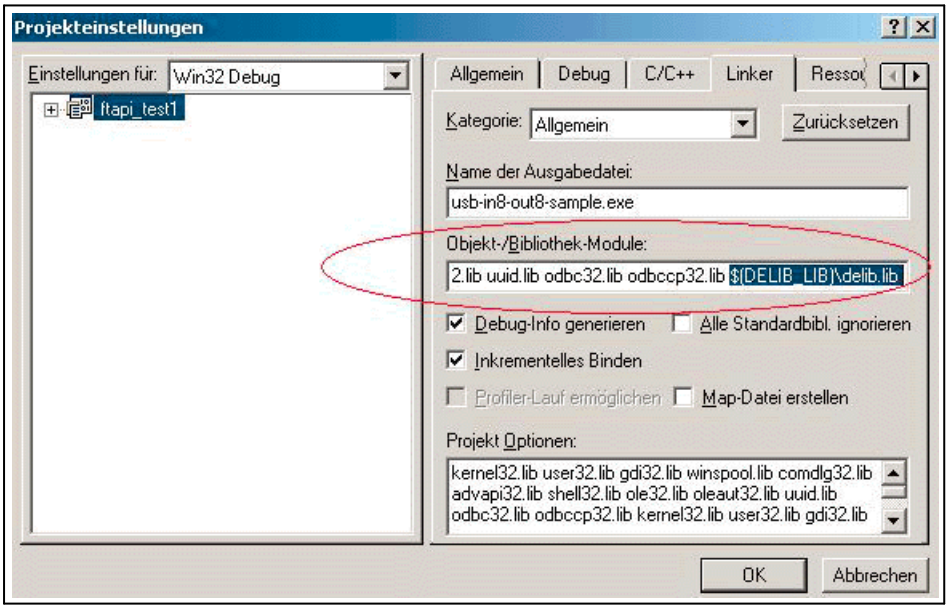
DELIB.H entry in the Visual-C/C++ project settings

Under the tab "C/C++" select the "Category" Preprocessor and enter "$(DELIB_INCLUDE)" under "Additional Include Directories.



DELIB.LIB entry in the Visual-C/C++ project settings

Under the "Linker" tab at "Object/Library Modules" extend the existing line with the extension "$(DELIB_LIB)\delib.lib".



4.1.6.2. Embedding the DELIB in Visual-C/C++ (Visual Studio 2015)

To make it easier to reference the DELIB include and DELIB lib directory, environment variables are defined when the DELIB is installed.

32 Bit DELIB Installation

DELIB_LIB = C:\Programs\DEDITEC\DELIB\lib

DELIB_INCLUDE = C:\Programs\DEDITEC\DELIB\include

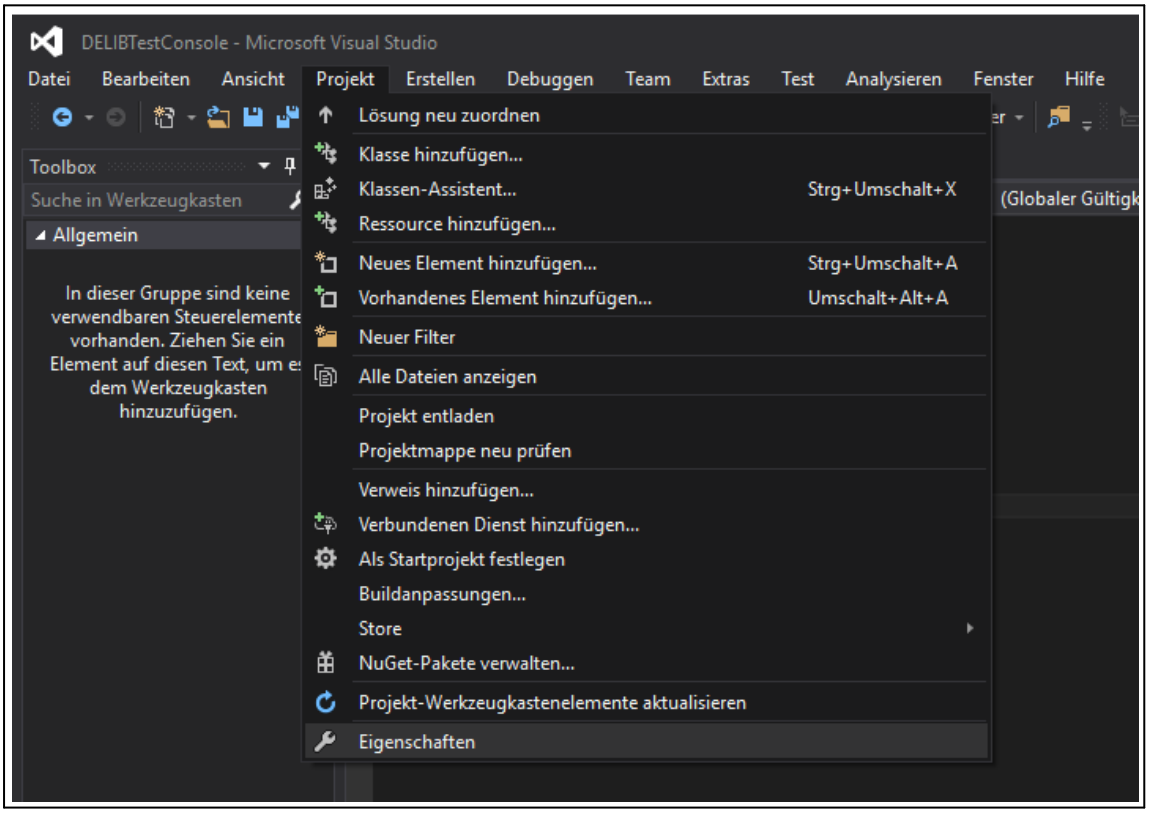
64 Bit DELIB Installation

DELIB64_LIB = C:\Programs\DEDITEC\DELIB64\lib

DELIB64_INCLUDE = C:\Programs\DEDITEC\DELIB64\include

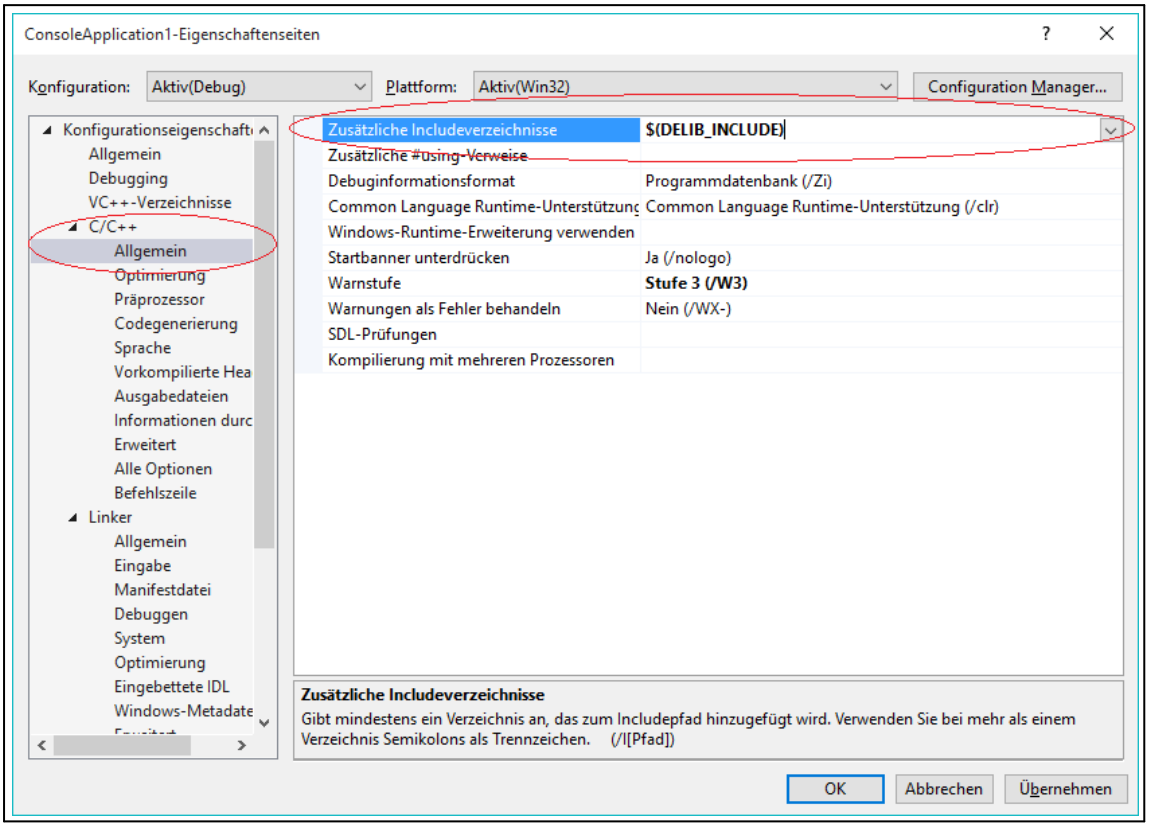
These are entered in the project settings of the compiler in the following.

Visual-C/C++ Start and open in the menu "Project → Properties.



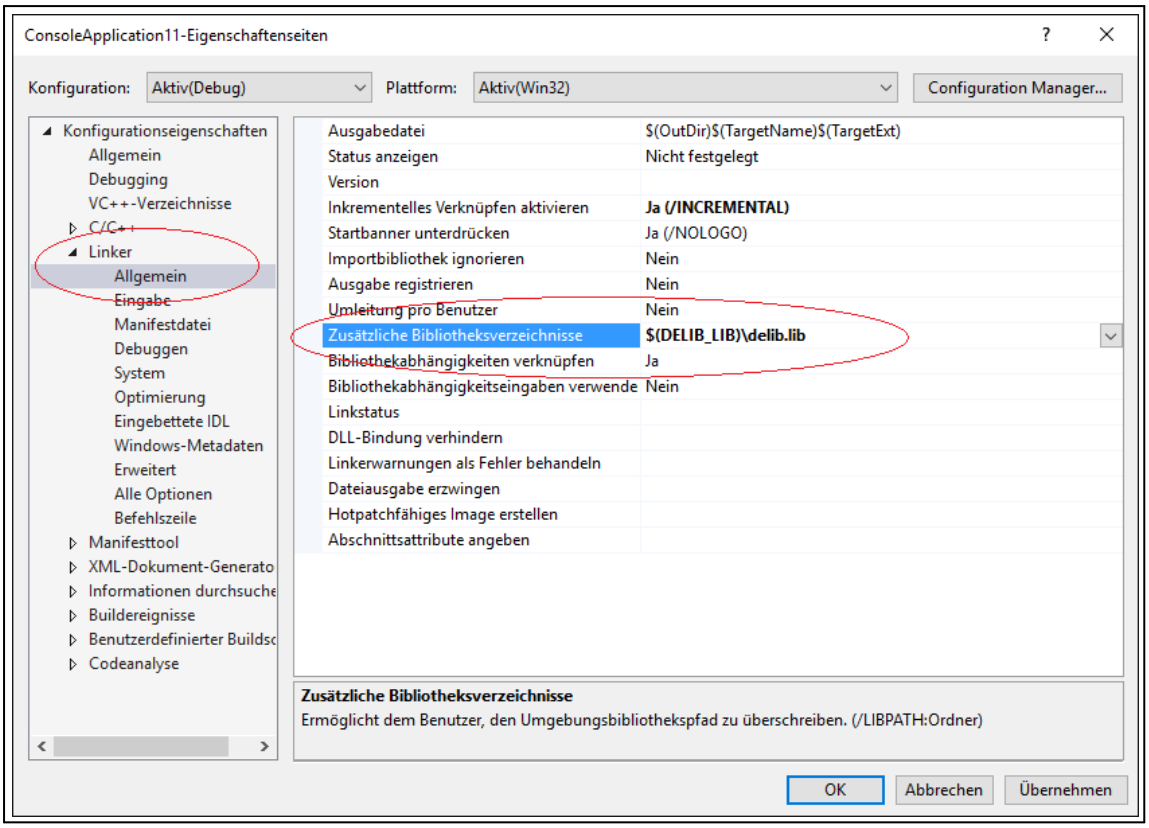
DELIB.H entry in the Visual-C/C++ project settings

Under the tab "C/C++" select the "Category" General and enter "$(DELIB_INCLUDE)" under "Additional Include Directories".



DELIB.LIB entry in the Visual-C/C++ project settings

Under the "Linker" tab, enter "$(DELIB_LIB)\delib.lib" for "General".

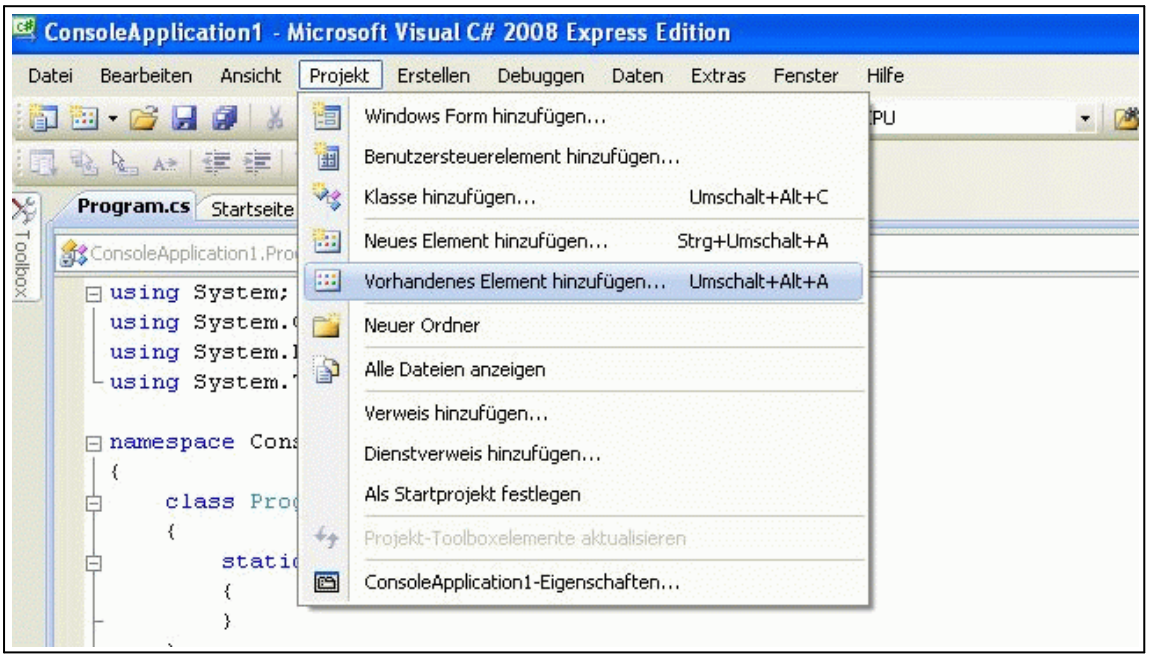

ConsoleApplication11-Eigenschaftenseiten
Konfiguration: Aktiv(Debug)
Plattform: Aktiv(Win32)
Configuration Manager...
Konfigurationseigenschaften
Allgemein
Debugging
VC++-Verzeichnisse
C/C++
Linker
Allgemein
Eingabe
Manifestdatei
Debuggen
System
Optimierung
Eingebettete IDL
Windows-Metadaten
Erweitert
Alle Optionen
Befehlszeile
Manifesttool
XML-Dokument-Generato
Informationen durchsuche
Buildereignisse
Benutzerdefinierter Buildsc
Codeanalyse
Ausgabedatei $(OutDir)$(TargetName)$(TargetExt)
Status anzeigen Nicht festgelegt
Version
Inkrementelles Verknüpfen aktivieren Ja (/INCREMENTAL)
Startbanner unterdrücken Ja (/NOLOGO)
Importbibliothek ignorieren Nein
Ausgabe registrieren Nein
Umleitung pro Benutzer Nein
Zusätzliche Bibliotheksverzeichnisse $(DELIB_LIB)\delib.lib
Bibliothekabhängigkeiten verknüpfen Ja
Bibliothekabhängigkeitseingaben verwende Nein
Linkstatus
DLL-Bindung verhindern
Linkerwarnungen als Fehler behandeln
Dateiausgabe erzwingen
Hotpatchfähiges Image erstellen
Abschnittsattribute angeben
Zusätzliche Bibliotheksverzeichnisse
Ermöglicht dem Benutzer, den Umgebungsbibliothekspfad zu überschreiben. (/LIBPATH:Ordner)
OK
Abbrechen
Übernehmen

4.1.6.3. Embedding the DELIB in Visual-C#

The required file for Visual-C# is located in the directory

C:\Programs\DEDITEC\DELIB\include.

Start Visual-C# and use the menu "Project → Add existing element" in the directory C:\Programme\DEDITEC\DELIB\include\ to open the file delib.cs for import.



Add the following reference in your program:

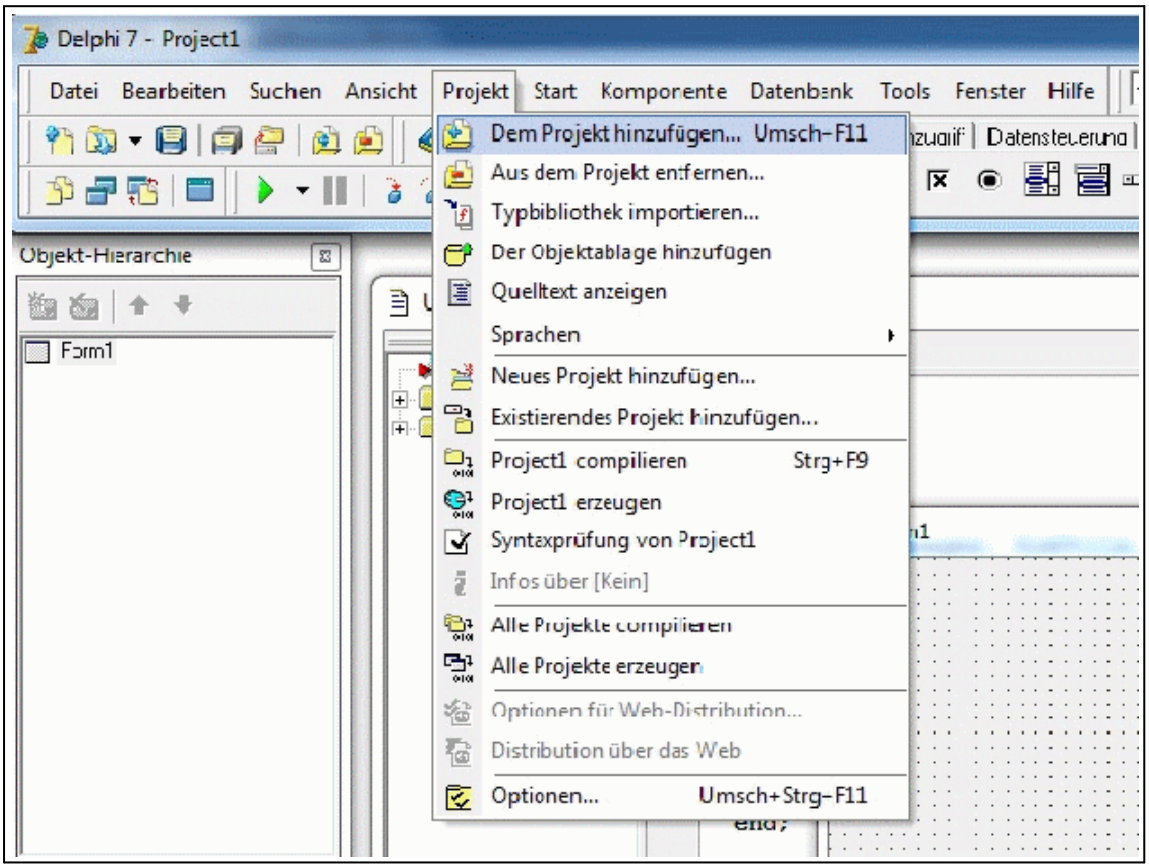
using DeLib;

4.1.6.4. Embedding the DELIB in Delphi

The required file for Delphi is located in the directory

C:\Programme\DEDITEC\DELIB\include.

Start Delphi and use the menu "Project → Add to project" in the directory C:\Programs\DEDITEC\DELIB\include\ to open the file delib.pas for import.

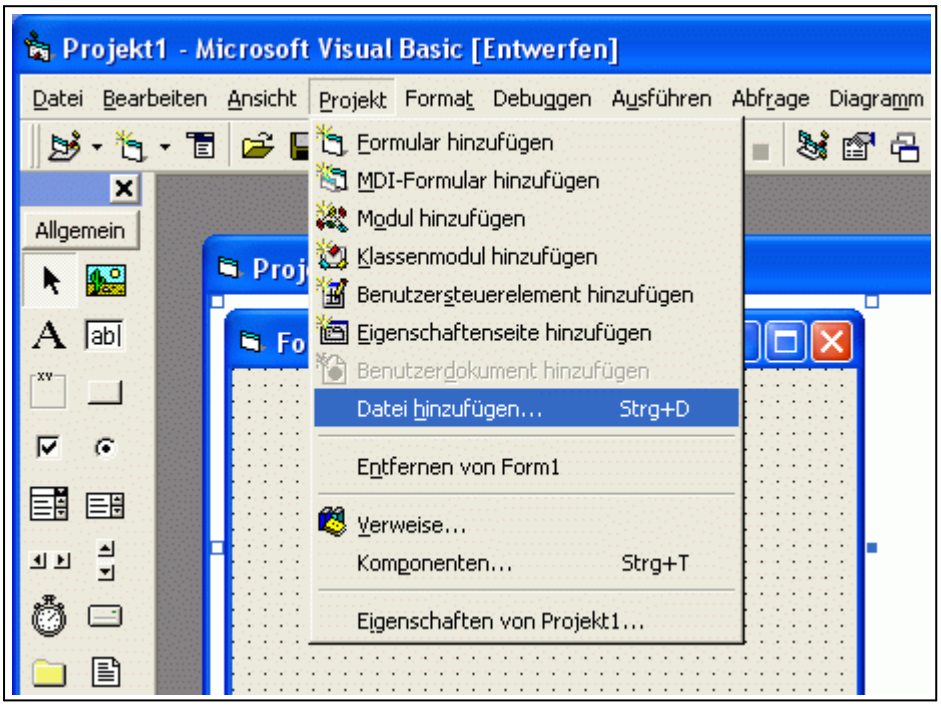


4.1.6.5. Embedding the DELIB in Visual-Basic (VB)

The required file for Visual-Basic is located in the directory

C:\Programs\DEDITEC\DELIB\include.

Start Visual Basic and use the menu "Project → Add file...". in the directory C: \Programme\DEDITEC\DELIB\include\ open the file delib.bas for import.

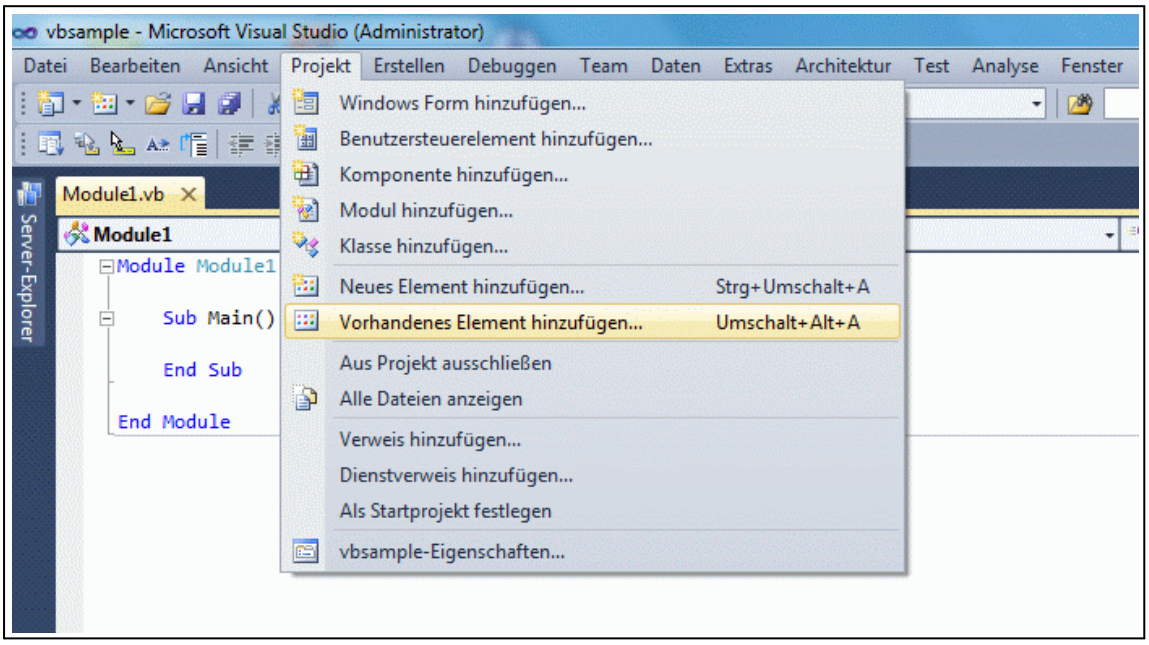


4.1.6.6. Embedding the DELIB in Visual-Basic.NET (VB.NET)

The required file for VB.NET is located in the directory

C:\Programme\DEDITEC\DELIB\include.

Start VB.NET and use the menu "Project → Add existing element" in the directory C:\Programme\DEDITEC\DELIB\include\ to open the file delib.vb for import.

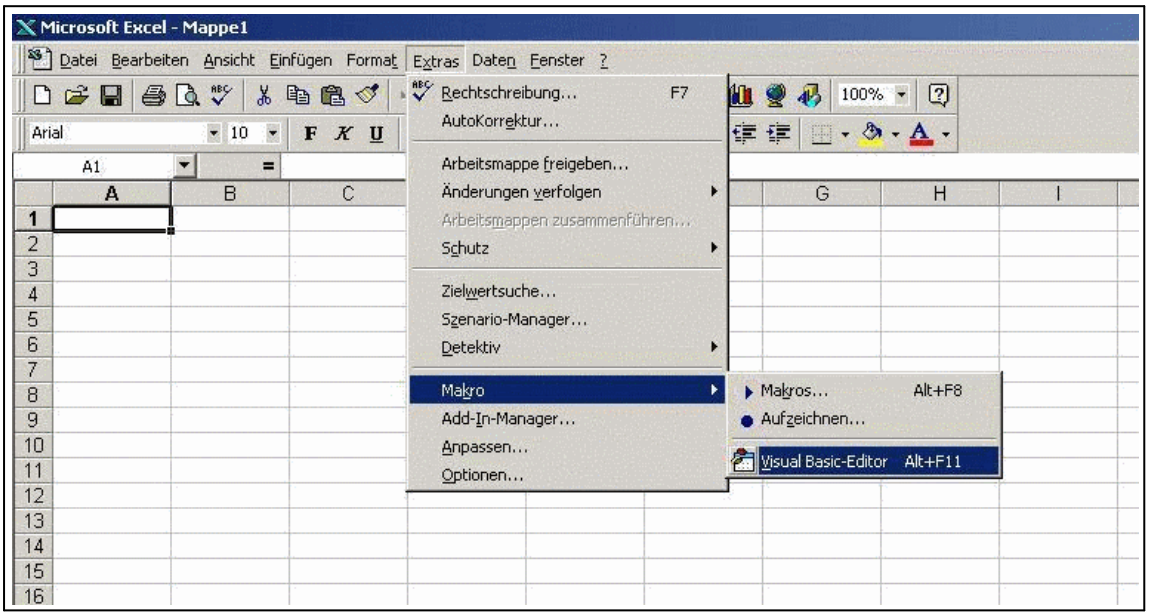


4.1.6.7. Embedding the DELIB in MS-Office (VBA)

The required file for VBA is located in the directory

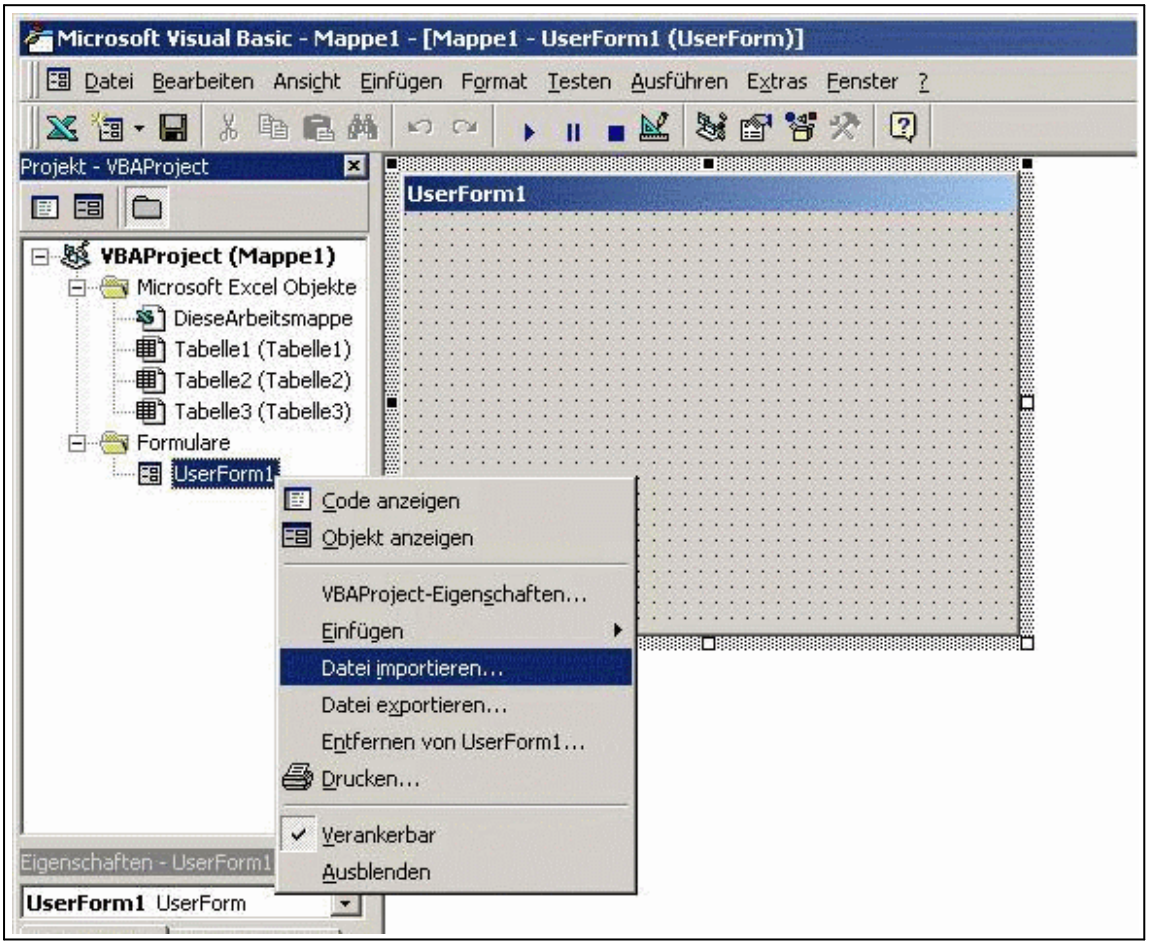
C:\Programs\DEDITEC\DELIB\include.

Start Microsoft Excel and open it via the menu "Tools → Macro → Visual Basic Editor".



Creating the UserForm

Create a new worksheet (UserForm) via the menu "Insert → UserForm". At the top left of the Project Manager, right-click on "UserForm → Import file". In the directory C:\Programme\DEDITEC\DELIB\include open the file delib.bas for import.



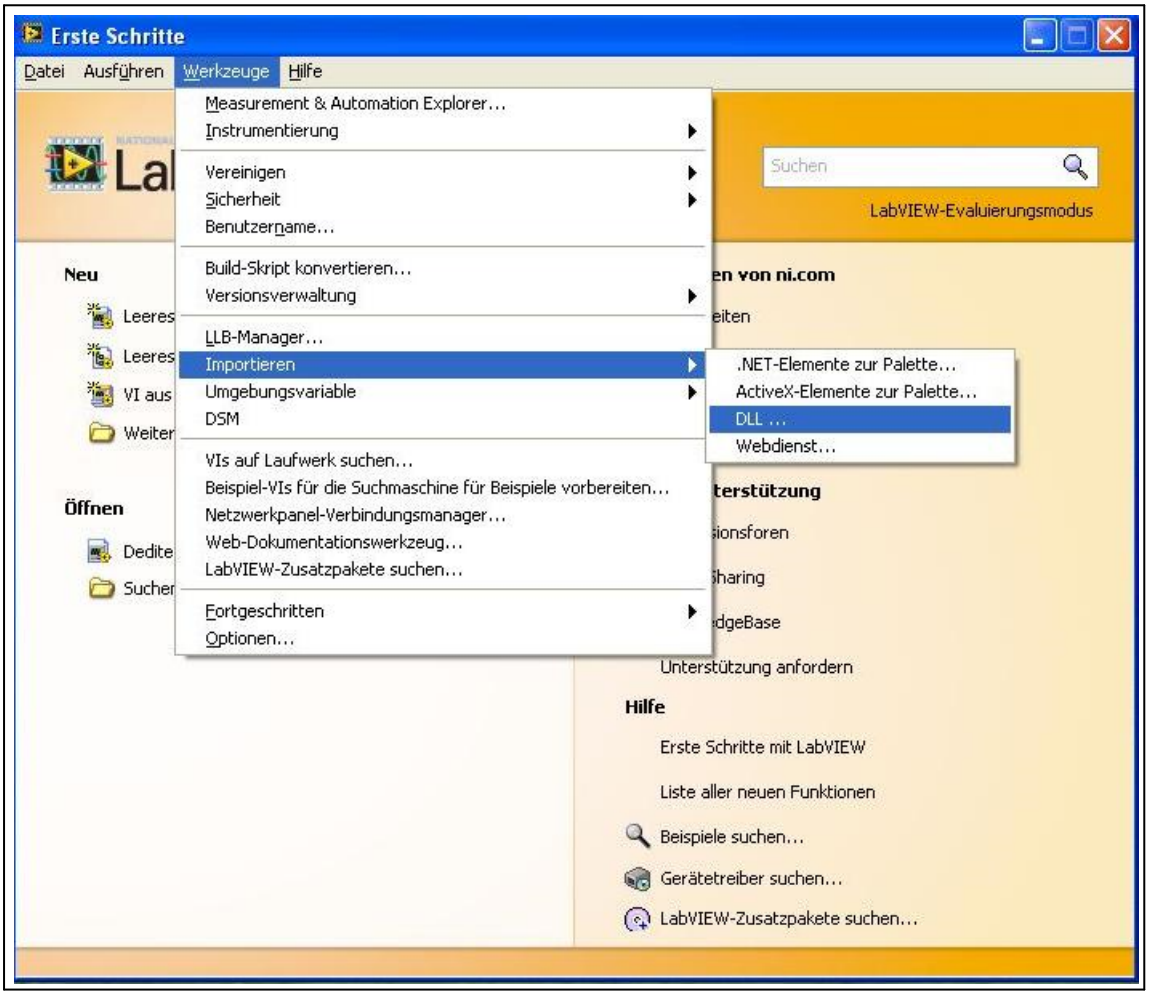
4.1.6.8. Embedding the DELIB in LabVIEW

4.1.6.8.1. Embedding the DELIB in LabVIEW

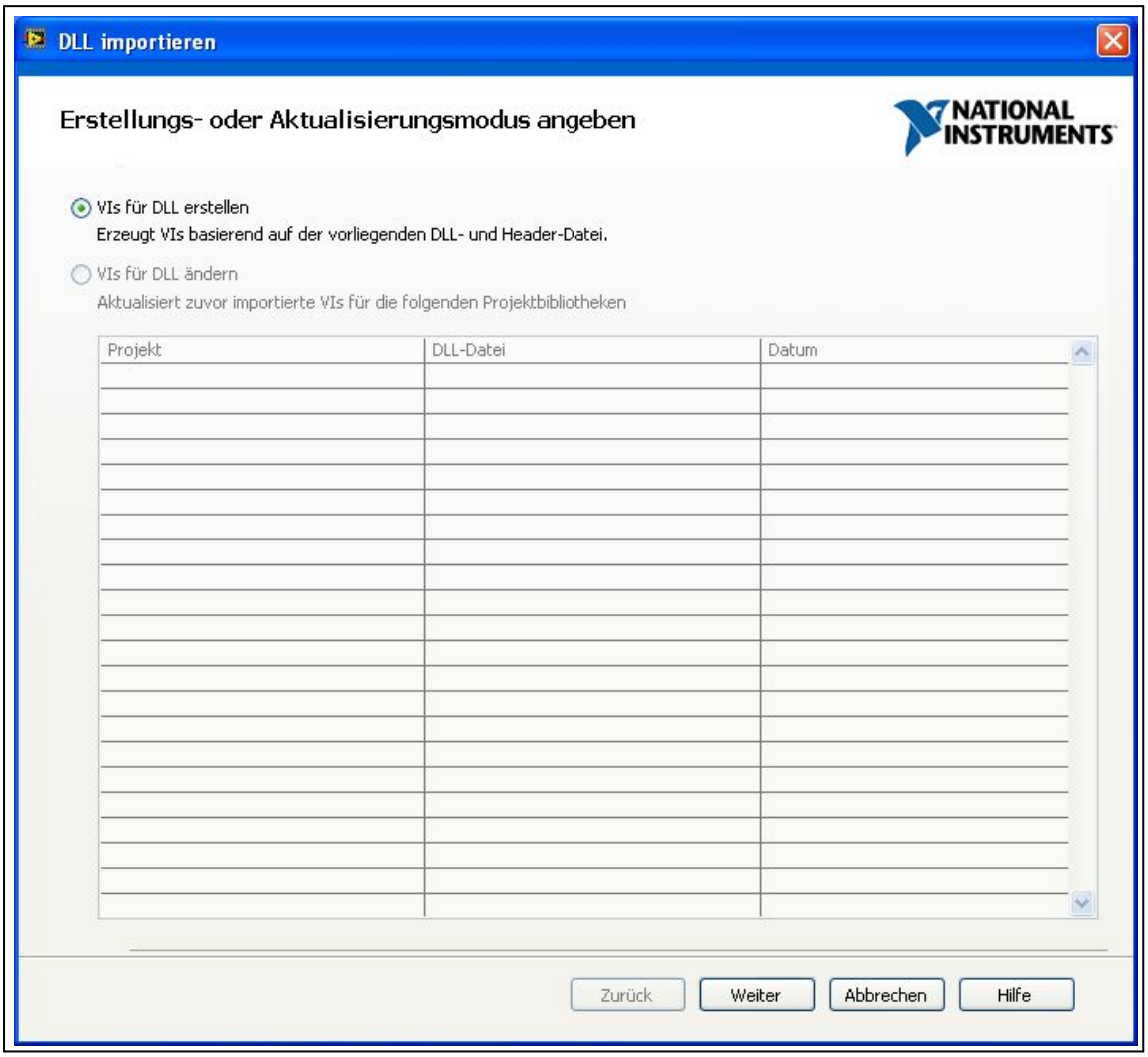
The LabVIEW sample program "Deditec_Modul_Control.vi" is not an EXE file and therefore requires the LabVIEW development environment for execution.

Description of the integration of the "delib.dll" in LabVIEW version 11

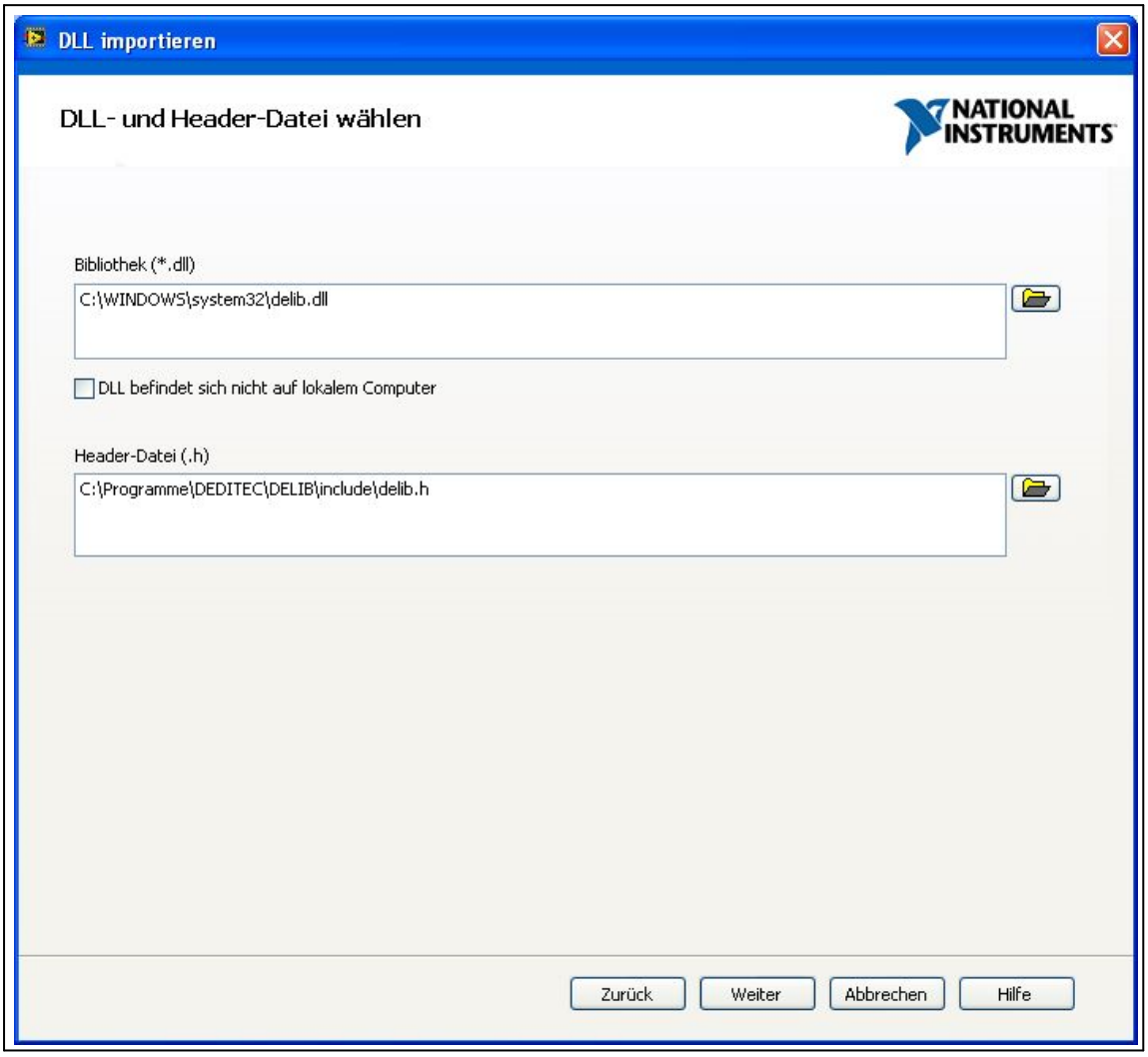
- The required files for LabVIEW are located in the directory C:\Windows\System32\delib.dll and in C:\Programs\DEDITEC\DELIB\include\delib.h
- Start LabVIEW and select the following option "Tools → Import → DLL ...".



- Select the item "Create VIs for DLL" and press "Next".

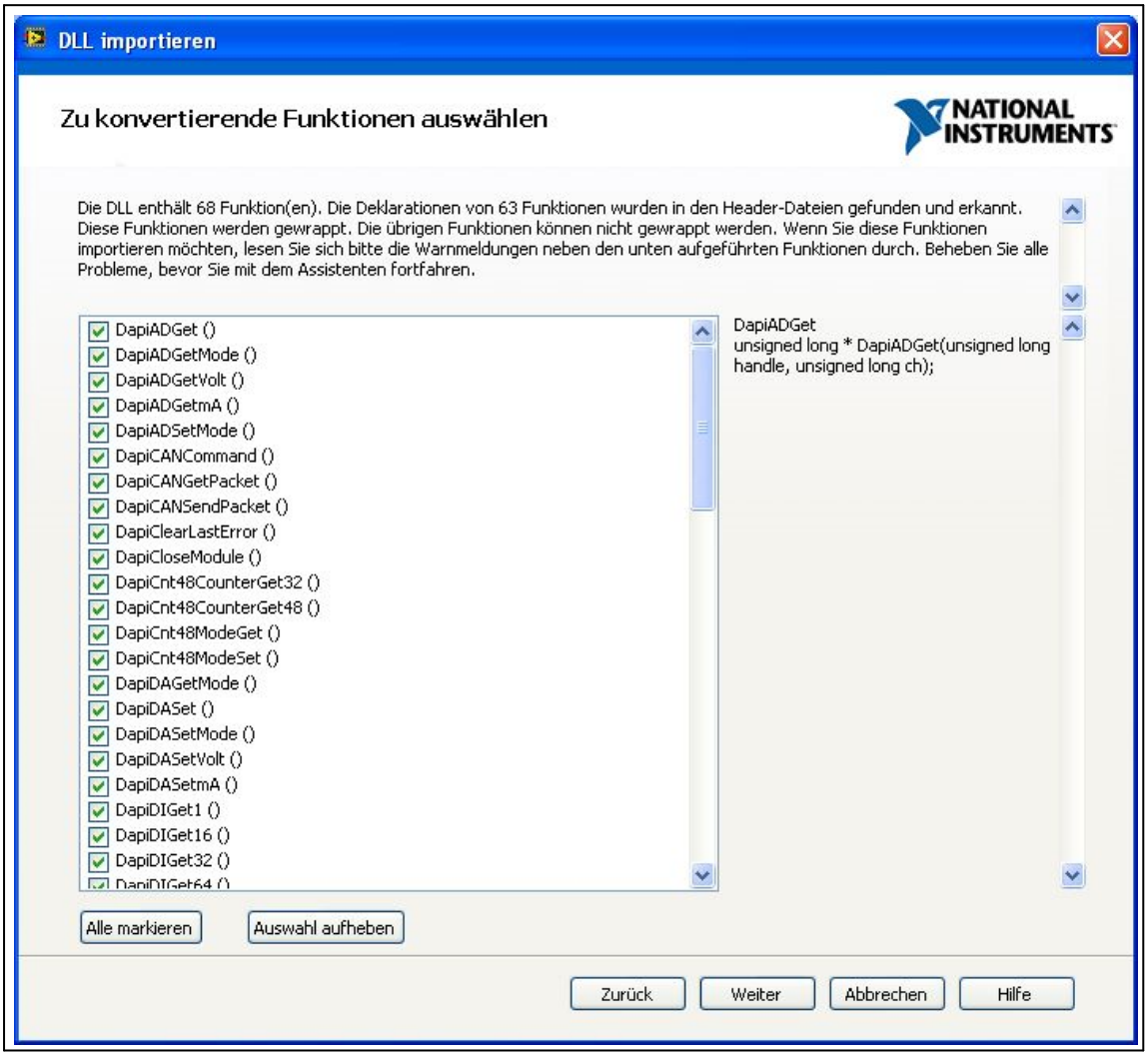


- In the next window, use the browser buttons to specify the location of the delib.dll and delib.h files and continue with "Next".

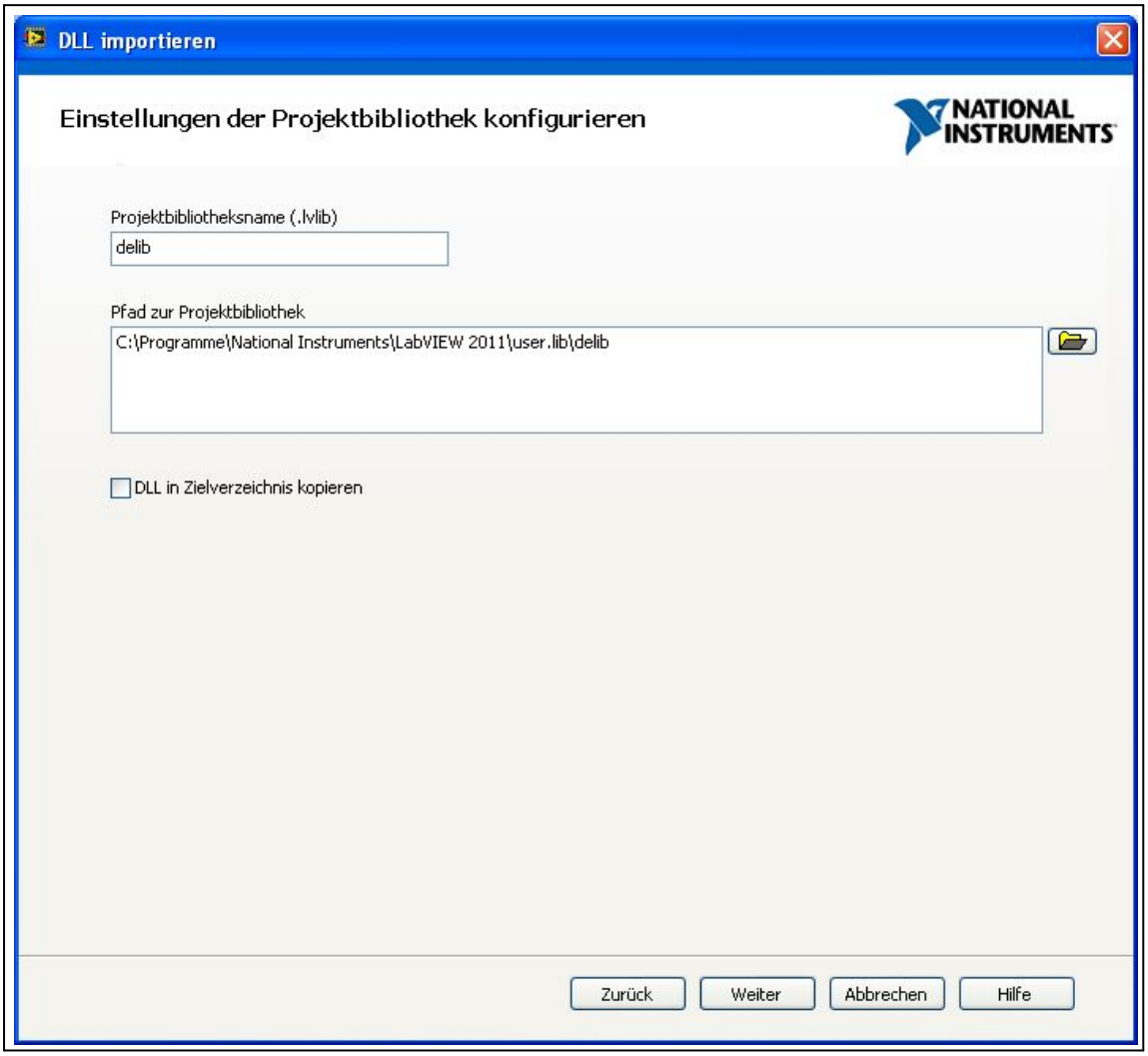


- Click on "Next" again to continue.

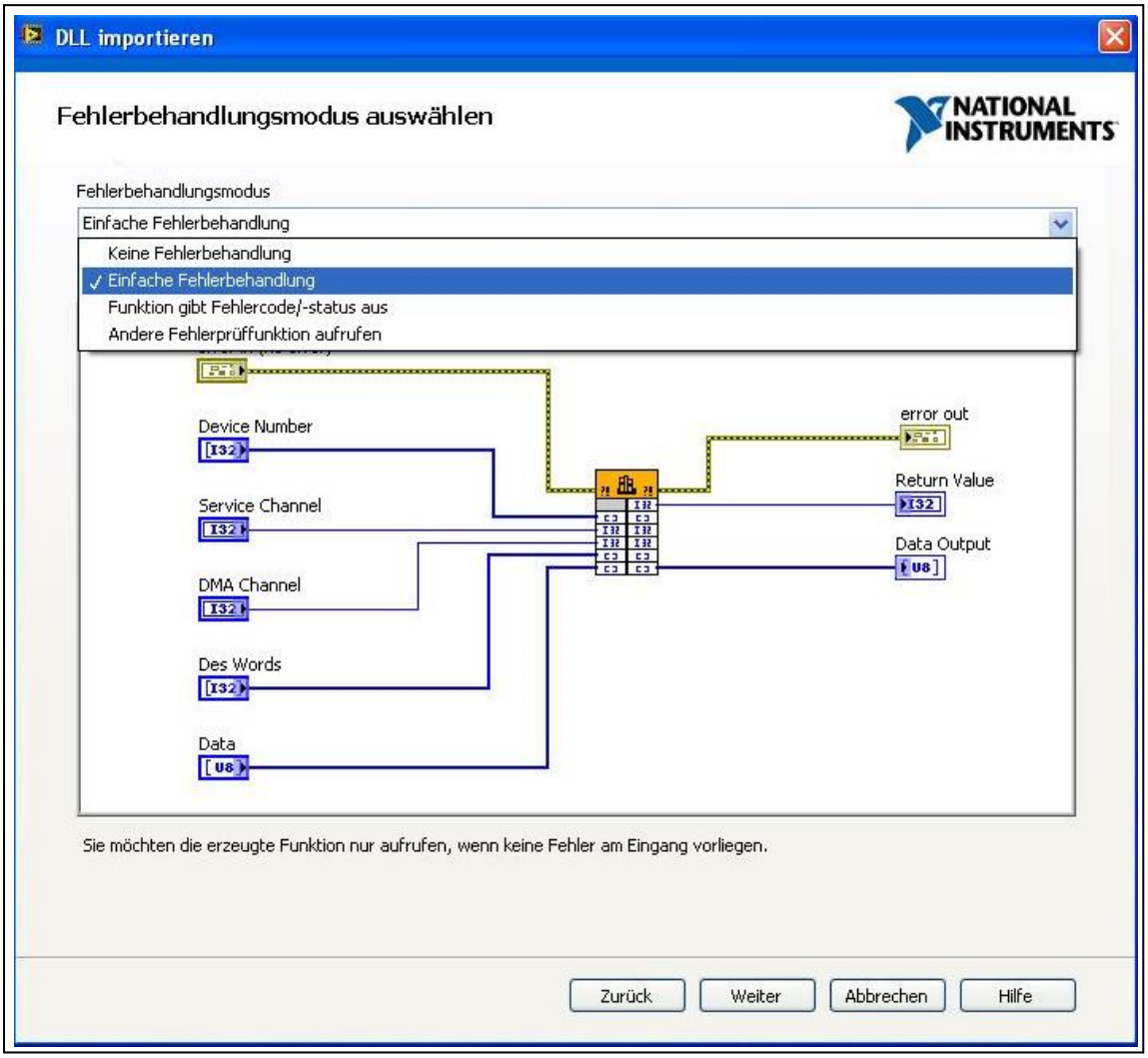
- The header file will now be analyzed. Then click "Next" again in the following window to continue.



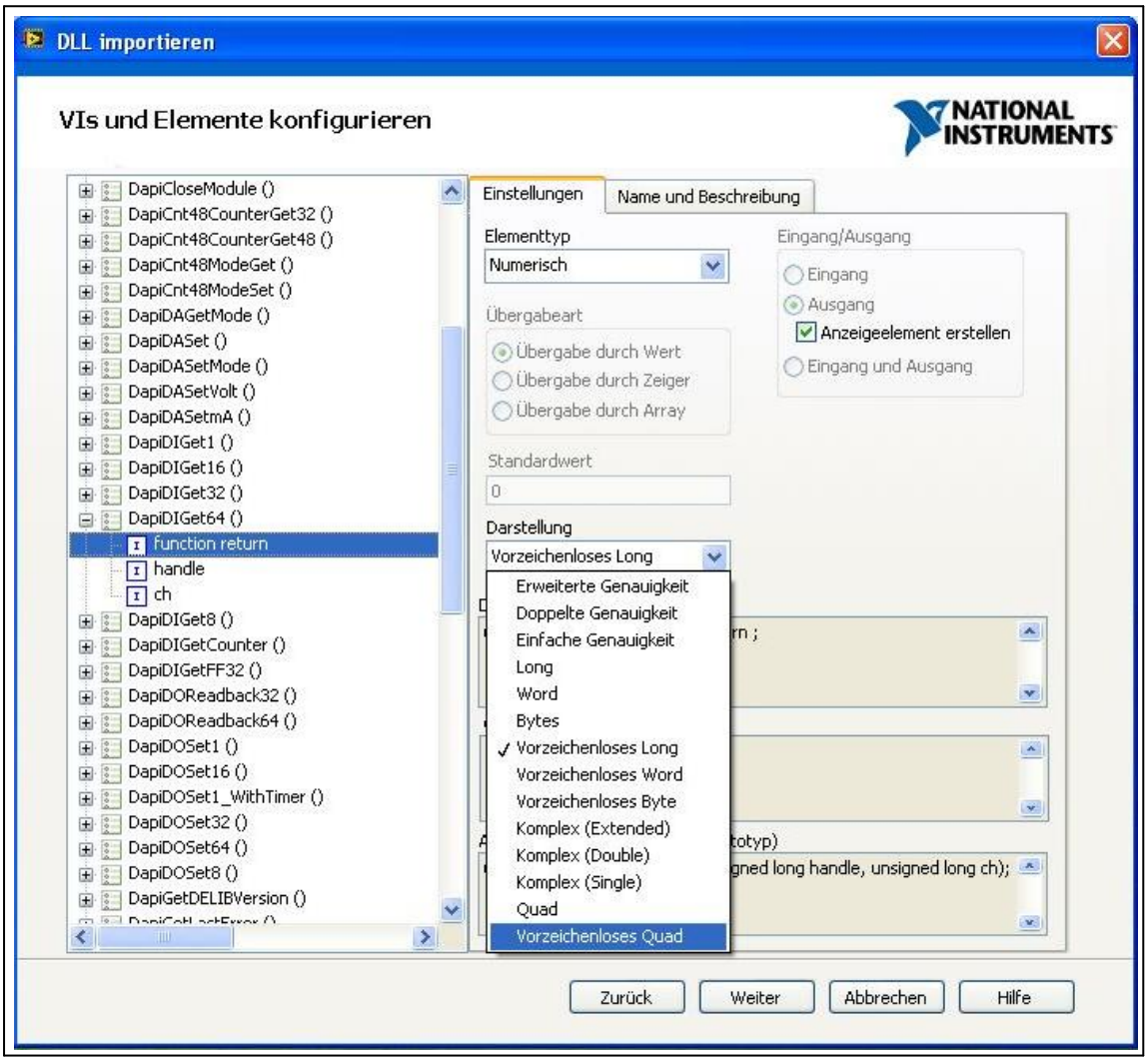
- Follow the further instructions or adjust the configuration and the location for the VIs.



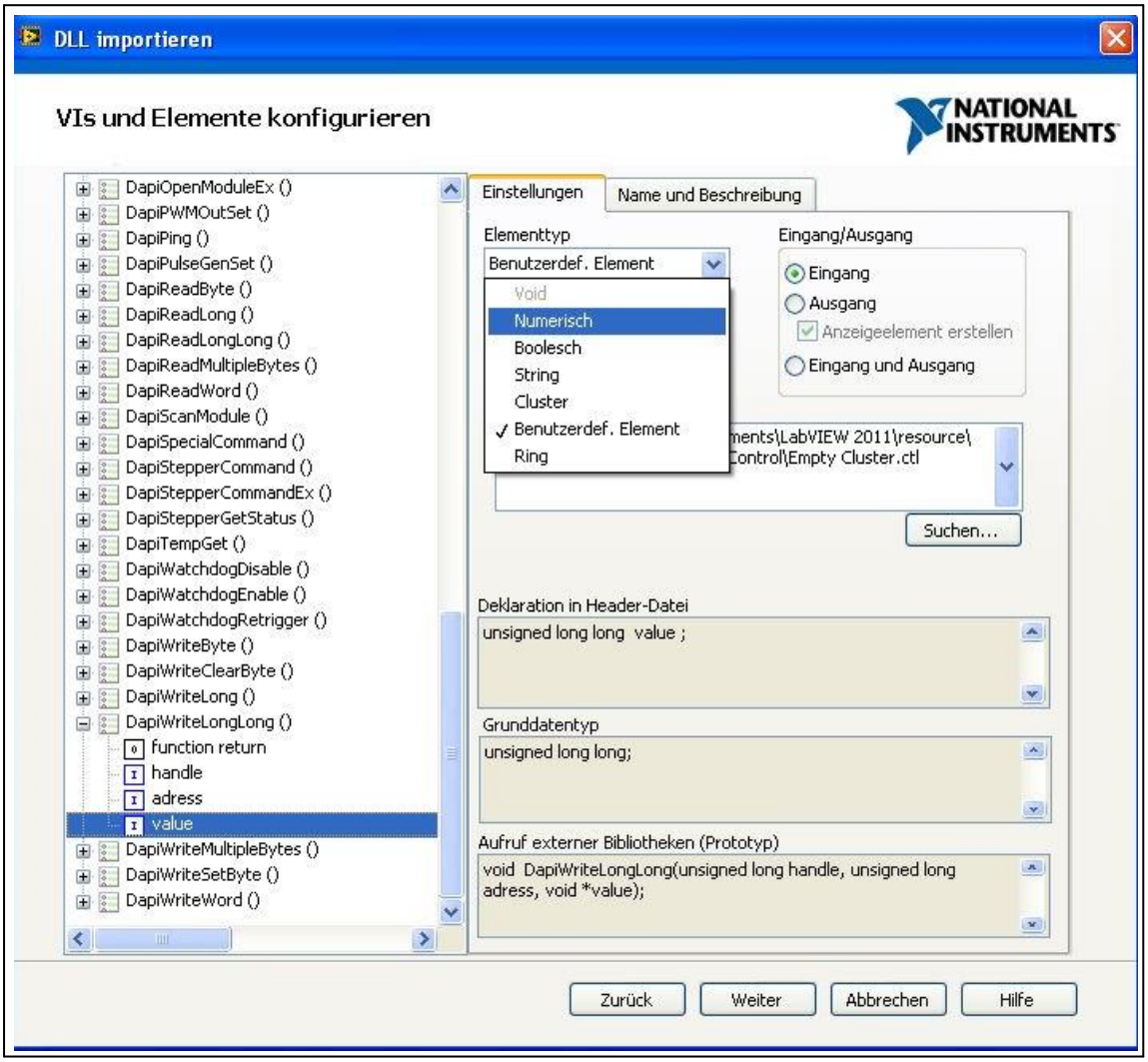
- In the following window, select the "Simple error handling" option from the drop-down menu and continue with "Next".



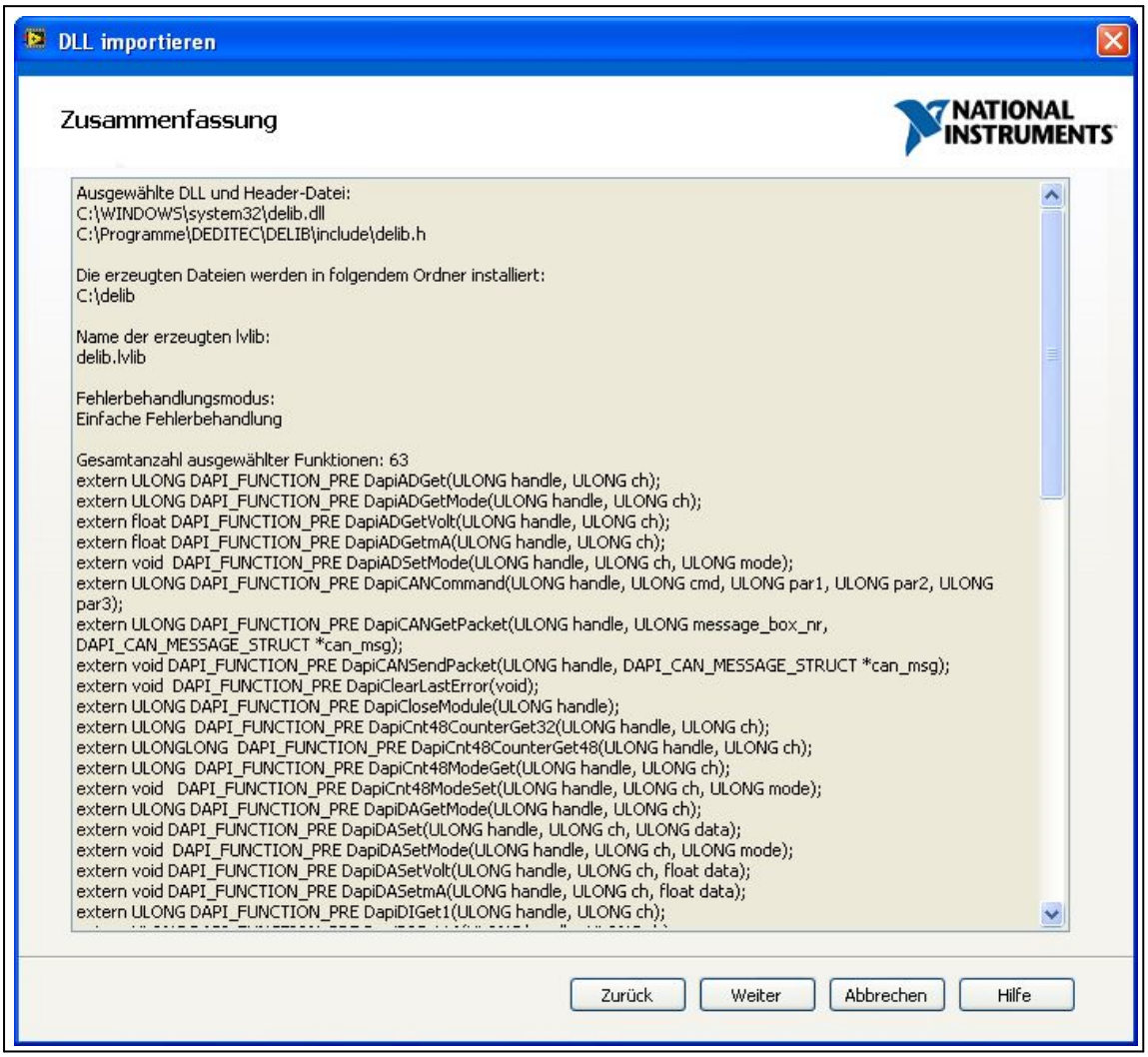
- For VIs that work with 64-bit values, the representation must be changed from "Unsigned Long" to "Unsigned Quad".
- The following VIs must be edited:
    - → DapiCNT48CounterGet48 (function return)
    - → DapiDIGet64 (function return)
    - → DapiDOSet64 (data)
    - → DapiDOReadBack64 (function return)



- For some VIs, the element type must also be changed to "Numeric" and then the representation must be changed to "Unsigned Quad".
- The following VIs must be edited:
    → DapiWriteLongLong (value)
    → DapiReadLongLong (function return)



- A summary of the steps performed appears.
- Press "Next" to continue.



- The VIs are now created and can be used.

4.1.6.8.2. Using the VIs in LabVIEW

In our sample programs, some functions use so-called defines as transfer parameters.

These defines are not supported in LabVIEW.

This example is intended to show how such functions can be used in LabVIEW.

As an example we use the function to configure the voltage range of an A/D converter.

The definition for the function is:

void DapiADSetMode(ULONG handle, ULONG ch, ULONG mode);

The voltage ranges for the function are already predefined in the DELIB driver library.

```
// ------------------------------------------------------------------------
// A/D and D/A Modes

#define ADDA_MODE_UNIPOL_10V                    0x00
#define ADDA_MODE_UNIPOL_5V                     0x01
#define ADDA_MODE_UNIPOL_2V5                    0x02
```

Sample code in C/C++:

DapiADSetMode(handle, 0, ADDA_MODE_UNIPOL_5V);

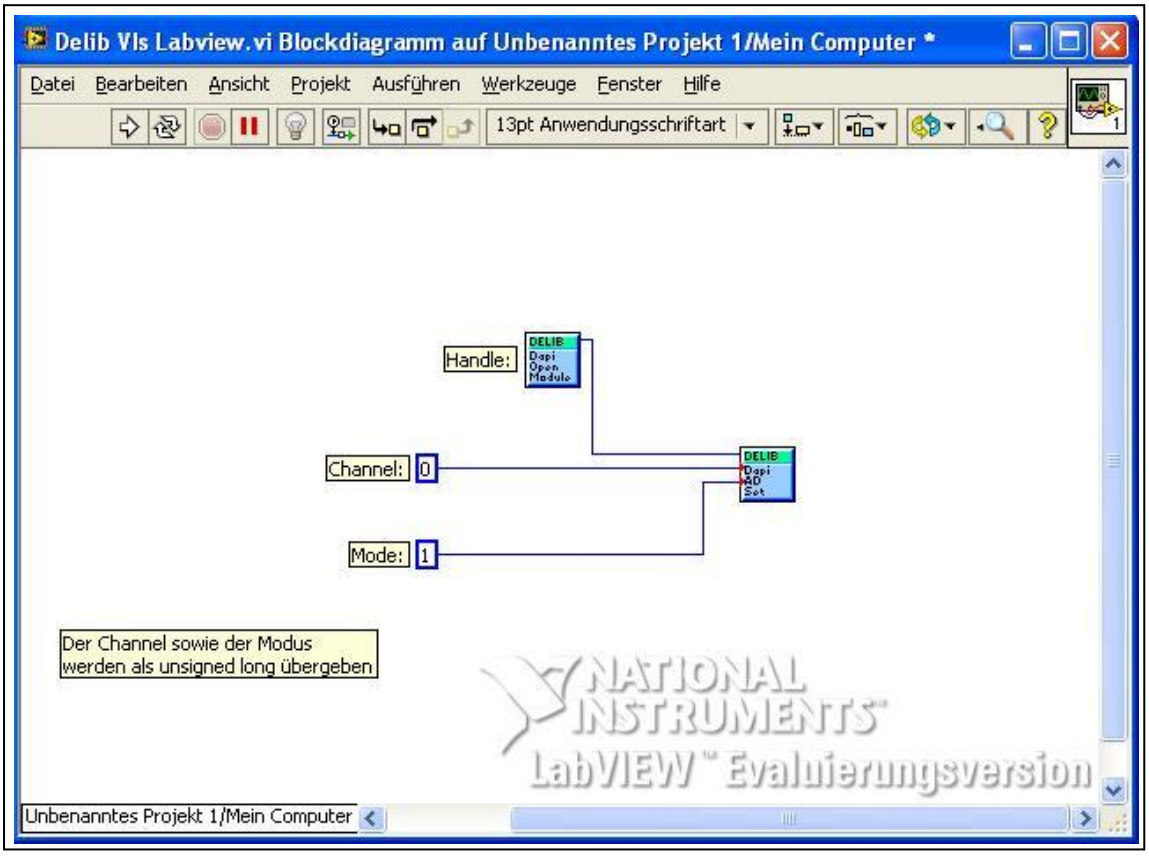
Alternatively, you can use the following notation:

DapiADSetMode(handle, 0, 0x01);

Here the hexadecimal value, which you can take from the delib.h file, was passed as parameter for the mode

The delib.h file can be found after the installation of the DELIB driver library in the directory C:\Programs\Deditec\DELIB\Include

In LabVIEW, the function could then look like this:



4.1.6.8.3. Setting the module ID in LabVIEW

In the following example the addressing of a RO-ETH module in LabVIEW is shown.

The connection to the module is established by means of the function DapiOpenModule.

The definition for this function is:

ULONG DapiOpenModule(ULONG moduleID, ULONG nr);

The module ID (e.g. "RO_ETH") of the used module is usually passed as parameter for moduleID.

An overview of all possible module IDs can be taken from the file "delib.h".

The delib.h can be found after the installation of the DELIB driver library in the directory C:\Programs\Deditec\DELIB\Include

```
// ****************************************************************************
// ****************************************************************************
//
//
#define DELIB_VERSION                                   0x0141      // Actual DELIB-Version


// all Modul-ID's
#define USB_Interface8                                  1           // USB-Controller8/USB-TTL-IN8-OUT8
#define USB_CAN_STICK                                   2           // USB-CAN-Stick
#define USB_LOGI_500                                    3           // USB-LOGI-500/USB-LOGI-250
#define RO_USB2                                         4           // RO-CPU2 / 480 MBit/sec
#define RO_SER                                          5           // RO-SER-Serie
#define USB_BITP_200                                    6           // USB-BITP-200
#define RO_USB1                                         7           // RO-USB-Serie
#define RO_USB                                          7           // RO-USB-Serie
#define RO_ETH                                          8           // RO-ETH-Serie
#define USB_MINI_STICK                                  9           // USB-MINI-Stick-Serie
#define USB_LOGI_18                                     10          // USB-LOGI-100
#define RO_CAN                                          11          // RO-CAN-Serie
#define USB_SPI_MON                                     12          // USB_SPI_MON
#define USB_WATCHDOG                                    13          // USB_Watchdog
#define USB_OPTOIN_8                                    14          // USB-OPTOIN8 / USB-RELAIS-8
#define USB_RELAIS_8                                    14          // USB-OPTOIN8 / USB-RELAIS-8
#define USB_OPTOIN_8_RELAIS_8                           15          // USB-OPTOIN-8-RELAIS-8
#define USB_OPTOIN_16_RELAIS_16                         16          // USB-OPTOIN-16-RELAIS-16
#define USB_OPTOIN_32                                   16          // USB-OPTOIN-16-RELAIS-16
#define USB_RELAIS_32                                   16          // USB-OPTOIN-16-RELAIS-16
#define USB_OPTOIN_32_RELAIS_32                         17          // USB-OPTOIN-32-RELAIS-32
#define USB_OPTOIN_64                                   17          // USB-OPTOIN-32-RELAIS-32
#define USB_RELAIS_64                                   17          // USB-OPTOIN-32-RELAIS-32
#define USB_TTL_32                                      18          // USB-TTL-32
#define USB_TTL_64                                      18          // USB-TTL-64

#define MAX_NR_OF_MODULES 18
```

Example in C:

handle = DapiOpenModule(RO_ETH, 0); // opens a RO-ETH module with module

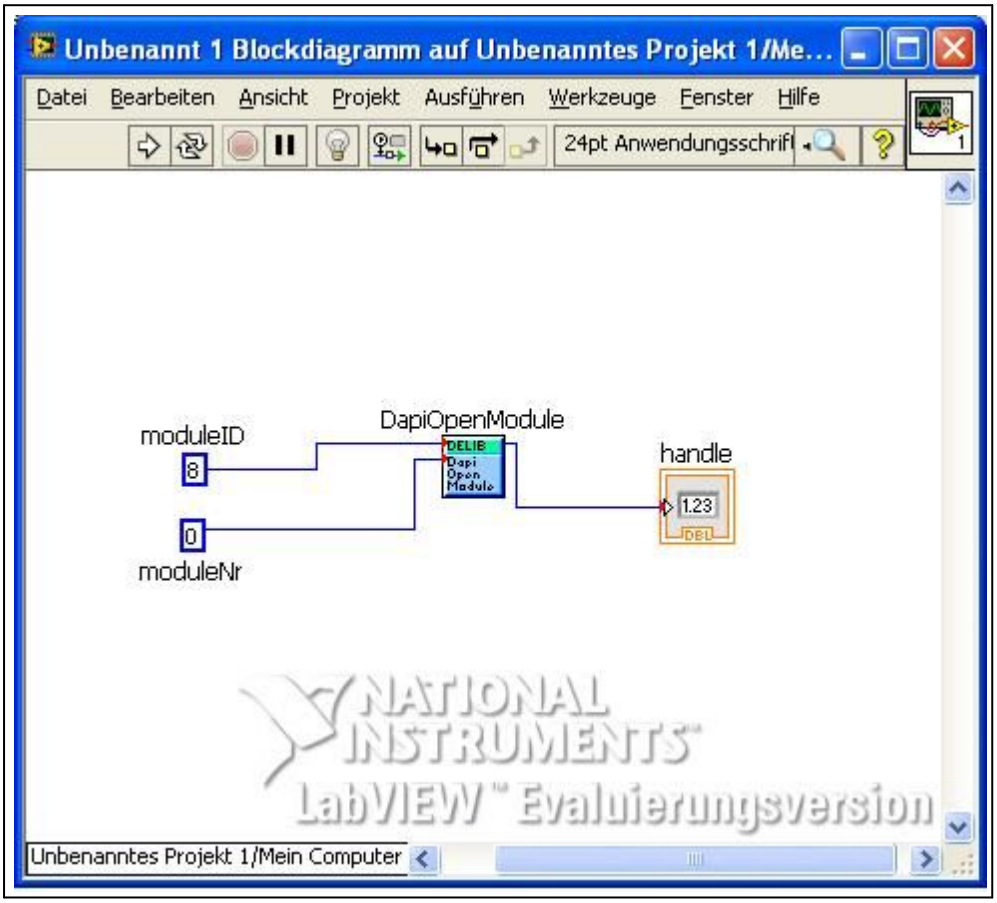
no. 0.

Alternatively, you can use the following notation:

handle = DapiOpenModule(8, 0);

Since it is not possible in LabVIEW to pass these "C-Defines" as parameters for the function DapiOpenModule, the alternative notation must be used here.

Example in Labview:



4.1.6.9. Embedding the DELIB in Java

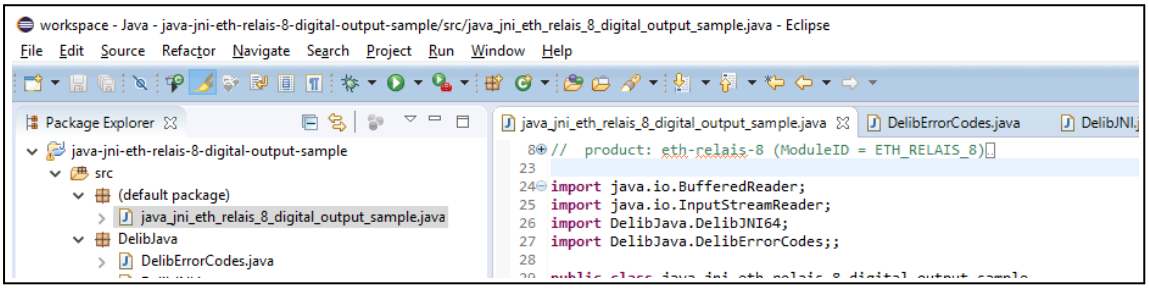
The required files for Java are located in the following directory, depending on the DELIB installation

C:\Program Files (x86)\DEDITEC\DELIB\include\DelibJava (32 bit installation)

C:\Program Files\DEDITEC\DELIB64\\include\DelibJava (64 bit installation)

If Eclipse is used, the DelibJava folder can be added to the project simply by dragging and dropping.

Afterwards the used modules still have to be imported.



4.2. DELIB driver library

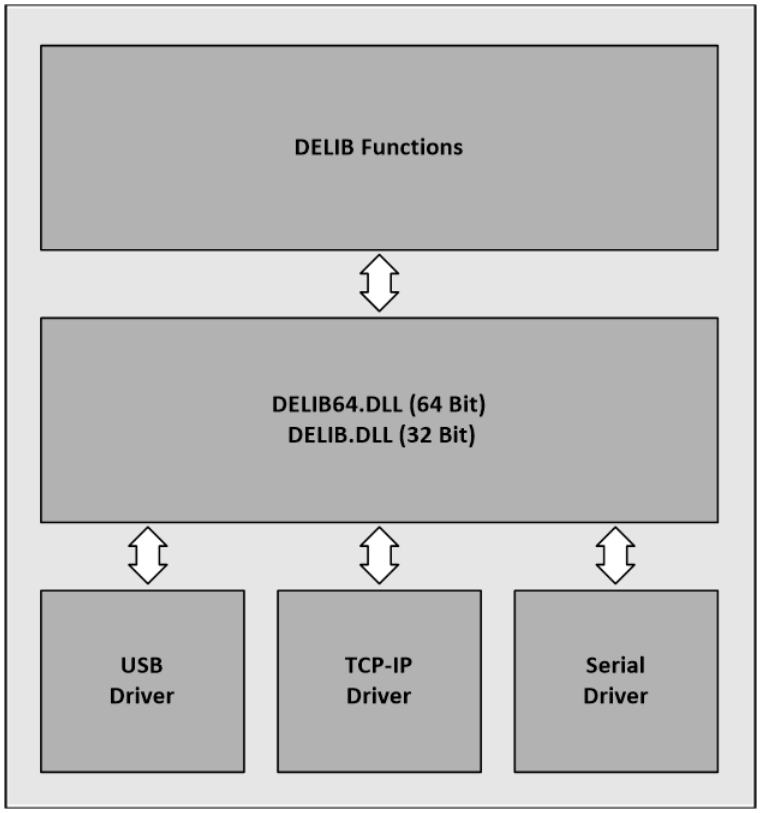
The DELIB driver library contains the DELIB API and various programs for the configuration test of our products.

Via the API you have access to all functions you need to communicate with our products.

In the chapter **DELIB API Referenz** you will find all functions of our driver library explained and provided with application examples.

4.2.1. Overview

The following figure explains the structure of the DELIB driver library



The DELIB driver library enables a uniform response of DEDITEC hardware, with special consideration of the following aspects:

- Operating system independent
- Programming languages independent
- Product independent

We offer these versions of the driver library:

- 32/64-bit DELIB driver library for Windows
- 32/64-bit DELIB driver library for Linux
- 32/64-bit DELIB Driver Library ETH

DELIB Driver library ETH

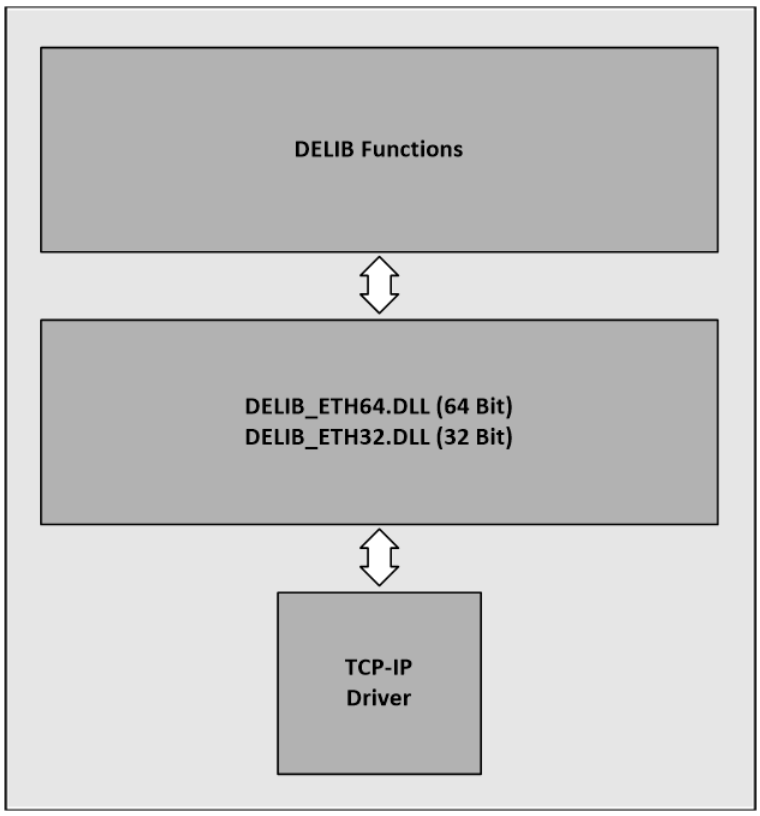
While the DELIB is available for ALL products, the DELIB-ETH does not access any other drivers (like USB).

This means that the DELIB-ETH does not need to be installed.

Customers who write their own applications do not have to create their own SETUP which also installs e.g. USB drivers.

The DELIB-ETH.dll file only has to be located in the program directory of the application and serves as interface between customer application and hardware.

The DELIB-ETH provides all DELIB commands and can easily be replaced by the old DELIB for Ethernet applications.



4.2.1.1. Supported programming languages

The following programming languages are supported by the DELIB driver library:

- C

- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office (VBA)
- Java (platform independent, only for Ethernet products)
- Java JNI (for Windows only, all products are supported)

If provided by the programming language/development environment, we support both 32-bit and 64-bit projects.

4.2.1.2. Supported operating systems

The following operating systems are compatible with our DELIB driver library:

32-Bit:

- Windows 10
- Windows 7
- Windows 8
- Windows Server 2012
- Windows Server 2008
- Windows Vista
- Windows XP
- Windows Server 2003
- Windows 2000
- Linux

64-Bit:

- Windows 10 x64
- Windows 7 x64
- Windows 8 x64
- Windows Server 2012 x64
- Windows Server 2008 x64
- Windows Vista x64
- Windows XP x64
- Windows Server 2003 x64
- Linux x64

4.2.1.3. SDK kit for programmers

Integrate the DELIB into your application. On request, we will provide you with installation scripts free of charge that enable you to include the DELIB installation in your application.

4.2.2. DELIB Setup

The DELIB Setup guides you through the installation of our DELIB driver library.

You will then be guided through the configuration process and function test for our various products.

Our current version of the DELIB setup for Linux and Windows in the 32-bit or 64-bit versions can be downloaded from our homepage.

Link: https://www.deditec.de/en/delib

The DELIB setup guides you step by step through the installation of the DELIB driver library.

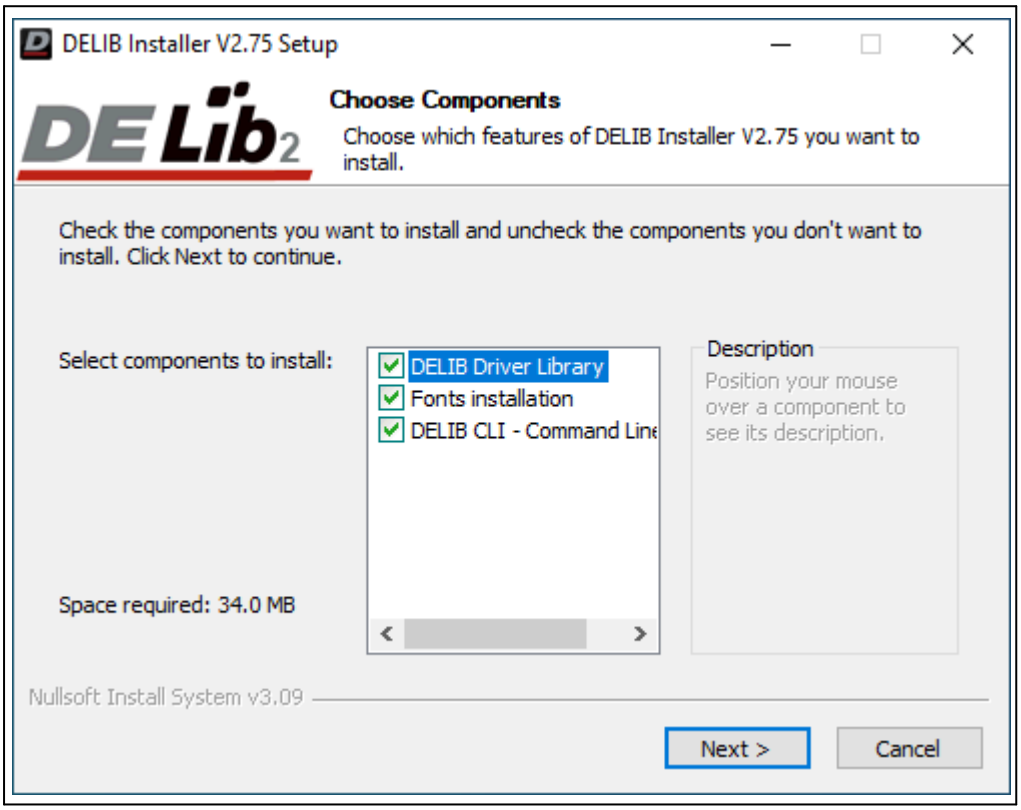


In addition to the DELIB driver library, you also have the option of installing our "DELIB CLI - Command Line" software.

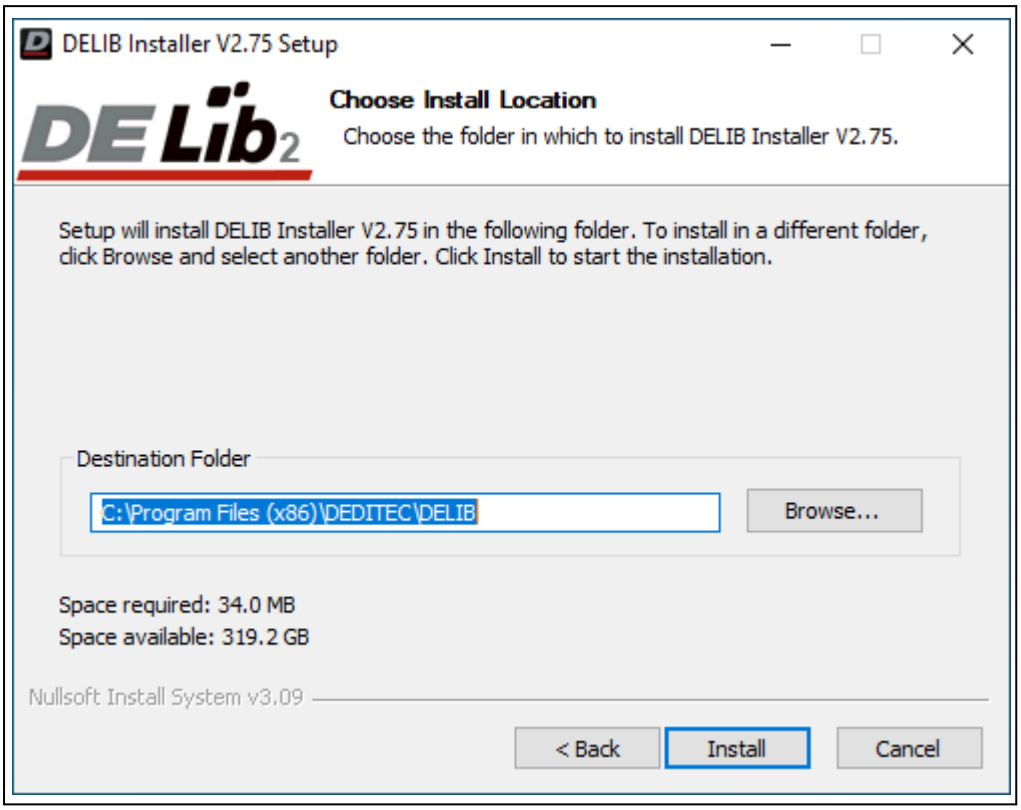
You can use this program to execute commands directly from the command line. You can find more information on this in the following chapter:

DELIB CLI (command-line interface) for Windows

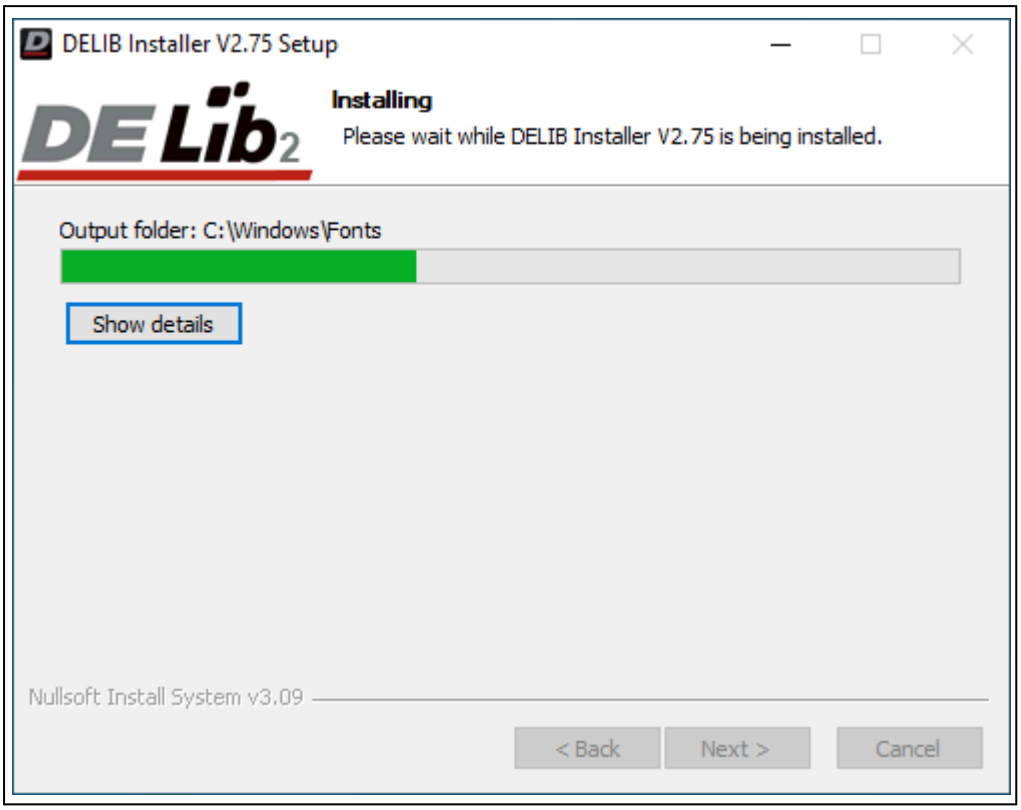
DELIB CLI (command-line interface) for Linux



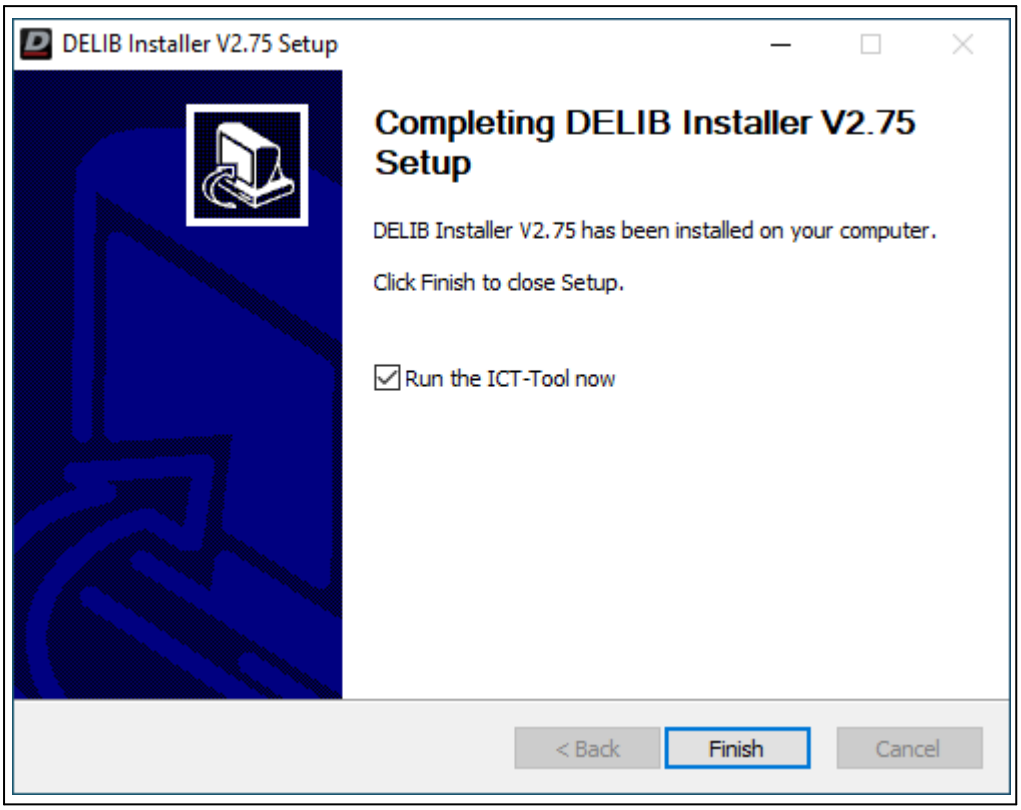
Select your desired installation path and confirm with the "Install" button



Installation progress of the driver library.



After completing the installation, close the setup with the "Finish" button



With "Run the ICT-Tool now" you have the option of starting the ICT-Tool directly after exiting the DELIB setup.

In addition, you will now find the ICT tool and the DELIB-CLI in your Windows "Start" directory.

4.2.3. ICT-Tool

With our new ICT tool, your module can be easily configured, tested, diagnosed and flashed.

A more detailed description of the individual functions can be found in the chapter **ICT-Tool.**

4.3. DELIB Sample Sources (Windows program examples)

The DELIB Sample Sources offer sample programs including source code for almost all DEDITEC products.

To simplify the quick start with our modules, you will find source codes for the following programming languages:

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

4.3.1. Installation DELIB Sample Sources

You can install the DELIB Sample Sources via a separate setup.

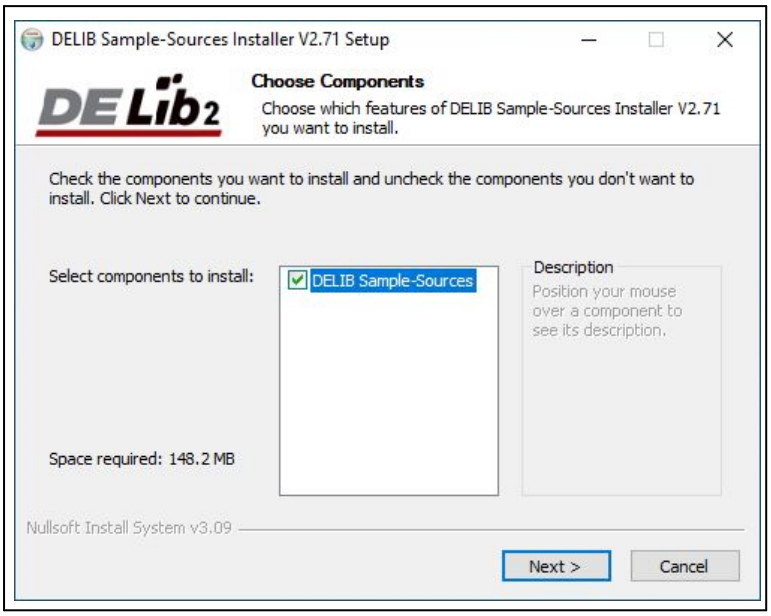
You can also find a current version of the sample sources on the Internet under the following link in the Samples tab:

https://www.deditec.de/en/downloads/software/

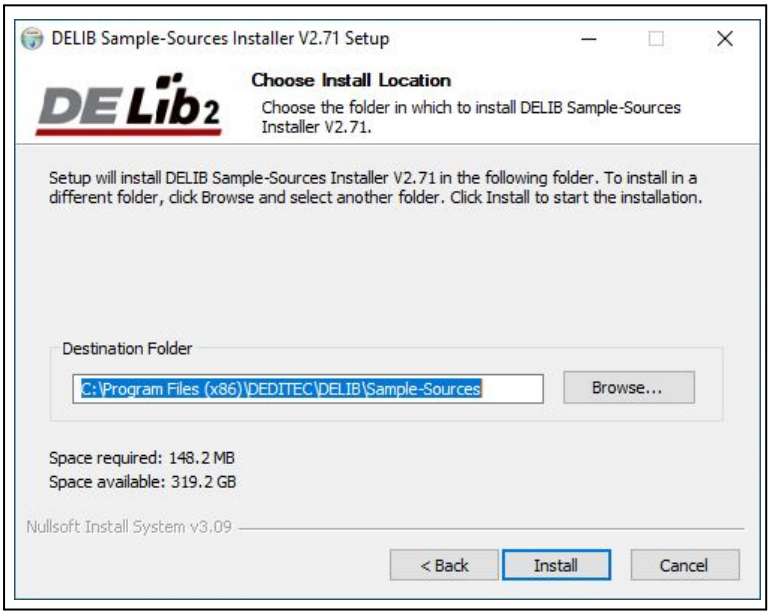
Startup screen of the DELIB Sample Sources Installer



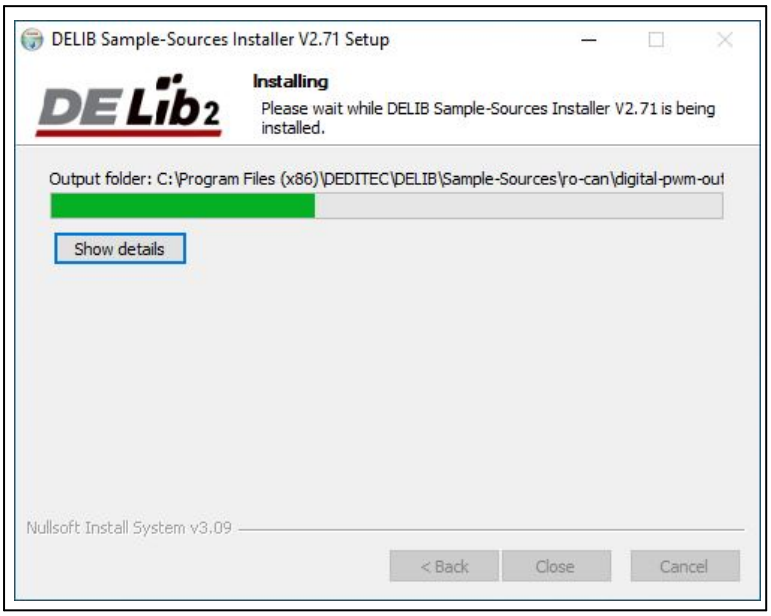
Press Next.



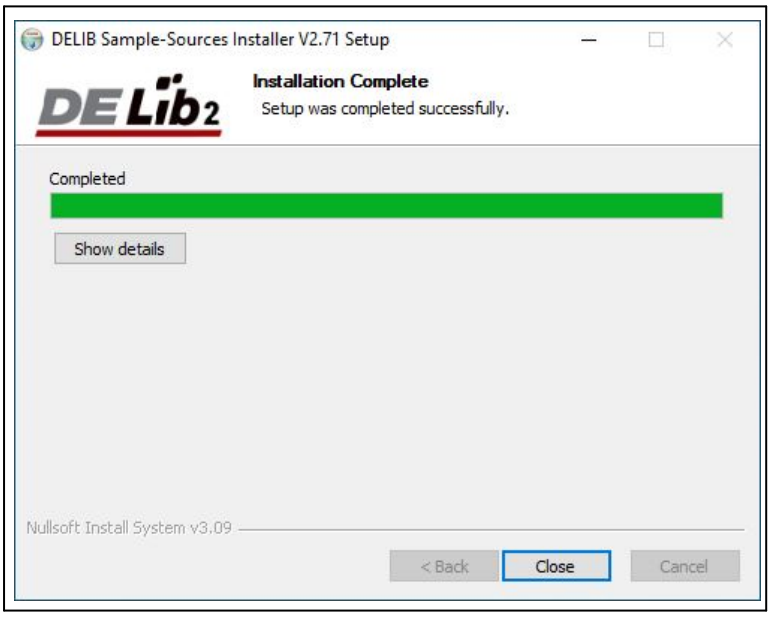
Select the installation folder and press Install.



The DELIB sample sources are now installed.



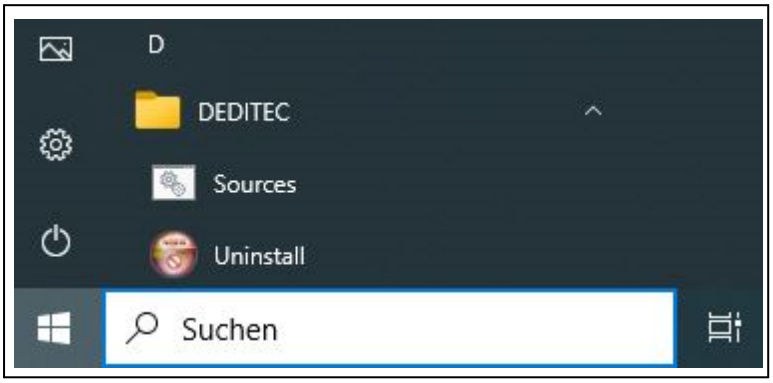
The DELIB Sample Sources have been successfully installed. Press Close to finish the installation.



4.3.2. Use of the DELIB Sample Sources

After installing the DELIB Sample Sources you can find them under

Start → Programs → DEDITEC → DELIB → Sample Sources → Sources

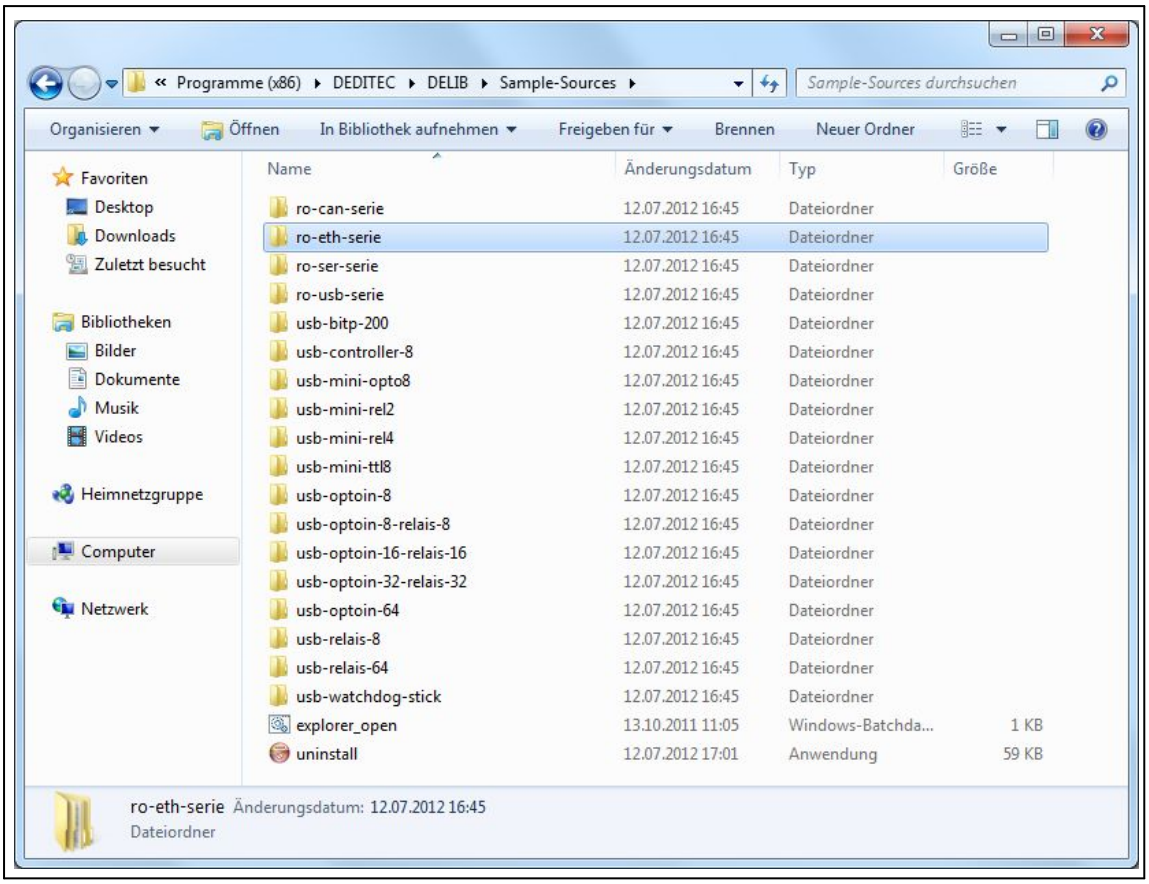


Now the Windows Explorer opens with an overview of all products for which a sample program is available.

4.3.2.1. Step 1 - Product selection

For example you need a help for programming the digital inputs of a RO-ETH module (e.g. RO-ETH-O16) in the programming language Visual-C.

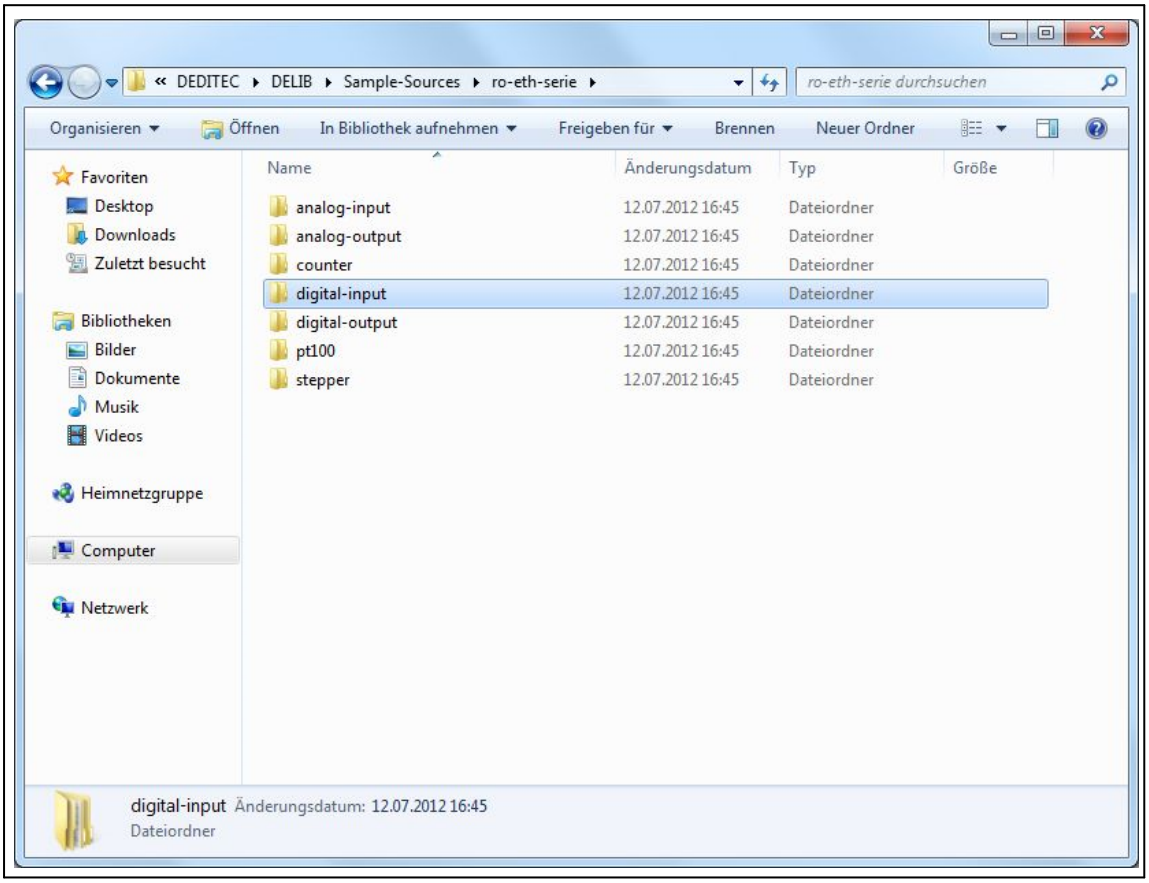
Since it is a RO-ETH product, select or open the ro-eth-series folder.


« Programme (x86) ▸ DEDITEC ▸ DELIB ▸ Sample-Sources ▸
Sample-Sources durchsuchen
Organisieren
Öffnen
In Bibliothek aufnehmen
Freigeben für
Brennen
Neuer Ordner
Favoriten
Desktop
Downloads
Zuletzt besucht
Bibliotheken
Bilder
Dokumente
Musik
Videos
Heimnetzgruppe
Computer
Netzwerk
Name
Änderungsdatum
Typ
Größe
ro-can-serie 12.07.2012 16:45 Dateiordner
ro-eth-serie 12.07.2012 16:45 Dateiordner
ro-ser-serie 12.07.2012 16:45 Dateiordner
ro-usb-serie 12.07.2012 16:45 Dateiordner
usb-bitp-200 12.07.2012 16:45 Dateiordner
usb-controller-8 12.07.2012 16:45 Dateiordner
usb-mini-opto8 12.07.2012 16:45 Dateiordner
usb-mini-rel2 12.07.2012 16:45 Dateiordner
usb-mini-rel4 12.07.2012 16:45 Dateiordner
usb-mini-ttl8 12.07.2012 16:45 Dateiordner
usb-optoin-8 12.07.2012 16:45 Dateiordner
usb-optoin-8-relais-8 12.07.2012 16:45 Dateiordner
usb-optoin-16-relais-16 12.07.2012 16:45 Dateiordner
usb-optoin-32-relais-32 12.07.2012 16:45 Dateiordner
usb-optoin-64 12.07.2012 16:45 Dateiordner
usb-relais-8 12.07.2012 16:45 Dateiordner
usb-relais-64 12.07.2012 16:45 Dateiordner
usb-watchdog-stick 12.07.2012 16:45 Dateiordner
explorer_open 13.10.2011 11:05 Windows-Batchda... 1 KB
uninstall 12.07.2012 17:01 Anwendung 59 KB
ro-eth-serie Änderungsdatum: 12.07.2012 16:45
Dateiordner

4.3.2.2. Step 2 - Category selection

In the next step, you will find an overview of the available categories for the selected product.

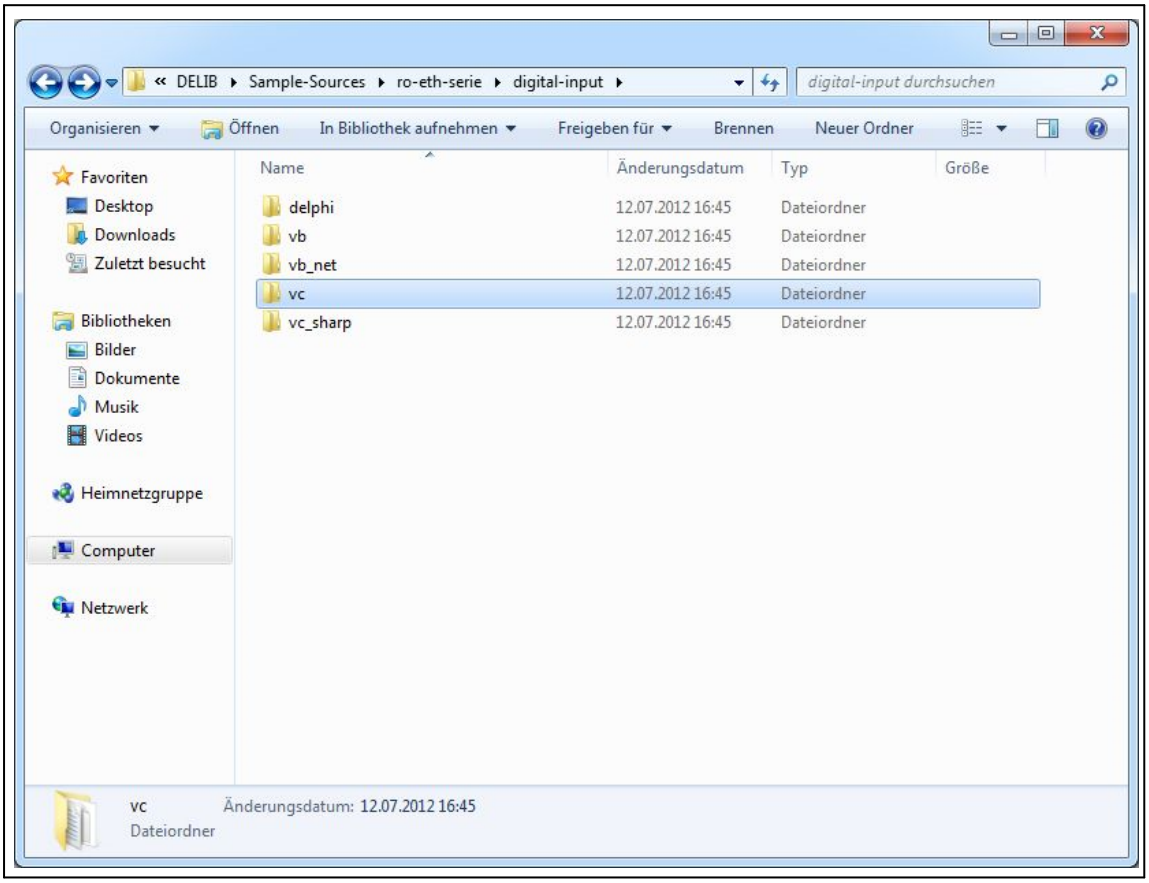
Since we focus on the digital inputs in this example, select the category digital-input



4.3.2.3. Step 3 - Programming language selection

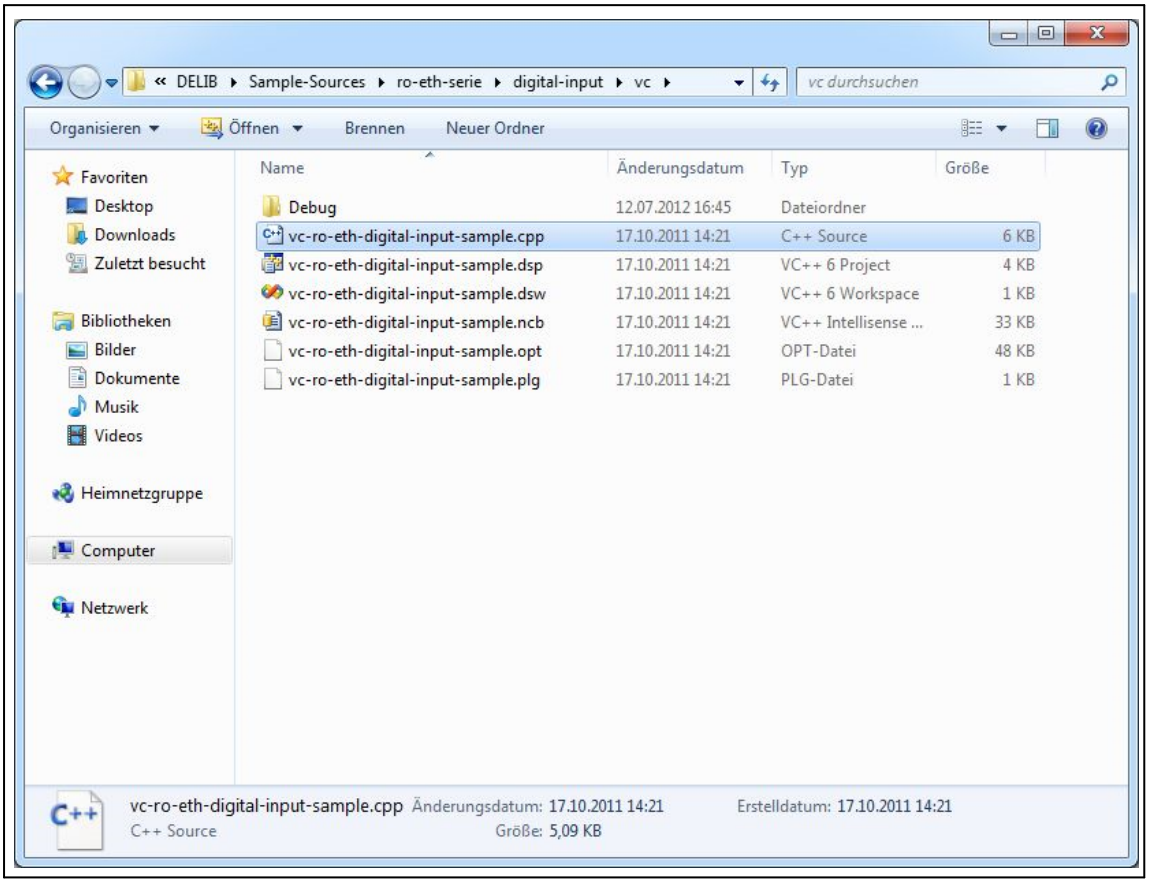
In this step you will see all available programming examples of the selected category, sorted by programming languages.

Since we are focusing on the Visual-C programming language in this example, open the vc folder.



4.3.2.4. Step 4 - Source code

After selecting the programming language you will get the following overview:



You can now open the source code of the sample program (in this case .cpp file) with any text editor.

vc-ro-eth-digital-input-sample.cpp - Editor

Datei Bearbeiten Format Ansicht ?

```
//*****************************************************************************************
//*****************************************************************************************

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

        // ---------------------------------------------------------------------------
        // Open Module

        handle = DapiOpenModule(RO_ETH,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

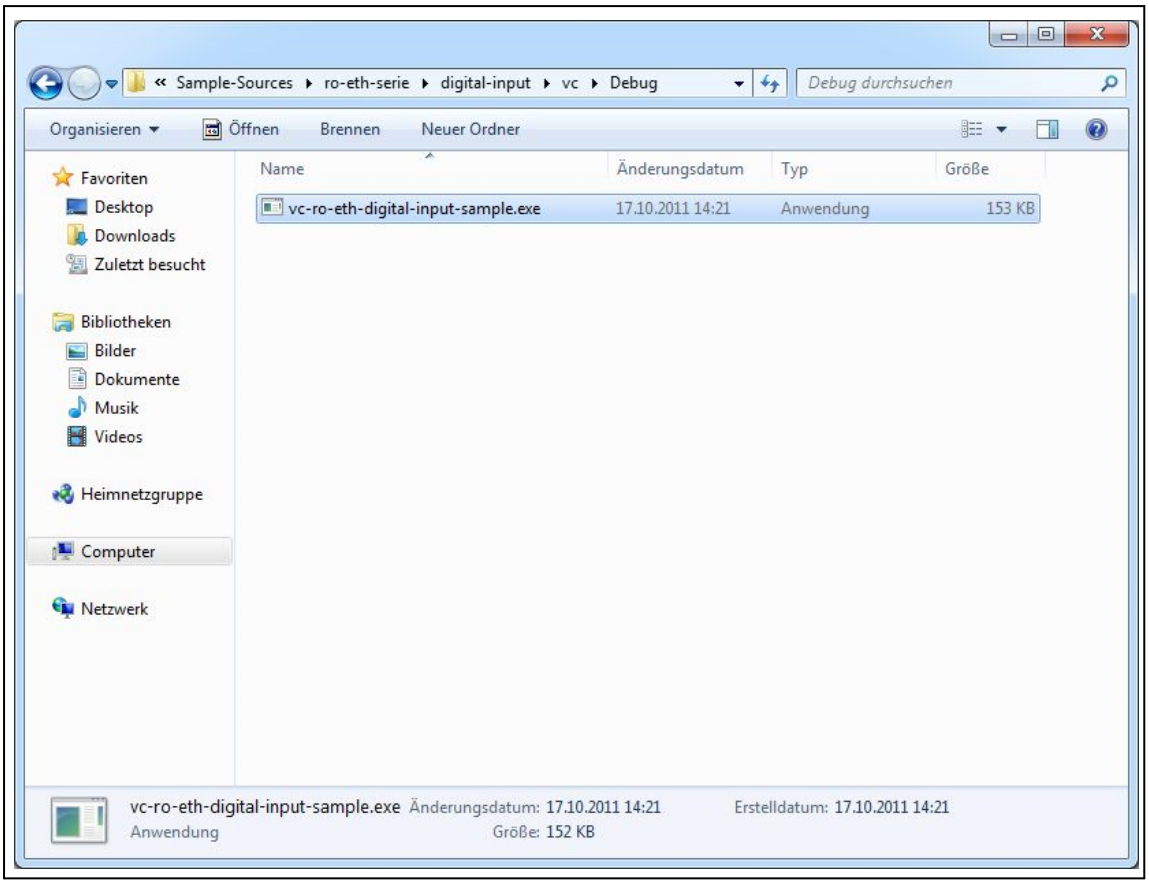
        value = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG, DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI, 0, 0);
        IsError();
        printf("Configuration of the module: no. of digital inputs %d\n", value);
        printf("Press any key to continue\n");
        getch();

        // ---------------------------------------------------------------------------
        // Read input channels

        ret = DapiDIGet1(handle, 0);
        IsError();
        printf("Reading input channel 0\n");
        printf("value = %d\n", ret);
        printf("Press any key to continue\n");
        getch();

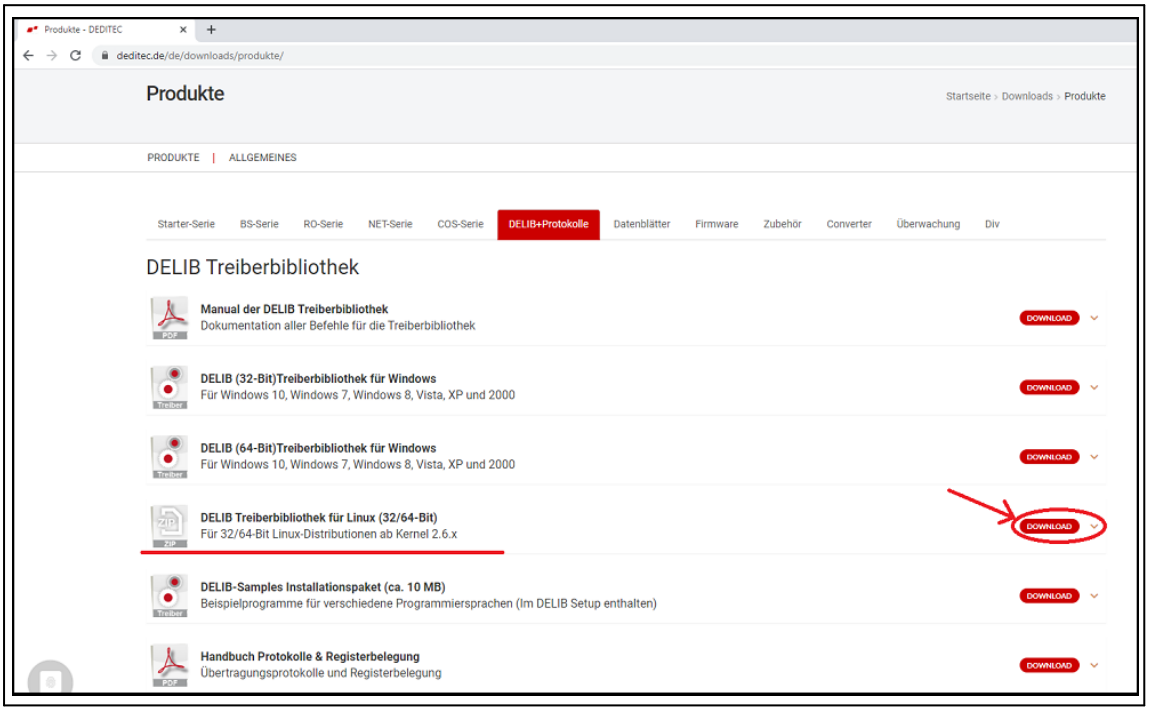
        ret = DapiDIGet1(handle, 1);
```

In addition, you will find an already compiled and executable program for this project in the debug folder.

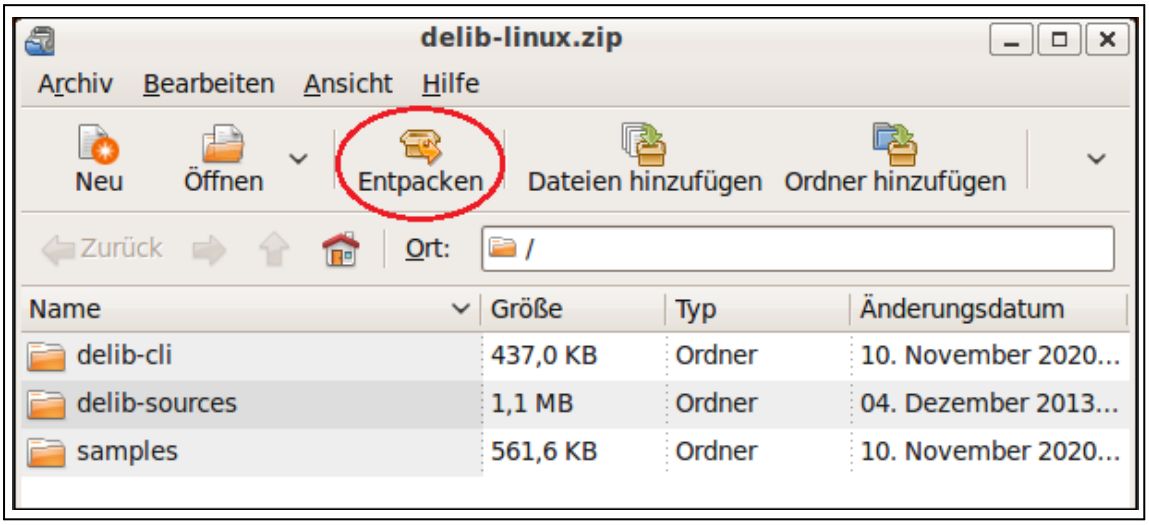

« Sample-Sources ▸ ro-eth-serie ▸ digital-input ▸ vc ▸ Debug
Debug durchsuchen
Organisieren
Öffnen
Brennen
Neuer Ordner
Favoriten
Desktop
Downloads
Zuletzt besucht
Bibliotheken
Bilder
Dokumente
Musik
Videos
Heimnetzgruppe
Computer
Netzwerk
Name
Änderungsdatum
Typ
Größe
vc-ro-eth-digital-input-sample.exe
17.10.2011 14:21
Anwendung
153 KB
vc-ro-eth-digital-input-sample.exe Änderungsdatum: 17.10.2011 14:21
Erstelldatum: 17.10.2011 14:21
Anwendung
Größe: 152 KB

4.4. DELIB for Linux

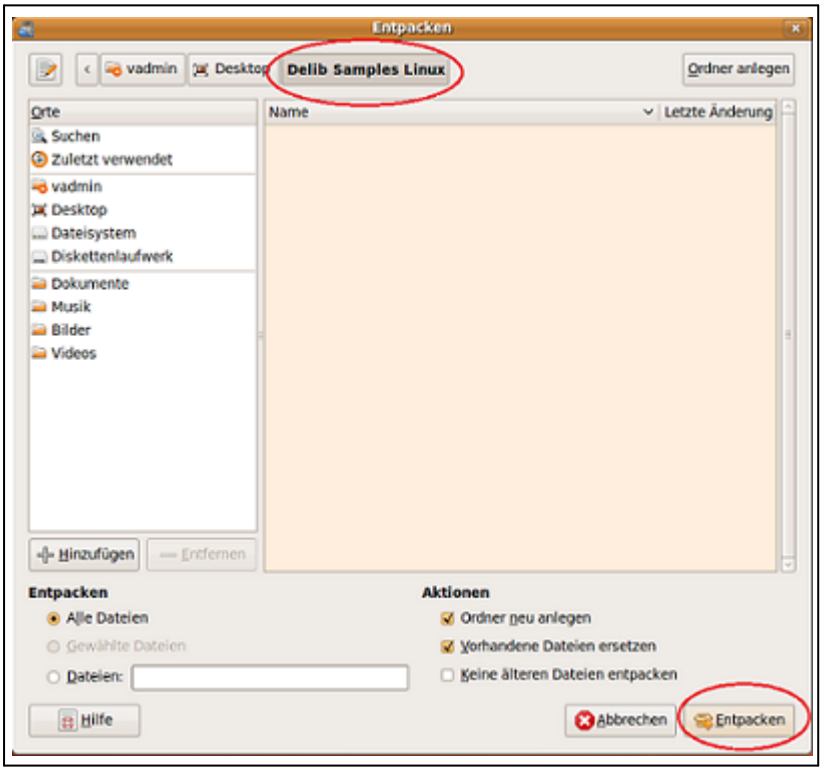
Download the Delib Linux driver library under "www.deditec.de/de/downloads/produkte/" in the tab "DELIB+Protocols" or under "www.deditec.de/media/zip/delib/delib-linux.zip" directly on your Linux system.



Unzip the "delib-linux.zip" to any destination folder. To do this, double-click the zip file and then use the "Unzip" button in the top menu bar.



Select your destination folder and then click the "Unzip" button.



4.4.1. Using the DELIB driver library for Linux

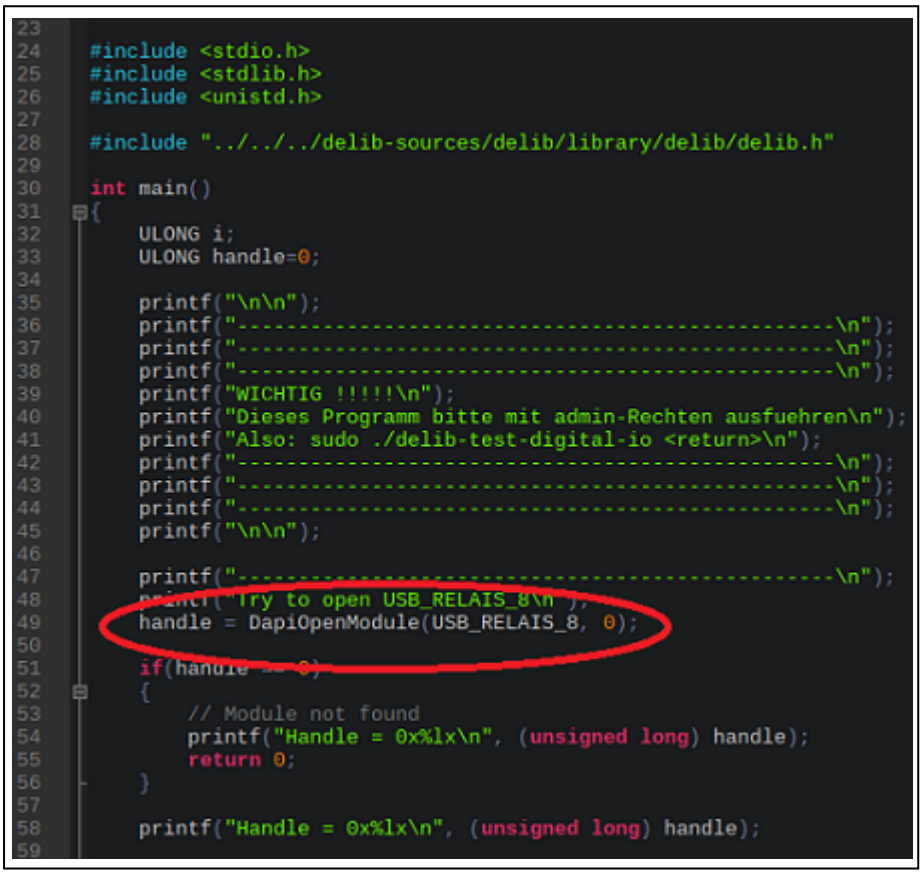
4.4.1.1. Delib USB sample in Linux

Presets

In this program example a USB_RELAIS_8 module is addressed. If you use another module, you have to enter the following in the file

"/samples/usb_sample/source/usb_sample.c" with the command "DapiOpenModule". The exact name can be found in "delib.h". You can find it in the directory

"/delib-sources/delib/library/delib/delib.h".



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#include "../../../delib-sources/delib/library/delib/delib.h"

int main()
{
    ULONG i;
    ULONG handle=0;

    printf("\n\n");
    printf("-------------------------------------------------\n");
    printf("-------------------------------------------------\n");
    printf("-------------------------------------------------\n");
    printf("WICHTIG !!!!!\n");
    printf("Dieses Programm bitte mit admin-Rechten ausfuehren\n");
    printf("Also: sudo ./delib-test-digital-io <return>\n");
    printf("-------------------------------------------------\n");
    printf("-------------------------------------------------\n");
    printf("-------------------------------------------------\n");
    printf("\n\n");

    printf("-------------------------------------------------\n");
    printf("Try to open USB_RELAIS_8\n");
    handle = DapiOpenModule(USB_RELAIS_8, 0);

    if(handle == 0)
    {
        // Module not found
        printf("Handle = 0x%lx\n", (unsigned long) handle);
        return 0;
    }

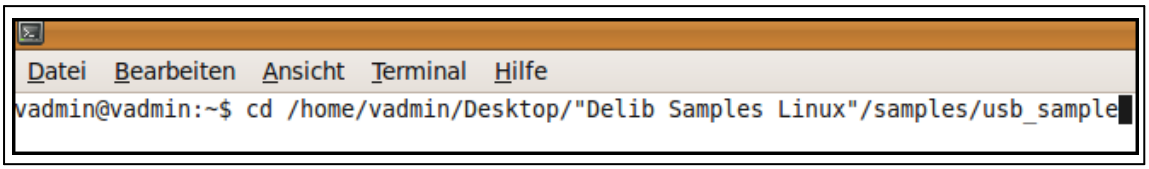
    printf("Handle = 0x%lx\n", (unsigned long) handle);
```

Compiling the USB sample

To compile the test program, open a terminal window and navigate with the command

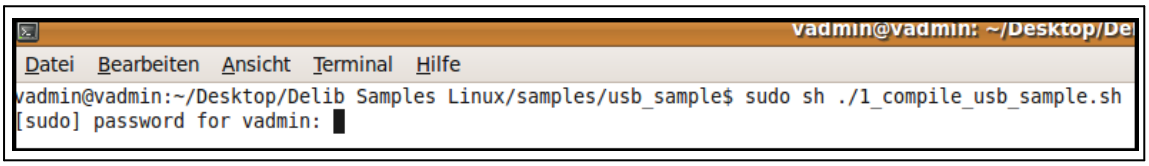
"cd /<directory path>" to the "/samples/usb_sample" directory.

Tip: If there are spaces in your folder name, enter them in " " as shown in the example below.



To compile, now open the shell script contained in it with the command "sudo sh ./1_compile_usb_sample.sh".

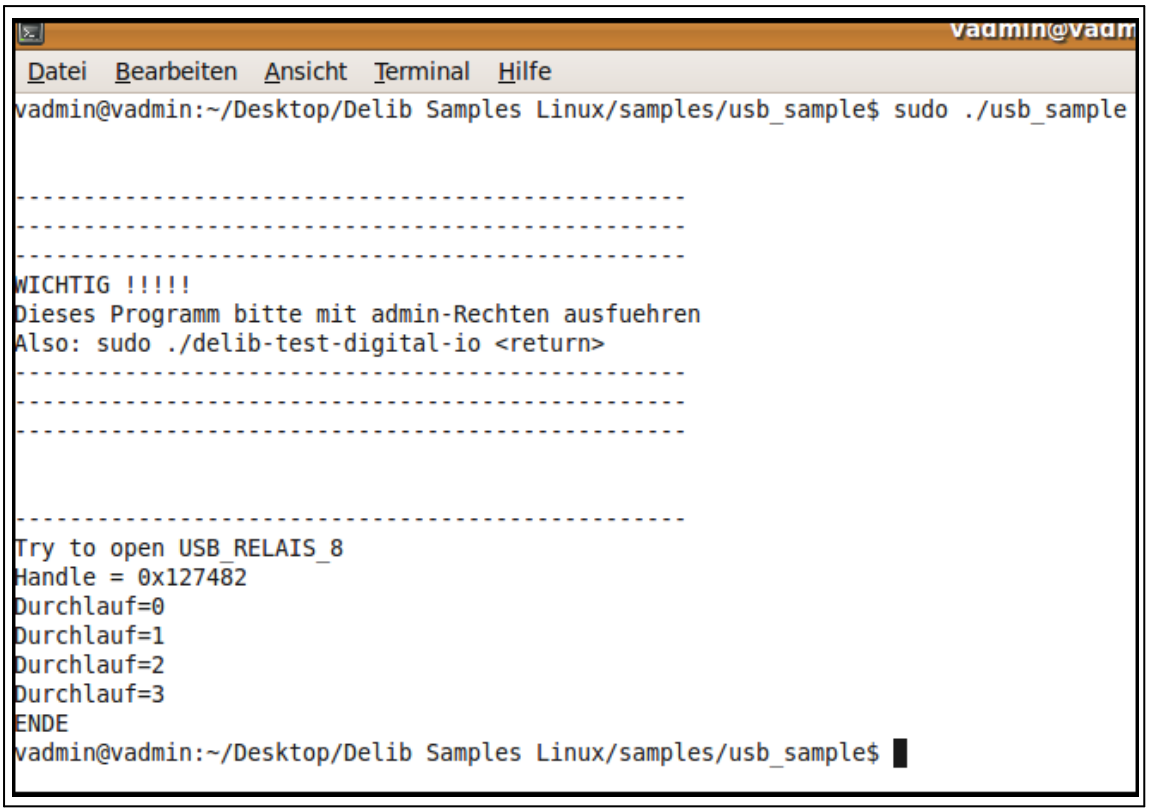
If necessary, enter your user password.



If the compilation was successful, "compiling successful" should now appear in the terminal window.
The file "usb_sample" has been added to the directory.

Now you can execute the sample program with "sudo ./usb_sample".

IMPORTANT!! You need admin rights to run it. Therefore use the command with "sudo".



The program is now executed.

In this example all digital outputs of the USB_RELAIS_8 are switched on and off again in a loop.

4.4.1.2. Delib ETH sample in Linux

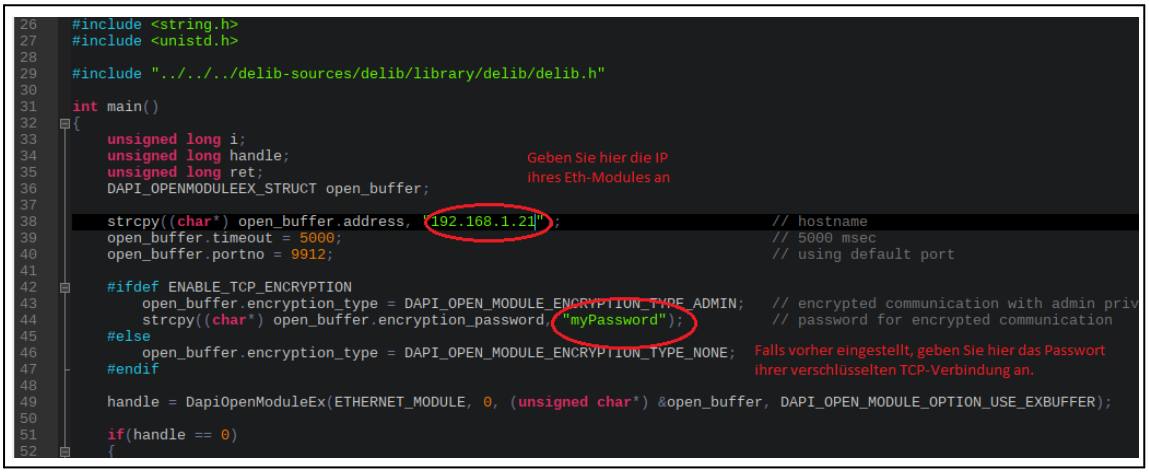
Presets

In this program example the module is addressed with the IP "192.168.1.21". You can change this in the file

"/samples/ethernet_sample/source/eth_sample.c" (see picture below).

If you have preset a password for an encrypted TCP connection, you can also enter it there (see picture below). If you have not specified a password, you can leave this line unchanged.

The configuration of the ETH modules can be set via the ICT-Tool, as well as via the web interface of the module.




```
#include <string.h>
#include <unistd.h>

#include "../../../delib-sources/delib/library/delib/delib.h"

int main()
{
    unsigned long i;
    unsigned long handle;
    unsigned long ret;
    DAPI_OPENMODULEEX_STRUCT open_buffer;

    strcpy((char*) open_buffer.address, "192.168.1.21");                    // hostname
    open_buffer.timeout = 5000;                                             // 5000 msec
    open_buffer.portno = 9912;                                              // using default port

    #ifdef ENABLE_TCP_ENCRYPTION
        open_buffer.encryption_type = DAPI_OPEN_MODULE_ENCRYPTION_TYPE_ADMIN;   // encrypted communication with admin priv
        strcpy((char*) open_buffer.encryption_password, "myPassword");          // password for encrypted communication
    #else
        open_buffer.encryption_type = DAPI_OPEN_MODULE_ENCRYPTION_TYPE_NONE;
    #endif

    handle = DapiOpenModuleEx(ETHERNET_MODULE, 0, (unsigned char*) &open_buffer, DAPI_OPEN_MODULE_OPTION_USE_EXBUFFER);

    if(handle == 0)
    {
```

If you use a module without digital inputs, you must comment out the lines in the same file as shown below.



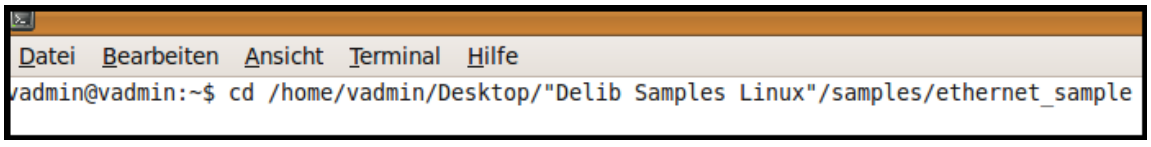
Compiling the ETH sample

For compiling the test program, open a terminal window and navigate with the command

"cd /<directory path>" to the "/samples/ethernet_sample" directory.

Tip: If there are spaces in your folder name, enter them as shown in the example below

in " " as shown in the example below.

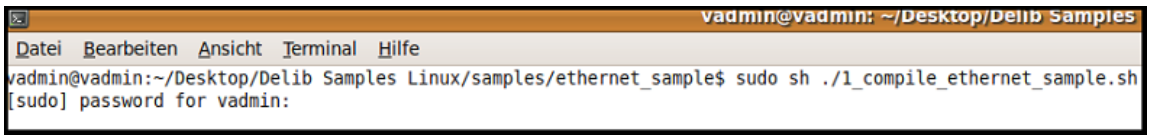


To compile, now open the desired shell script with the command

"sudo sh ./<DATEINAME>"

- If you want to access the module via an unencrypted TCP connection, use the file "1_compile_ethernet_sample.sh".
- If you want to control the module over an encrypted TCP connection, use the file "2_compile_ethernet_sample_with_encryption.sh".

If necessary, enter your user password.

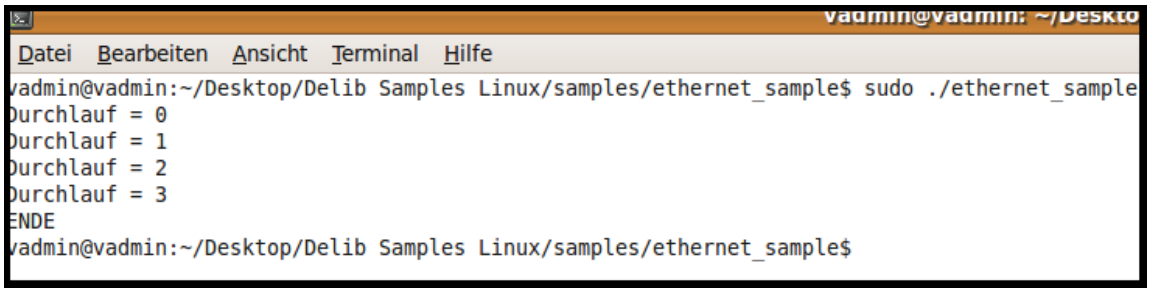


If the compilation was successful, "compiling successful" should now appear in the terminal window.

The file "ethernet_sample" has been added to the directory.

Now you can execute the sample program with "sudo ./ethernet_sample".

IMPORTANT!! You need admin rights to run it. Therefore use the command with "sudo".



The program is now executed.

In this example, all outputs of the module are switched on and off again in a loop.

4.4.2. DELIB CLI (command-line interface) for Linux

The DELIB CLI command for Linux is located in the folder /deditec-cli/ after unpacking the zip archive "delib-linux-cli".

Definition for USB modules (Linux)

sudo delib_cli [command] [channel] [value | unit ["nounit"]]

Definition for ETH modules (Linux)

delib_cli [command] [channel] [value | unit ["nounit"]]

Note:

The individual parameters are separated only by a space.

Upper and lower case are not considered here.

Parameter

Command	Kabal	Value		unit	nounit
di1	0, 1, 2, ...	-		hex	nounit
di8	0, 8, 16, ...				
di16					
di32					
ff	0, 32, ...	-		hex	nounit
do1	0, 1, 2, ...	0/1 (1-bit command)		-	-
do8	0, 8, 16, ...	8-bit value	(Bit 0 for channel 1, Bit 1 for channel 2, ...)		
do16		16 bit value			
do32		32-bit value			
ai	0, 1, 2, ...	-		hex, volt, mA	nounit
ao	0, 1, 2, ...	Integer or hexadecimal number (starting with 0x).		-	-

Return value

State of the read digital inputs

In combination with parameter unit "hex" the state is read as hex

State of the FlipFlips of the digital inputs

In combination with parameter unit "hex" the state is read as hex

Status of the read analog inputs

In combination with parameter unit "hex" the state is read as hex

In combination with parameter unit "volt" the voltage is read

In combination with parameter unit "mA" the current is read

4.4.2.1. Configuration of the DELIB CLI

Presets

Before using the DELIB CLI for the first time, the "delib_cli.cfg" must be edited with a text editor.

You can find the "delib_cli.cfg" in the directory "/delib_cli/".

Contents of the "delib_cli.cfg":

```
moduleID=14;
moduleNR=0;
RO-ETH_ipAddress=192.168.1.11;
```

moduleID

The corresponding number of the hardware used must be entered as moduleID.

This number can be taken from the "delib.h".

Under Linux you find this in the zip archive of the "delib-linux" under the path "delib-sources\delib\library\delib".

moduleNR

The moduleNR is assigned in the ICT-Tool.

This number is used to identify identical hardware.

The default value is 0.

RO-ETH_ipAddress

This entry is only required for the connection to our ETH modules.

The IP address of the ETH modules can be set via the ICT-Tool as well as via the web interface of the module.

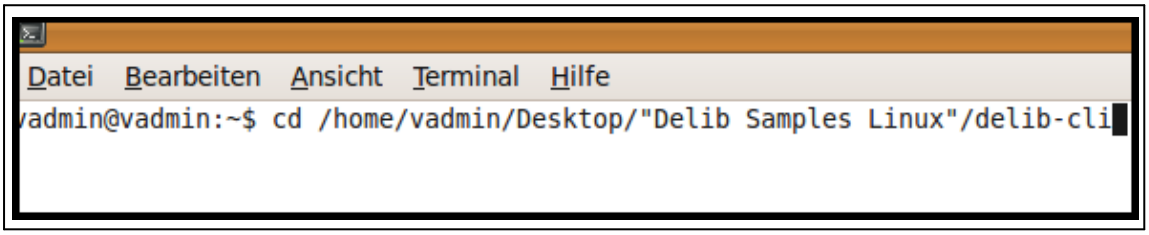
Compiling the Delib CLI sample

To compile the test program, open a terminal window and navigate with the command

"cd /<directory path>" to the "../delib_cli/" directory.

Hint: If there are spaces in your folder name, enter them as shown in the example below

in " " as shown in the example below.

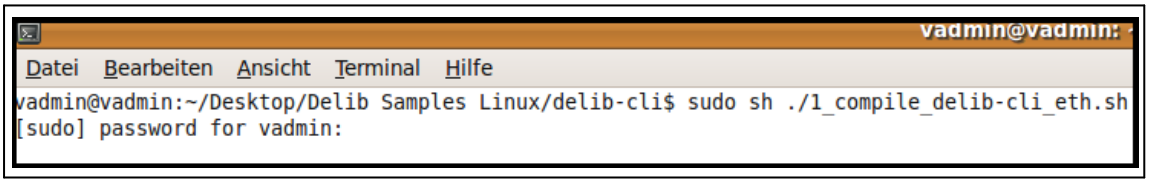


To compile, now open the desired shell script with the command

"sudo sh ./<DATEINAME>"

- ETH - "1_compile_delib-cli_eth.sh"
- USB - "2_compile_delib-cli_usb.sh"

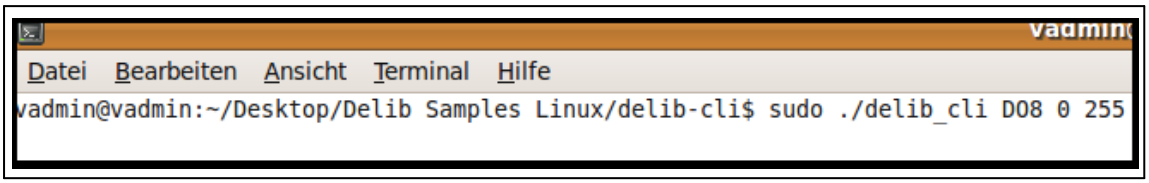
If necessary, enter your user password.



If the compilation was successful, "compiling successfull" should now appear in the terminal window. The file "delib_cli" has been created in the directory. Now you can compile the sample program with

"sudo ./delib_cli [command] [channel] [value | unit ["nounit"]] ".

IMPORTANT!! You need admin rights to run it. Therefore use the command with "sudo".


Datei Bearbeiten Ansicht Terminal Hilfe
vadmin@vadmin:~/Desktop/Delib Samples Linux/delib-cli$ sudo ./delib_cli DO8 0 255

4.4.2.2. DELIB CLI Examples

Digital outputs

```
sudo delib_cli DO1 17 1
```

→ switches on the 18th digital relay of a USB module

```
sudo delib_cli  DO1 3 0
```

→ switches off the 4th digital relay of a RO-ETH module

Digital inputs

```
sudo delib_cli DI1 3
```

Example of a return value: 1

→ read the state of the 4th digital input of a USB module and return it

```
sudo delib_cli DI8 0 hex
```

Example of a return value: 0xFF

(a signal is present on channels 1 to 8)

→ read the value of digital input 1-8 of a RO-ETH module as hexadecimal number

```
sudo delib_cli FF  0
```

Example of a return value: 192

(a change of state has been detected on channels 7 and 8).

→ read the value of the FlipFlops of the digital inputs 1-32

```
sudo delib_cli FF  32
```

Example of a return value: 65535

(a change of state has been detected on channels 33 to 64).

→ read the value of the FlipFlops of the digital inputs 33-64

```
sudo delib_cli FF  0 hex
```

Example of a return value: 0xD00

(a change of state was detected on channels 9, 11 and 12)

→ read the value of the FlipFlops of the digital inputs 1-32 as hexadecimal number

Analog outputs

```
sudo delib_cli  AO 7 4711
```

→ sets the decimal value 4711 to the 8th analog output of a USB module

```
sudo delib_cli AO 6 0x4711
```

→ sets the hexadecimal value 0x4AF1 to the 7th analog output of a RO-ETH module

Analog inputs

```
sudo delib_cli AI 2
```

Example of a return value: 1234

→ reads the value of the 3rd analog input as decimal number of a USB module

```
sudo delib_cli AI 2 hex
```

Example of a return value: 0x1FA

→ reads the value of the 3rd analog input as hexadecimal number of a RO-ETH module

4.5. Web interface

Enter the IP address (delivery state 192.168.1.1) of the module in an Internet browser.



The integrated web server of the module requires an authentication

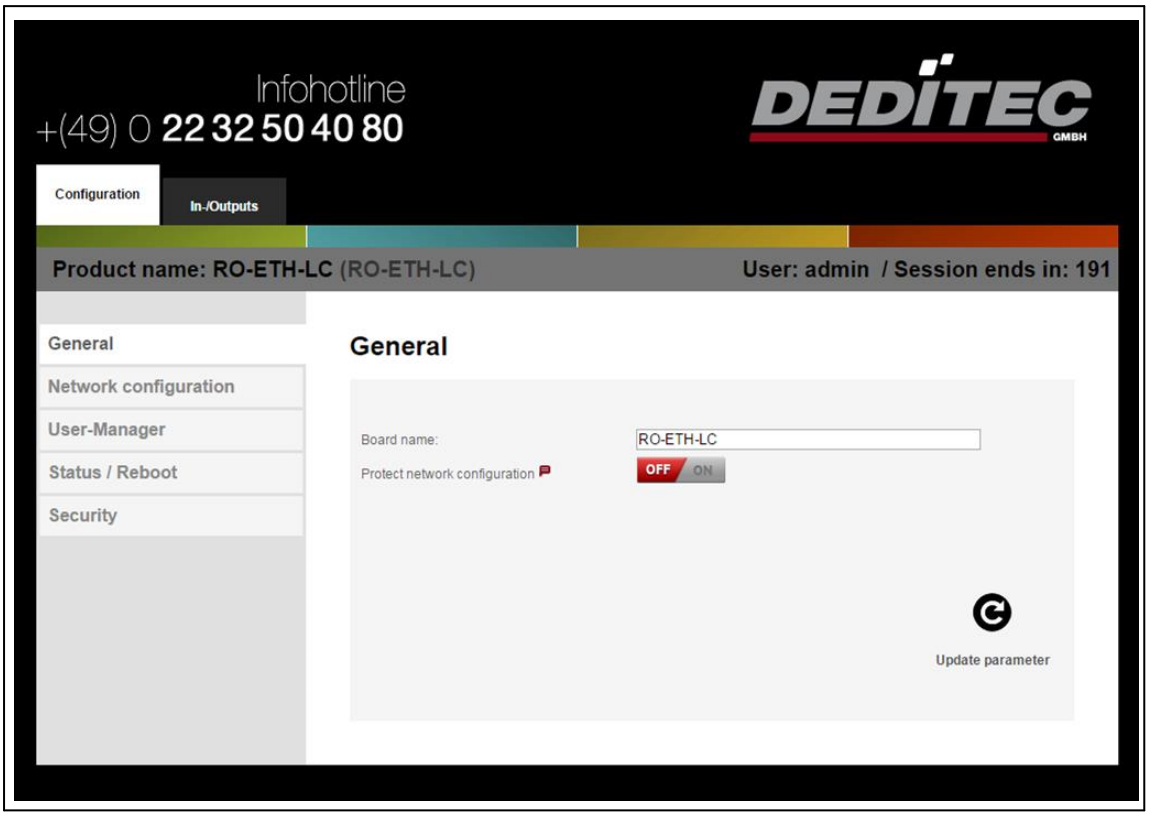
to protect the module from unauthorized access.

By default, the following user is set up:

Username	Password	Rights
admin	admin	Administrator rights

4.5.1. Configuration

4.5.1.1. General



Boardname

This is the name of the module, which is also displayed in the ICT tool.

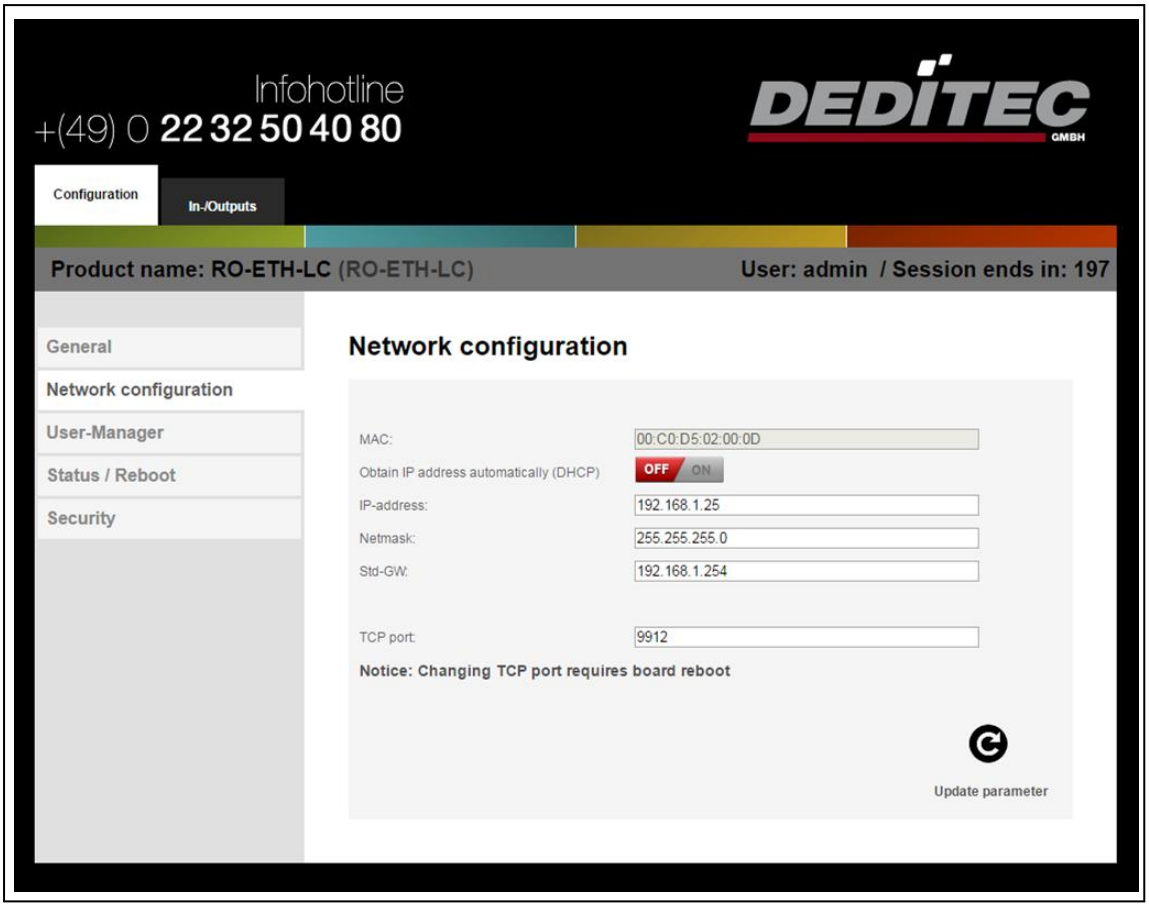
This name is used to identify several DEDITEC Ethernet modules in the network.

Protect network configuration

If this option is activated, the network configuration can only be changed via the module's web interface.

Changing this configuration (e.g. via the ICT tool) is prevented in this case.

4.5.1.2. Network configuration

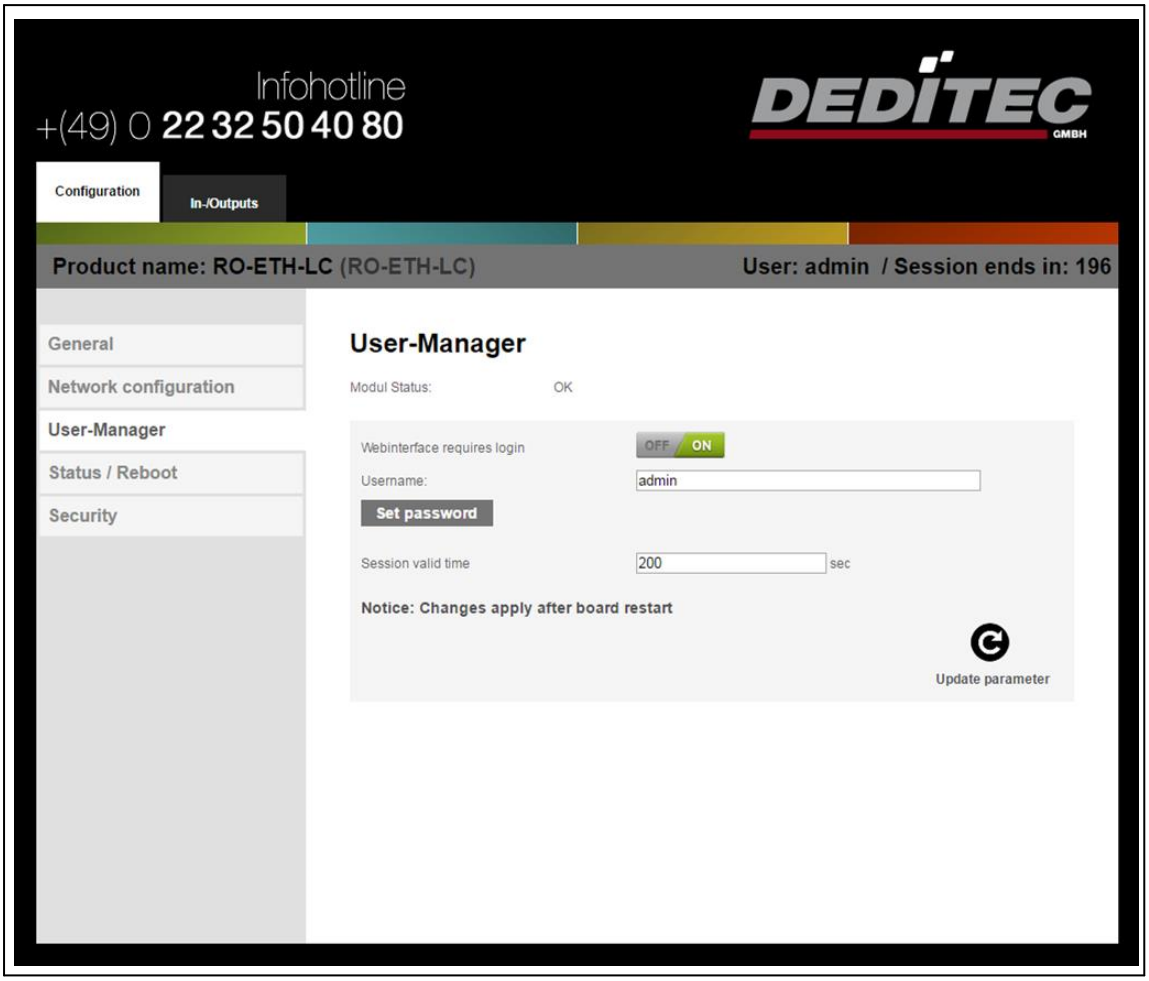


The network configuration of the module can be changed here.

If this configuration is changed, you will be automatically forwarded to the new IP address, provided it can be reached from the PC.

Changing the port requires a restart of the module.

4.5.1.3. User Manager



Remove

Deletes the corresponding user account.

Add User

Creates a new user account. The input mask prompts you to enter a user name and password.

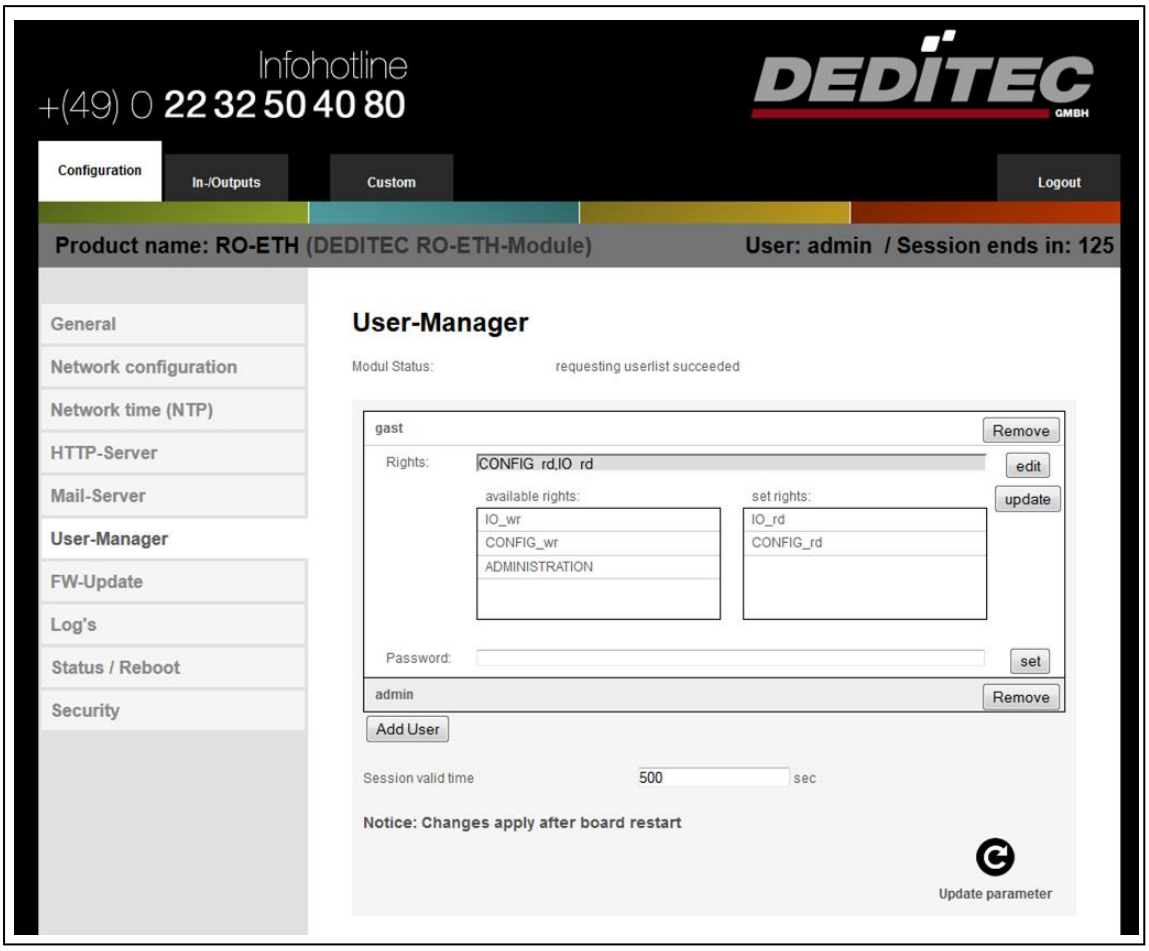
Session valid time

Specifies the time for which a registration is valid. If this time expires, the user must log in again.

Attention:

Changing this time requires a restart of the module.

Click on a user account (e.g. guest) to change the settings.



Edit

Changes the access permissions of the current user account. Click on an access authorization to either select or deselect it.

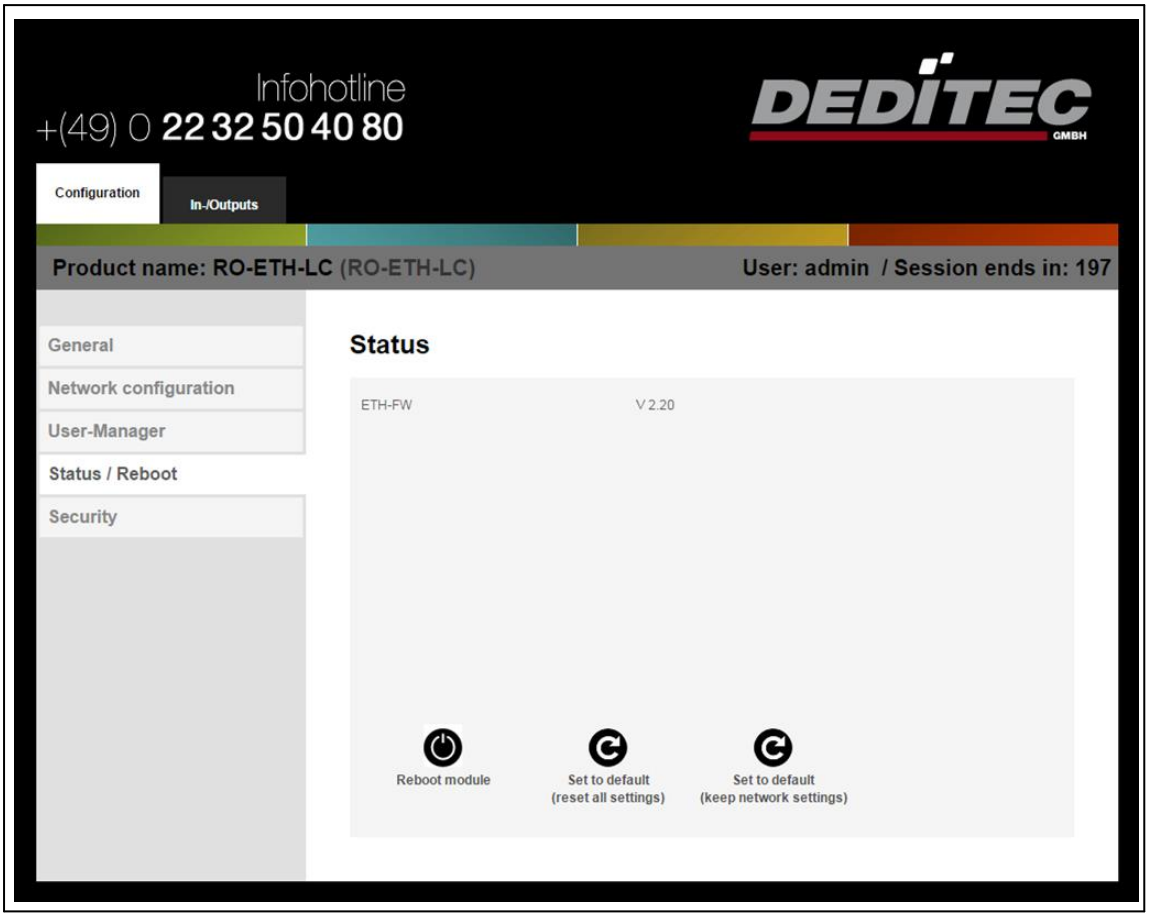
Update

Activates the access authorizations of the current user account.

Password

A new password can be set here for the current user account, which must be confirmed with set.

4.5.1.4. Status



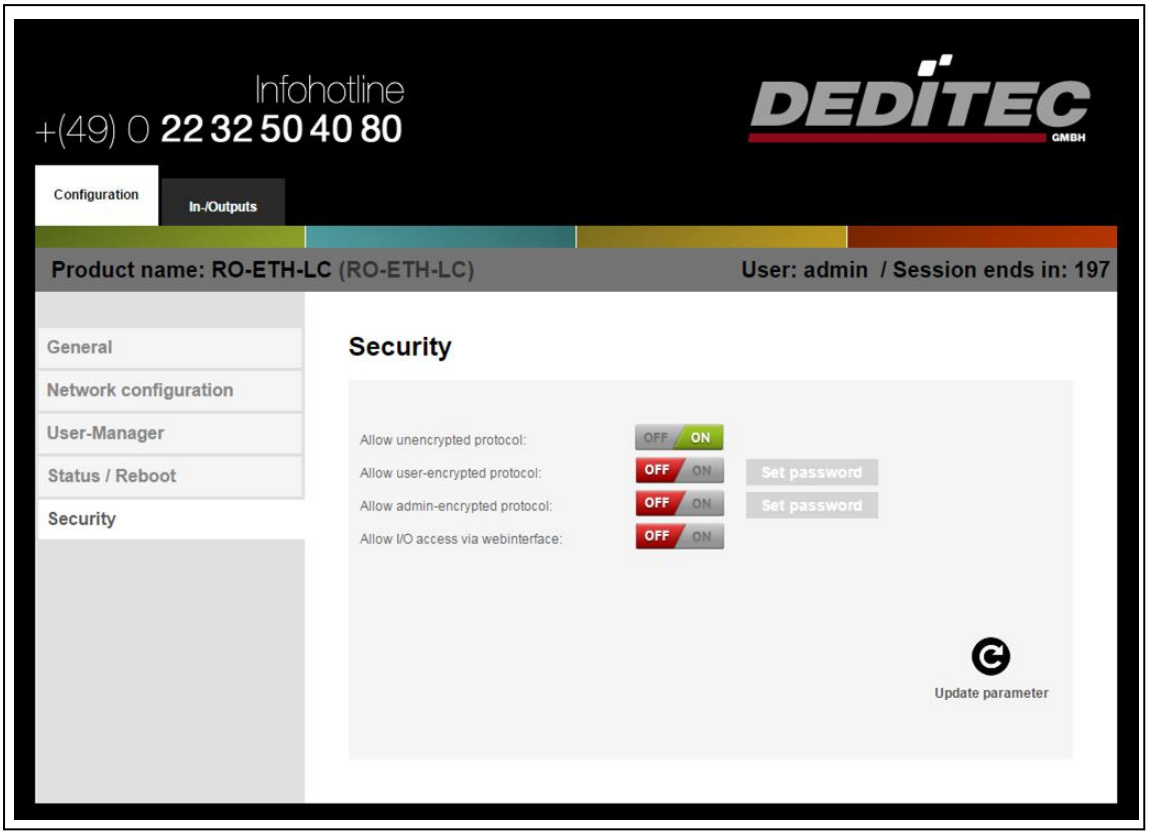
On the status page, you can see the revision numbers of the most important system processes.

In addition, the module can be restarted here with a click or the network settings can be reset to the factory settings.

Attention

Restarting the module and resetting to factory settings require administrator authorizations.

4.5.1.5. Security



Allow unencrypted protocol

This option determines whether access to the module with an unencrypted protocol is permitted.

Allow user-encrypted protocol

This option determines whether access to the module is permitted with a user-encrypted protocol.

This protocol has restricted access rights and is recommended for users whose communication is encrypted but whose system settings are not to be changed.

Allow admin-encrypted protocol

This option determines whether access to the module is permitted with an admin-encrypted protocol.

This protocol is required, for example, to read or change the names of the connected inputs/outputs.

Allow I/O access via webinterface

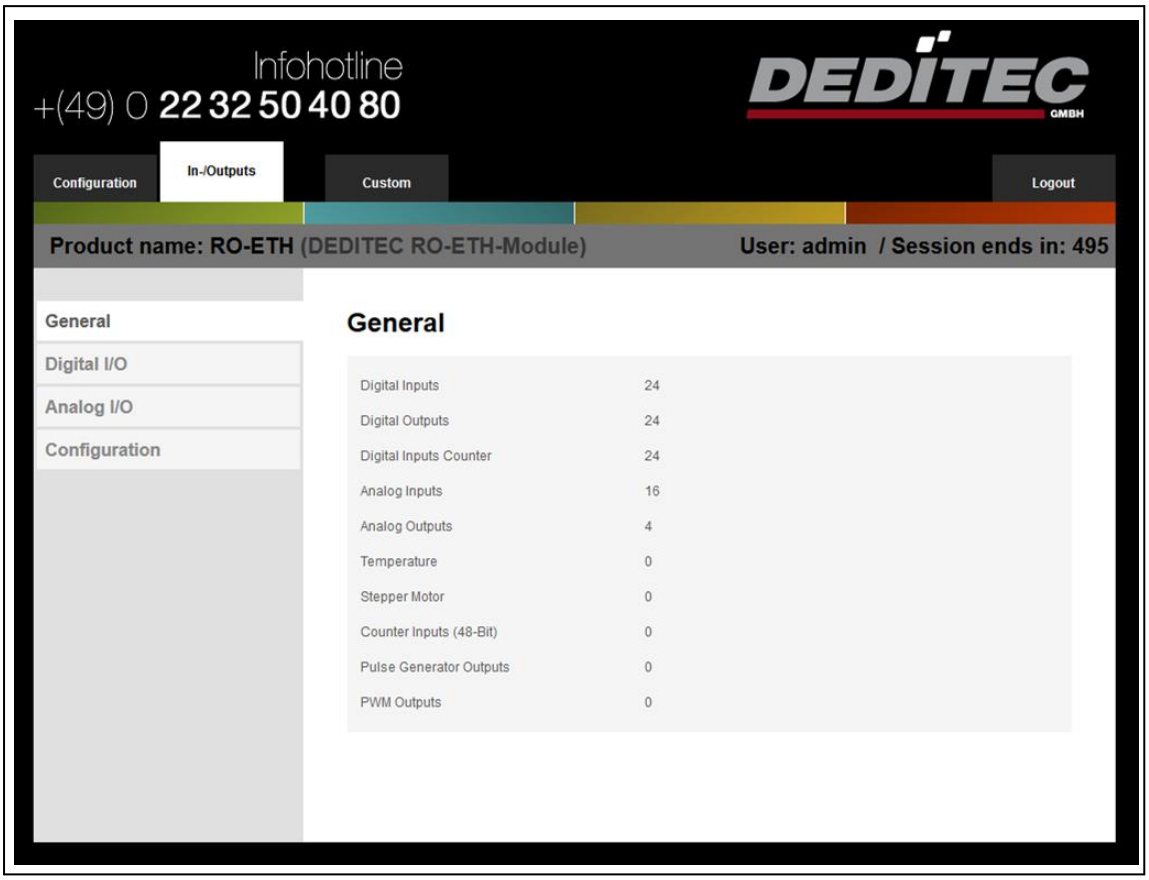
This option determines whether connected inputs/outputs can be read/switched from the web interface.

Attention:

If the password for user or admin encryption is changed, this must also be entered in the ICT-Tool.

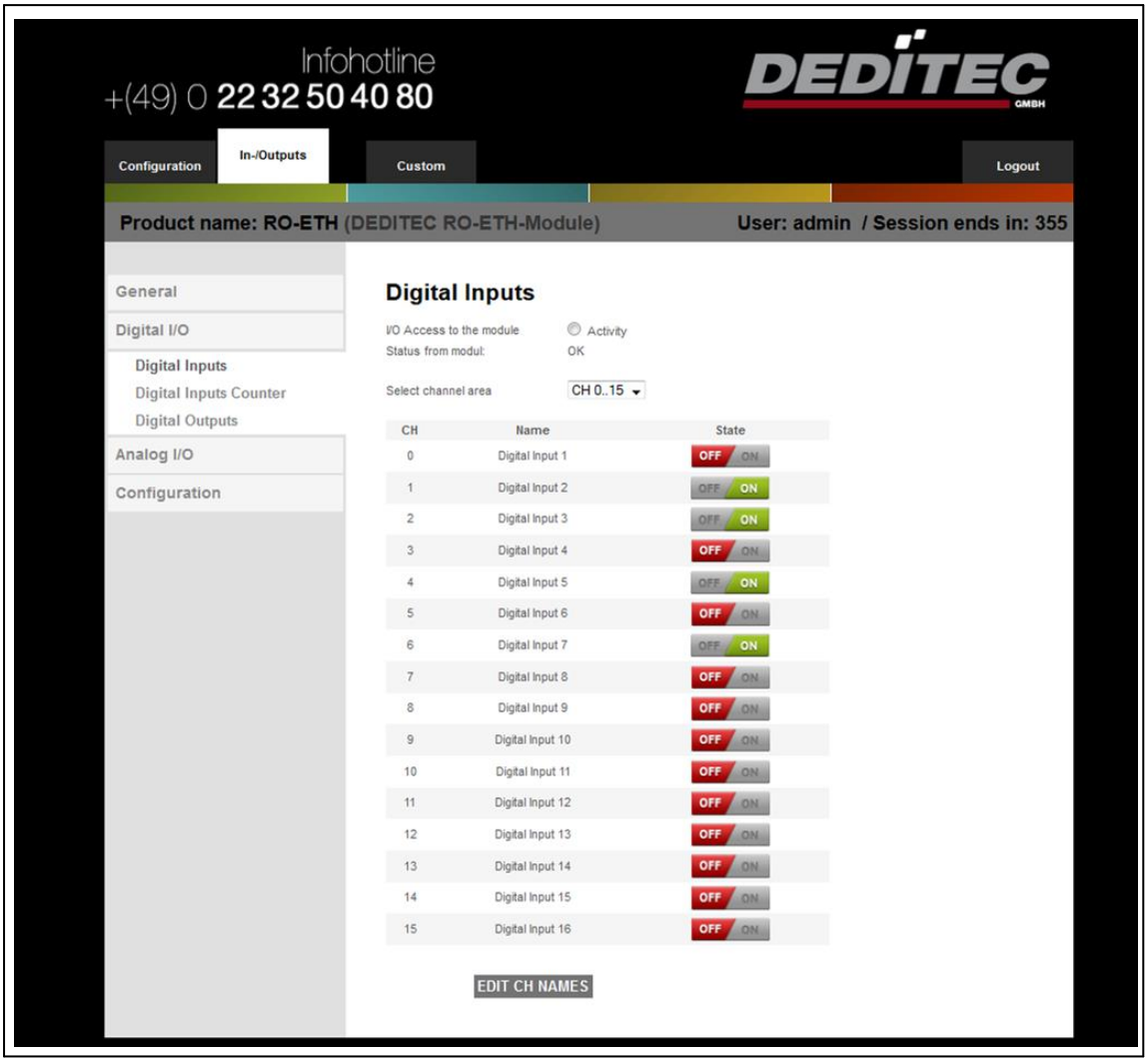
4.5.2. Inputs/outputs

4.5.2.1. General



Here you can see a list of the connected inputs/outputs

4.5.2.2. Digital inputs



The module's inputs are read at regular intervals on this page.

Select channel area

The displayed channel range (blocks of 16) can be defined here.

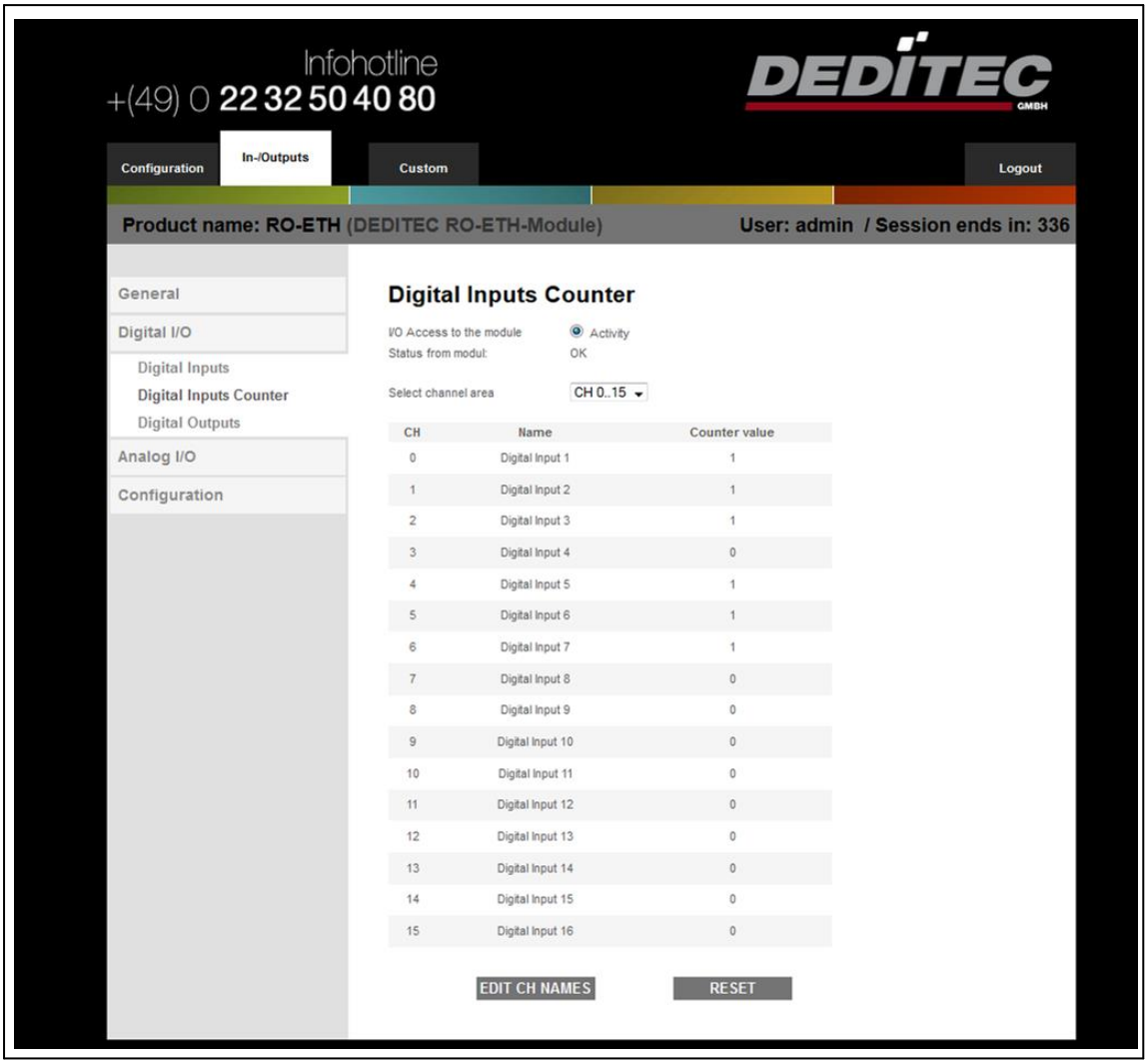
State

Status of the individual inputs

Edit CH names

For a better overview, a name can be assigned to each channel here.

4.5.2.3. Digital inputs Counter



The module's input counters are read at regular intervals on this page.

Select channel area

The displayed channel range (blocks of 16) can be defined here.

Counter value

Current status of the input counters

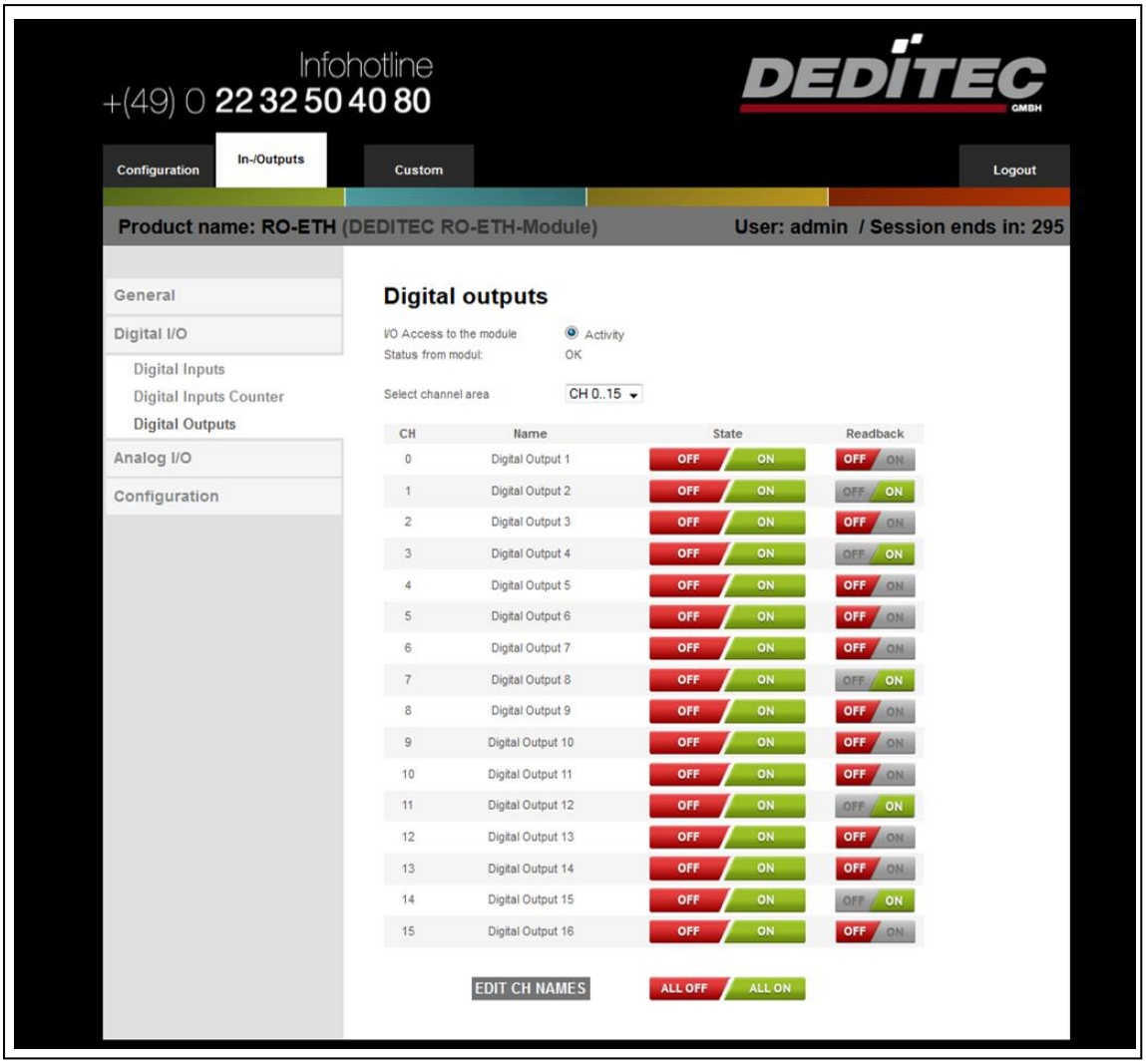
Edit CH names

For a better overview, a name can be assigned to each channel here.

Reset

Resets all input counters of the current channel range

4.5.2.4. Digital outputs



On this page, the module's outputs are read back at regular intervals. In addition, each channel individually or the current channel range can be switched on/off.

Select channel area

The displayed channel range (blocks of 16) can be defined here.

State

Switch this channel or all channels on/off

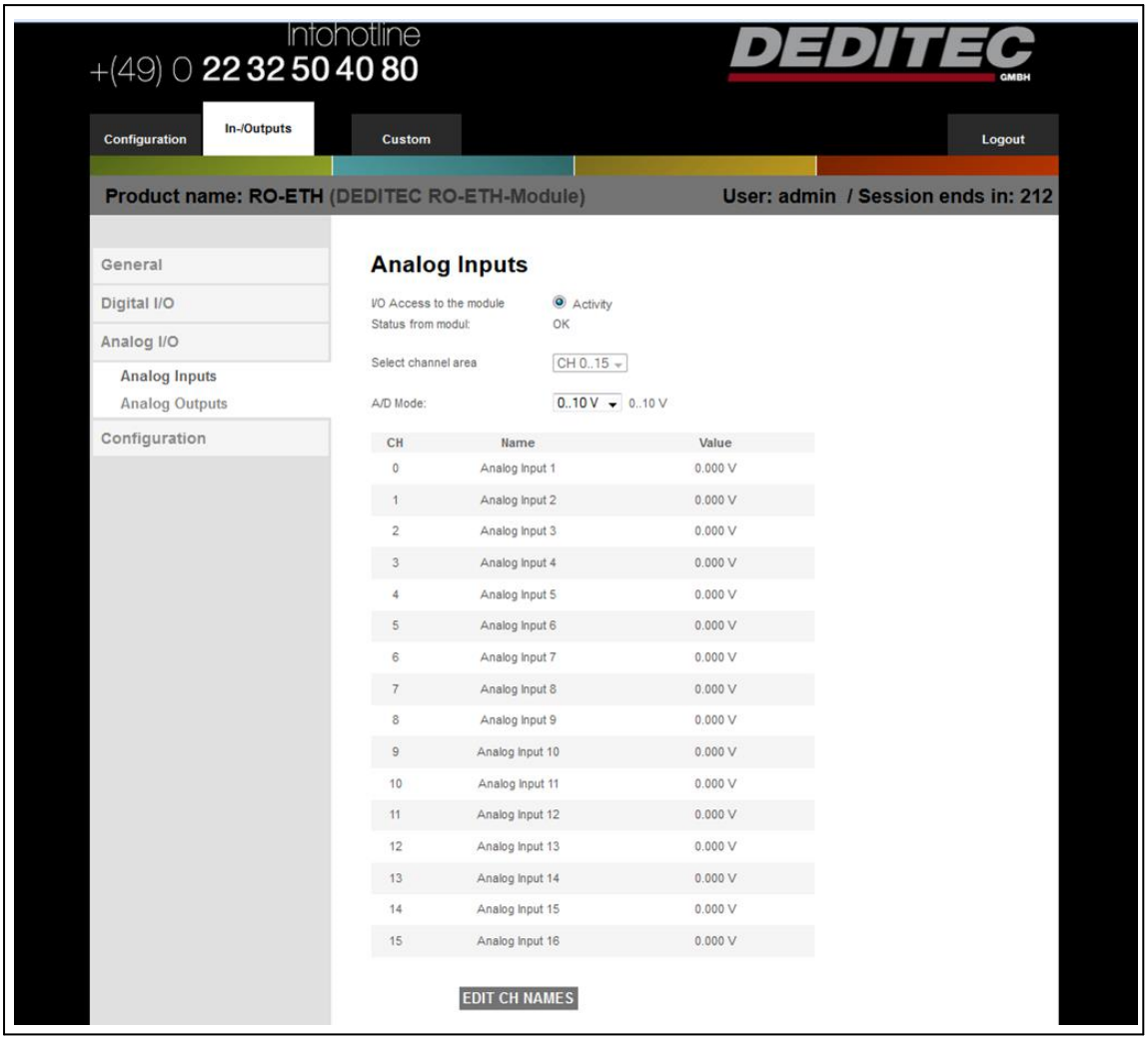
Readback

Current status of the output

Edit CH names

For a better overview, a name can be assigned to each channel here.

4.5.2.5. Analog inputs



On this page, the analog inputs of the module are read at regular intervals.

Select channel area

The displayed channel range (blocks of 16) can be defined here.

A/D Mode

Current A/D mode of the channel range.

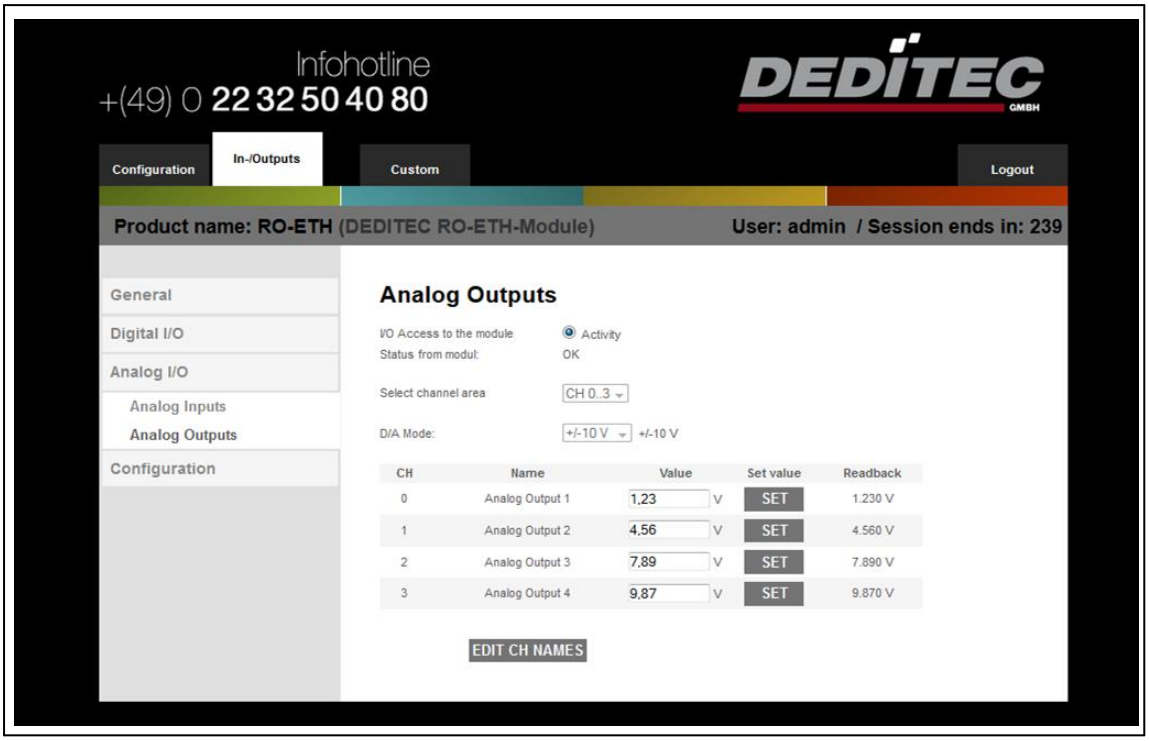
Edit CH names

For a better overview, a name can be assigned to each channel here.

Value

Current analog value.

4.5.2.6. Analog outputs



Analog outputs can be set on this page.

Select channel area

The displayed channel range (blocks of 16) can be defined here.

D/A Mode

Current D/A mode of the channel range.

Edit CH names

For a better overview, a name can be assigned to each channel here.

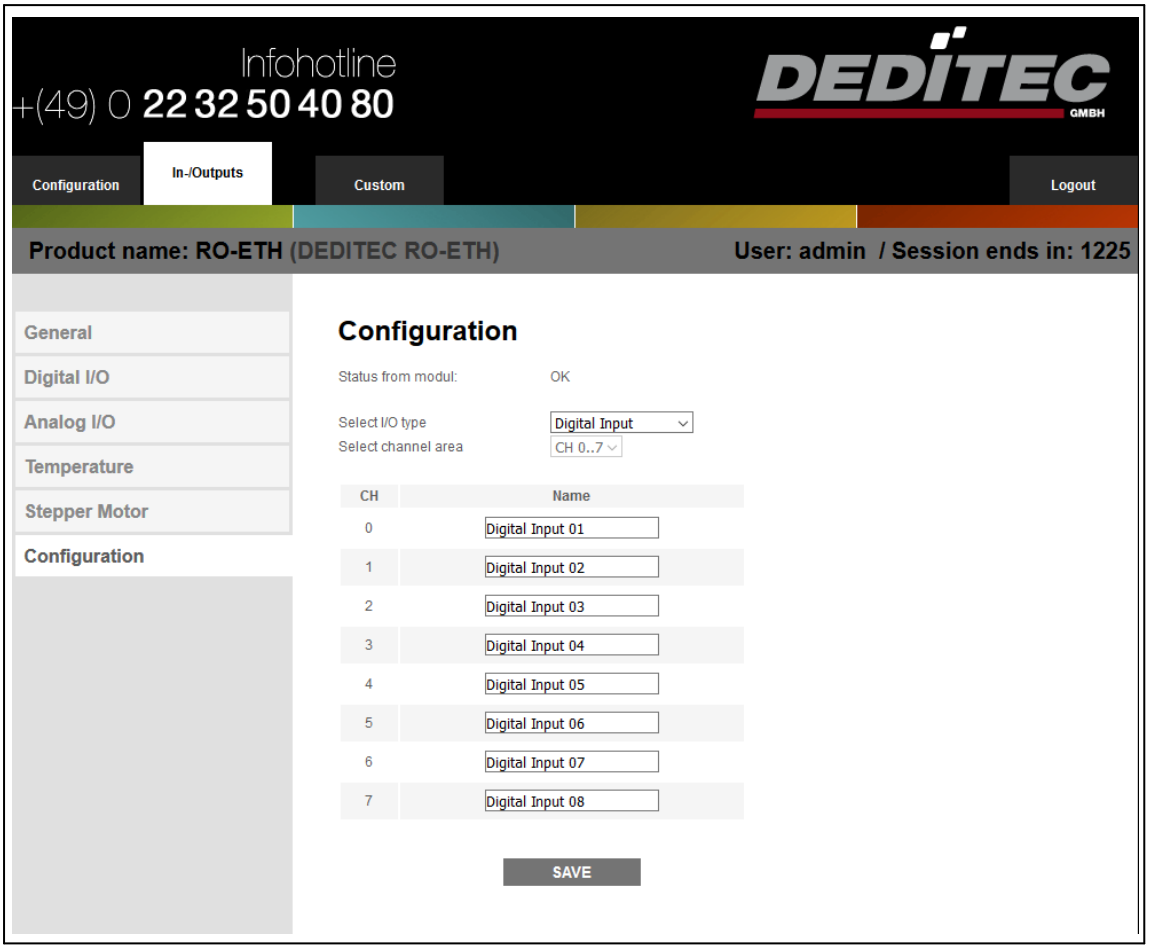
Value

Analog Value to be output.

Readback

Current analog value

4.5.2.7. Configuration



All names of the connected I/O units can be edited on this page.

Select I/O type

The I/O type can be selected here.

Select channel area

If more inputs/outputs are connected than can be displayed on this form, the channel range can be selected here.

Save

This is used to save the names for this channel range.

Note:

The channel name may be a maximum of 16 characters long.

ICT-Tool



5. ICT-Tool

The ICT-Tool gives you full control over your DEDITEC modules. With this all-in-one software,

you can flash, configure and test your module without additional software. This is particularly important for the initial commissioning of the module.

In the following chapters, the various areas of the ICT-Tool are described in more detail.

5.1. General descriptions

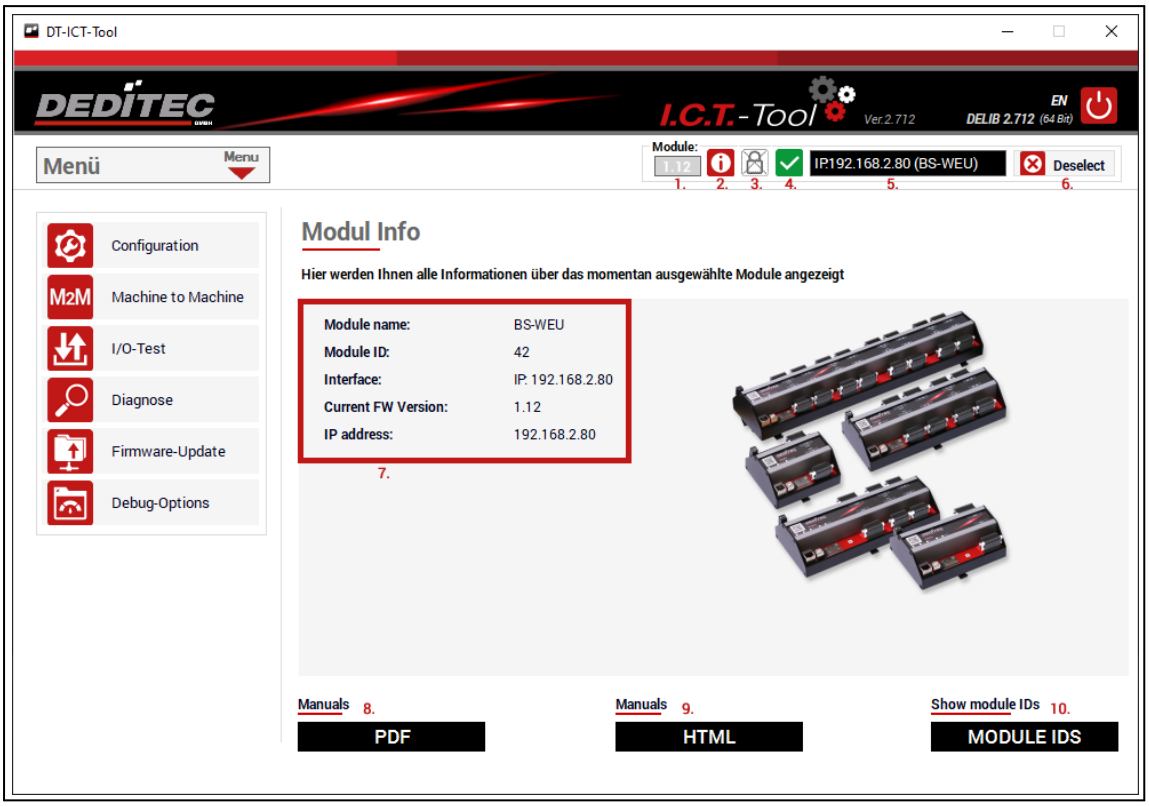
Description Program start:



1. Displays the name of the DEDITEC software used
2. Displays the currently used version number of the software
3. Displays the DELIB version currently in use
4. The list of added modules is updated here
5. Click on the EN or DE to change the language between English and German
6. Closes the program

7. In the Selectors module, you can switch between the individual areas of the software (a module must first be added)
8. You can add a module to the software using the + symbol. You can find more information on this in the chapter: **Add module**

Beschreibung bei erfolgreicher Modulverbindung



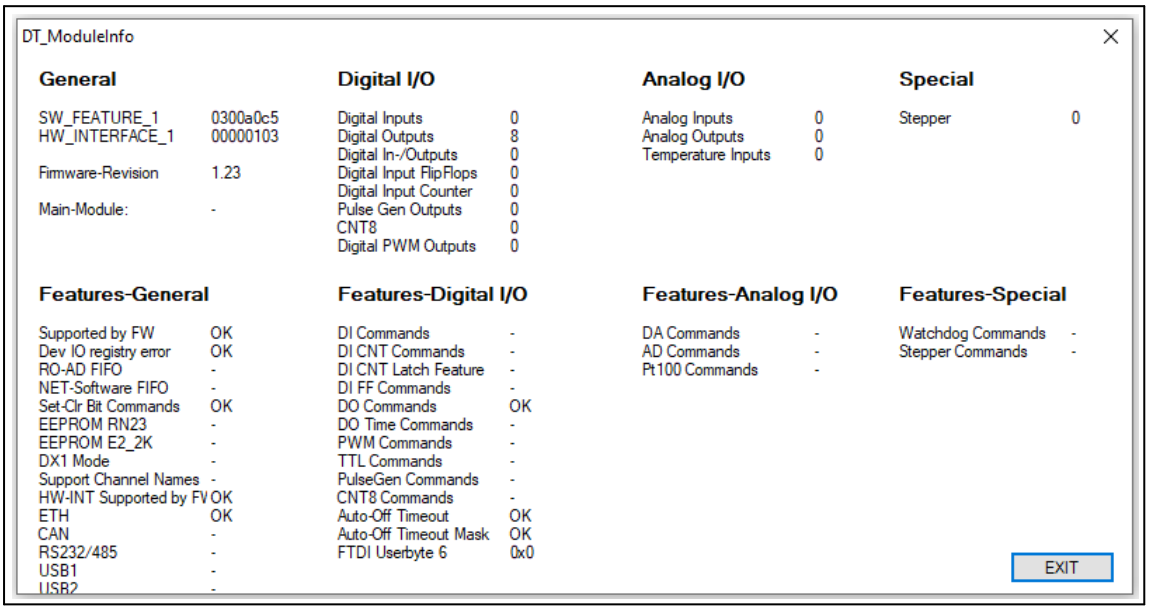
1. Displays the current firmware version of the interface module
2. Calls up the information window (see below). There you will find all I/O-specific information
3. Indicates whether the module is encrypted or unencrypted
4. Indicates whether there is a connection to the module
5. Displays the IP address of the module (via Ethernet) or the module number (via USB)
6. Closes the module and takes you back to the module selector
7. Displays the current network configuration, firmware and module ID of your module
8. Opens the manual for your module locally as a PDF
9. Opens the manual for your module in the browser as an HTML version
10. Shows you all available module IDs. This is required for programming your

own projects.

Information window

Depending on the connected module, information on the interface and submodules used is displayed here.

Among other things, you can see the number of connected inputs and outputs and which DEDITEC commands are supported.



In this example, a WEU-RELAIS-8 from our Startet series with 8 digital outputs was connected via Ethernet.

5.2. Add module

To be able to use our products with the ICT-Tool, they must be added to the start screen of the program.

USB modules

ModulesWith our modules with USB interface, this is done automatically via plug and play. The modules are displayed directly on the start screen and can be used immediately.

CAN modules

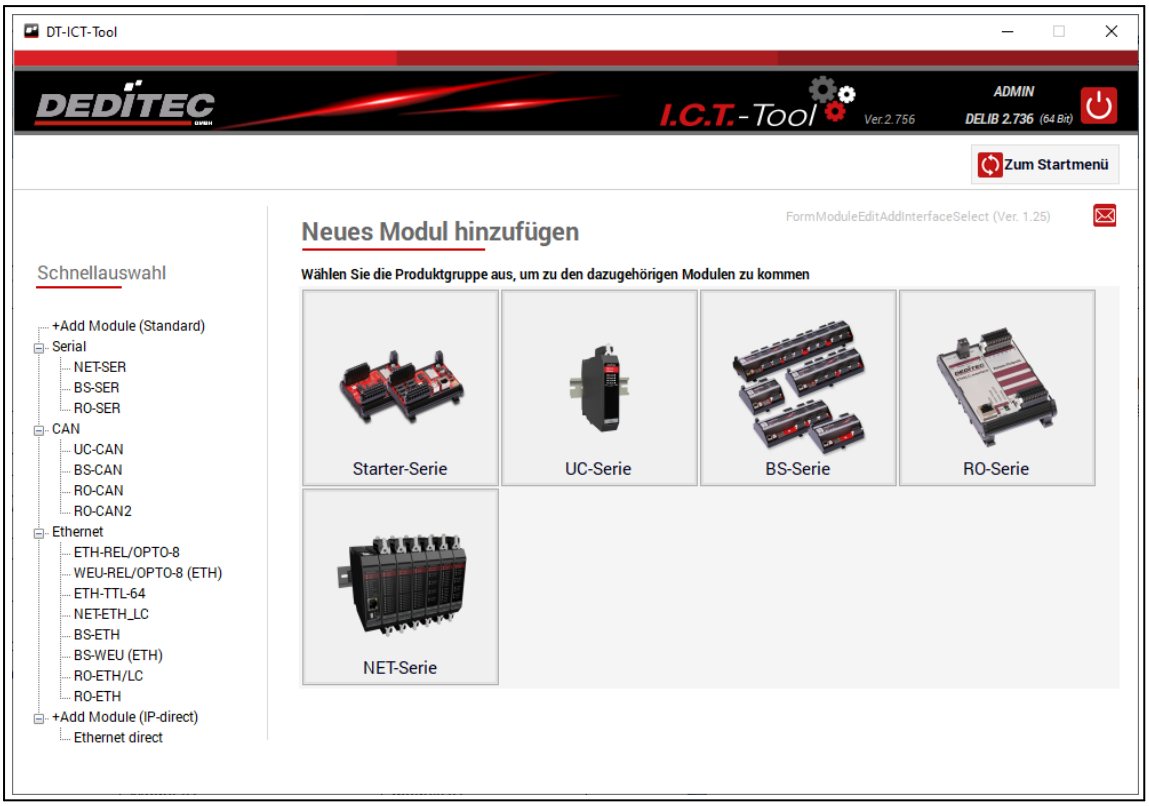
As our CAN modules are also configured via the USB interface, no further settings are necessary here either.

This does not apply to our older RO-CAN1 series modules, which are also integrated in the same way as our Ethernet and serial modules, as explained in the next step.

Ethernet and serial modules

Click on the + symbol to add our products with an Ethernet or serial interface and our modules from the old RO-CAN series to the ICT-Tool.

In the following window, you have the option of selecting the module directly using the quick selection on the left-hand side of the screen or you can navigate to the desired product using the series preview images.



In the following chapters you will find more detailed information on integrating and configuring the interface of your module.

5.2.1. USB Module

Our USB modules are connected via plug and play. The modules are automatically recognized and can be used immediately.

If there are several USB modules of the same series, the module number of the module must also be changed.

Further information on this can be found in the chapter: **USB configuration**



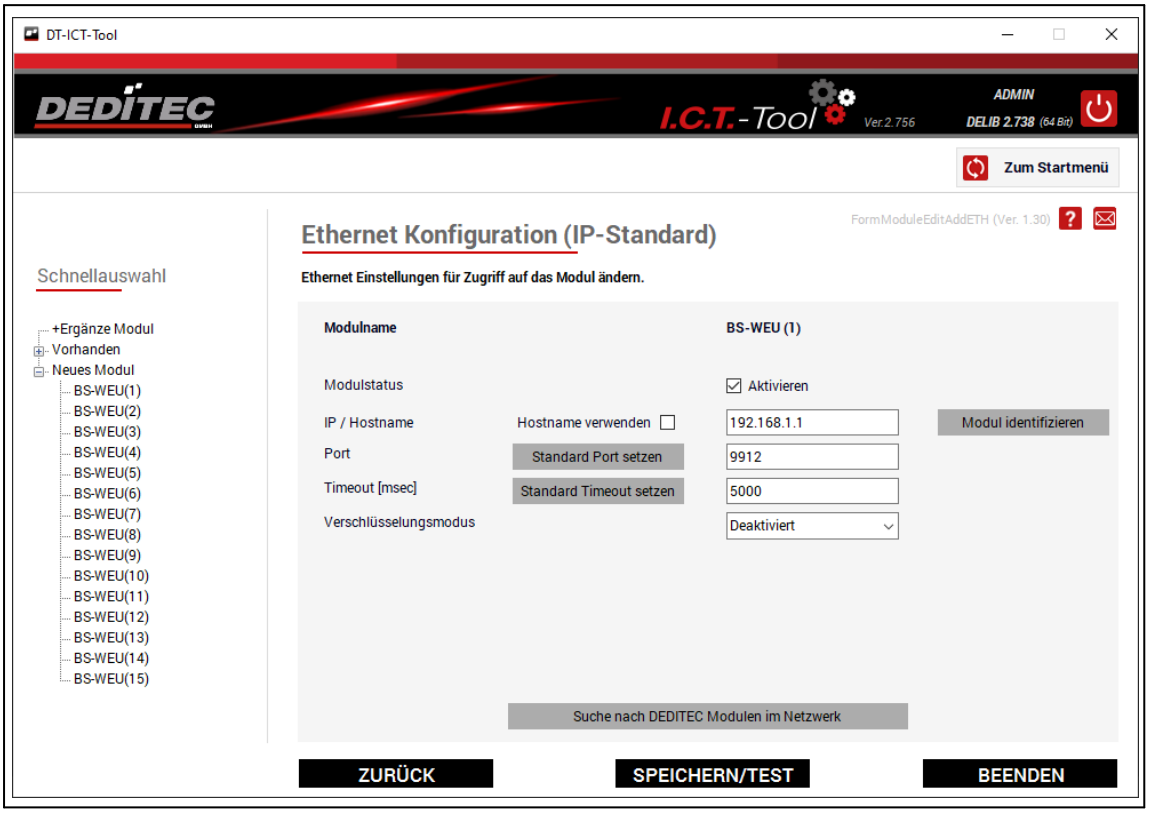
5.2.2. ETH Module

To add Ethernet modules, you have the option of doing this via a registry entry or directly via the IP address of the module. The following table illustrates the respective advantages and disadvantages.

	Ethernet Standard	Ethernet direct
General	• Administration rights may be required	• No Administration rights are required
Installation	• The required .DLL and driver files are installed using the DELIB setup	• No installation necessary. The corresponding DLL only needs to be in the program directory
Configure modules	• Simple configuration using the ICT tool • Module configurations are entered in the Windows registry	• Simple configuration using the ICT tool • Module configuration on the computer is not necessary, as the IP address is entered directly into the application software
Addressing the modules via your own software	• Simple integration into a project, as ModuleID and ModuleNR are used (DapiOpenModule) • Application software does not need to know the IP address, as this is stored on the computer in the registry. This makes it easier to distribute software to different computers	• DapiOpenModuleExt opens the module directly, specifying the IP address etc.
Third-party applications (e.g. LabView)	• DapiOpenModule command requires only 2 parameters, therefore easy to integrate	• is often more difficult in our experience

5.2.2.1. Ethernet standard method

Here you can add your module via a registry entry.



Quick selection

Available

Modules that have already been added are listed here. The number in brackets corresponds to the assigned module number

New module

All unused module numbers are listed here. The number in brackets corresponds to the assigned module number.

Configuration

Module status

With this option, you can add an ethernet module to the ICT tool. If the check mark is not set, the module is removed again.

IP/Hostname

You can enter the IP address of the module here. If the "use Hostname" checkbox is ticked, the configured hostname is used.

Port

The port number of the module is entered here. The "default port" button writes the default port "9912" in the field

Timeout

The timeout of the module can be set here. (milliseconds)

Encryption type

Here you can set whether the module should be used unencrypted or encrypted (User/Admin).

Search via Broadcast

Available modules in the network can be found here. More on this in the chapter: **Search via broadcast**

Identify Module

You can use the Identify function to visually check on the board which module is being addressed with the IP entered.

When starting the identification, the status LED of the respective module starts to flash very quickly. This is particularly helpful when using several

several modules.

SAVE/TEST

Here you can save the settings you have made and test communication with the module.

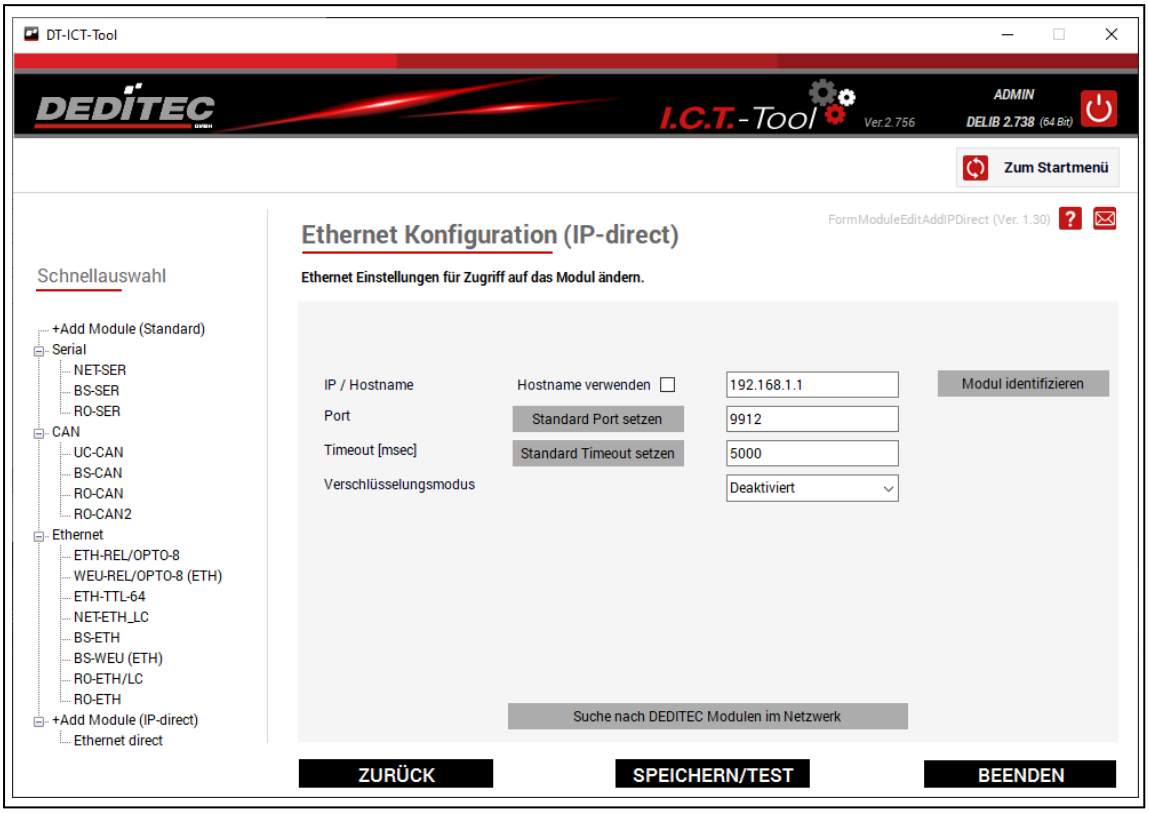
(Administration rights may be required)

EXIT

This takes you back to the module selector

5.2.2.2. Ethernet Direct method

Here you can insert your module directly via the IP address.



Configuration

IP/Hostname

You can enter the IP address of the module here. If the "use Hostname" checkbox is ticked, the configured hostname is used.

Port

The port number of the module is entered here. The "default port" button writes the default port "9912" in the field

Timeout

The timeout of the module can be set here. (milliseconds)

Encryption type

Here you can set whether the module should be used unencrypted or encrypted (User/Admin).

Search via Broadcast

Available modules in the network can be found here. More on this in the chapter: **Search via Broadcast**

Identify Module

You can use the Identify function to visually check on the board which module is being addressed with the IP entered.

When starting the identification, the status LED of the respective module starts to flash very quickly. This is particularly helpful when using several

several modules.

SAVE/TEST

Here you can save the settings you have made and test communication with the module.

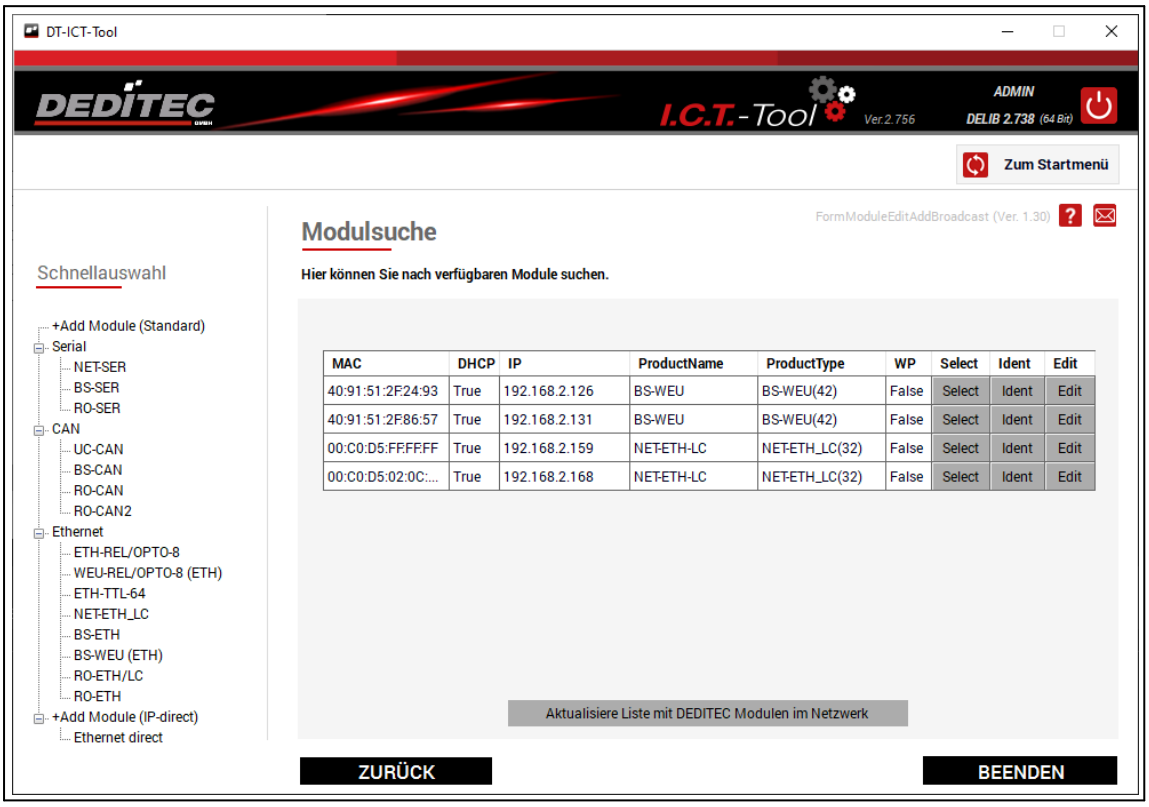
(Administration rights may be required)

EXIT

This takes you back to the module selector

5.2.2.3. Search via Broadcast

You can use the broadcast function to find and add modules that are connected to the network.



Select

This accepts the module's preconfigured network settings.

Ident

You can use the Identify function to visually check on the board which module is being addressed with the selected IP.

When starting the identification, the status LED of the respective module starts to flash very quickly. This is particularly helpful when using several

several modules.

Edit

Here you can directly change the network settings of the selected module and save them in the module.

5.2.3. CAN Module

CAN modules are connected to the PC via the USB interface for configuration.

Settings on the CAN interface and the TX and RX packets can be made in the configuration area.

Further information can be found in the chapter: **CAN configuration**

5.2.4. SER Module

Here you can assign serial modules to a free COM port.

The serial interface can be configured in the configuration area. You can also set there

whether you want to use the module via the serial interface with the ICT tool.

Further information on this can be found in the chapter: **Serial configuration**



Quick selection

Available

Modules that have already been added are listed here. The number in brackets corresponds to the assigned module number

New module

All unused module numbers are listed here. The number in brackets corresponds to the assigned module number.

Configuration

Module name

Displays the selected module

Module status

With this option, you can add a serial module to the ICT tool. If the check mark is not set, the module is removed again.

COM-Port

Here you can assign a free COM port to the module.

Communication attempts in the event of an error

Here you can set how often attempts should be made to access the interface in the event of an error before actions are terminated.

SAVE/TEST

Here you can save the settings you have made and test communication with the module.

(Administration rights may be required)

EXIT

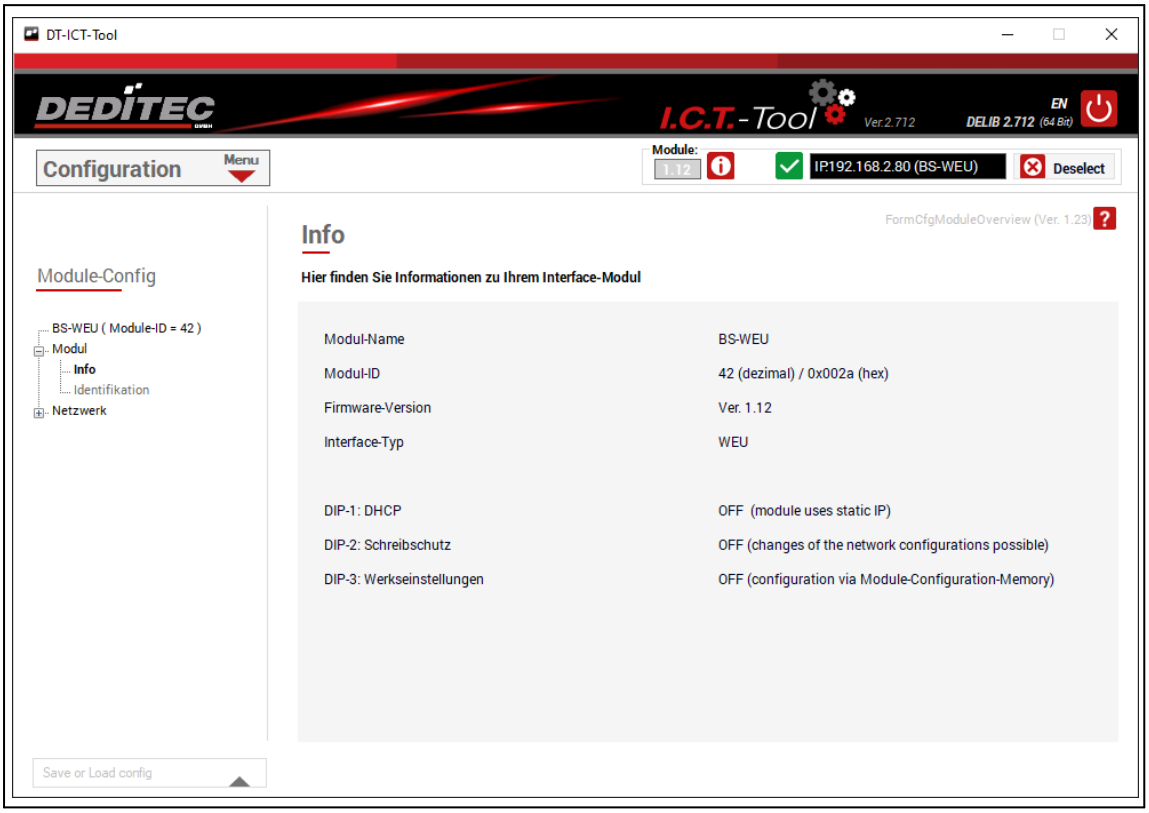
This takes you back to the module selector

5.3. Configure module

The module's configuration settings can be viewed or changed in the configuration area.

5.3.1. Module info

The ICT-Tool not only allows you to configure your module quickly and easily, you can also display a lot of important module information at a glance.



Module name

Displays the name of the DEDITEC module currently in use.

Module ID

Displays the ID of the module you are using. This is required for programming your own software with DELIB commands.

Firmware version

Displays the current firmware version installed on the module.

DIP switch positions

Displays the current DIP switch setting of your ETH interface.

5.3.2. Module identification

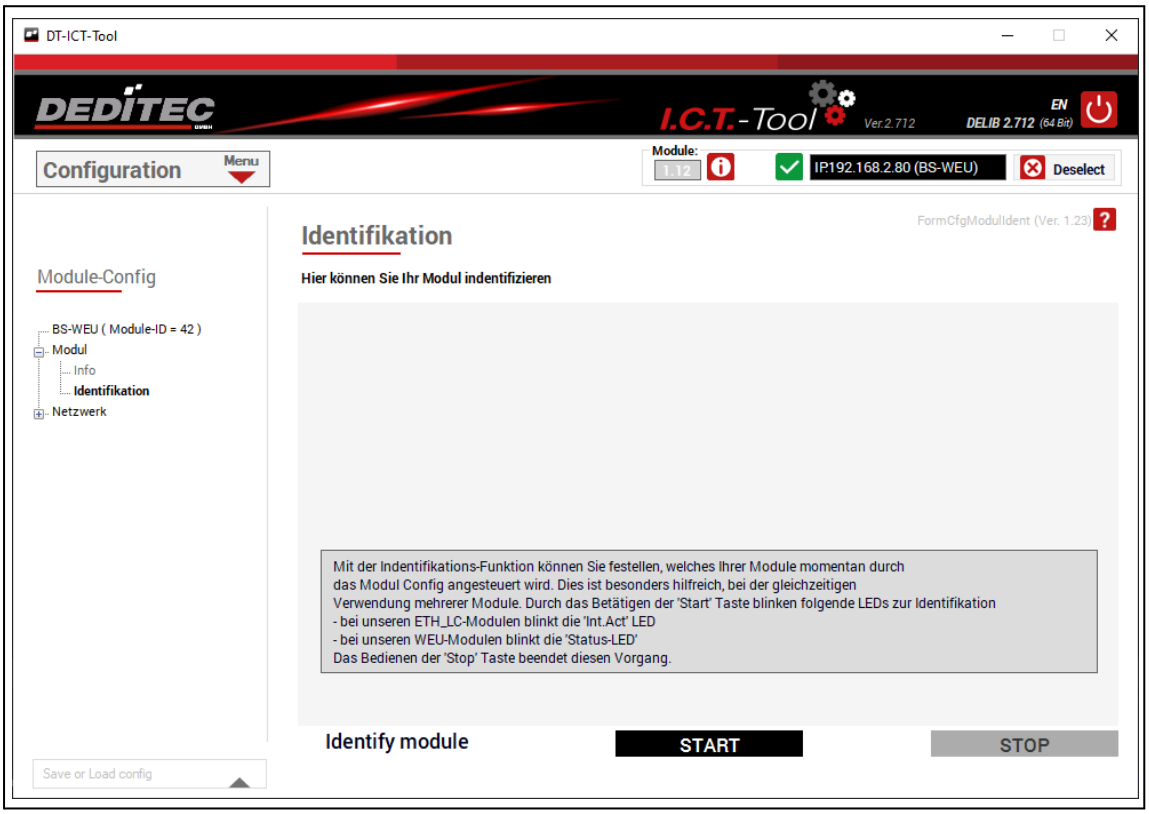
Identifies the module that you are currently addressing with the ICT-Tool to prevent confusion.

This is particularly helpful if several modules are in operation at the same time.

Identification is started by pressing the “Identify module” button.

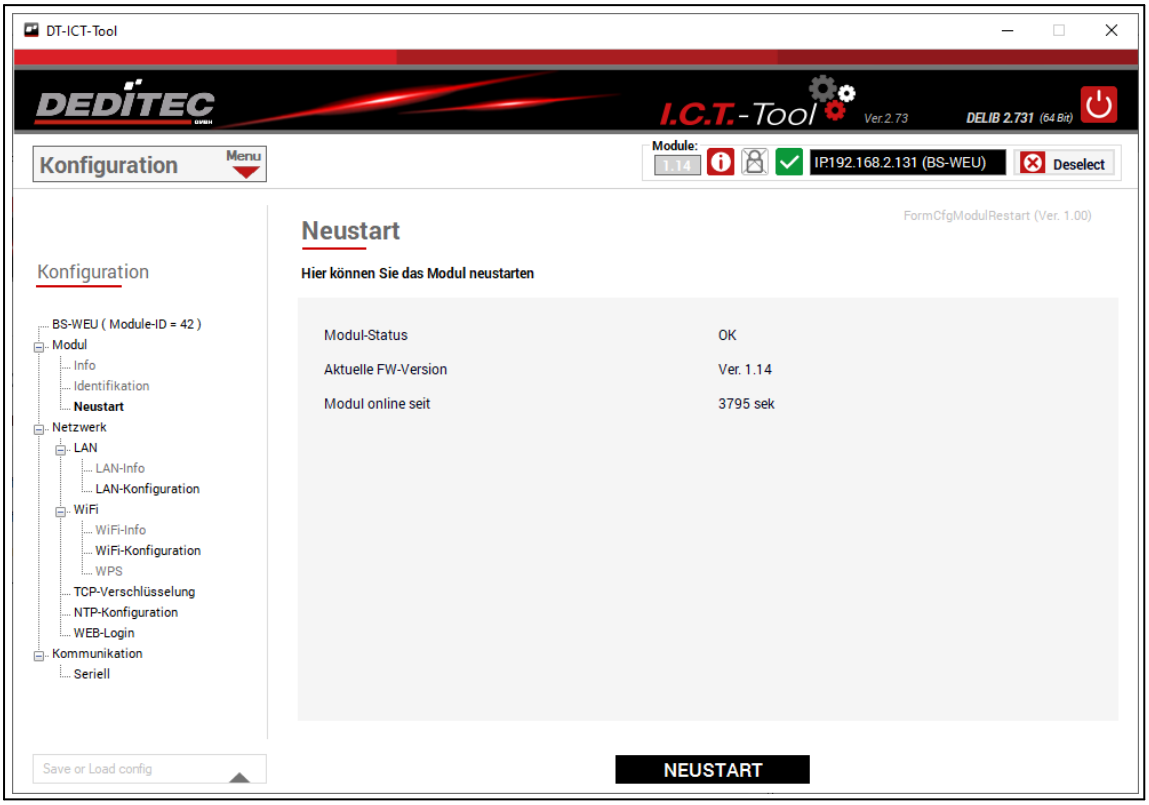
Depending on the module series, the following LEDs flash on your module:

- The “Int.Act” LED flashes on our ETH_LC modules
- The “Status” LED flashes on our WEU modules



5.3.3. Module restart

You can restart your module here.



Module status

Indicates whether the connection to the module has been successfully re-established after the restart.

OK: Connection was successfully established

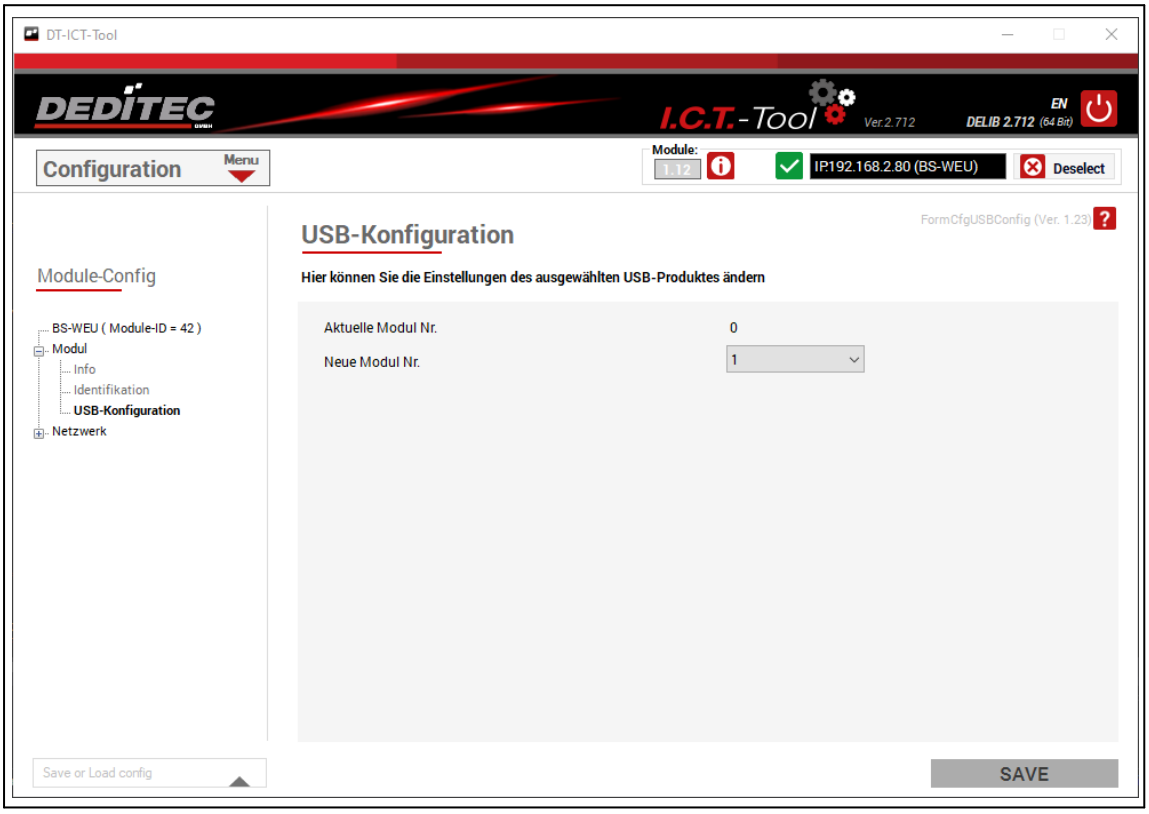
Error: Module not found after restart

5.3.4. USB configuration

You can change the module number of your USB module here.

This is necessary if you want to operate several modules of the same type.

To be able to open the correct module, the corresponding module number is required in addition to the module ID.



Current module no.

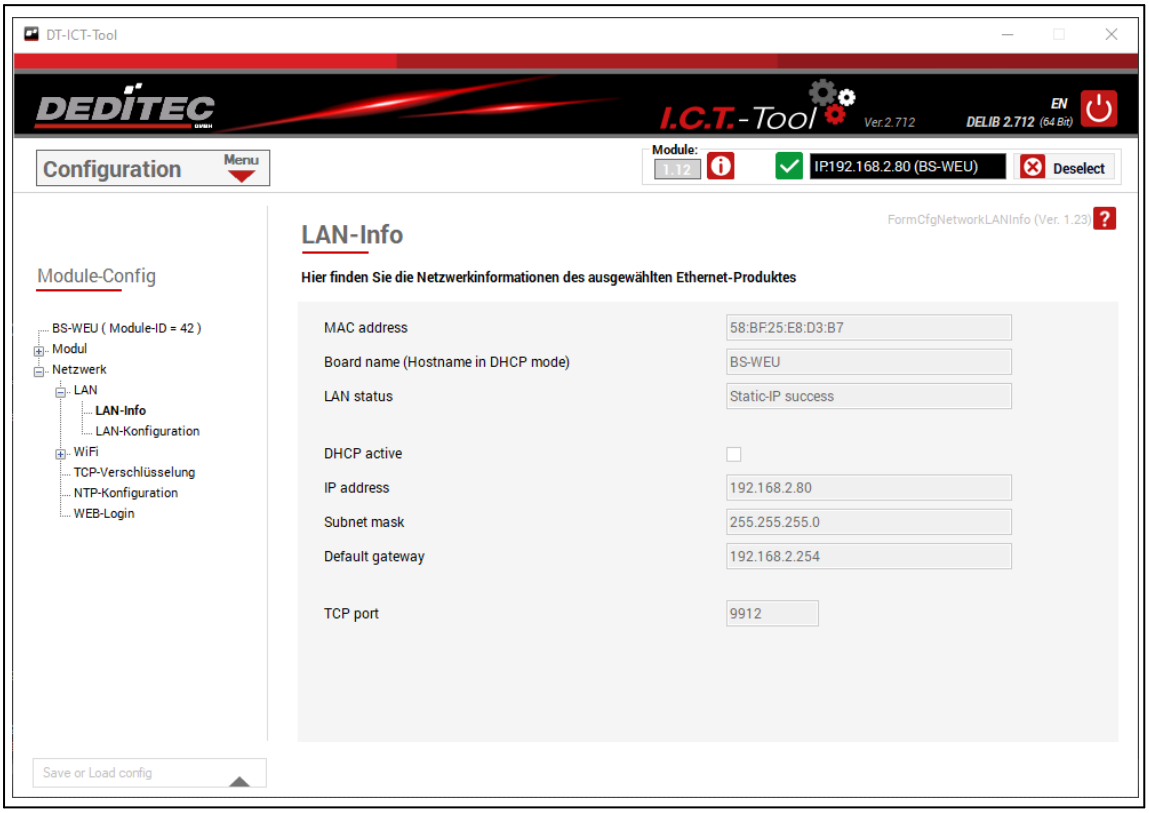
Shows which module number is currently assigned

New module no.

You can assign a new module number to the module here.

5.3.5. LAN network information

Here you will find all the important LAN network information for your module at a glance.



MAC address

The MAC address is the physical address of the product and is permanently linked to the hardware.

Board Name

Displays the current board name of your module.

LAN status

The connection status of your connected module is displayed here.

If the status "Query not supported (FW update)" is displayed, your module requires a more up-to-date firmware.

DHCP active

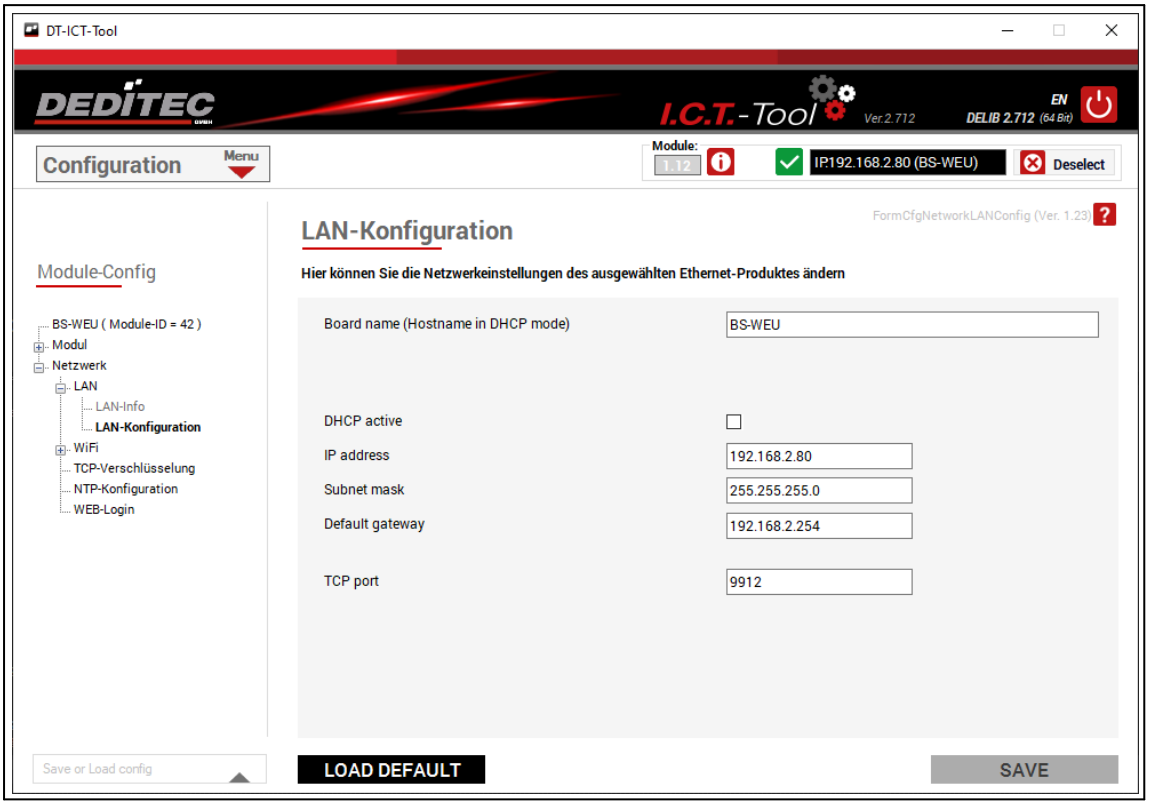
Indicates whether the module is connected via DHCP.

IP address, netmask, standard gateway and TCP port

Displays the current network configuration to which the module is connected.

5.3.6. LAN network settings

Here you can configure the LAN network settings of your module.



Board Name

The board name can be used for device identification. If DHCP is active, the board name is used as the host name.

This option is particularly useful when using several modules.

For example, you can assign a special board name such as "Garage" or "Gazebo" to a module. You can then control the module directly in the module selector under this name.

More information on connecting the module by board name

see chapter: **Using the module selector**

DHCP active

If this option is activated, the device attempts to obtain a valid IP address from a DHCP server in the network at startup.

The board name is used as the host name. If DHCP is activated via the DIP switch, this setting is ignored.

IP address, subnet mask, default gateway and TCP port

These settings are used if DHCP is deactivated.

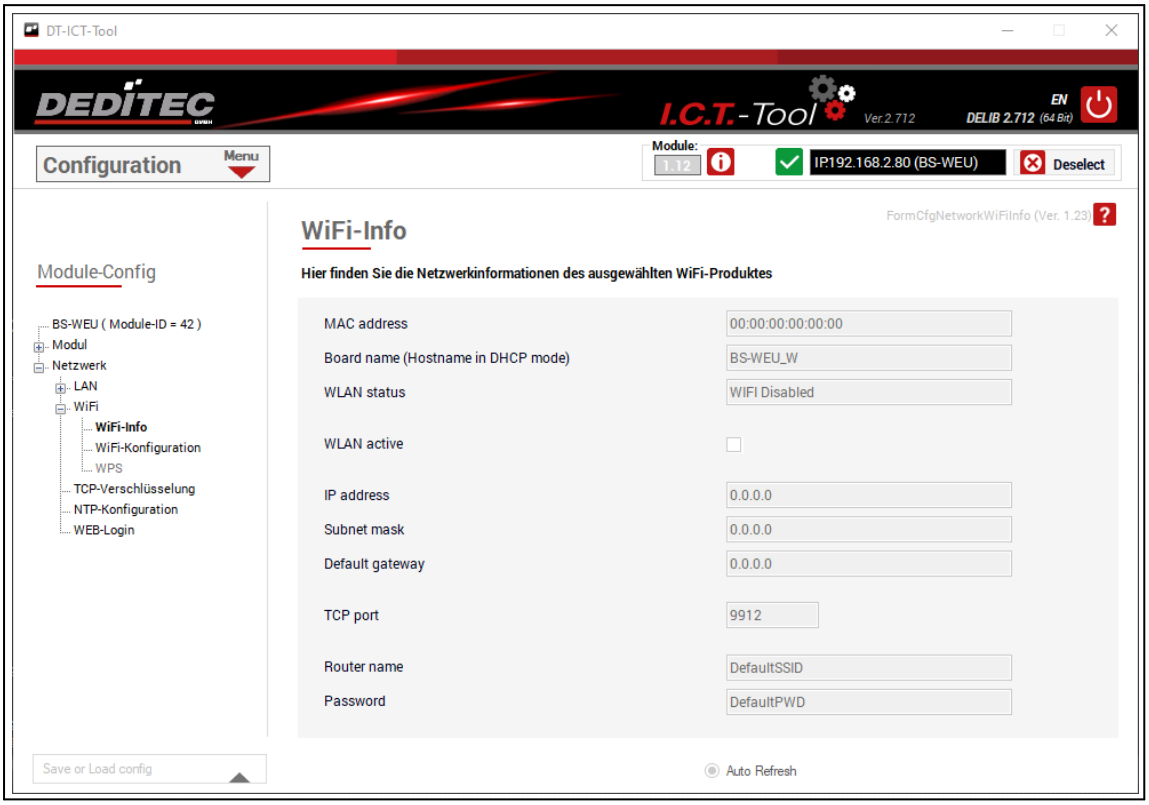
Load factory settings

The IP configuration is reset to the factory settings here. These look as follows:

Factory settings	
Board name	Module dependent
DHCP	off
IP-Adresse	192.168.1.1
Subnet mask	255.255.255.0
Default gateway	192.168.1.254
TCP Port	9912

5.3.7. WiFi network information

Here you will find all the important WiFi network information for your module at a glance.



MAC address

The MAC address is the physical address of the product and is permanently linked to the hardware.

Board Name

Displays the current board name of your module.

WLAN status

The connection status of your connected module is displayed here.

If the status "Query not supported (FW update)" is displayed, your module requires a more up-to-date firmware.

WLAN active

Indicates whether the module is connected via WLAN.

IP address, netmask, standard gateway and TCP port

Displays the current network configuration to which the module is connected.

Router name

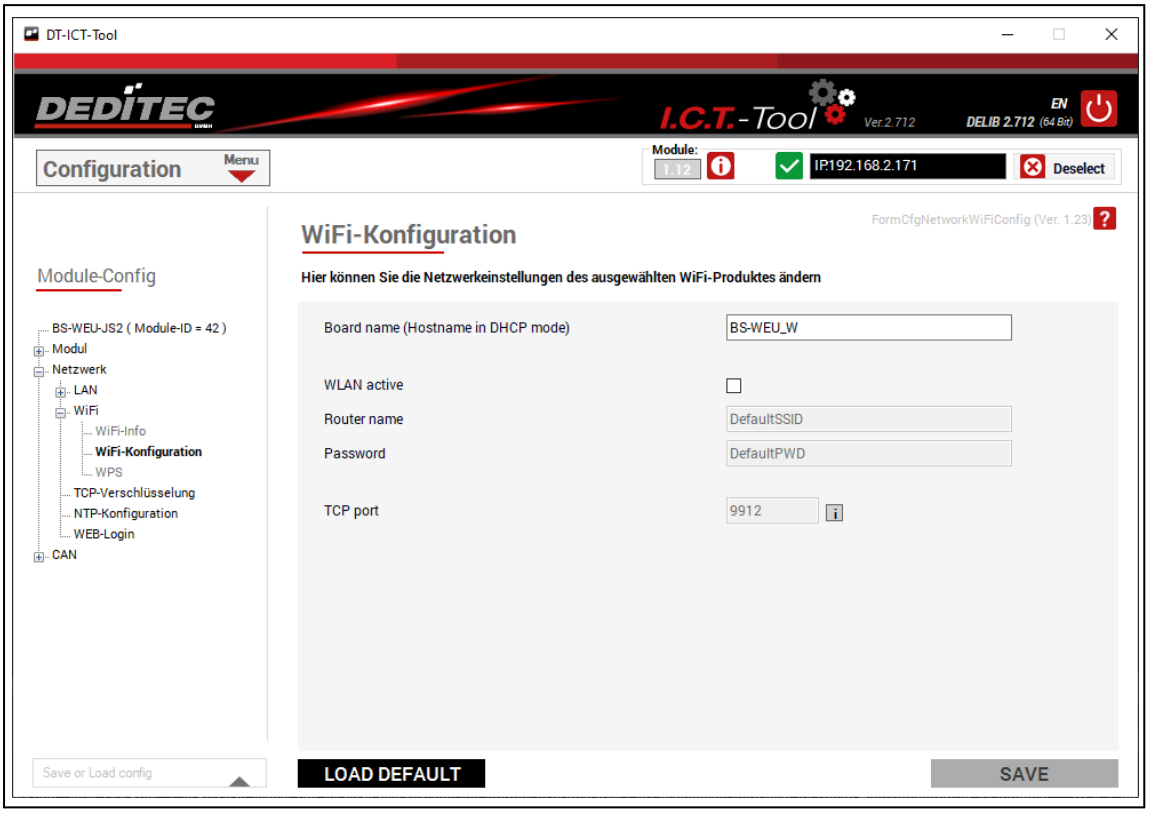
Shows which router name is used to connect via WLAN.

Router password

Displays the router password used.

5.3.8. WiFi network settings

Here you can make changes to the WiFi settings of your module.



Board name

The board name can be used for device identification. If DHCP is active, the board name is used as the host name.

This option is particularly useful when using several modules.

For example, you can assign a special board name such as "Garage" or "Gazebo" to a module. You can then control the module directly in the module selector under this name.

For more information on connecting the module by board name, see chapter: **Using the module selector**

WLAN active

You can use this option to activate or deactivate your module's WLAN.

Router name

Here you can enter the router name to be used for a connection via WLAN.

Router password

You can enter the router password of the router used here.

If necessary, ask your system administrator.

TCP port

The TCP port used is shown here. The port can only be changed for LAN network configurations.

Load factory settings

The WiFi configuration is reset to the factory settings here. These look as follows:

Factory settings	
Board name	Module dependent
WLAN active	off
Router name	DefaultSSID
Password	DefaultPWD

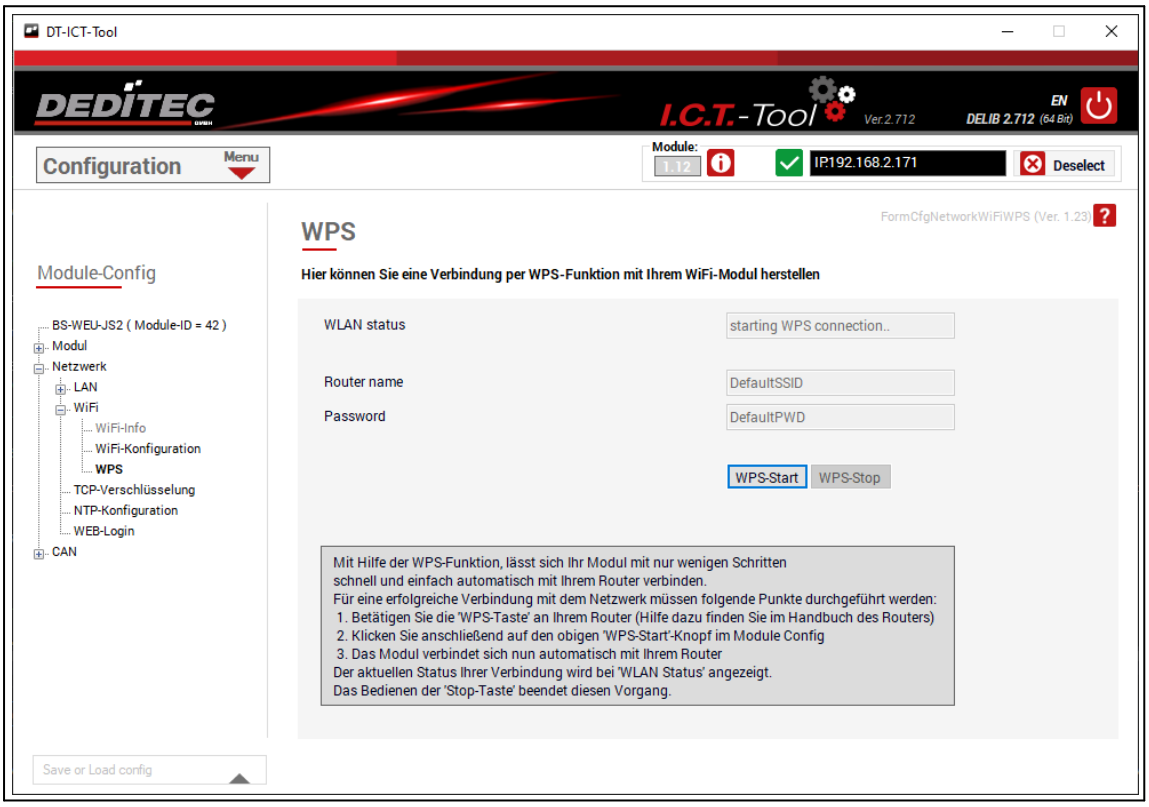
5.3.9. WiFi WPS connection

Here you can connect your module to your PC network using the WPS function.

To do this, press the “WPS Start” button in the ICT-Tool (see image below) **or** press and hold the Config button on your module for 7 seconds.

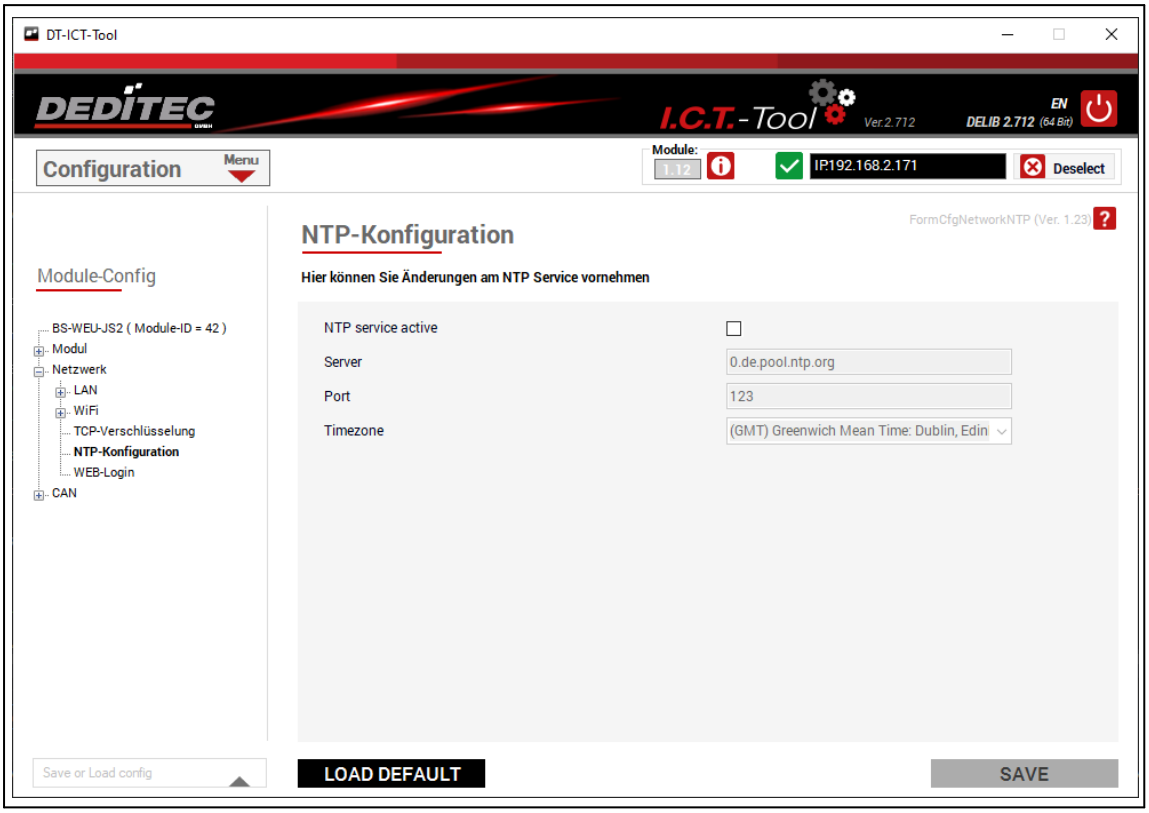
The WPS status should now change to “starting WPS connection”. Now start the WPS search on your router.

If the connection is successful, the WLAN status changes to “DHCP is ON”. The router name and password used will also appear.



5.3.10. NTP configuration

Changes to the NTP service can be made here.



NTP service active

If this option is activated, the NTP service is activated.

Server

Here you can set the NTP server to be used.

Port

Here you can set the NTP port to be used.

Timezone

Here you can set the time zone to be used by the module.

Load factory settings

The TCP encryption settings are reset to the factory settings here. These look as follows:

Factory settings	
NTP service active	on
Server	0.de.pool.ntp.org
Port	123
Timezone	(GMT) Greenwich Mean Time: Dublin, Edinburgh, Lisbon, London

5.3.11. TCP encryption

1.Access and recovery options

To ensure tap-proof TCP communication between the user and the module, this can be encrypted using a password. Encryption takes place automatically in delib.dll.

The user's entries are transmitted in encrypted form, decrypted by the module itself and then executed.

In order to also write the password securely to the module, in addition to direct configuration via USB, it is also possible to encrypt the password in the ICT tool via Ethernet using the temporary admin mode.

When configuring with the temporary admin mode, switches or buttons must be physically operated directly on the board to ensure that no unauthorized person can carry out this configuration.

The following encryption options are available:

Encryption type	Meaning
unencrypted	The module can be accessed unencrypted
user encryption	• TCP transmission is encrypted • The I/Os can be accessed, but no changes can be made to the module configuration
admin encryption	• TCP transmission is encrypted • Full access to I/Os and module configuration
temp admin	Temporary encryption mode to transfer the set password to the module in encrypted form

Attention: If the options "Allow unencrypted protocol" and "Allow admin-

encrypted protocol" are deactivated, you can no longer make any settings to the module configuration.

If you no longer have access to the module, you can change the encryption settings as follows:

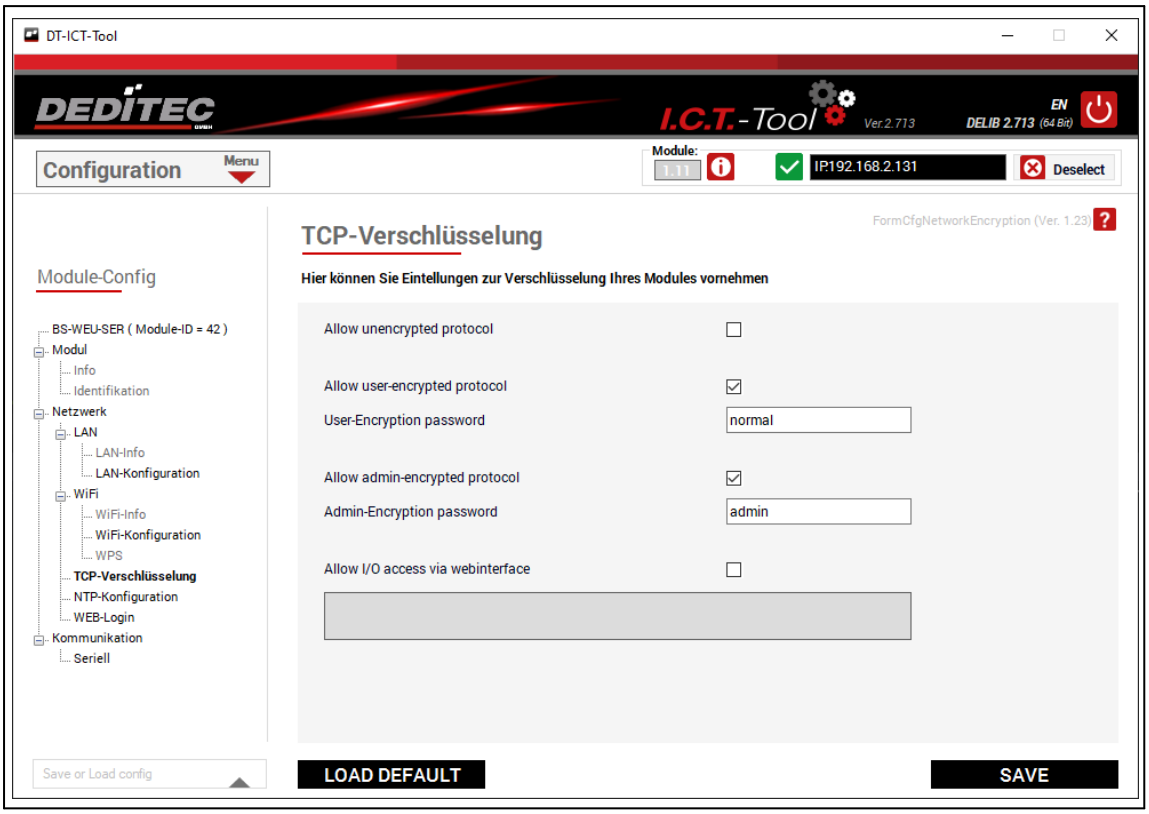
a) via USB: For modules with USB and Ethernert interface, you can make settings to the module configuration at any time via USB cable.

b) via web interface: You can make and save encryption settings via the web interface.

c) via DIP switch: With DIP-3 you can activate the factory settings of the module. This allows you to make and save changes to the encryption settings.

Remark

Please note that changes to the DIP switches only take effect after the module has been restarted.



2. General information

Allow unencrypted protocol

This configuration is used to activate or deactivate unencrypted access.

on = allows unencrypted access

off = only encrypted access is permitted

Allow user-encrypted protocol

With this option, you allow access to the module via user protocol.

Users who have this access can only operate the I/Os of the module.

Changes to the module configuration, e.g. LAN configuration, cannot be made.

A user password is required for this option.

Allow admin-encrypted protocol

With this option, you allow access to the module via admin protocol.

Users who have this access can change the module configuration and operate the I/Os of the module.

An admin password is required for this option.

Allow I/O access via webinterface

Use this option to activate or deactivate access to the module via the web interface.

3. setting the encryption

Select one or more encryption options as required and, if necessary, assign a secure password.

In the image above, it has been specified that the module can be opened either via user or admin access.

The unencrypted and web interface access have been deactivated.

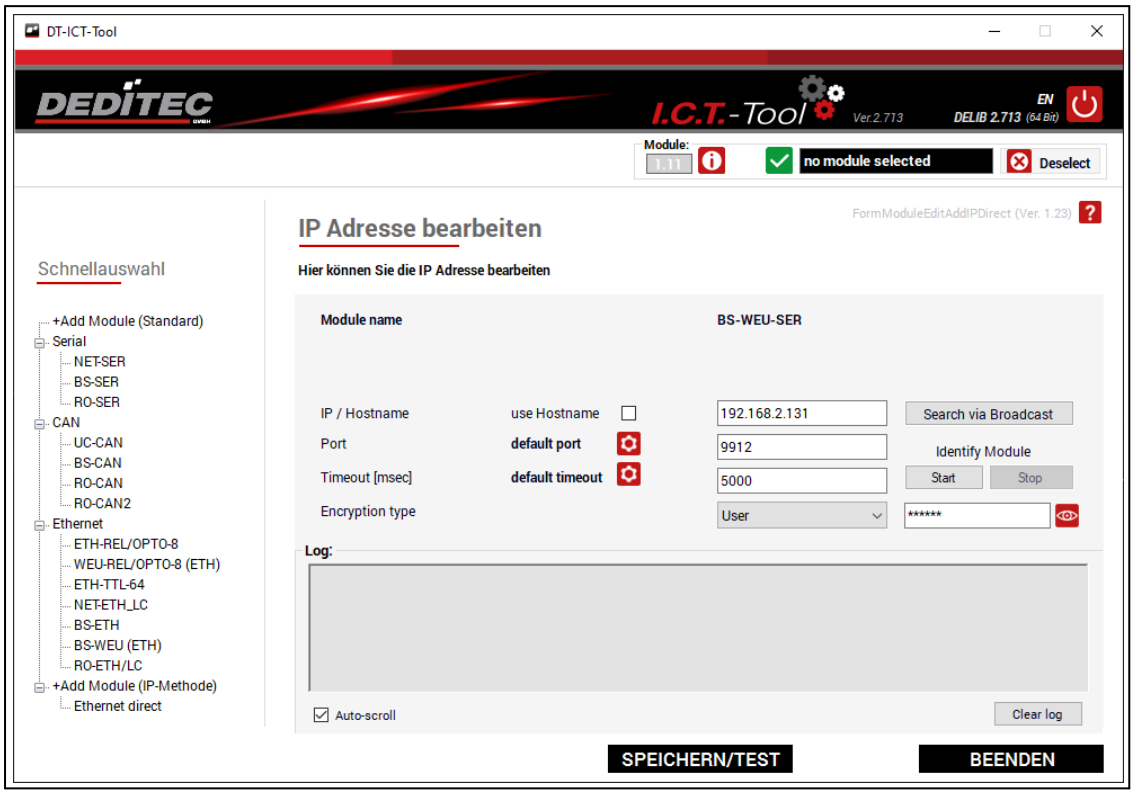
Next, go back to the module selection and click on the cogwheel of the now encrypted module.

Using the "Encryption type" option, you can now specify whether the module should be opened as User or Admin.

Enter the previously defined password in the text field to the right and save the change.

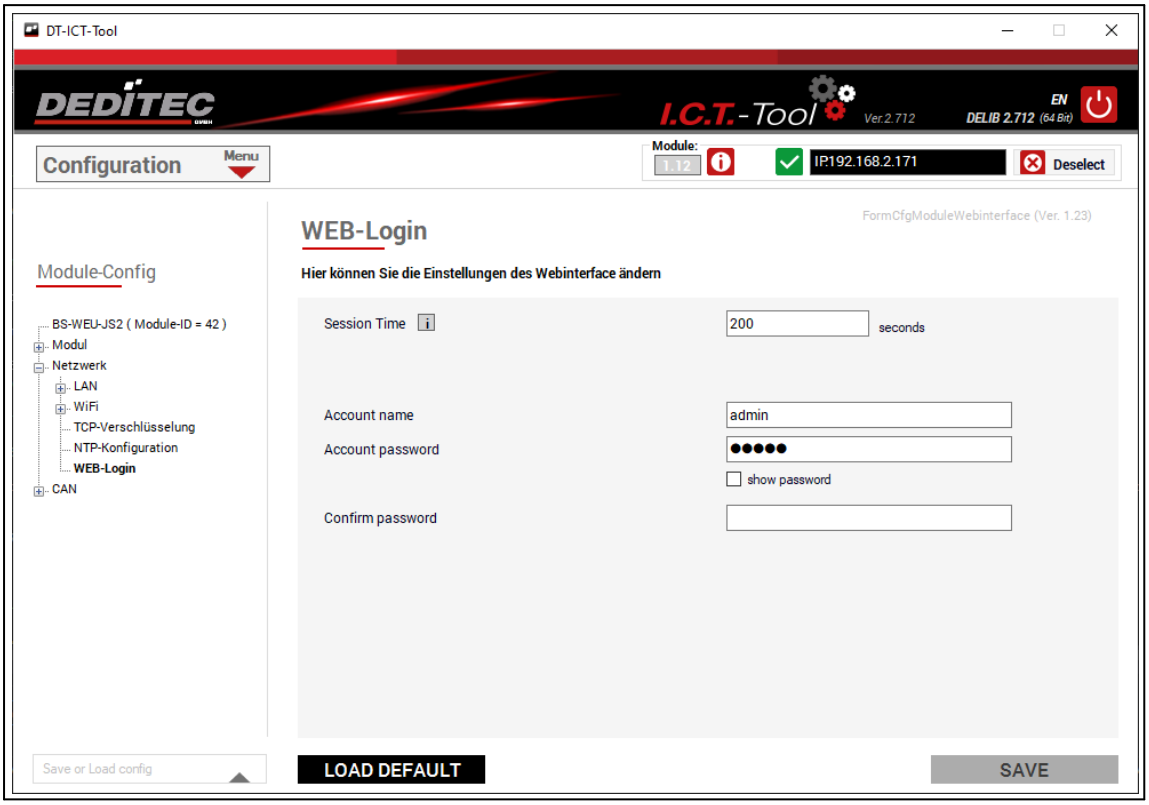
The module now starts with the defined access rights.

The image below shows that the module should be opened with user access.



5.3.12. WEB-Login

Here you can make settings on the module's web interface.



Session Time

Specifies the number of seconds of inactivity on the web interface after which the user is automatically logged out.

Setting from 10 - 65535 seconds possible.

Account name

Specifies the user name for logging in to the web interface.

Please note that it is case-sensitive.

Account password

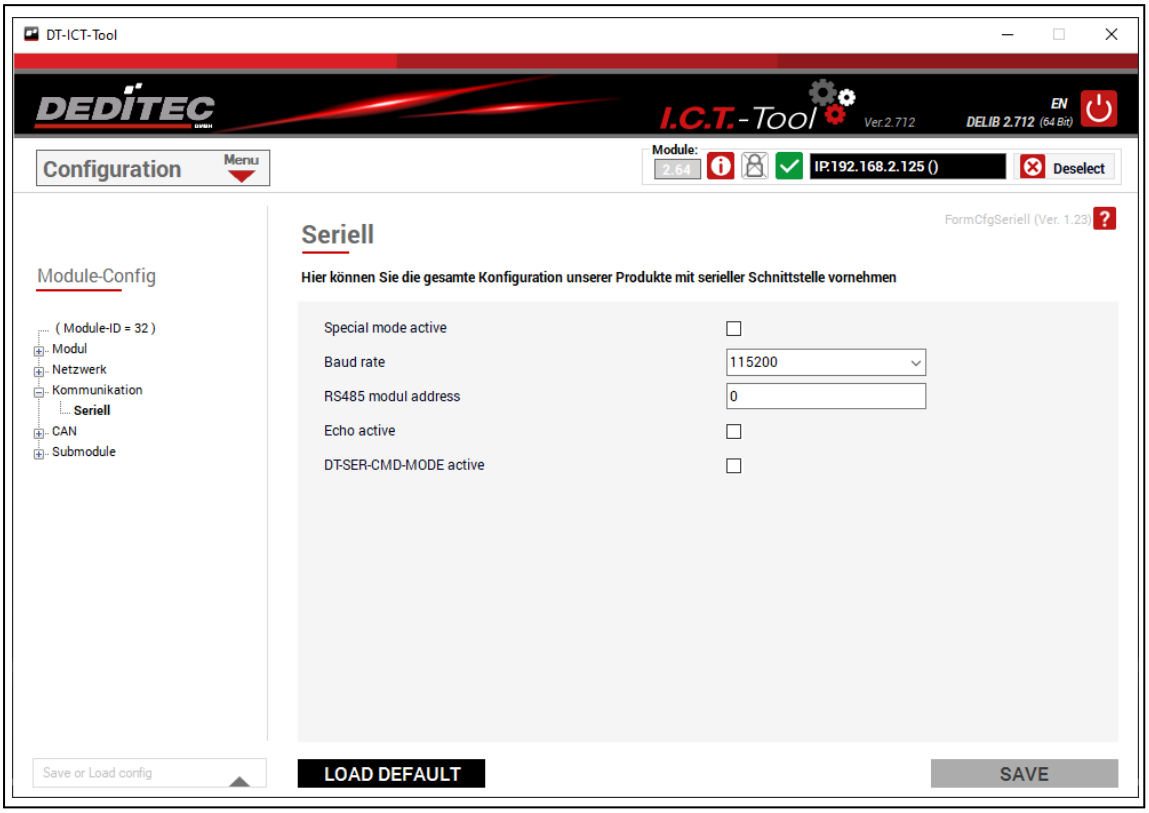
Specifies the password for logging in to the web interface.

For security reasons, the password must also be confirmed by re-entering it.

Please note that the password is case-sensitive.

5.3.13. Serial configuration

Here you can carry out the entire configuration of our products with serial interface.



Preferred mode (Special mode)

In preferred mode, the module is automatically operated with the following settings:

Baud rate: 115200

Modul-Nr. 0

Echo = Off

Register-Mode = On

Baud rate

If the preferred mode is deactivated, the communication speed can be set.

625000

250000

125000

115200

57600

50000

38400

19200

9600

4800

2400

1200

600

300

RS485 modul address

Address for identification in the RS485 bus.

Echo

Characters received serially are sent back by the module.

DT-SER-CMD-MODE

Deactivate the register mode to activate the text mode.

This option must be activated in order to access the ICT tool via the serial interface.

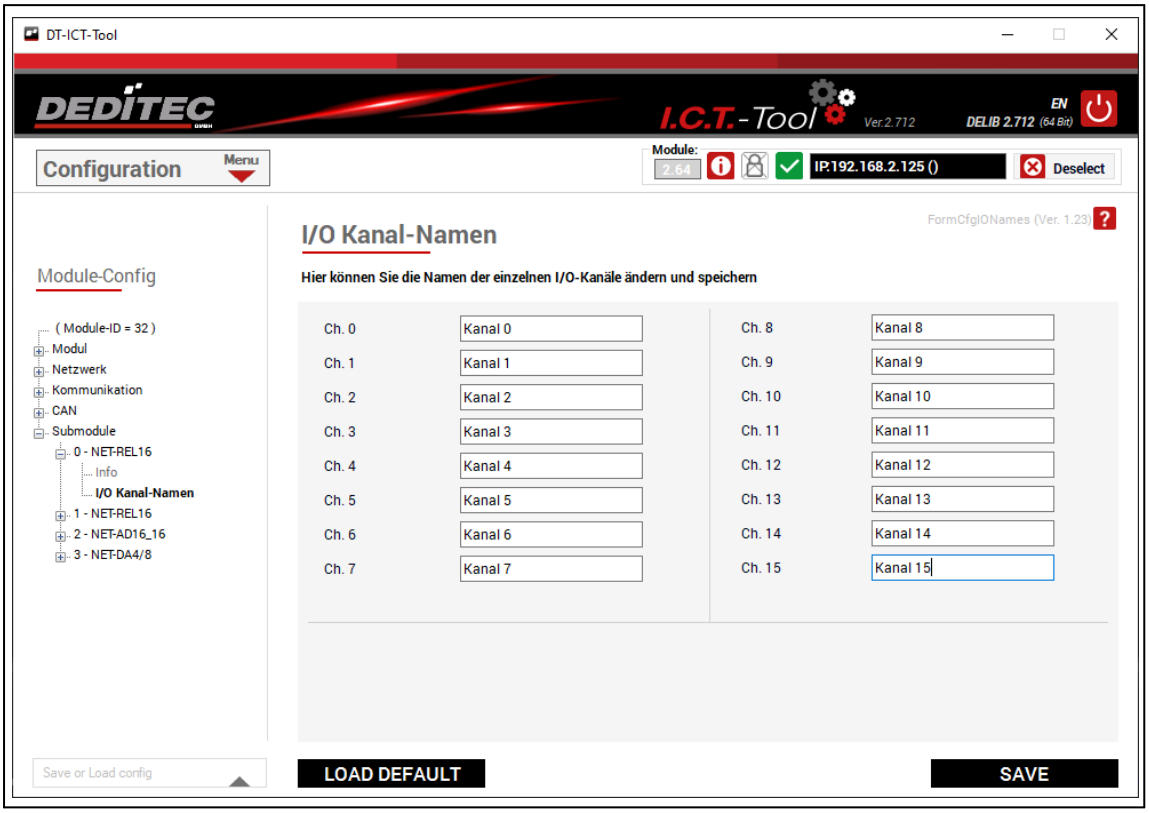
Load factory settings

Hier wird die serielle Konfiguration auf die Werkseinstellungen zurückgesetzt. Diese sehen wie folgt aus:

Factory settings	
Special mode activate	OFF
Baud rate	115200
RS485 module adress	0
Echo activate	OFF
DT-SER-CMD-MODE activate	OFF

5.3.14. I/O channel names

You can set the channel names of your main module or submodule here.



Note:

The channel name may be a maximum of 16 characters long.

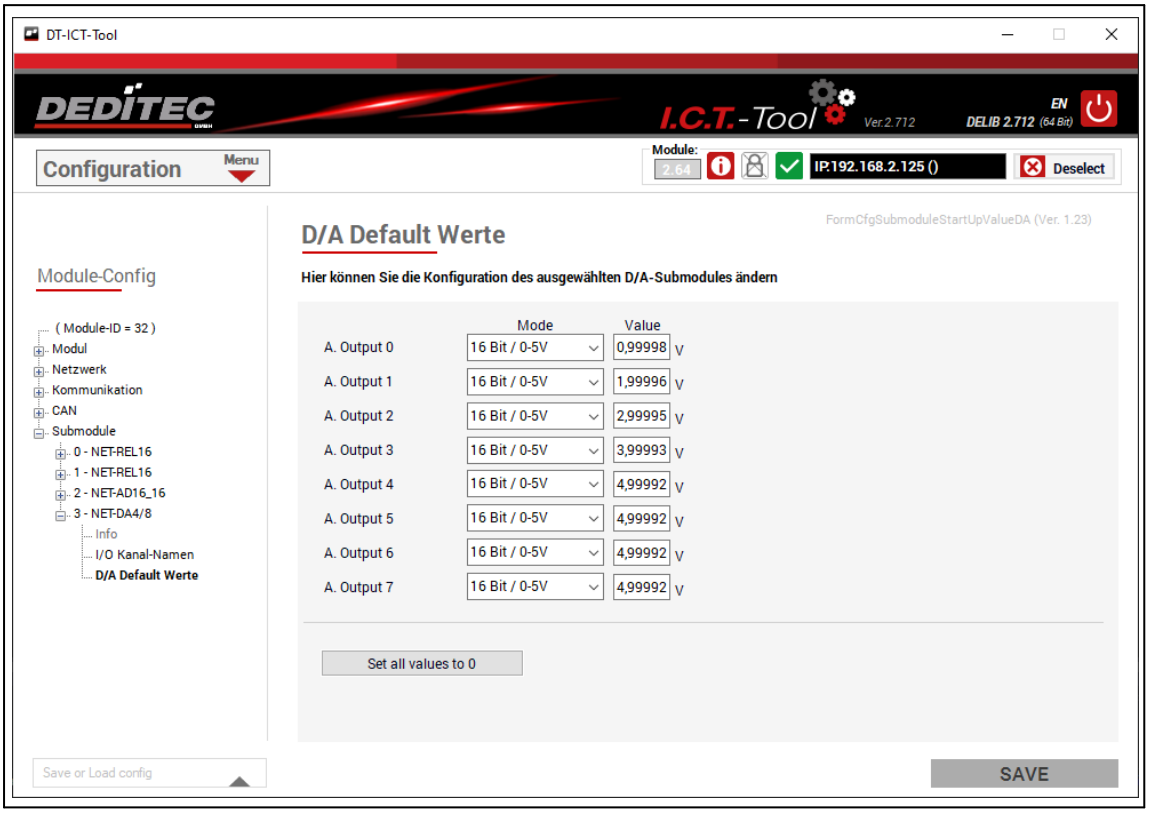
Set default channel name

Write the text shown above as a suggested name in the text fields.

You must also save it to apply it to the module.

5.3.15. D/A default values

Here you can set the D/A mode and value of the respective channels of your module with which the module is to be started.



Mode

Here you can set the D/A mode that the respective channel should have at module start.

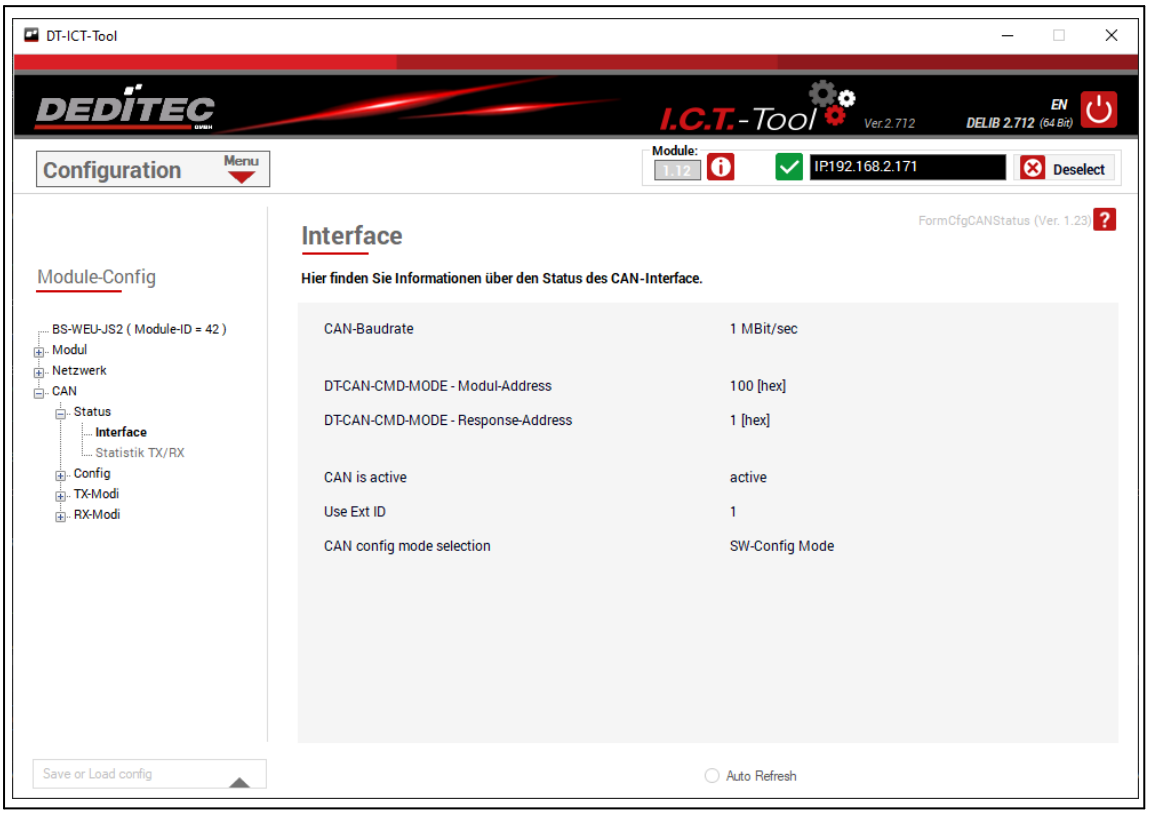
Value

Here you can set the D/A value that the respective channel should have at module start.

5.3.16. CAN configuration

5.3.16.1. CAN status interface

All important information about your CAN interface is displayed here



CAN baud rate

Displays the currently set baud rate of your CAN interface

DT-CAN-CMD-MODE - Module address [hex]

Determines the address under which the module is identified in the CAN bus.

DT-CAN-CMD-MODE - Response address[hex]

Specifies the module address to which a confirmation is sent as soon as a packet is received.

CAN is active

(only displayed if the module supports this function)

Use EXT ID

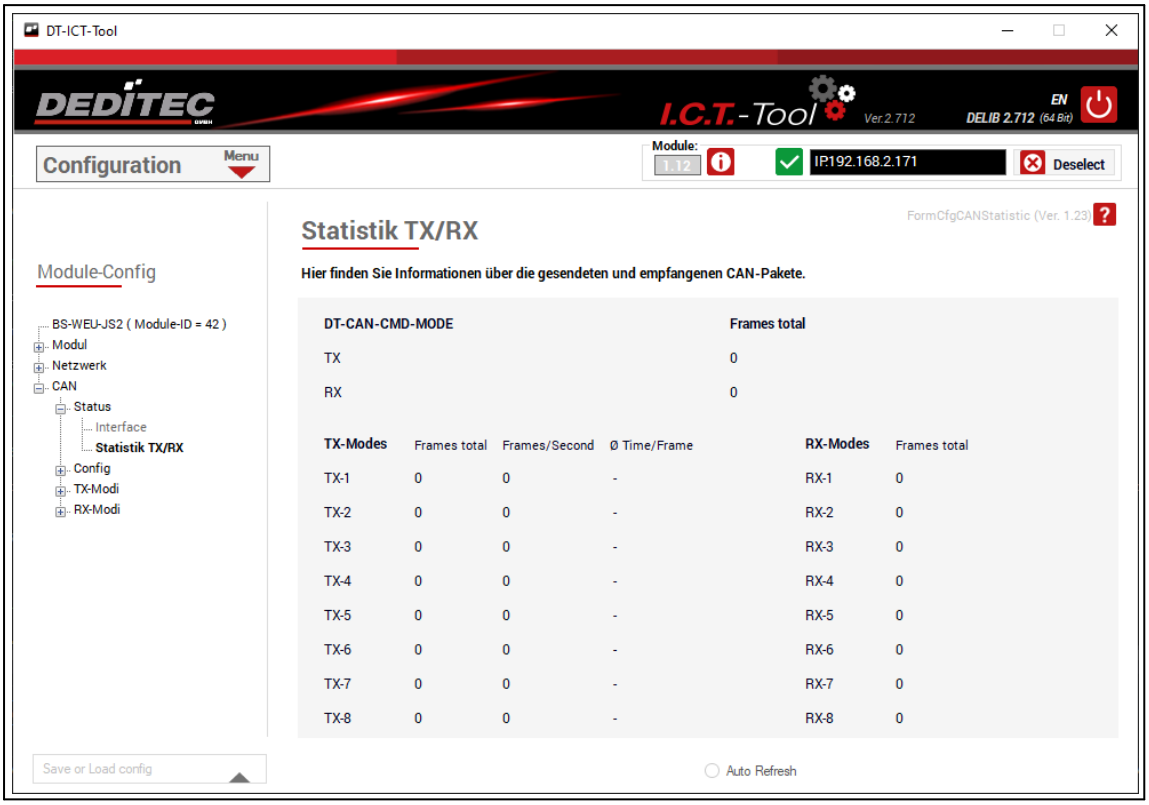
Displays the extended ID of the interface module used

CAN config mode selection

Displays the CAN configuration mode used

5.3.16.2. CAN statistics TX/RX

All important statistics for the TX and RX packages are displayed here.



The following TX information is displayed:

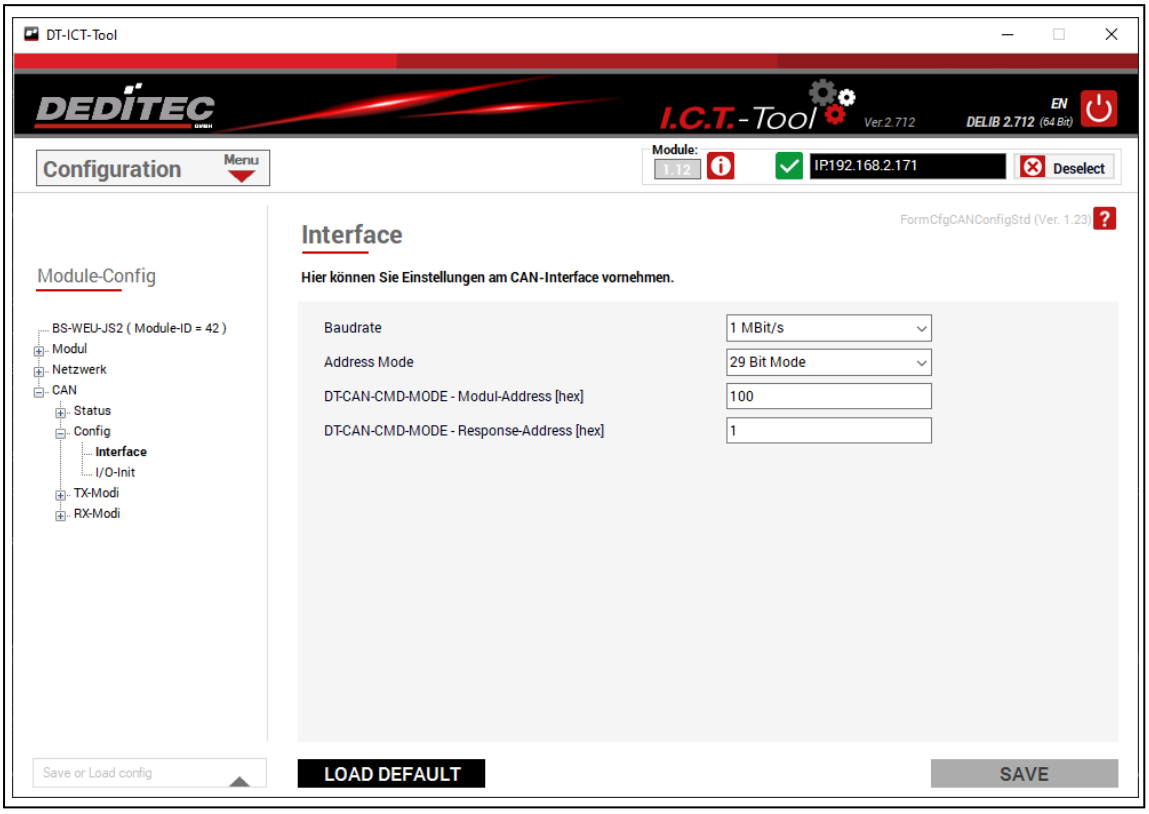
- Total number of packages sent
- Number of packets sent per second
- Average time required to send a parcel

The following RX information is displayed:

- Total number of packages received

5.3.16.3. CAN Main Interface

These settings can be used to configure the CAN interface.



Baud rate

The baud rate at which the module should communicate can be set here.

Address Mode

The address mode specifies how many bits are used for addressing.

DT-CAN-CMD-MODE - Modul-Address[hex]

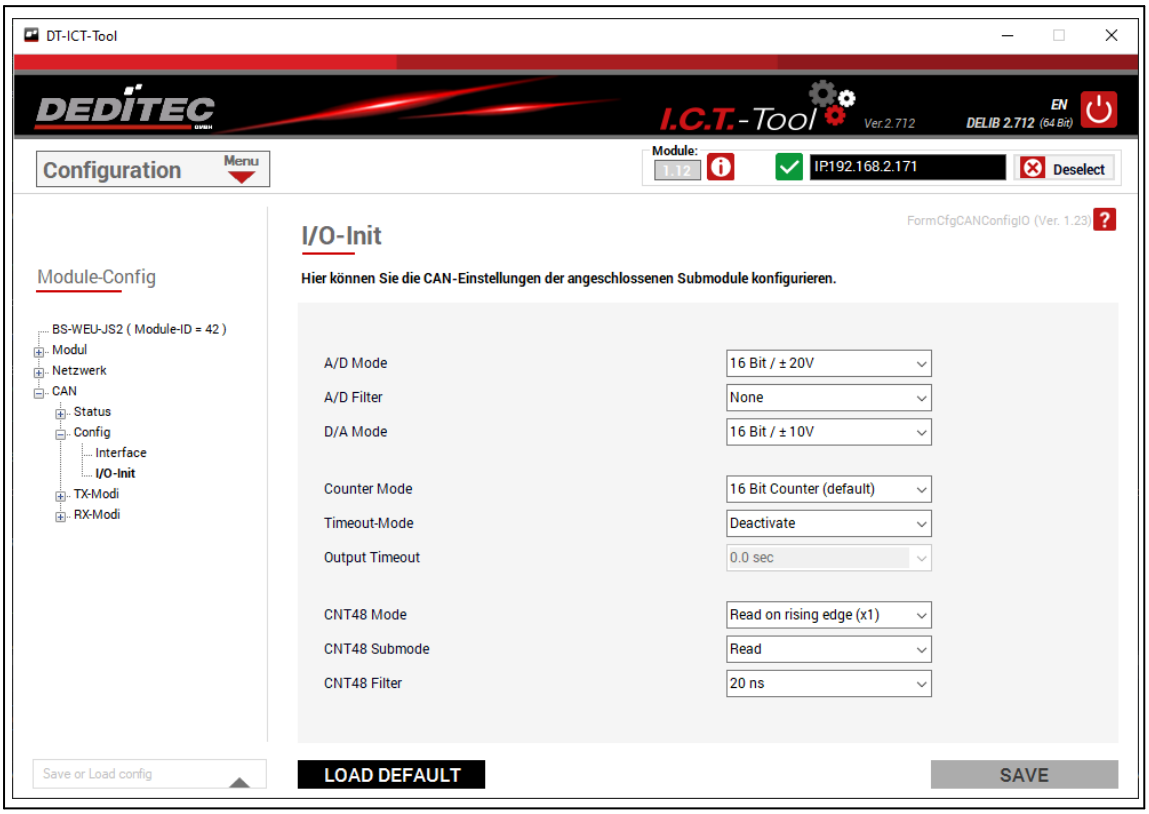
Determines the address under which the module is identified in the CAN bus.

DT-CAN-CMD-MODE - Response-Address[hex]

Specifies the module address to which a confirmation is sent as soon as a packet is received.

5.3.16.4. CAN Main I/O Init

These settings are used to configure the connected submodules. The respective filter / mode in which the connected submodules are started can be set.



A/D mode

The value range specifies the range in which analog signals are converted digitally (e.g. in the 0-10V range).

A/D filter

The A/D filter can be set here.

D/A mode

The value range specifies the range in which digital signals are converted to analog (e.g. in the 0-10V range).

Counter mode

Is responsible for the counter mode. You can choose to count up with 16 bits or count up and down with 8 bits each.

Timeout mode

The timeout mode can be set here. More information on the individual modes can be found in the chapter: DapiSpecialCMDTimeout. If no timeout is required, it can be deactivated with "Deactivate".

Output Timeout

Specifies the time after which the outputs switch off if a module can no longer be reached.

CNT48 mode

Sets which counter mode is to be used. There are 6 different modes to choose from.

CNT48 submode

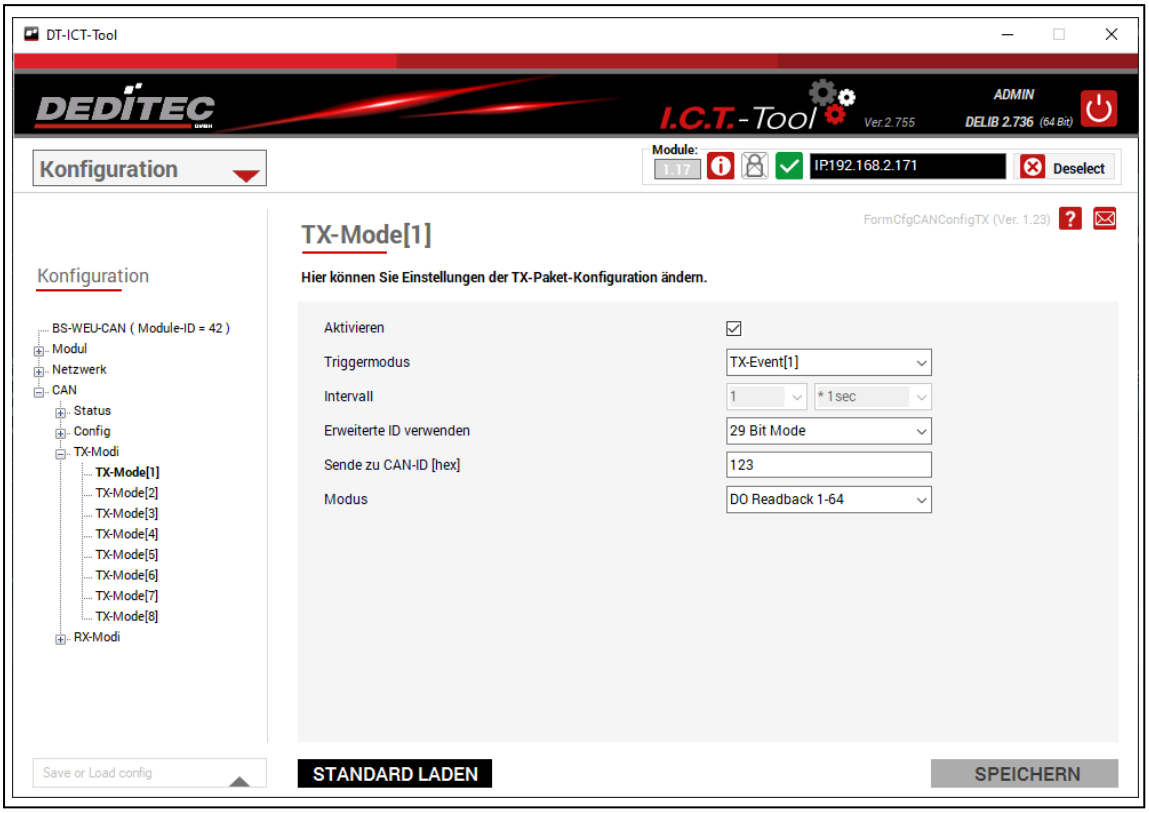
Depending on the mode selected under "CNT48 Mode", corresponding submodes are available for selection here.

CNT48 filter

Sets the filter for how long a signal must be at least for it to be recognized as high. You can choose from 16 predefined filters between 20ns and 5ms.

5.3.16.5. CAN TX Mode

Here you can make settings for the TX packages.



Activate

Activates this TX mode

Trigger mode

Specifies which mode should be used for sending. The following modes are available for selection: "Interval" , "RX event" and "Fast as possible" modes.

Interval

If the interval mode is set, you can also specify the time interval at which the transmission should take place.

Address mode

Specifies the bit mode to be used. You can choose between 11-bit mode and 29-bit mode.

Send to CAN-ID[hex]

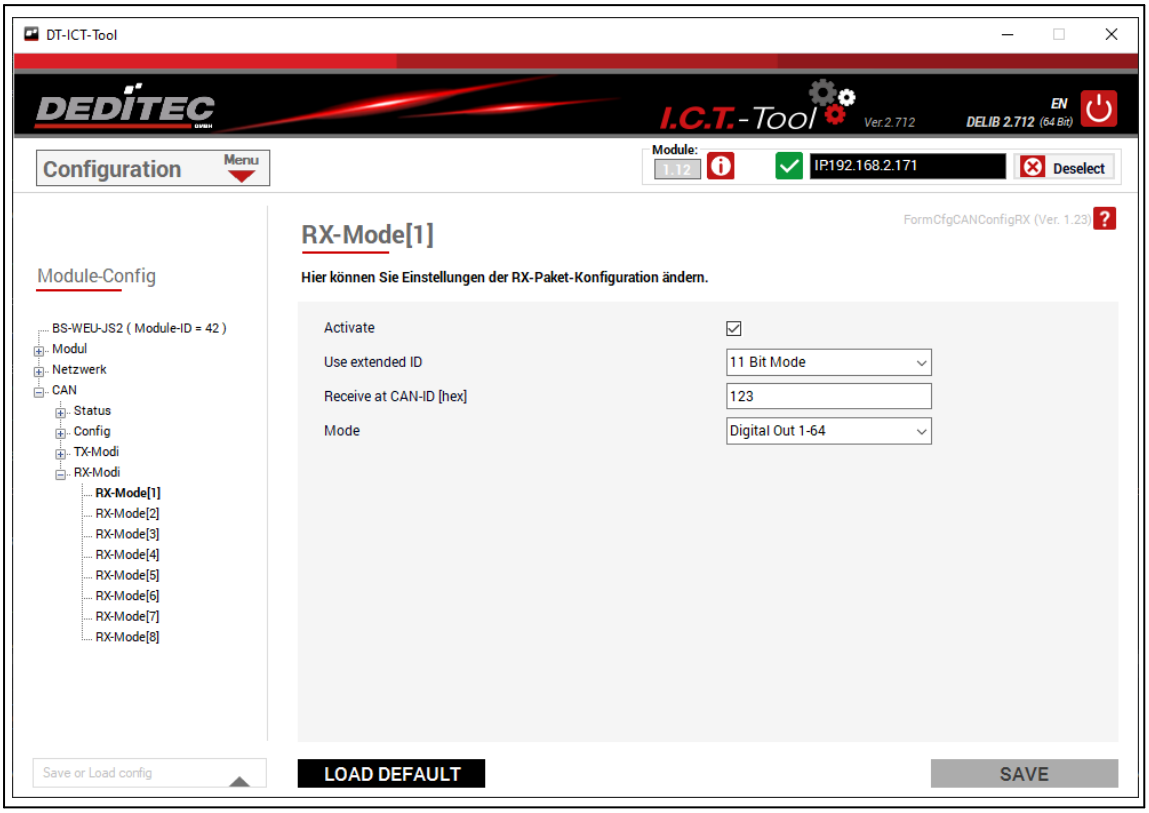
Sends the CAN packets to this address. In the example above, the packet is sent to address 0x200.

Mode

Here you can specify which data should be sent to the previously set address.

5.3.16.6. CAN RX Mode

You can make settings to the RX packages here.



Activate

Activates this RX mode

Address mode

The 11-bit or 29-bit address mode can be set here.

Receive at CAN-ID[hex]

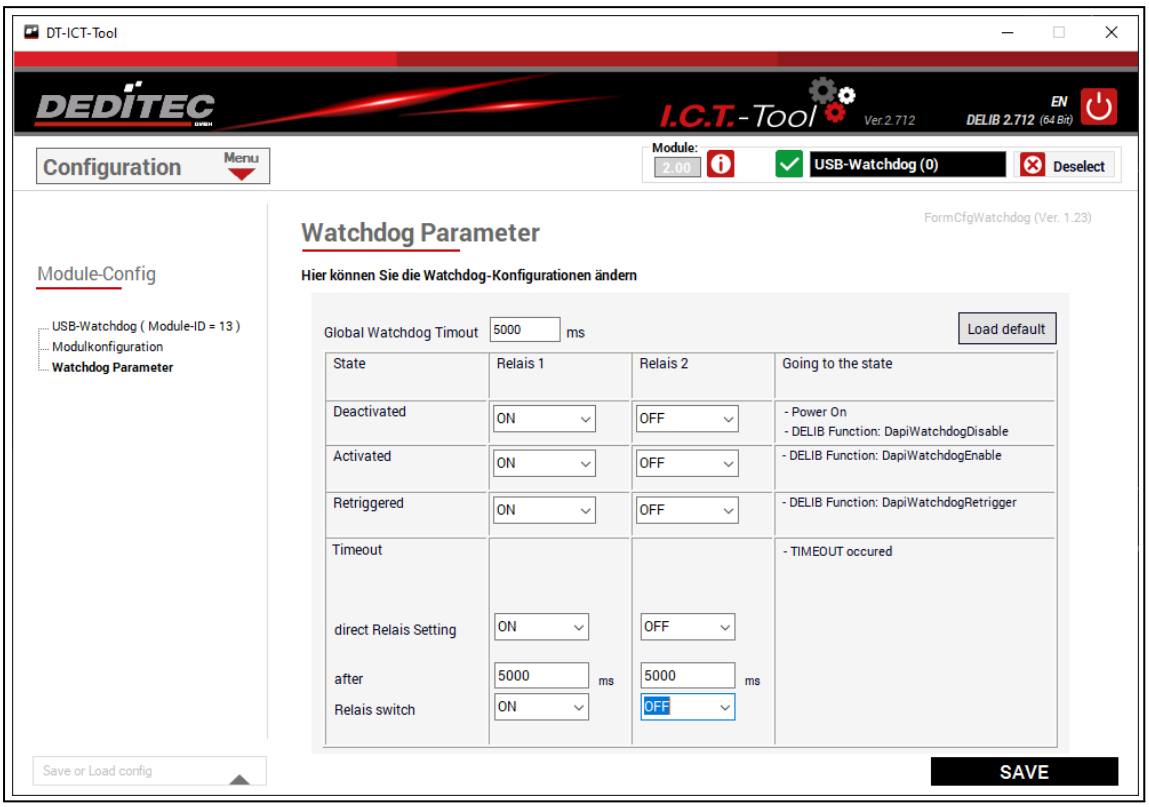
Specifies the recipient address. In the example above, a CAN packet is received at address 0x100.

Mode

If a packet is received at the set address, the content of the data packet is forwarded to the digital outputs 1-64, whereupon the outputs are switched on or off.

5.3.17. Watchdog configuration

You can make settings on your watchdog stick here.



Deactivated

Relay 1 and 2 monitoring not active

Activated

Relay 1 and 2 monitoring active

Retriggered

Watchdog time relay 1 and 2 is reset

Timeout

Watchdog time relay 1 and 2 was not reset

Relais Setting after x msec

Time-delayed switching of the relays after a timeout

5.4. M2M configuration

M2M stands for “Machine-to-Machine” and represents direct communication between two or more devices.

It enables, for example, the automated transmission of measured values or remote monitoring of machines and systems.

The possibility of M2M transmission between our DEDITEC modules is supported by two interfaces, Ethernet and CAN.

For M2M communication via Ethernet, it must be noted that the transmitter module must explicitly have the M2M function.

However, all DEDITEC products with an Ethernet interface can be used as receiver modules.

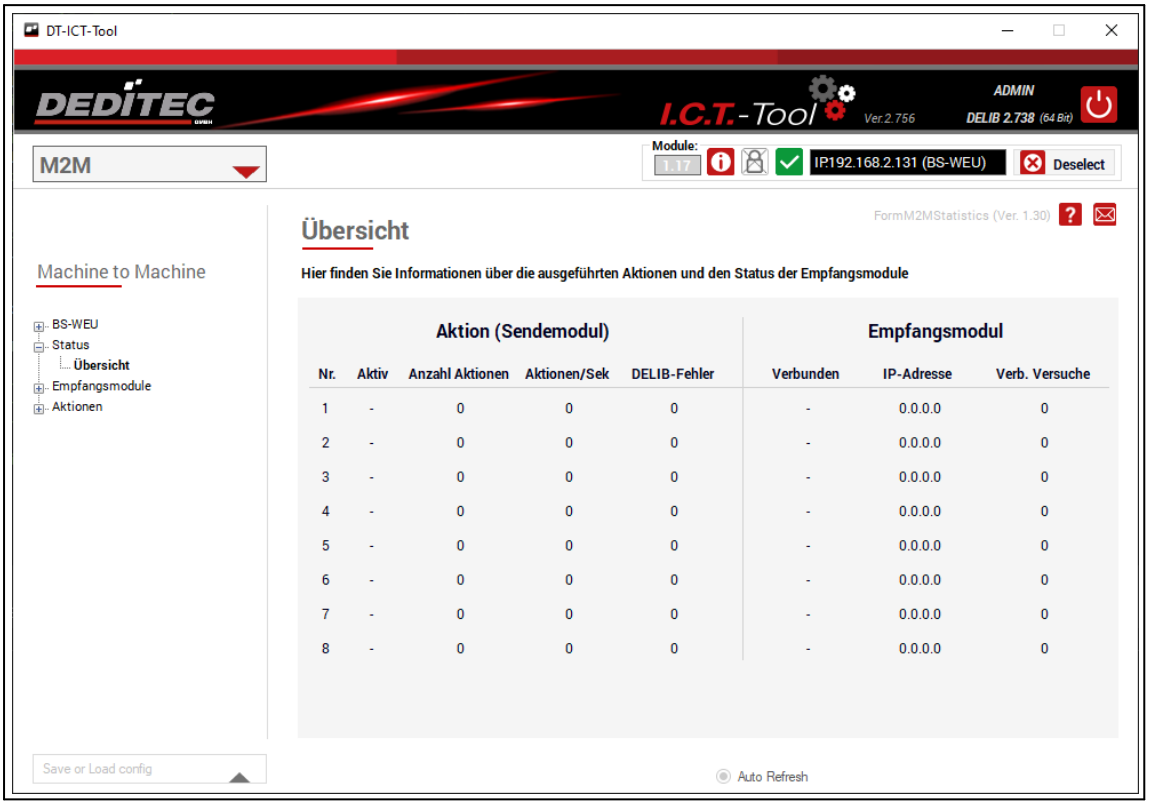
M2M communication via the CAN interface is supported by all our CAN modules.

Settings can be made in the CAN configuration area of the ICT tool.

You can find more information on this in the chapter: **CAN-Configuration**

5.4.1. Statistics

Here you get an overview of the activity of the transmitter and receiver module



The following transmitter information is displayed:

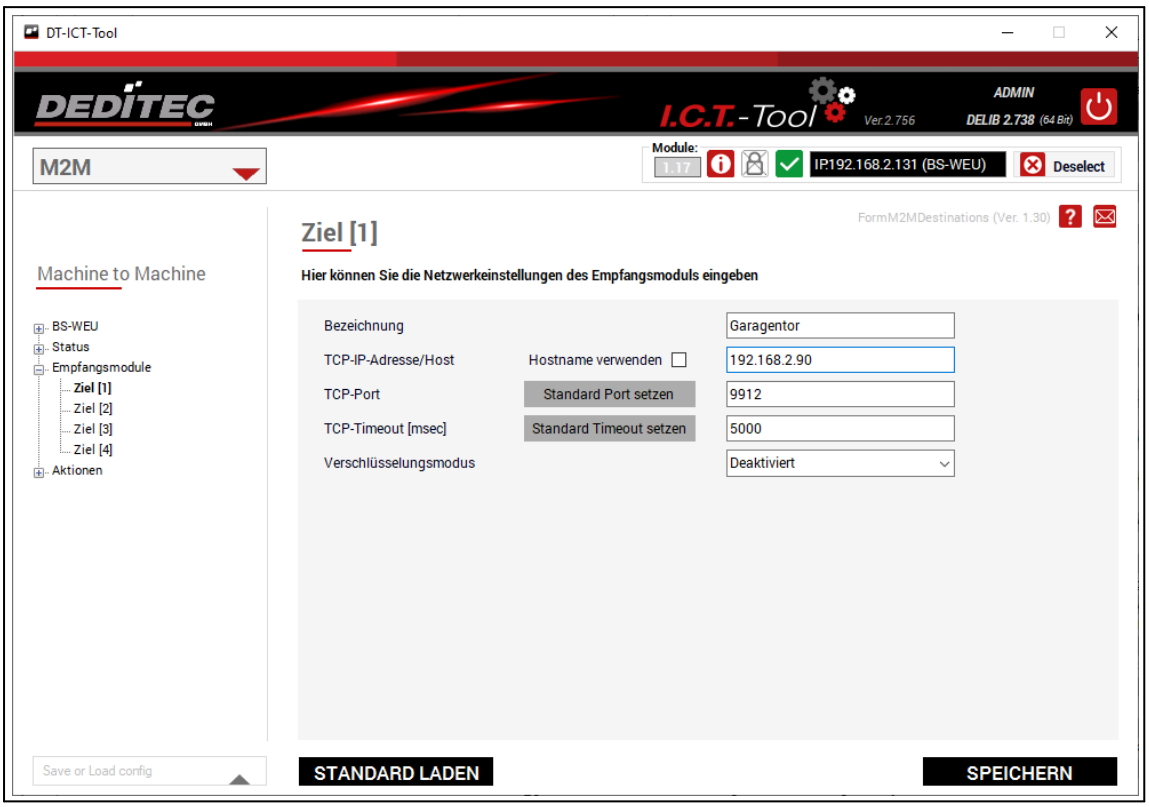
- Activity status: If the check mark is set, this action is currently being carried out
- Total number of actions performed. Is reset to 0 after a module restart
- Number of actions per second
- Number of DELIB errors occurring when executing commands from the DEDITEC driver library. You can find help for debugging errors in the chapter: **DELIB-DebugView-Global**

The following receiver module information is displayed:

- Connected: If the check mark is set, the transmitter and receiver modules are connected to each other
- IP address of the receiver module
- Number of connection attempts

5.4.2. Receiver module

You can set the network settings of the receiver module here.



Designation

Enter the desired name for the receiver module here.

IP address / Port

You can specify the network settings of the receiver module here.

Timeout[msec]

The timeout specifies how long an attempt should be made to establish a connection to the module. If this time is set too short, the timeout may be triggered before the module has been opened properly. The duration of the connection attempts after a connection to the module is lost is also specified with the timeout.

Encryption type / pw

Here you can specify the encryption settings of the receiver module.

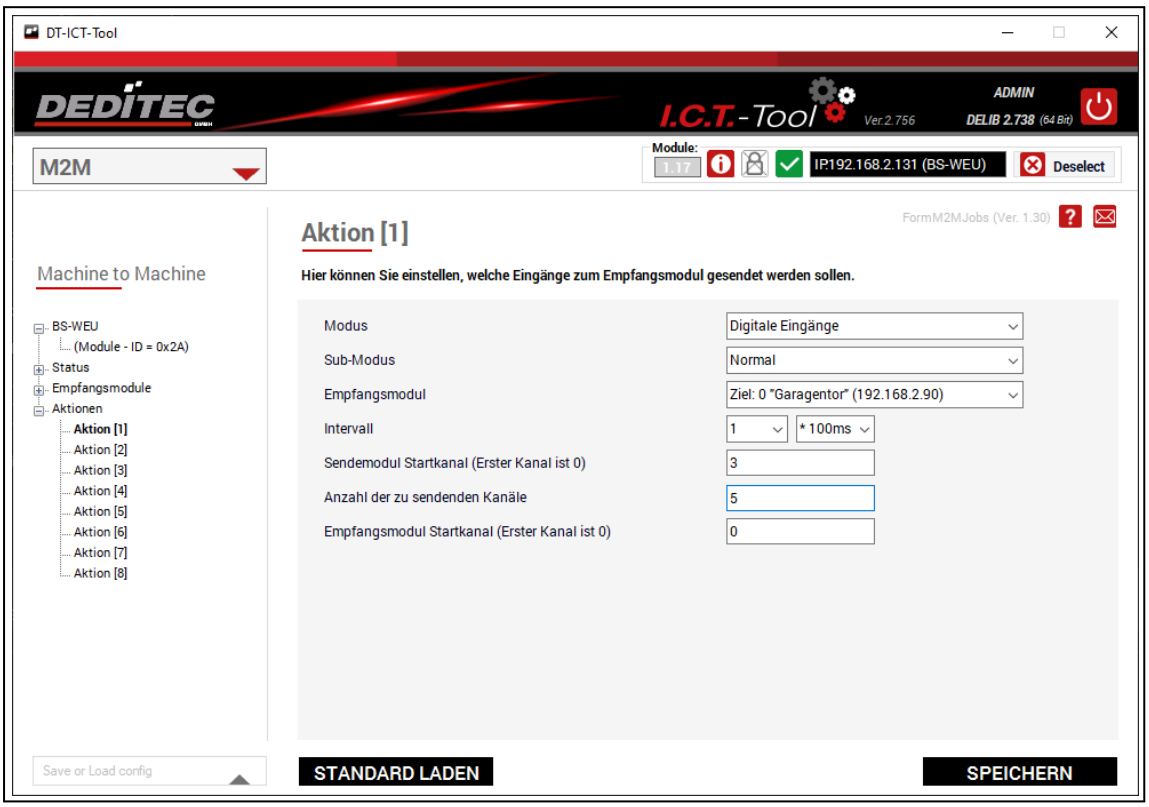
These settings are used to establish the connection to the receiver module. Make sure that these settings have also been made in the receiver module.

You can find more information on this in the chapter: **TCP encryption**

5.4.3. Jobs

Use an job to define which input signal is to be sent from the transmitter module to which output of a receiver module.

In this image example, DI data is sent every 100ms from channel 3(CH Start) to channel 8 ((CH Start) + (CH Count)) to channels 0-5 of the receiver module.



Job

Here you can specify whether you want to send digital or analog input signals.

Mode

In normal mode, the current states of the input channels are transmitted to the receiver module.

In test mode, alternating high / low packets are transmitted to the receiver module. This makes it easier for you to check your transmit/receive configuration.

Receiver module

Here you can select which receiver module should receive the information. The receive module is defined in the "Receive modules" tab.

Interval

Set the time intervals at which the transmitter module should send the information.

Transmitter module start channel

Set the input channel from which the information is to be sent to the receiver module.

Number of channels to be transmitted

Enter the number of input channels to be transmitted to the receiver module here, starting with your entry in the &quot field;Transmitter module start channel"

Receiver module start channel

Set here from which output channel of the receiver module the information is to be output by the transmitter module.

5.5. Event control

The event control can be used to configure actions that are automatically executed by the module as soon as a certain condition is met.

Both the event condition and the action can be set separately.

Which events and actions can be configured depends on the module.

You can find help on configuring the event here: **Configuration events**

You can find help on configuring the action here: **Configuration actions**

The following module series are supported by Event Control:

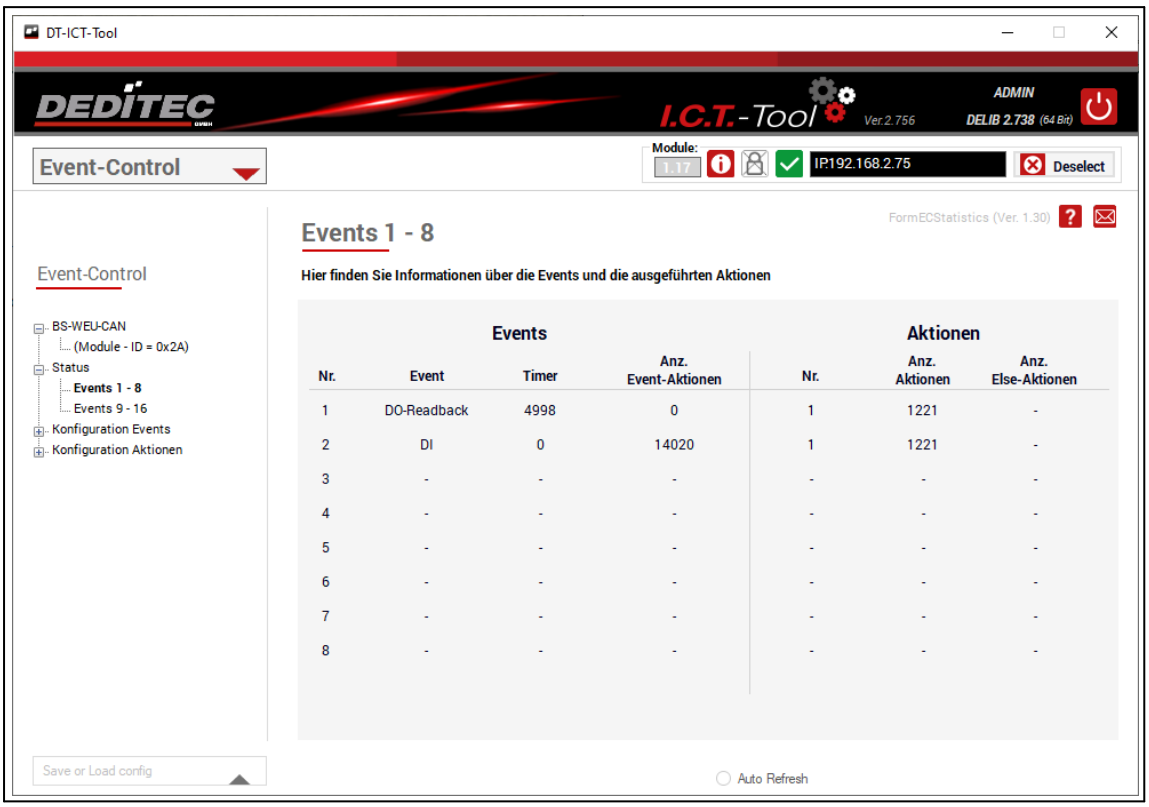
- BS-WEU series
- BS-USB series
- NET series
- UC series
- CANBox series

The following module series are **not** supported by Event Control:

- Starter series
- RO series
- COS series

5.5.1. Statistics

Various information about the events and actions is displayed here.



5.5.2. Event configuration

With the help of an event, a certain action is executed as soon as a condition is fulfilled.

For example, voltages can be automatically output via the D/A converter or relays can be switched when a previously defined switching threshold is reached at the A/D inputs,

or a signal is detected at a digital input.

You can also specify how long the condition must be fulfilled before an action can be carried out.

These settings are saved directly in the **Modul-Configuration-Memory** of the module.

Depending on the features of your product, you can choose from different modes:

"DI"

High or low signal of a digital input.

"DI flip-flop"

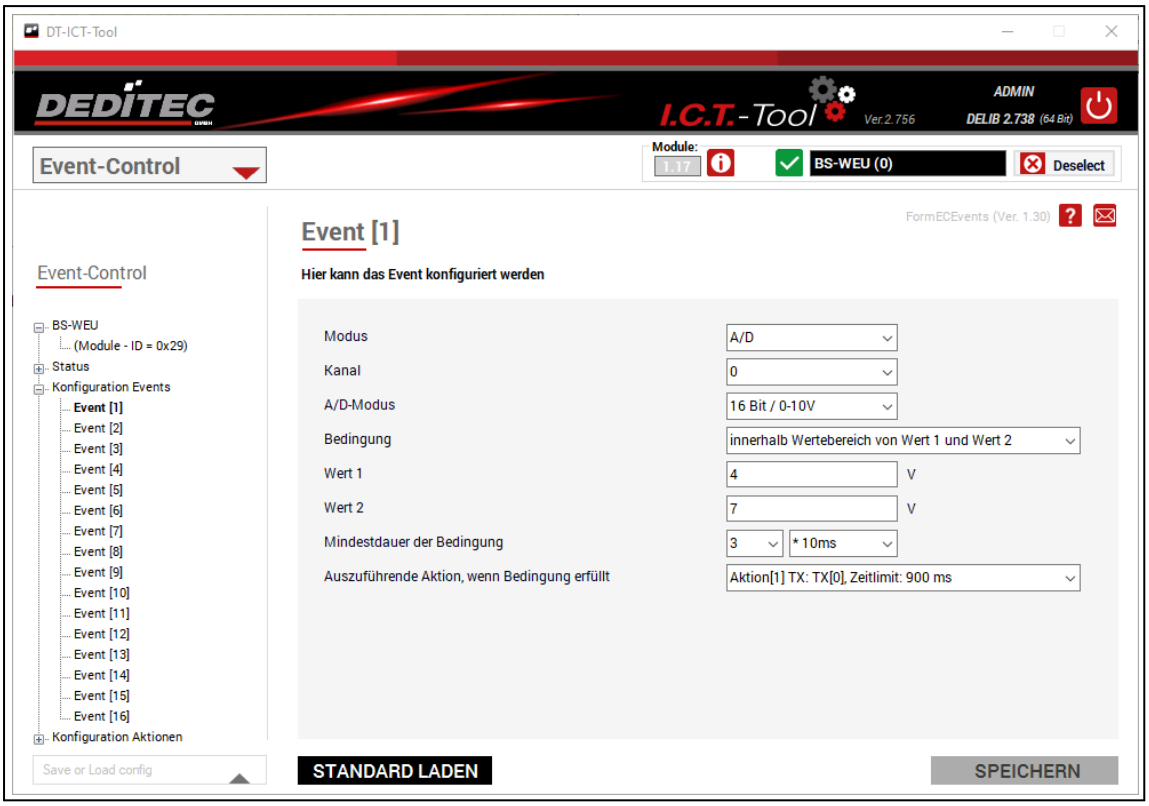
If the status of the digital input changes, the flip-flop of the channel is set and then reset again.

"DO readback"

The status of the output channel is read out and returned using a DO readback command.

"A/D"

The current voltage of the analogue input is read out.


DT-ICT-Tool
DEDITEC
I.C.T.-Tool
Ver.2.756
ADMIN
DELIB 2.738 (64 Bit)
Event-Control
Module:
1.17
BS-WEU (0)
Deselect
FormECEvents (Ver. 1.30)
Event [1]
Event-Control
Hier kann das Event konfiguriert werden
BS-WEU
(Module - ID = 0x29)
Status
Konfiguration Events
Event [1]
Event [2]
Event [3]
Event [4]
Event [5]
Event [6]
Event [7]
Event [8]
Event [9]
Event [10]
Event [11]
Event [12]
Event [13]
Event [14]
Event [15]
Event [16]
Konfiguration Aktionen
Save or Load config
Modus
A/D
Kanal
0
A/D-Modus
16 Bit / 0-10V
Bedingung
innerhalb Wertebereich von Wert 1 und Wert 2
Wert 1
4
V
Wert 2
7
V
Mindestdauer der Bedingung
3
* 10ms
Auszuführende Aktion, wenn Bedingung erfüllt
Aktion[1] TX: TX[0], Zeitlimit: 900 ms
STANDARD LADEN
SPEICHERN

5.5.3. Actions

An action defines what should happen within an event once a condition has been met.

Different modes can be defined depending on the features of your product:

"Set DO output"

Switches a digital output on or off.

"Set D/A output"

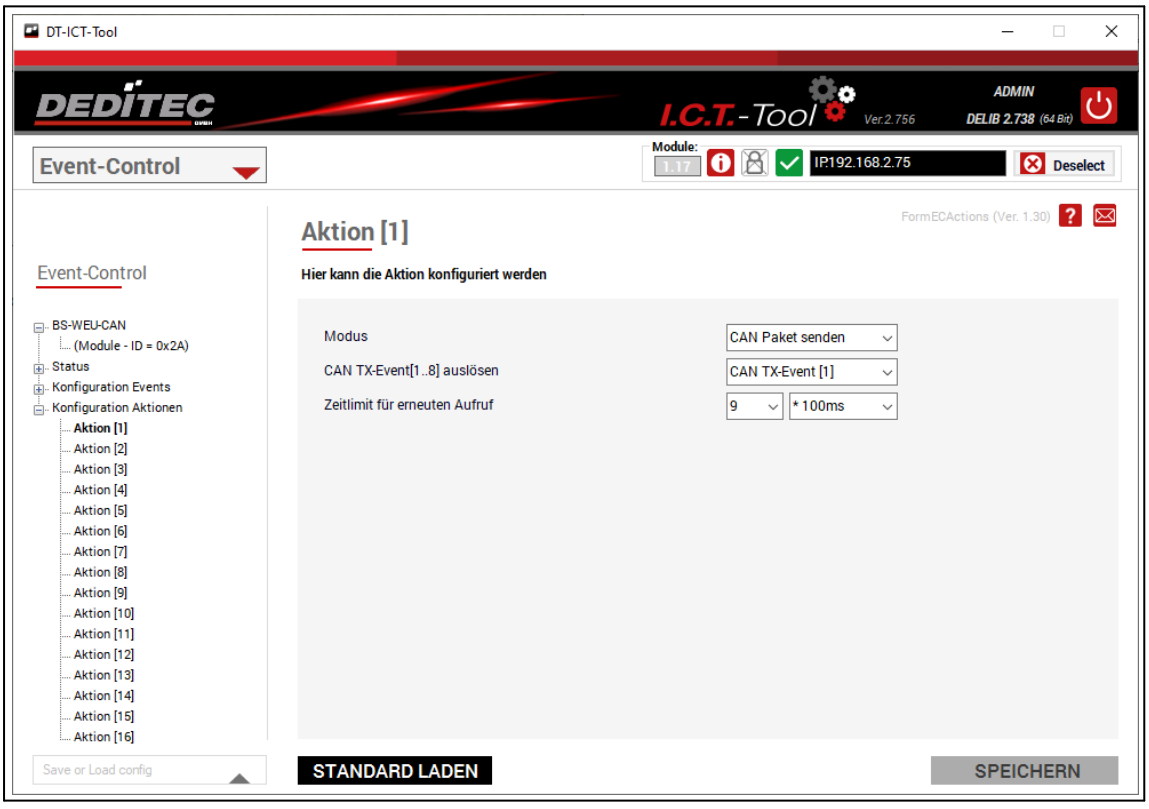
Outputs voltage at an analogue output.

"Send CAN packet"

The set CAN TX event is triggered.

For this function, the trigger mode in the CAN TX package must be set to "TX-Event".

You can find more information on configuring the TX mode in the chapter: **CAN TX-Mode.**



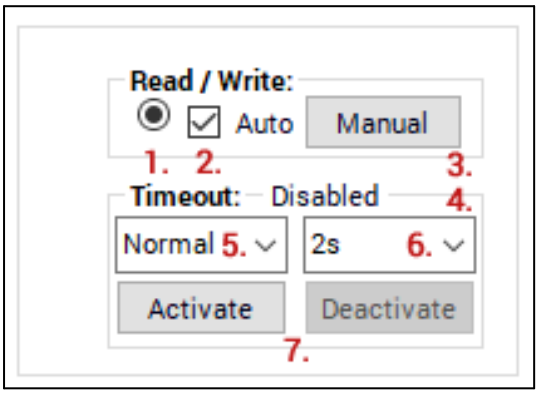
5.6. Test module

Tests can be carried out on the modules in the I/O area.

5.6.1. Timeout test function

Timeout settings can be made in the "Read/Write" area.

These functions can be used to trigger a timeout case.



1 The field flashes repeatedly to indicate whether there is a connection to the module. If the field remains empty, communication is interrupted.

2 Unchecking the box interrupts the connection and triggers a timeout. Ticking the box again restores the connection.

3 As the user interface is not updated automatically in the event of a timeout, this can be triggered by clicking on the "Manual" button.

4 The current timeout status is displayed here.

5 The desired timeout mode can be set here. For more information, see chapter: **DapiSpecialCMDTimeout**.

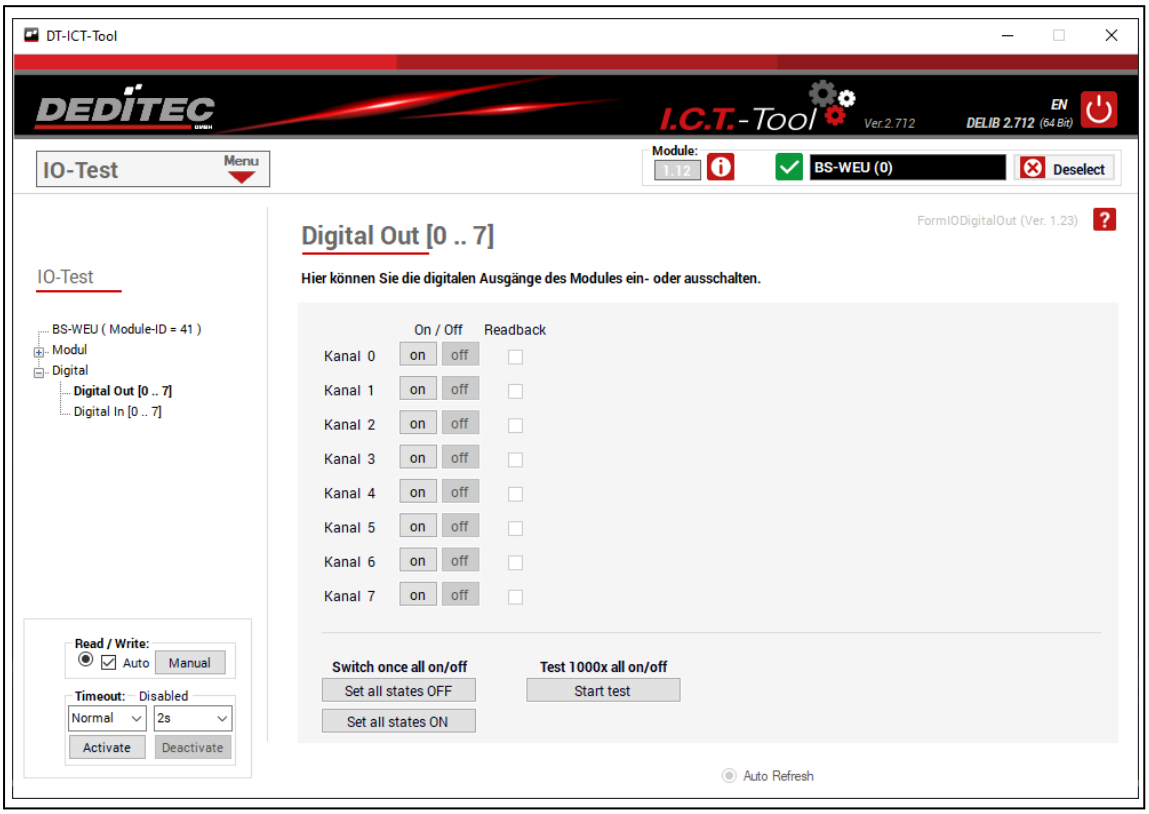
6 Here you can set the time at which the timeout should be triggered.

7 "Activate" activates the timeout. Deactivate" deactivates it.

5.6.2. Digital Out

Here you can switch the individual digital outputs of your module on and off.

An LED on each output relay on the board of your module indicates the current status of the output (LED on = relay on).



On/Off

Switches the respective output relay on or off.

Readback

Displays the current status of the respective relay (on or off).

Switch all states OFF / Switch all states ON

With these buttons, all outputs of the module can be switched on or off simultaneously.

Switch channels with timer function

(Only displayed if supported by the module)

In the timer area, enter a time (in seconds) after which the relays should be switched on or off. Press "set" to start the timer.

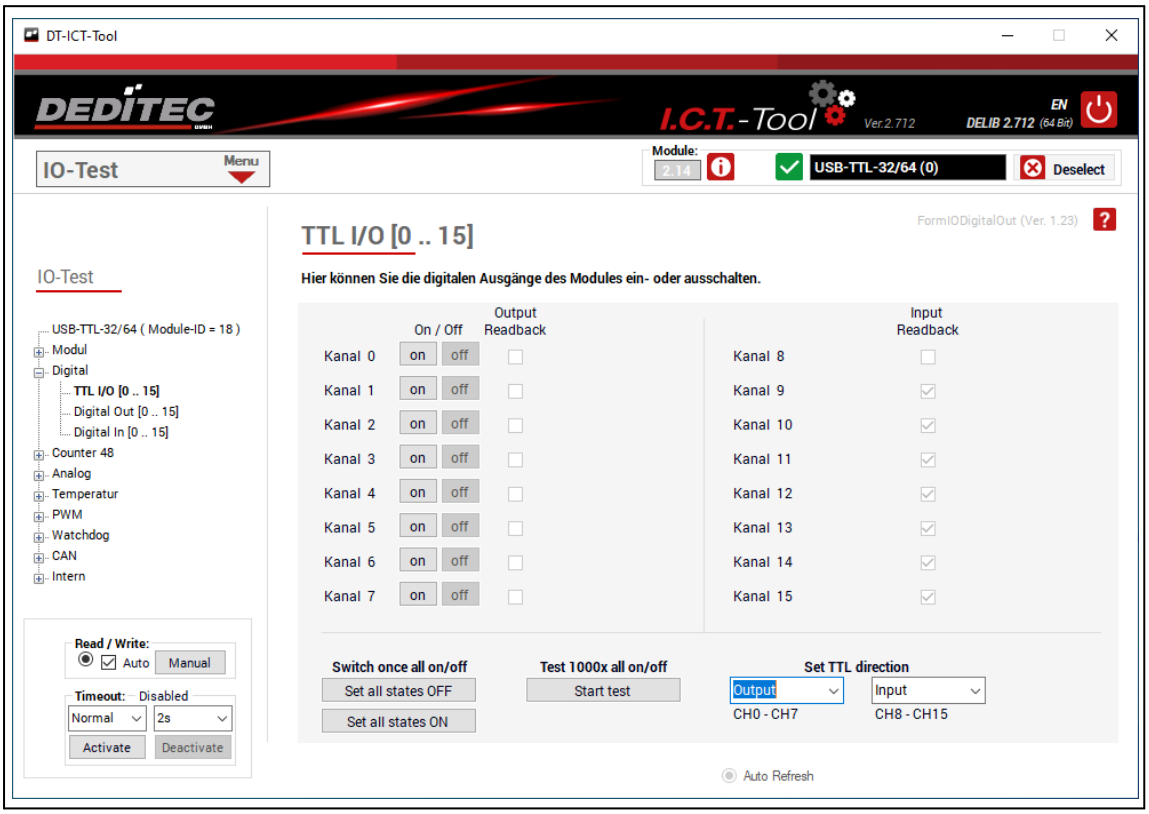
Invert DO-Timer

If this option is activated, the relay is deactivated after the timer has expired. If this option is deactivated, the relay is activated after the timer has expired.

5.6.3. Digital TTL

Here you can switch the individual digital outputs of your module on and off.

An LED on each output relay on the board of your module indicates the current status of the output (LED on = relay on).



On/Off

Switches the respective output relay on or off.

Readback

Displays the current status of the respective relay (on or off).

Switch all states OFF / Switch all states ON / Test 1000x all on/off

These buttons can be used to switch all of the module's outputs on or off simultaneously.

With the "Test 1000x ll on/off" test button, the outputs of the module are automatically switched on and off 1000 times.

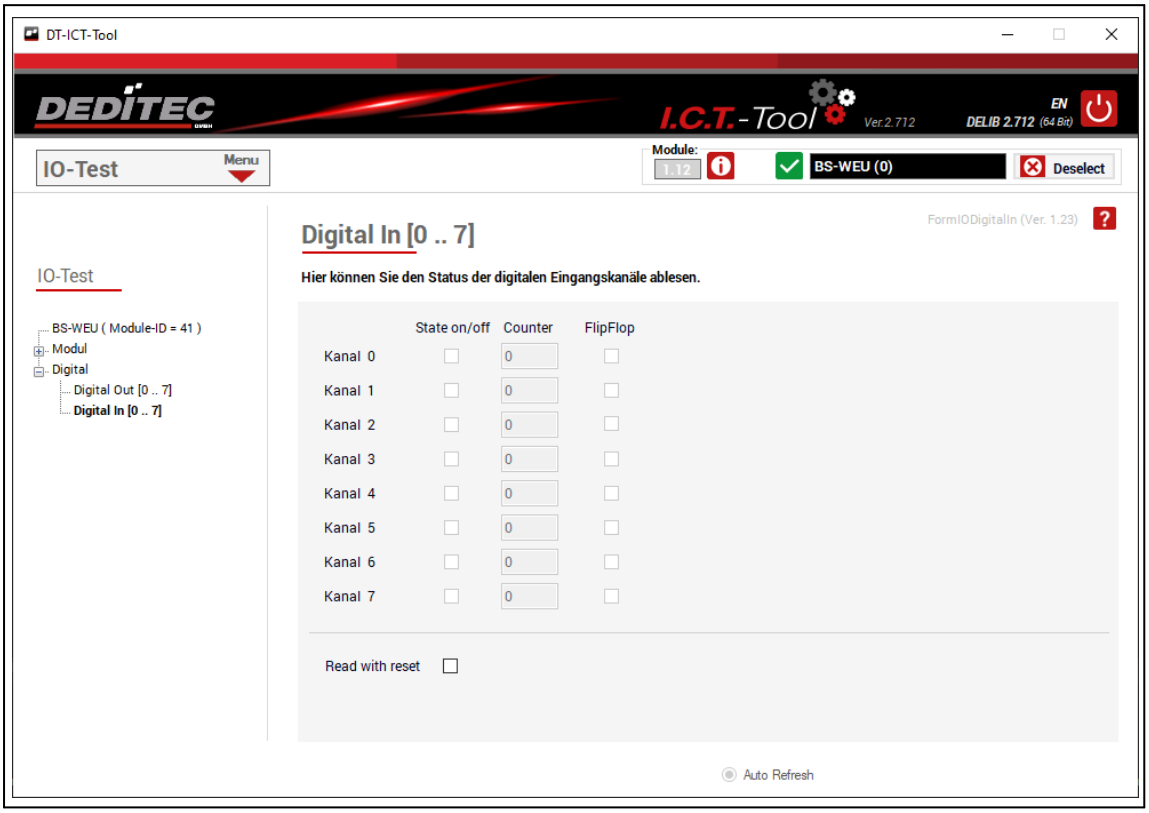
Set TTL direction

Here you can set the direction (input/output) of your module 8-channel by 8-channel.

These settings are retained until the module is restarted.

5.6.4. Digital In

Here you will find information on the digital inputs, as well as the input counter and the flip-flop filter.



State on/off

Shows the current status of the individual input channels.

Counter

Displays the counter readings of the input counters.

FlipFlop

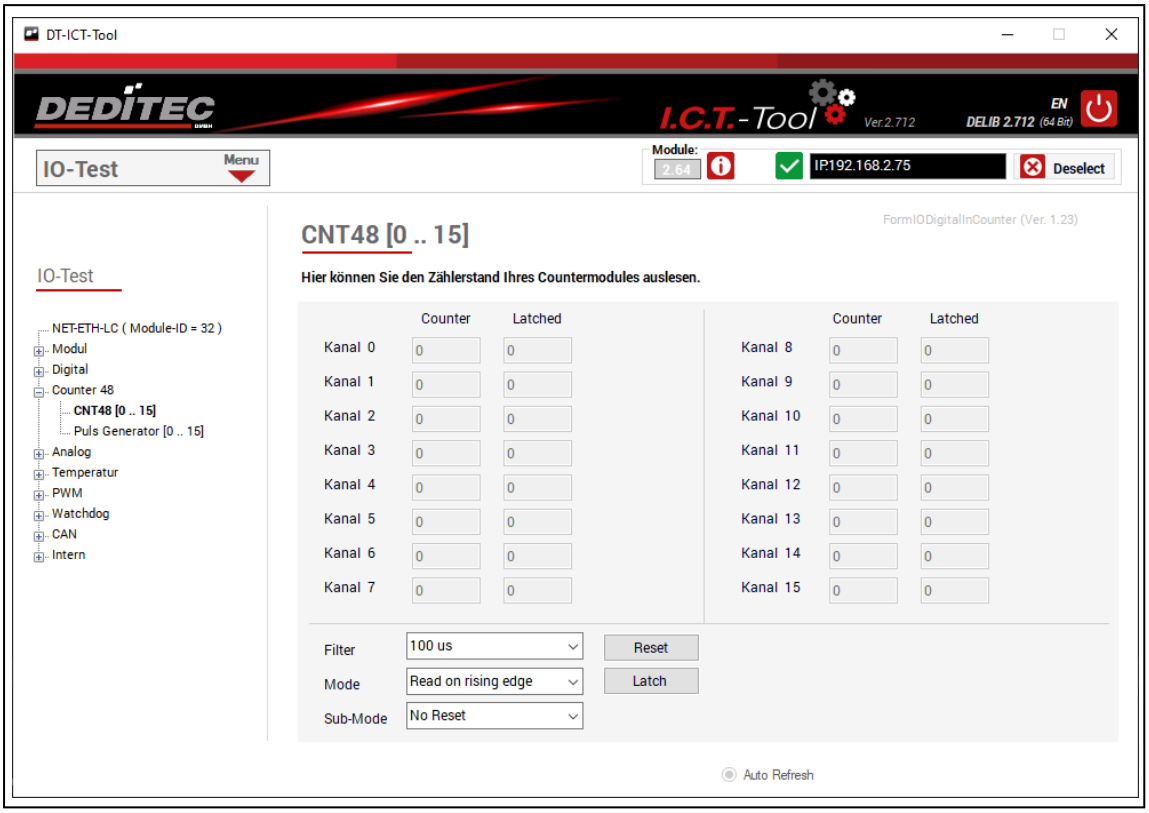
Displays the change in input status since the last readout.

Read with reset

This option is used to specify whether the counters are to be reset the next time they are read.

5.6.5. Digital In Counter

You can test your Digital In Counter module here



Counter

Displays the current counter reading of the channel.

Latched

Displays the current counter reading of the buffer memory.

Filter

The input filter can be set here.

Mode

The counter mode can be set here.

Sub-Mode

The counter-sub mode can be set here.

Reset

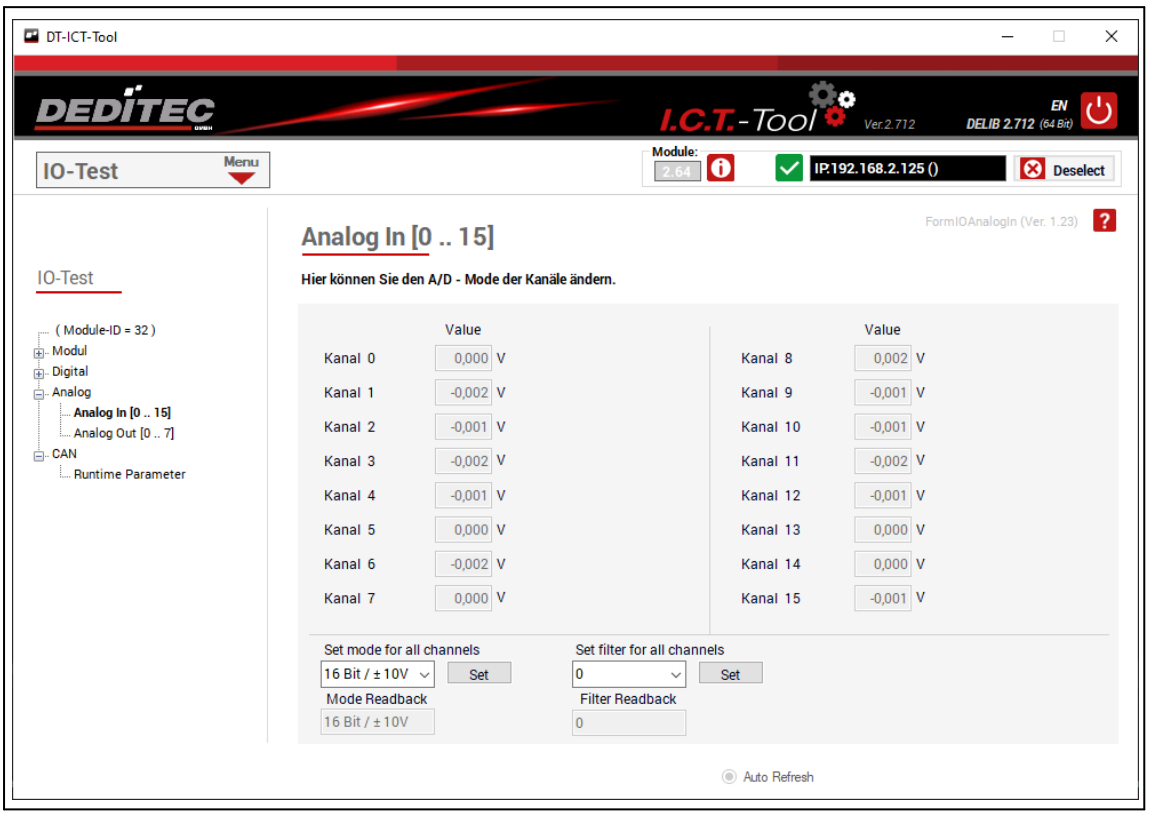
All meter readings are reset here.

Latch

Reads all meter readings from the buffer memory and displays them.

5.6.6. Analog In

Here you can change and test settings in A/D mode.



Value

Reads out the value on the respective A/D channel.

Set mode for all channels

Selection of the voltage/current range for all channels in which measurements are to be taken. Only modes that are supported by your module are displayed.

Mode readback

Reads the currently used A/D mode from the module.

Set filter for all channels

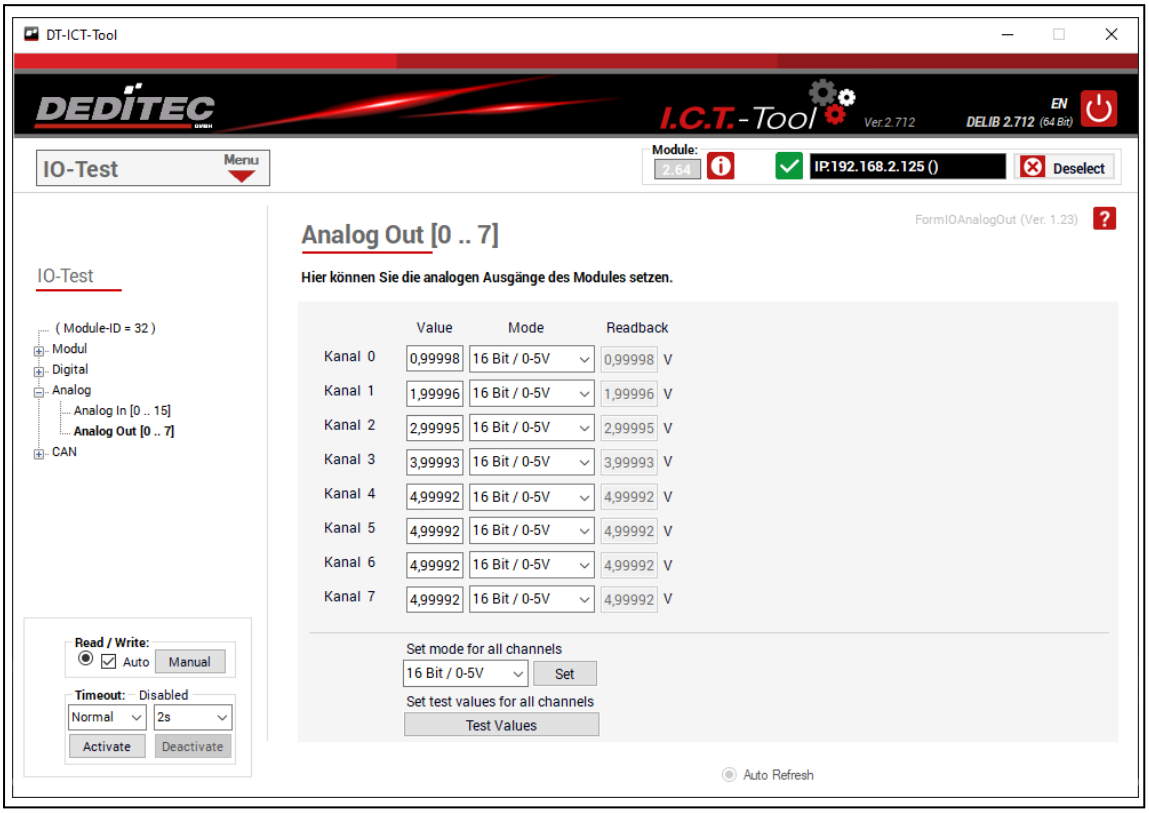
Selection of the A/D filter to be used for all channels.

Filter readback

Reads the currently used A/D filter from the module.

5.6.7. Analog Out

In this area, you can make and test settings on the D/A channels of your module.



Value

The value to be output on the respective D/A channel can be entered here.

Mode

Selection of the voltage/current range in which the value of the respective channel is to be output. Only modes that are supported by the module can be selected.

Readback

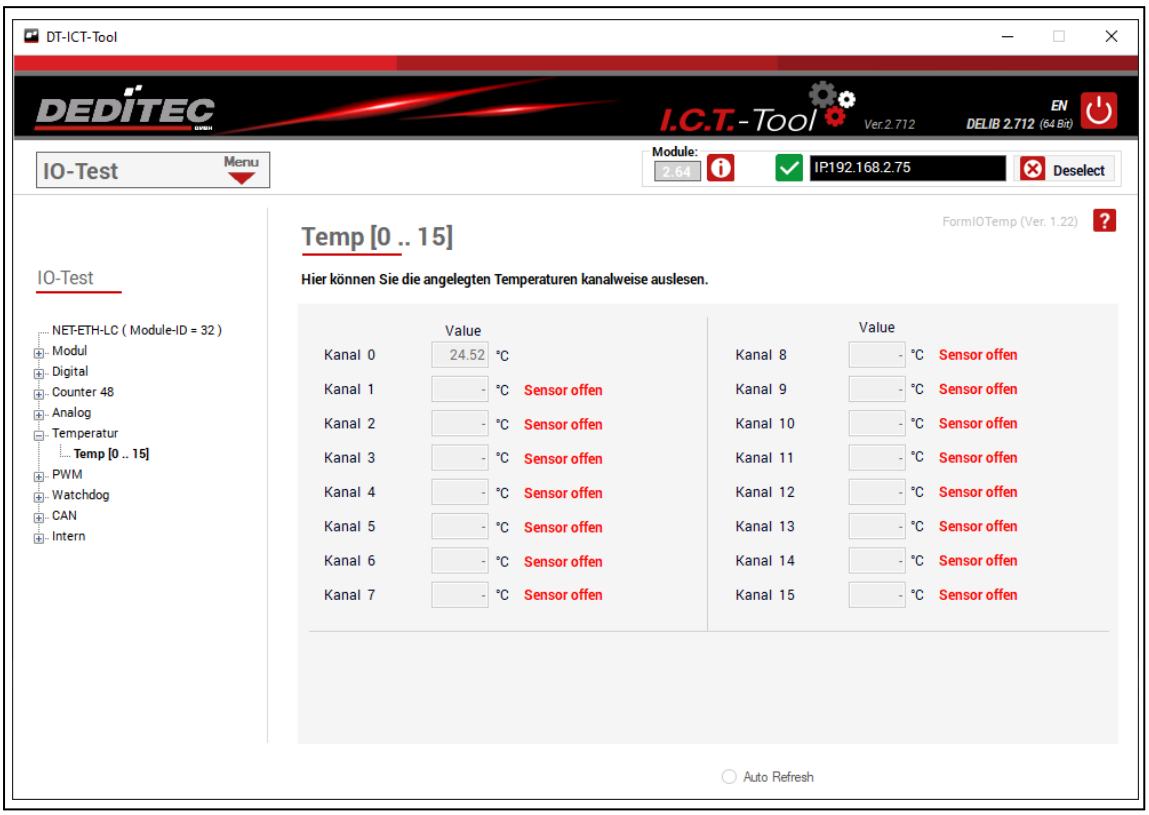
Reads back the actual value of the respective D/A channel.

Set mode for all channels

This function can be used to change the voltage/current range for all available channels at once.

5.6.8. Temp

Here you can read out the applied temperatures and the status of the sensor.



Value

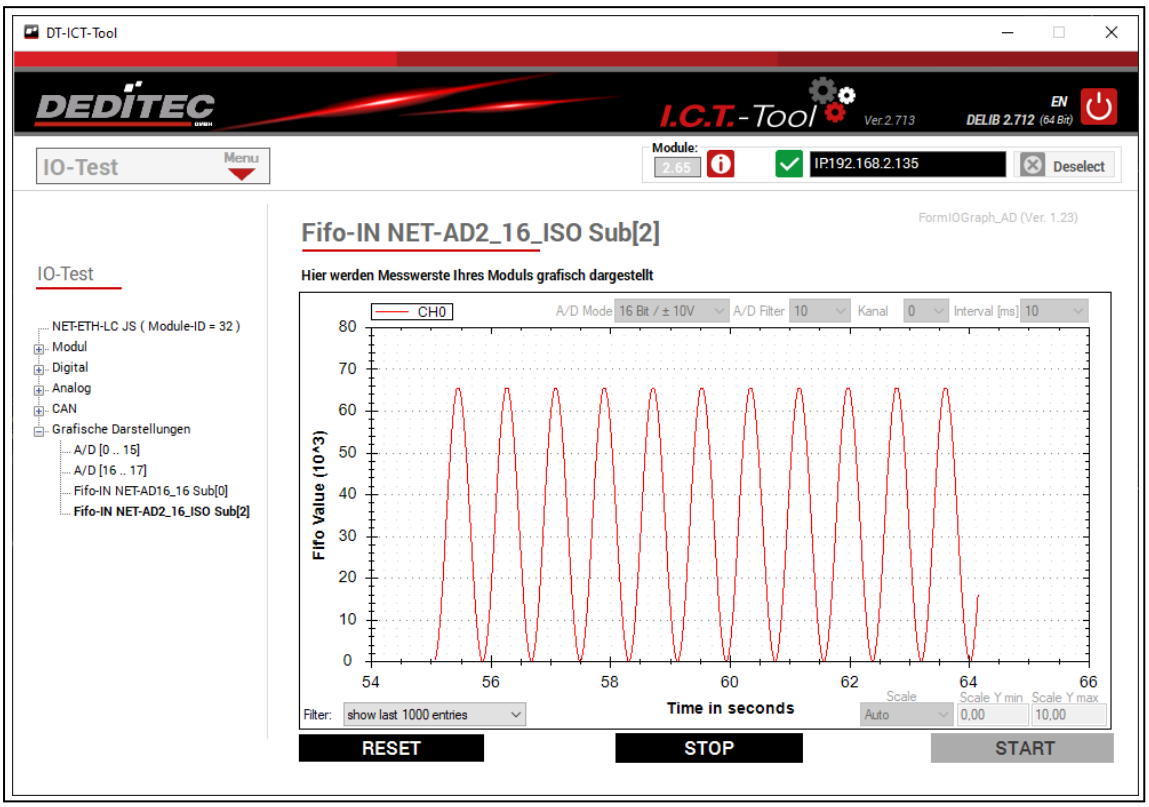
Indicates the current temperature.

Status

Displays the current status of the temperature sensor.

5.6.9. Graphical representation

Here you can display read A/D, temperature and FIFO values graphically.



A/D Mode

Here you can specify the A/D mode of the A/D converter

A/D Filter

Here you can specify which A/D filter is to be used

Channel

Specifies which channel is to be displayed graphically

Interval [ms]

Specifies the interval at which the data should be read

Filter

Specifies how much data is to be displayed

Start / Stop

Starts or ends the graphical output

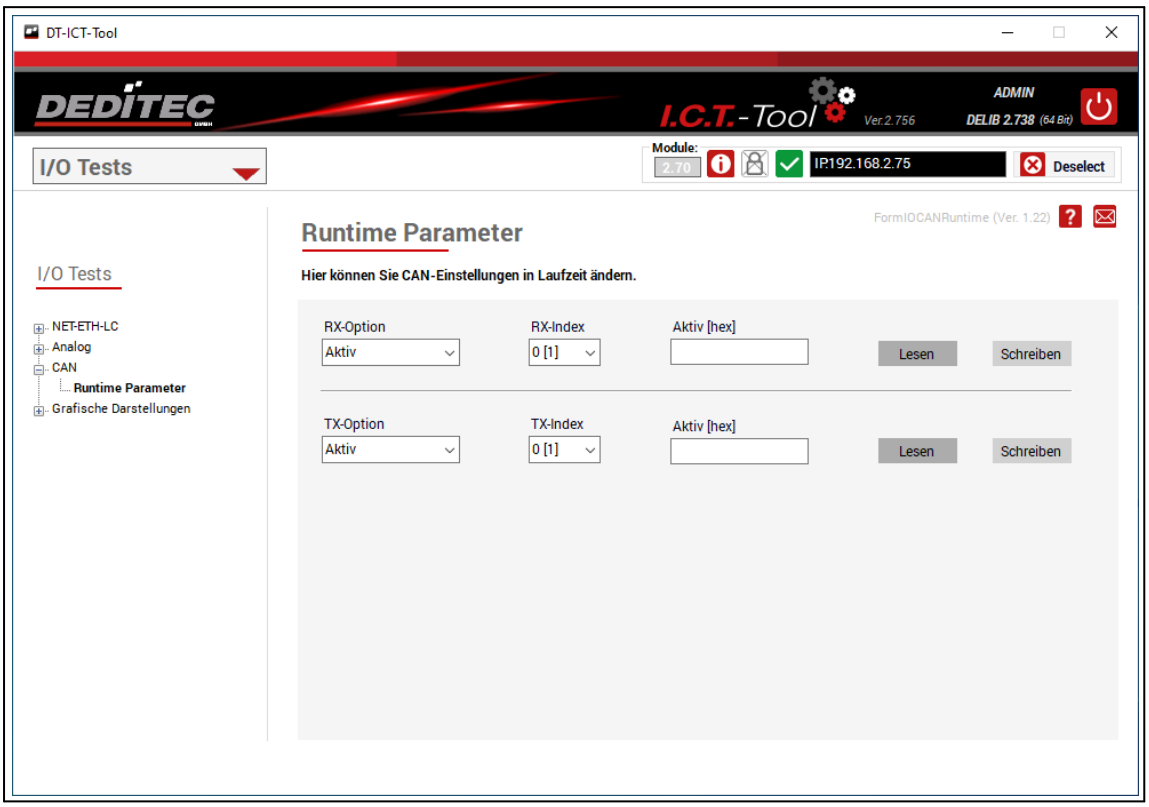
Reset

Deletes the current issue

5.6.10. CAN Runtime Parameter

Here you can make CAN settings on the TX and RX packets in runtime.

These changes are not saved in the module and are lost after a module restart.



Read

Reads out the current configuration of the respective parameter

Write

Saves the current configuration of the respective parameter

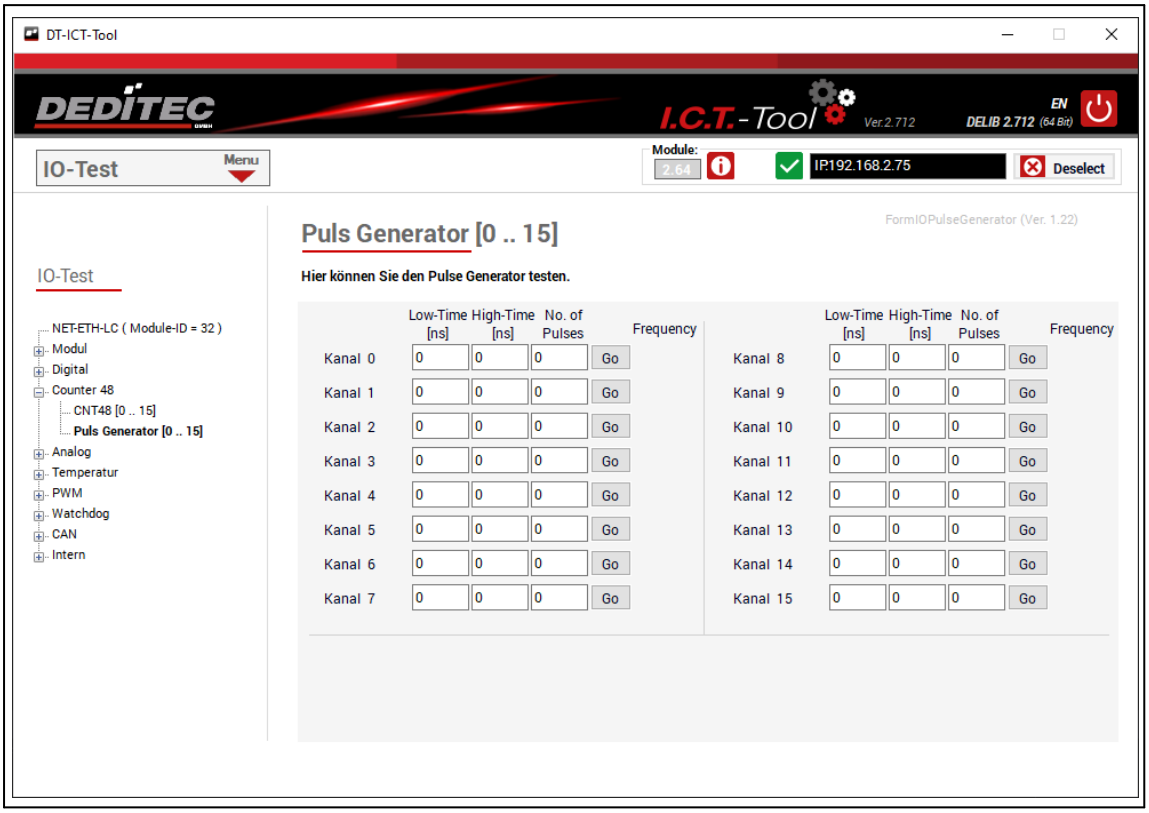
RX-/TX-Option

Here you can select which parameter you want to change.

The new value must be specified in HEX.

5.6.11. Puls Generator

You can test the Pulse Generator here.

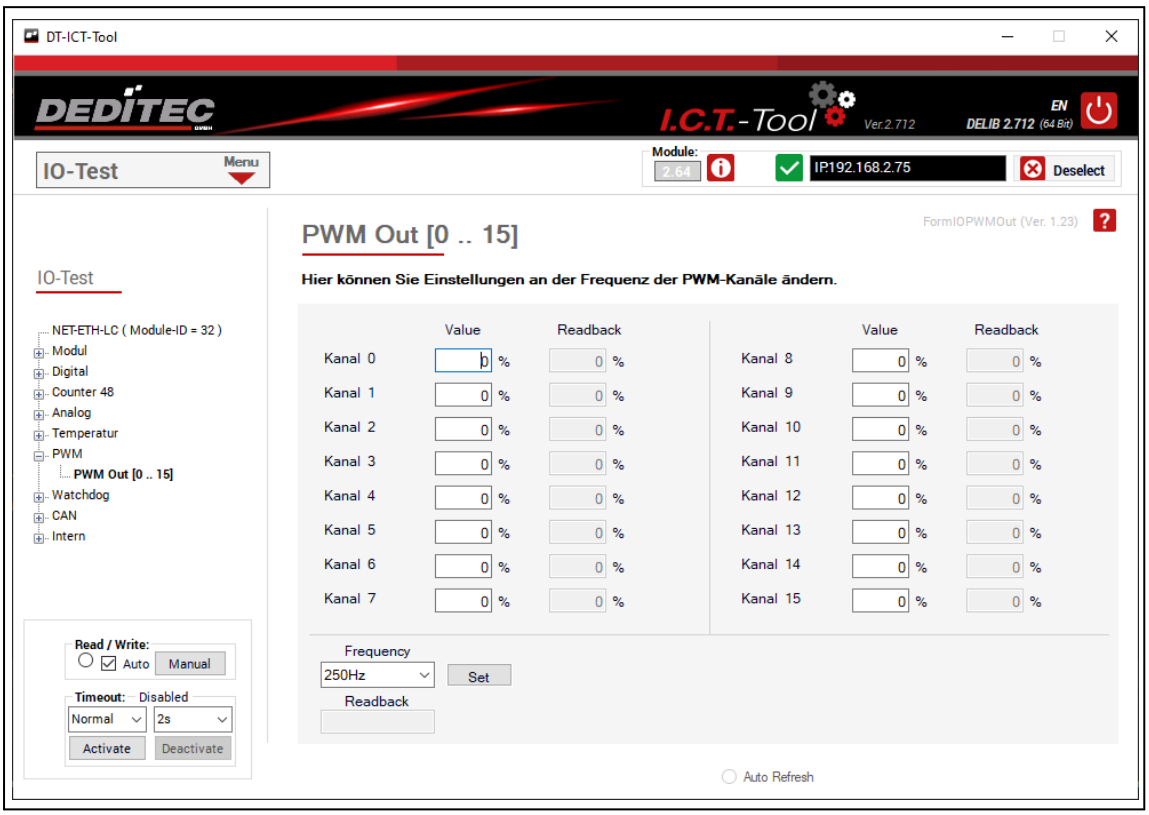


Low-Time[ns], High-Time[ns], No. of Pulses

The low time, high time and the number of pulses to be generated can be set here.

5.6.12. PWM Out

You can make settings to your PWM module here.



Value

Here you can set the PWM ratio of a channel.

PWM ratio from 0% to 100% in 1% steps

Readback (Value)

Indicates the PWM ratio of the respective channel.

Frequency

Here you can set the PWM frequency of the module

Readback (Frequency)

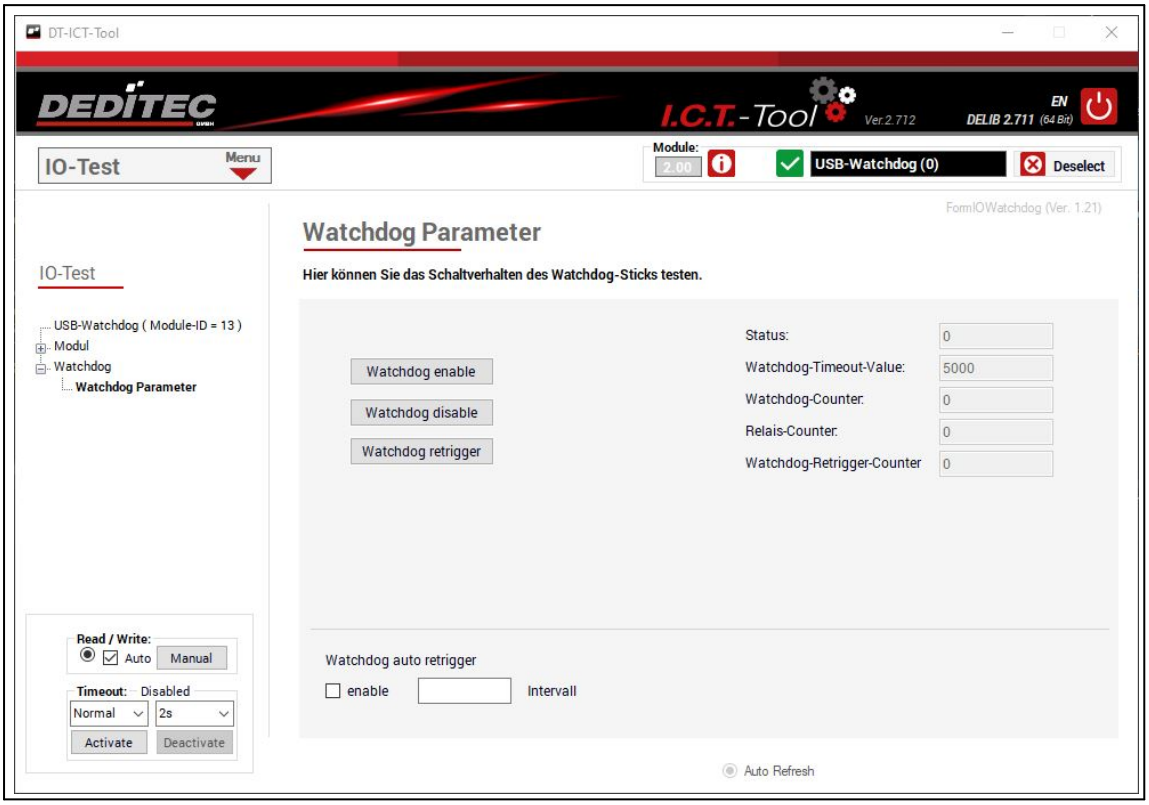
Returns the set PWM frequency.

Set mode for all channels

This function can be used to change the voltage/current range for all available channels at once.

5.6.13. Watchdog switching behavior

You can test the switching behavior of the watchdog stick here



Watchdog enable

Activates the watchdog

Watchdog disable

Deactivates the watchdog

Watchdog retrigger

Resets the watchdog stick status

Watchdog auto retrigger

Automatically resets the status of the watchdog stick after the specified time

5.7. Diagnose module

In the Diagnostics area, you can diagnose the functionality of your I/Os.

You can run function tests and display the evaluation.

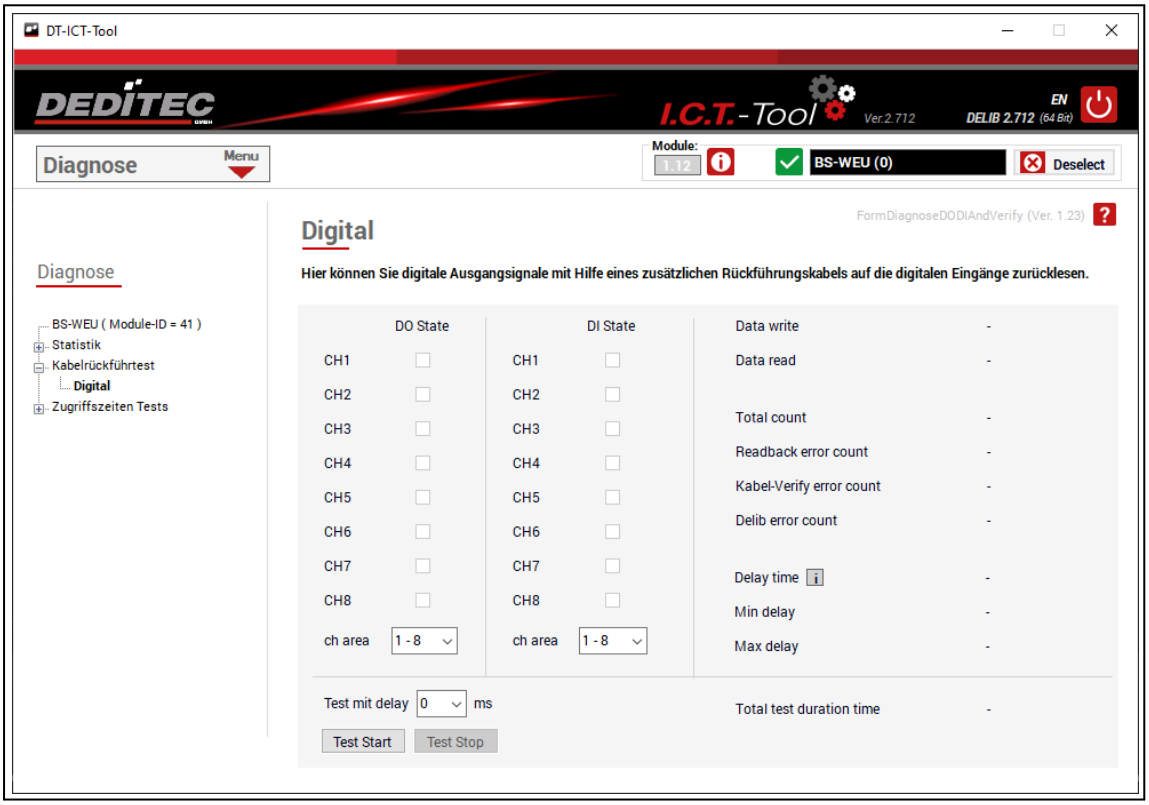
Additional cabling is required for the cable return tests. You can configure these cables yourself,

or order them directly from our store.

5.7.1. Digital cable return test

In this test, digital outputs are switched in 8 blocks. The voltage is then sent to the respective 8 digital inputs via a read-back cable and then read out.

It is important that the selected I/Os correspond to the wired inputs and outputs. **This test only works with an additional test cable.**



DO State

Displays the status of the current output. Switching is automatic.

DI State

Displays the status of the current input. Switching is automatic.

ch area

Specifies which I/Os are to be used for the test.

delay (in ms)

Here you can set the delay time to be waited between the write and read command.

This is required for relays that have a longer switching time.

Data write / Data read

Shows which value was written and read back. This happens automatically.

If necessary, check your wiring in the event of deviations

Total count

Shows how many commands have been executed.

Readback error count

Indicates whether there was an error during the test when reading back the written value directly. This value is independent of the wiring.

Kabel-Verify error count

Indicates whether there was a deviation between the value sent and received via the feedback cable during the test.

Delib error count

Indicates whether there was an error when executing a DELIB command during the test.

Delay Time, min, max

Displays the time required for a DapiDOSet8, DapiDOReadBack64 and DapiDIGet8 command.

Min/Max shows the lowest or highest value during the tests.

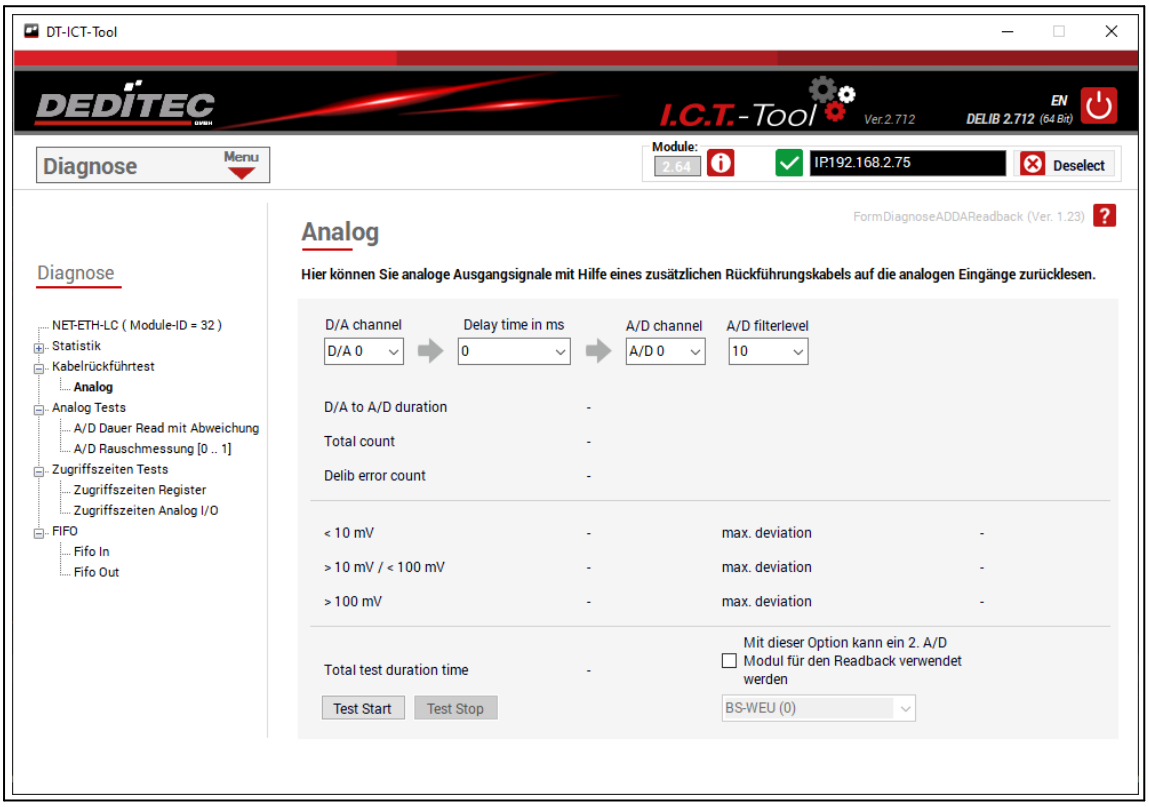
5.7.2. Analog cable return test

In this test, voltages in the range of +9/-9 V are applied to the selected D/A channels and read out at the corresponding A/D channels using a feedback cable.

It is important that the selected I/Os correspond to the wired inputs and outputs.

This test only works with an additional test cable.

An additional A/D module can be used as a reference module for this test.



D/A channel

Specifies the D/A channel to be used for the test

Delay time in ms

Specifies the time to wait between a write and read command. This is required for slower A/D converters.

A/D channel

Specifies the A/D channel to be used for the test

A/D filterlevel

Indicates which A/D filter level is to be used for this test

D/A to A/D duration

Shows how much time is required for a DapiDASet and DapiADGet command (incl. delay time)

Total count

Shows how many commands have been executed.

Delib error count

Indicates whether there was an error when executing a DELIB command during the test.

[< 10 mV], [>10 mV / < 100 mV], [> 100 mv]

Shows how often the respective deviation range has occurred.

Max. deviation

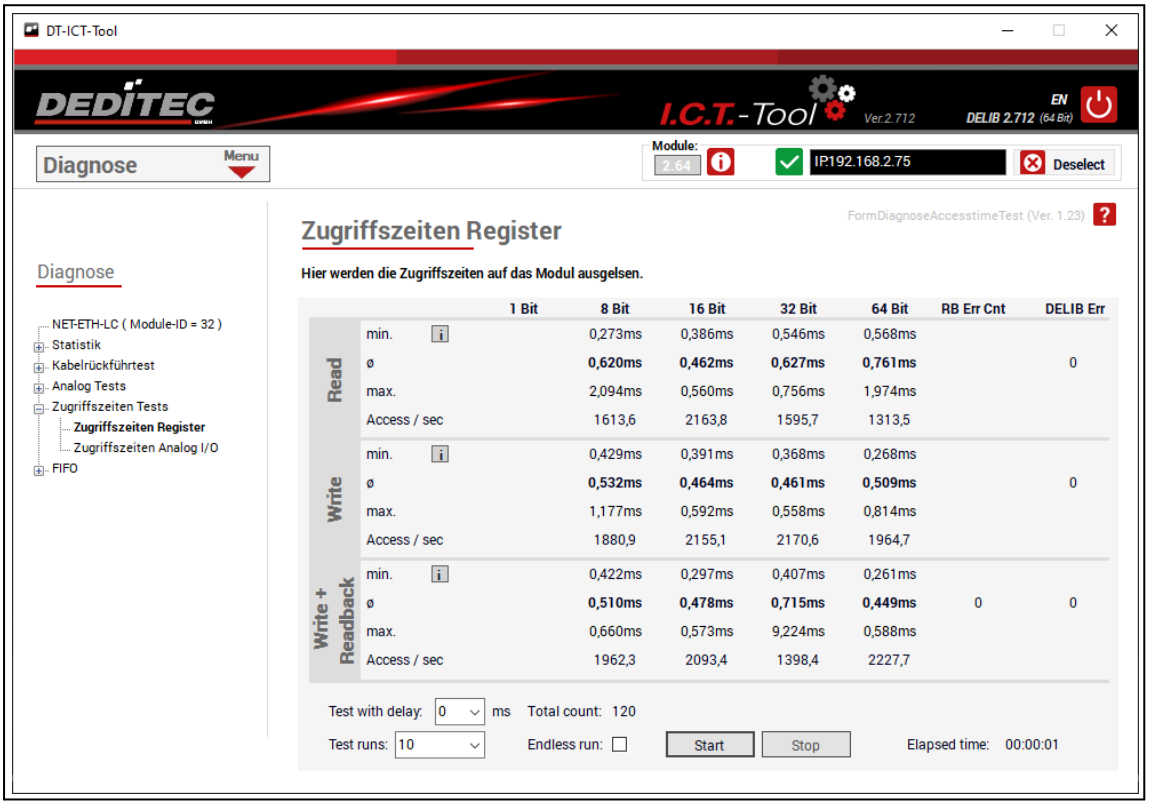
Displays the highest deviation within a deviation range.

5.7.3. Access time tests

Depending on the module configuration, you can perform different access time tests. In addition to register accesses, accesses to the digital and analog I/Os of your module can also be tested.

The time required to send commands to the module or to request values from the module is measured.

The test is carried out automatically with different bits or channels. Depending on the interface used, different time measurements may occur.



Test evaluation

The lowest, highest and average time required is determined here. The access time per second is also determined.

Test with delay

Here you can set the delay time between the commands with which the test is to be executed

Total Count

Indicates how many commands have been executed

Test runs

Here you can select how often commands should be executed per bit or channel

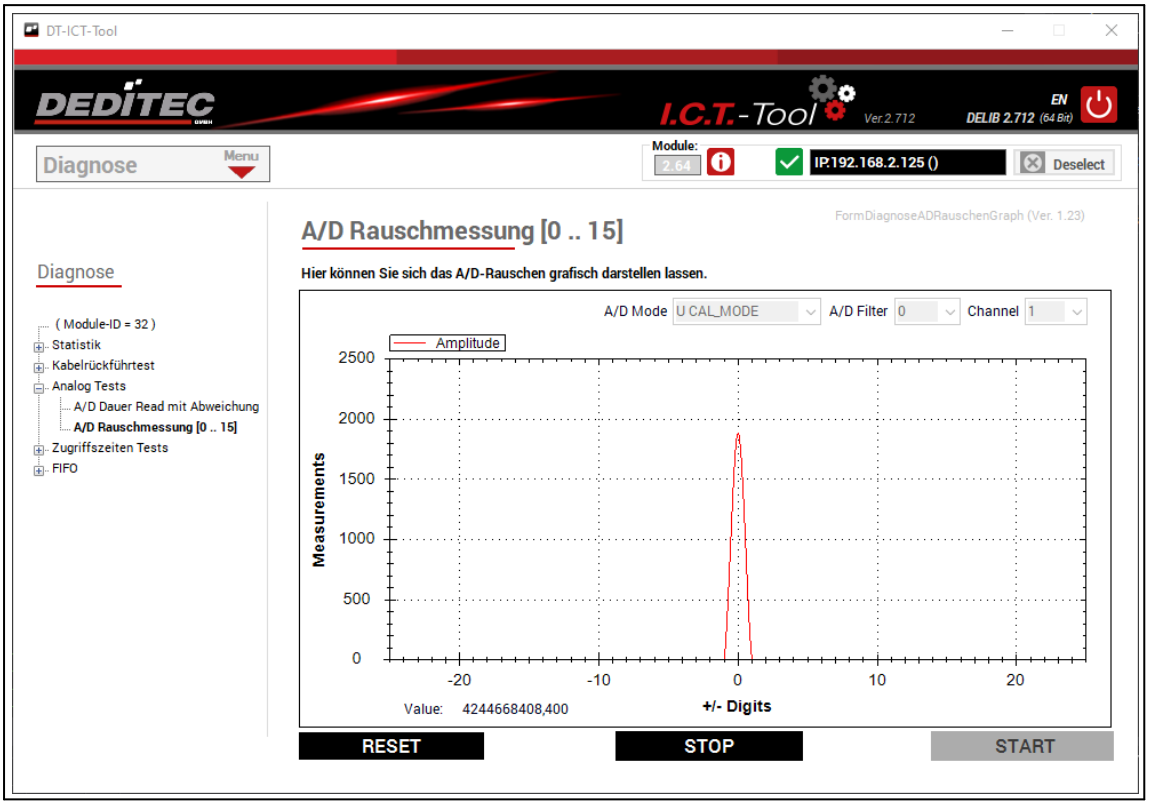
Endless run

If this box is ticked, the test starts again from the beginning after each run.

The evaluations are saved and adjusted.

5.7.4. A/D noise graph

Here you can display the noise of an A/D channel graphically.



A/D Mode

Specifies which A/D mode is to be used for the test

A/D Filter

Specifies which A/D filter is to be used for the test

Channel

Specifies which channel should be displayed

Start

Starts the graphical output

Stop

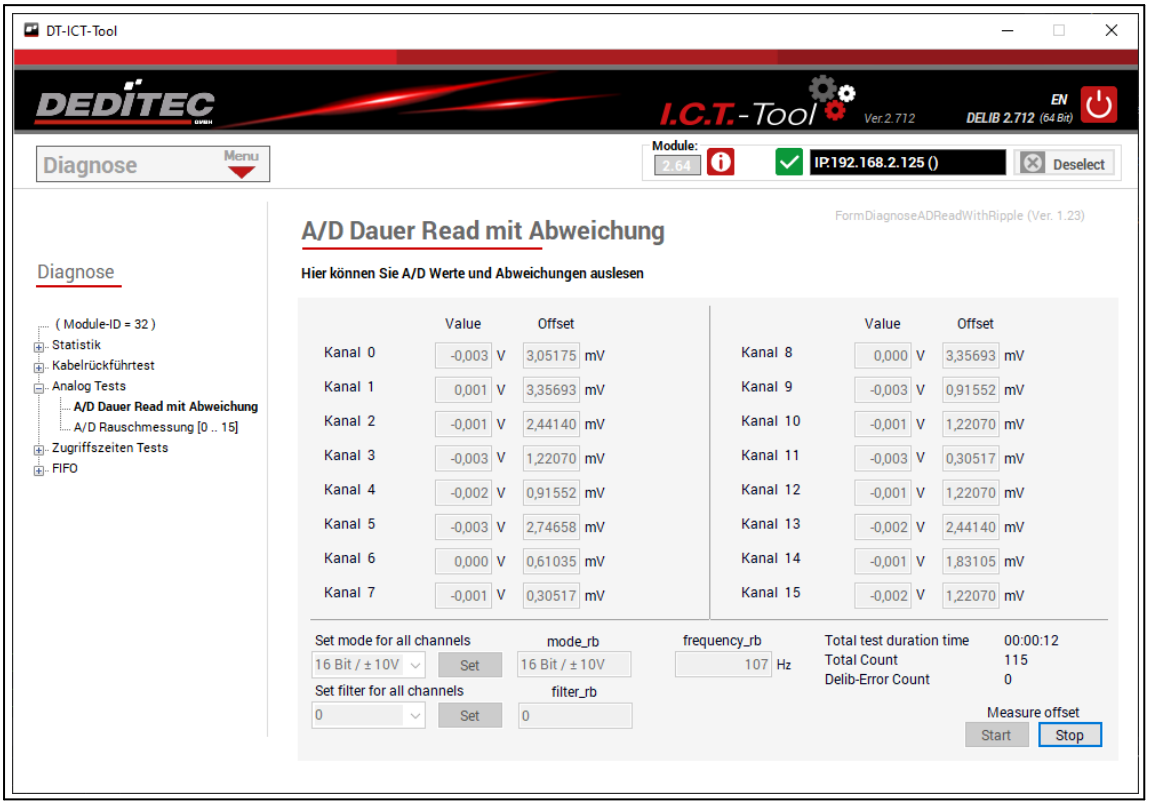
Ends the graphical output

Reset

Resets the graphical output

5.7.5. A/D read with ripple

Here you can read out the A/D channels of your module and display the deviations.



Value

Displays the voltage connected to the channel.

Offset

Displays the last deviation when measuring the voltage.

Set mode for all channels

Here you can change the A/D mode for all channels

mode_rb: Returns the written mode

Set filter for all channels

Here you can change the A/D filter for all channels

filter_rb: Returns the written filter level

Frequency_rb

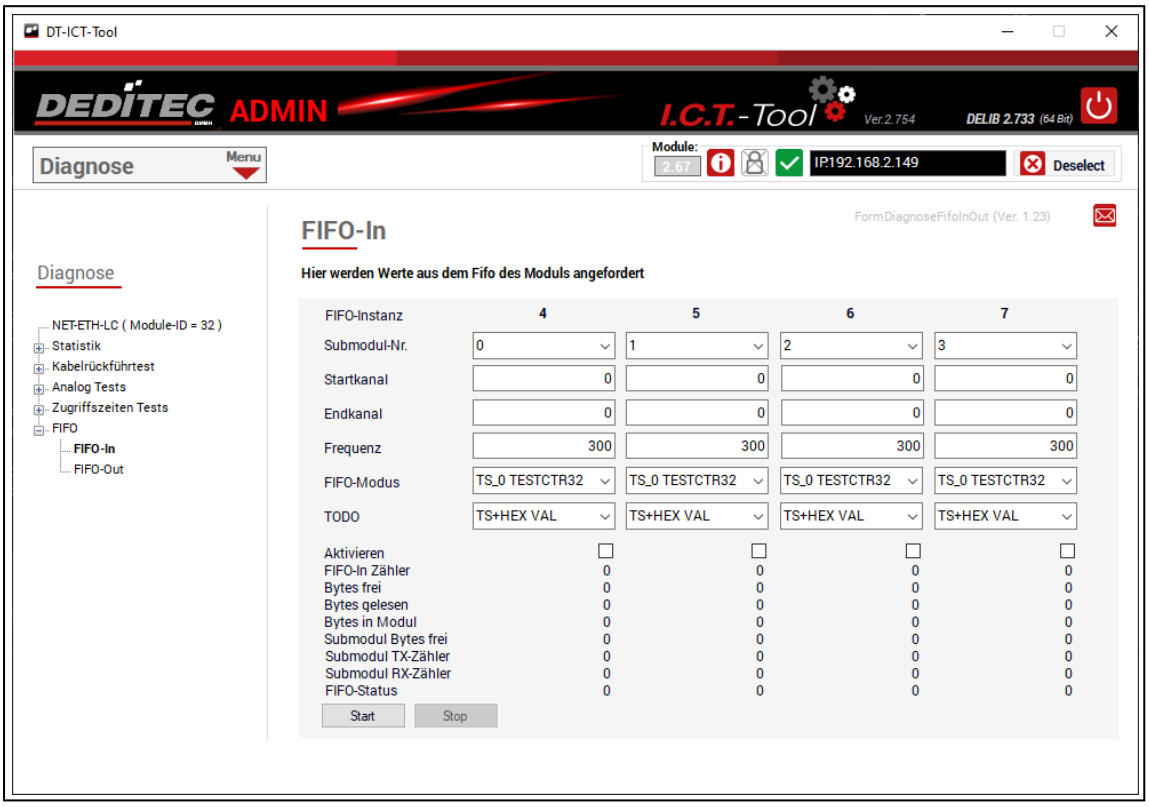
Indicates the current frequency with which the deviation is measured.

Start/Stop

Start or end the test

5.7.6. FIFO In/Out

Here you can test the FIFO-In or FIFO-Out of the module



FIFO instance

Indicates which instance is to be used (FIFO-Out: 0-3, FIFO-In: 4-7)

Submodule nr.

Specifies which submodule is to be used for the FIFO read/write

Start channel

Specifies the start channel for the FIFO

End channel

Specifies the end channel for the FIFO

Frequency

Specify the frequency at which the FIFO is to be read/written

FIFO mode

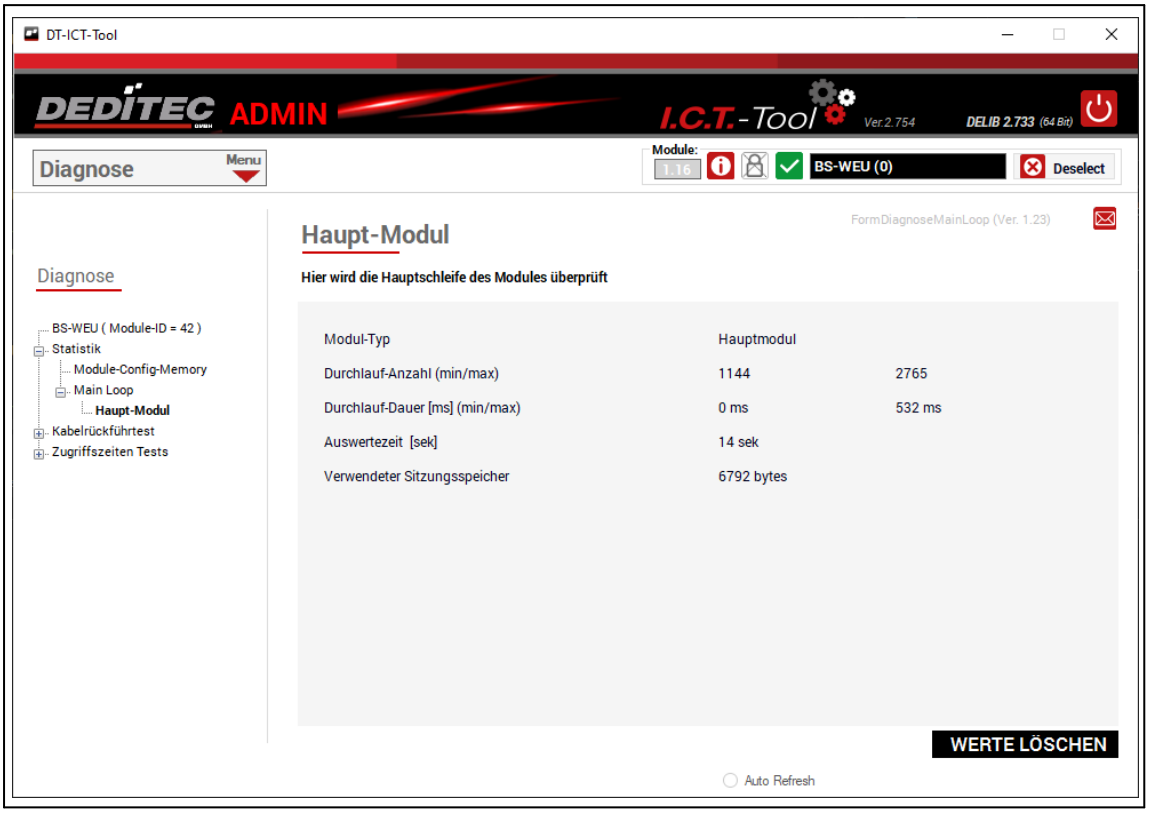
Specifies which mode is to be used for the FIFO

TODO

How the FIFO should be read/written

5.7.7. Main Loop

Information on the main loop of your module is displayed here



Module type

Indicates whether it is an interface/submodule.

Number of runs (min/max)

Shows how often the main loop has been run through since the module was started.

Run duration [ms] (min/max)

Shows how long a main loop run takes.

Evaluation time [sec]

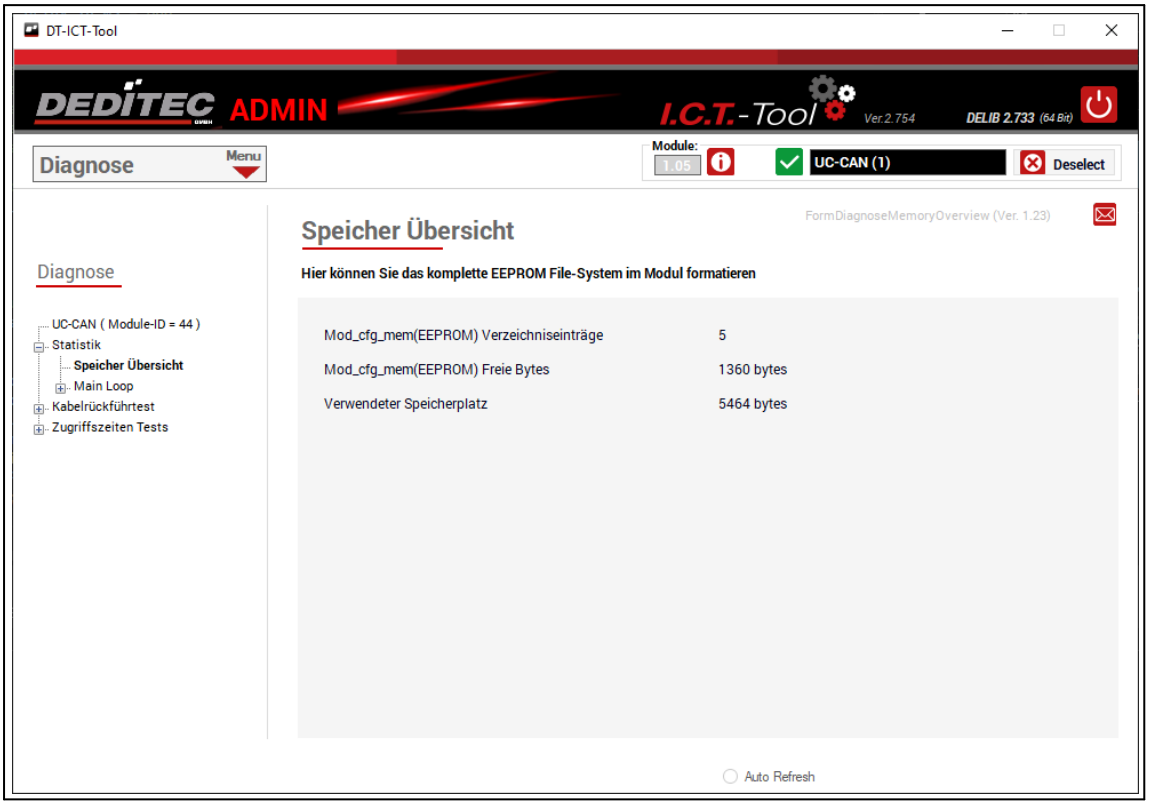
Shows how many seconds the measurement has been running.

Session memory used

Shows the amount of session memory used.

5.7.8. Module-Config-Memory

Information on the **Module-Config-Memory** is displayed here



Mod_cfg_mem(EEPROM) Directory entries

Indicates how many directory entries exist in the **Module-Config-Memory**

Mod_cfg_mem(EEPROM) Free bytes

Specifies the number of free bytes in the **Module-Config-Memory**

Sess memory used

Indicates the number of bytes used in the memory

5.8. Firmware update

5.8.1. About DEDITEC firmware

Most DEDITEC products have their own microcontroller. This processor is responsible for controlling all hardware processes.

The flasher integrated in the ICT-Tool can be used to subsequently change the firmware required for the processor. With this tool, the customer has the option of transferring newly published firmware versions directly to the module on site.

Note:

As new firmware versions usually "unlock" new functions for your product, we therefore recommend that you regularly update the firmware of your DEDITEC products.

Please note that the flashing process can only be carried out via one interface. This may vary depending on the product series.

Install Flashfiles manually

In some cases it is necessary to update the flash files manually, e.g. if administrator rights are not available on the PC.

Step 1

Download the latest version of the Flash files at

http://www.deditec.de/zip/deditec-flash_files.zip

Step 2

Unpack the downloaded ZIP archive into the following directory, depending on the DELIB installation:

x86

C:\Program Files(x86)\DEDITEC\DELIB\programs\

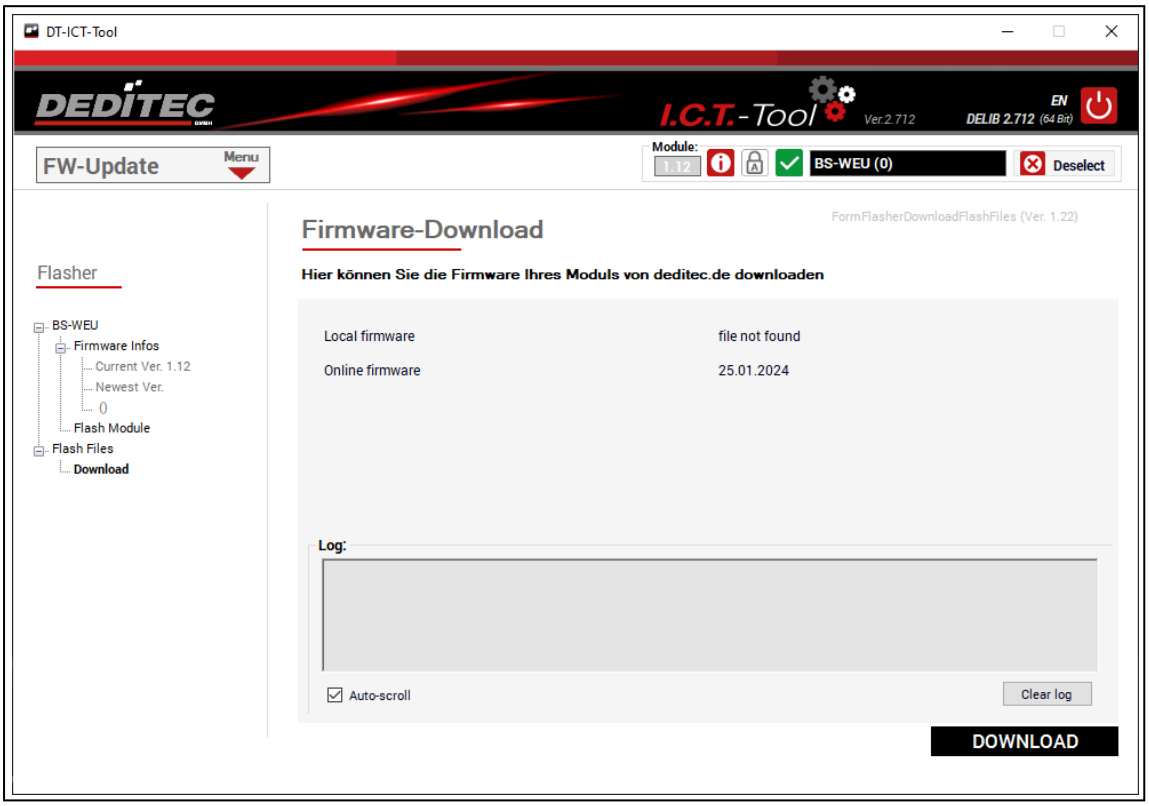
x64

C:\Program Files\DEDITEC\DELIB\programs

5.8.2. Update flash files via ICT tool

You can download the latest firmware for your module here.

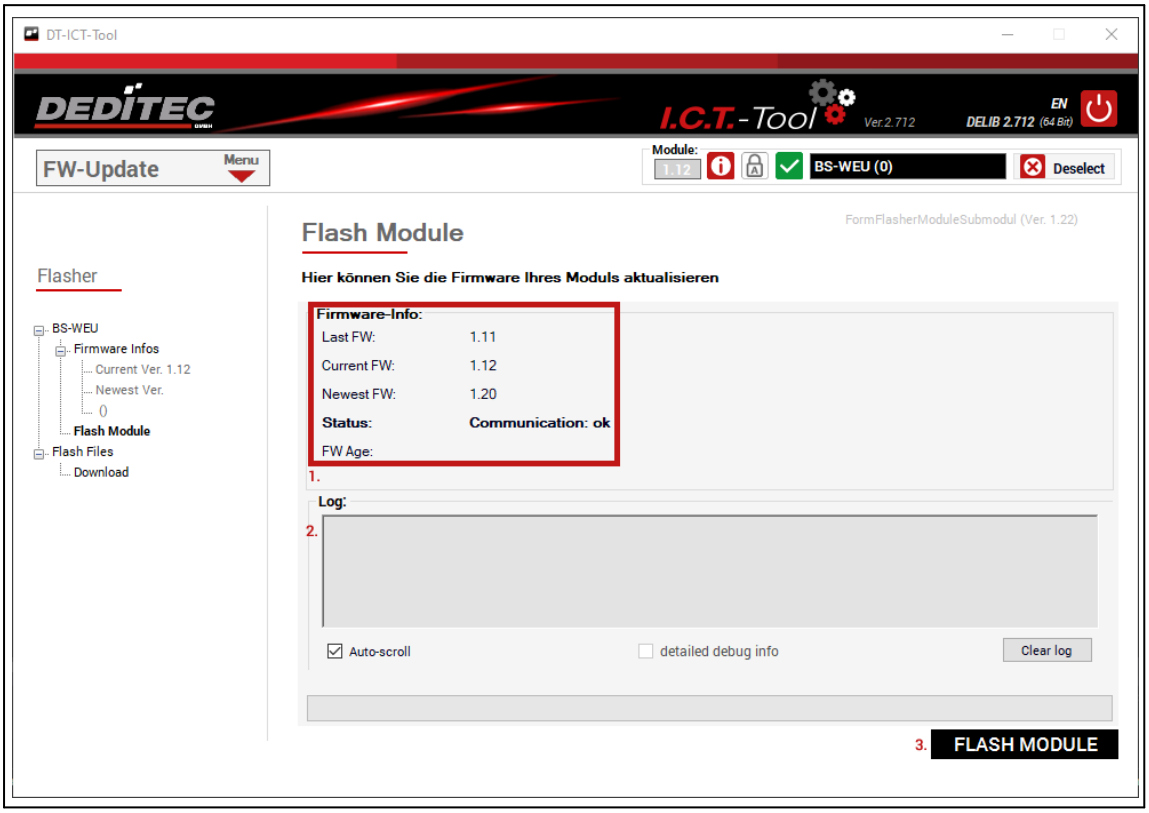
Under "Online firmware" you can see when the firmware was last updated on our homepage.



5.8.3. Perform firmware update

You can update the firmware of your interface and submodule here.

To benefit from the latest functions and prevent errors, we recommend that you always keep your DEDITEC product up to date.



1. Here you will find all the necessary information on the various firmware versions. Under "Newest FW" you can see whether there is a newer firmware version for your module.

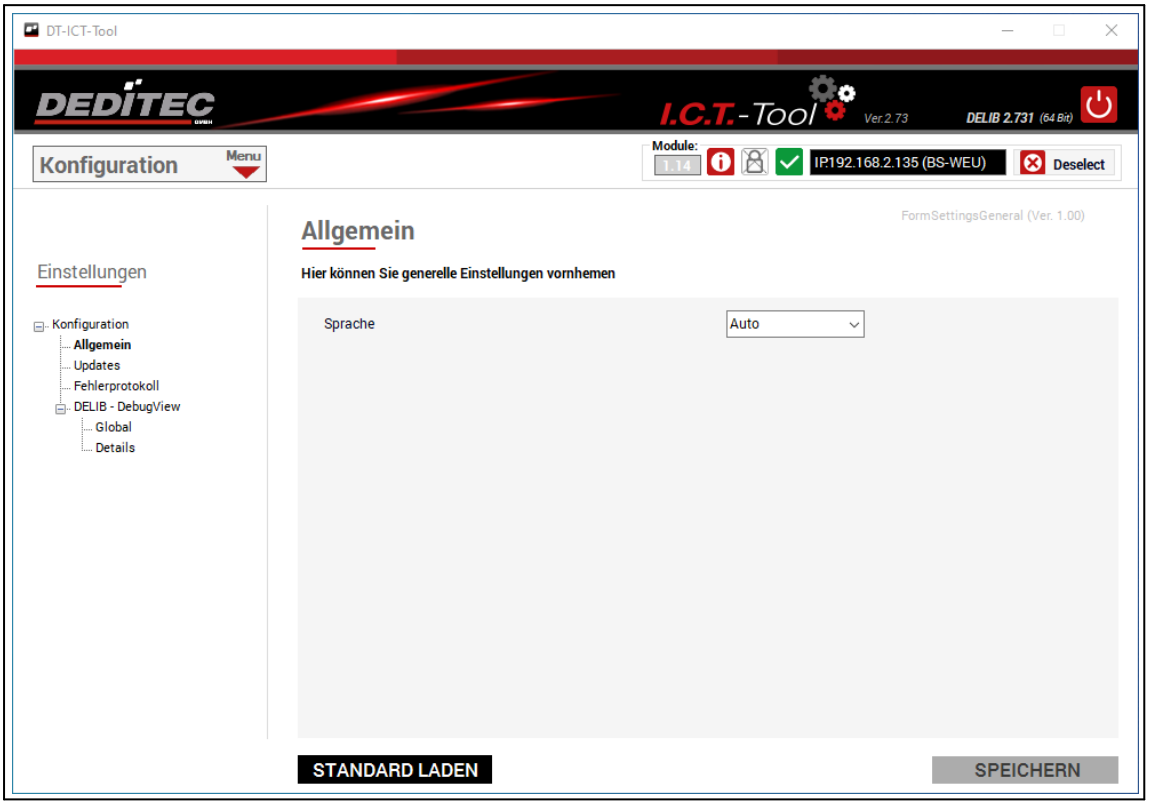
2. All important information, error and status messages that occurred during the flash process are entered here.

3. The flash process is started with the "Flash Module" button.

5.9. Settings

5.9.1. General

Here you can make general settings for the ICT-Tool.



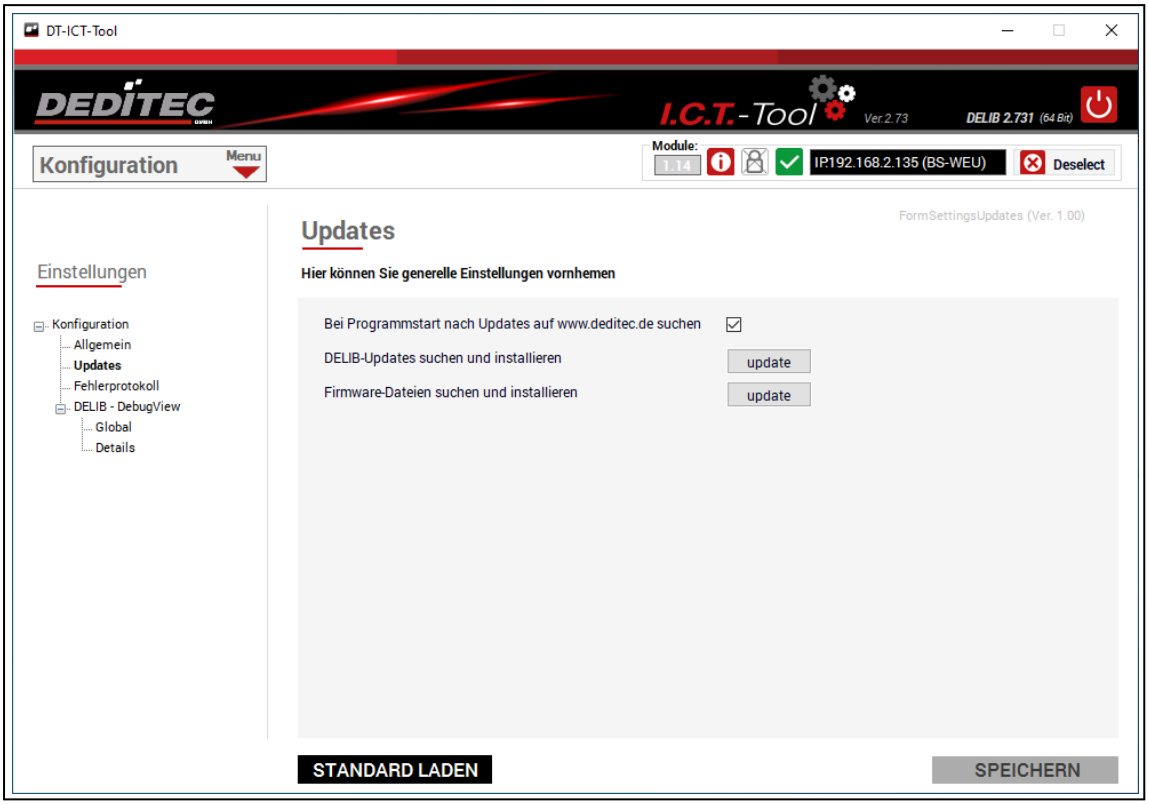
Language

Specifies the language in which the ICT-Tool should be displayed.

Currently, only German and English can be selected.

5.9.2. Updates

Here you can search for the latest DELIB and firmware updates.



Check for updates on www.deditec.de when starting the program

Specifies whether to check for updates when the program is started.

A functioning Internet connection is required for this.

Search for and install DELIB updates

Here you can search for the latest DELIB updates

Search for and install firmware files

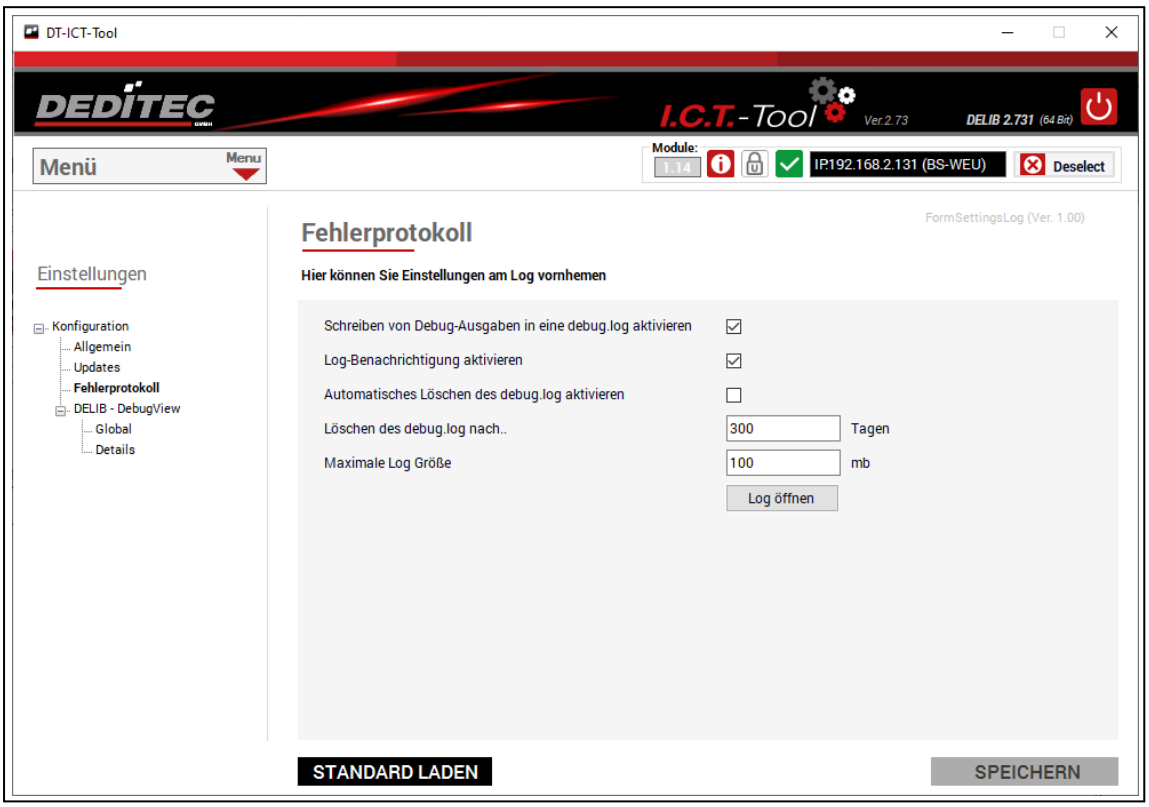
You can search for the latest firmware files here

5.9.3. Error log

Here you can make settings to the deditec_debug.log.

This is saved under the following path:

C:\Users\?USER?\AppData\Local\DEDITEC



Enable writing debug output to a debug.log

Specifies whether debug output should be written to an external .log

Activate log notification

Here you can set whether you want to be notified if the .log is too large or too old

cccDelete the debug.log after...

Here you can set whether the debug.log should be deleted automatically if the size or age of the file is exceeded

Delete the debug.log after...

Here you can set after how many days the log should be deleted

Maximum log size

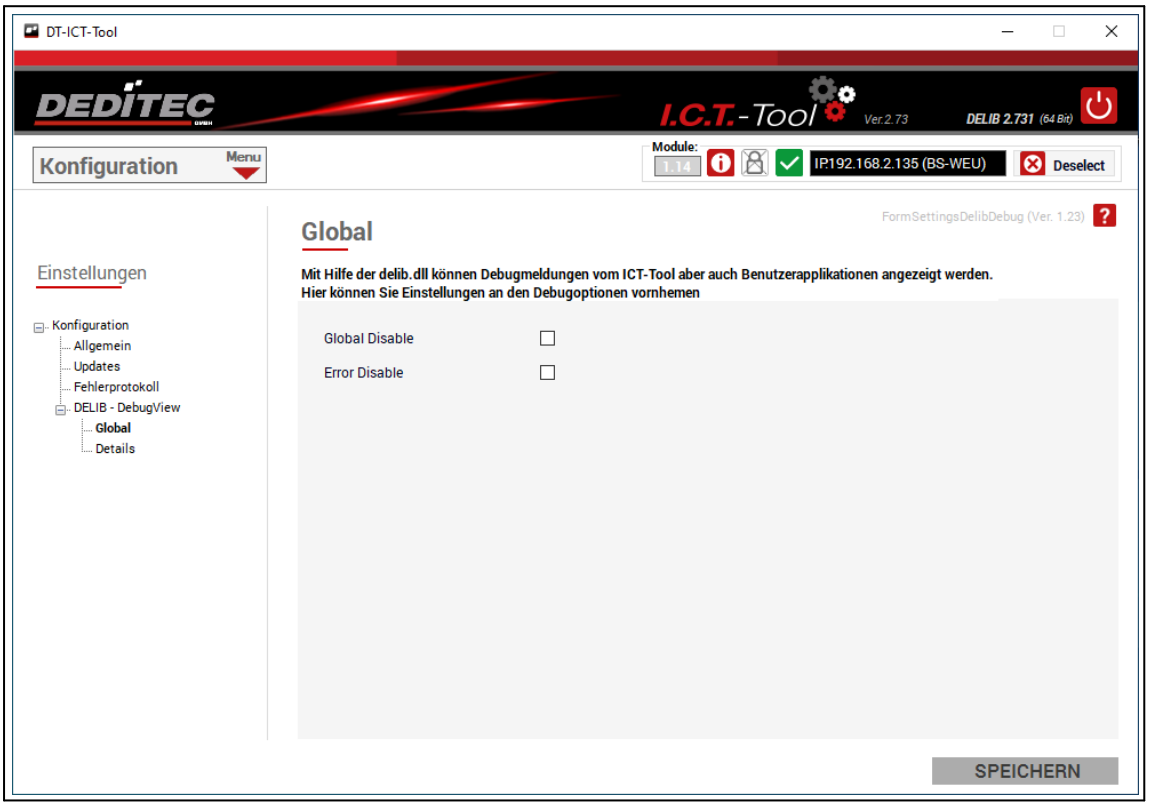
Here you can set how large the log may be before it is deleted

5.9.4. DELIB-DebugView-Global

You can activate the debug output here. You can also set whether any errors that occur should be displayed.

This is particularly helpful in the event of an error so that the module can be debugged more quickly.

The information is written to a separate .log file and can also be output by monitoring tools such as DEBUGView.

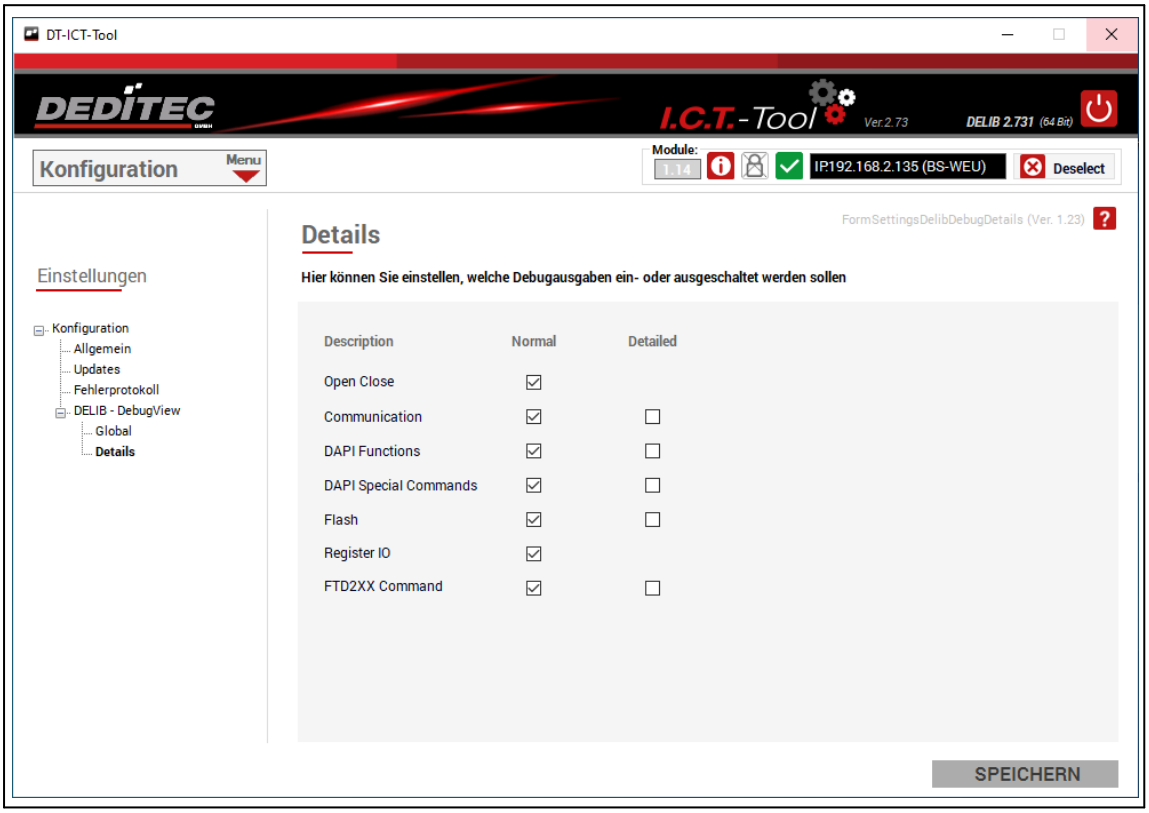


5.9.5. DELIB-DebugView-Details

Here you can set which and how detailed information should be recorded.

This is particularly helpful in the event of an error so that the module can be debugged more quickly.

The information is written to a separate .log file and can also be output by monitoring tools such as DEBUGView.



DELIB API Reference



6. DELIB API Reference

6.1. Available DEDITEC module IDs

Here you can find a list with all available module IDs.

This ID is needed for example to open the module and get a "handle".

More information can be found in the chapter **DapiOpenModule.**

Modul Name	ID
USB_Interface8	1
USB_CAN_STICK	2
USB_LOGI_500	3
USB_SER_DEBUG	4
RO_SER	5
USB_BITP_200	6
RO_USB1	7
RO_USB	7
RO_ETH	8
USB_MINI_STICK	9
USB_LOGI_18	10
RO_CAN	11
USB_SPI_MON	12
USB_WATCHDOG	13
USB_OPTOIN_8	14

Modul Name	ID
USB_RELAIS_8	14
USB_OPTOIN_8_RELAIS_8	15
USB_OPTOIN_16_RELAIS_16	16
USB_OPTOIN_32	16
USB_RELAIS_32	16
USB_OPTOIN_32_RELAIS_32	17
USB_OPTOIN_64	17
USB_RELAIS_64	17
BS_USB_8	15
BS_USB_16	16
BS_USB_32	17
USB_TTL_32	18
USB_TTL_64	18
RO_ETH_INTERN	19
BS_SER	20
BS_CAN	21
BS_ETH	22
NET_ETH	23
RO_CAN2	24
RO_USB2	25

Modul Name	ID
RO_ETH_LC	26
ETH_RELAIS_8	27
ETH_OPTOIN_8	27
ETH_O4_R4_ADDA	28
ETHERNET_MODULE	29
ETH_TTL_64	30
NET_USB2	31
NET_ETH_LC	32
NET_USB1	33
NET_SER	34
NET_CAN_OPEN	35
NET_RAS_PI	36
USB_CANOPEN_STICK	37
ETH_CUST_0	38
WEU_RELAIS_8	39
WEU_OPTO_8	39
WEU_E_RELAIS_8	40
BS_WEU	41
BS_WEU_E	42
UC_USB	43

Modul Name	ID
UC_CAN	44
BS_USB2	45
CAN_BOX	46

6.2. DELIB directory structure

After successful installation the following directory tree is available:

```
$DELIB_DIR
  ¦
  ¦ - > include              Includes for programming languages
  ¦ - > lib                  Library
  ¦ - > lib\bc               Borland Compiler Library
  ¦ - > prod_pics            Product images
  ¦ - > programs             Modul-Testprogramme
  +- > USB-Driver            Driver for USB modules
```

Please note that the "$DELIB_DIR" folder may vary, depending on the operating system and DELIB version.

DELIB Installation	Windows Installation	Path
32 Bit	32 Bit	C:\Programs\DEDITEC\DELIB\
32 Bit	64 Bit	C:\Programs (x86)\DEDITEC\DELIB\
64 Bit	64 Bit	C:\Programs\DEDITEC\DELIB64\

In addition, the following files are installed in the Windows System folder:
$SYSDIR\delib.dll, or $SYSDIR\delib64.dll (32 Bit, or 64 Bit DELIB version)
$SYSDIR\delibJNI.dll, or $SYSDIR\delibJNI64.dll (32 Bit, or 64 Bit DELIB version)

$SYSDIR\ftbusui.dll
$SYSDIR\ftd2xx.dll
$SYSDIR\FTLang.dll
$SYSDIR\drivers\ftdibus.sys

Please note that the "$SYSDIR" folder may vary, depending on the operating system and DELIB version.

DELIB Installation	Windows Installation	Path
32 Bit	32 Bit	C:\Windows\System32
32 Bit	64 Bit	C:\Windows\SysWOW64
64 Bit	64 Bit	C:\Windows\System32

6.2.1. Include directory

The include directory created for the DELIB contains the files which describe the corresponding library functions. These are given for the programming languages C (.h), Delphi (.pas) and Visual Basic (.bas).

6.2.2. Library directory

This contains the file "DELIB.lib". It serves as a link for compiling own programs which use the "DELIB.dll".

6.2.3. Library directory for Borland

For Borland Compiler there is a separate DELIB.lib, which is located in the subdirectory "bc". This also serves as a link for compiling own programs that use the "DELIB.dll".

6.2.4. Environment variables

Two environment variables point to important directories that contain files for the C, Delphi and Visual Basic programming languages.

“DELIB_INCLUDE” points to the include directory.

%DELIB_INCLUDE% à c:\Programs\DEDITEC\DELIB\include"

“DELIB_LIB" points to the library directory.

%DELIB_LIB% à c:\ Programs\DEDITEC\DELIB\lib

6.3. Administrative functions

6.3.1. DapiOpenModule

Description

This function opens a specific module.

Definition

ULONG DapiOpenModule(ULONG moduleID, ULONG nr);

Parameter

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

nr=Specifies which one (in case of multiple modules) should be opened.

nr=0 → 1. Module

nr=1 → 2. Module

Return value

handle=Corresponding handle for the module

handle=0 → Module was not found

Comment

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

Programming example

```
// Open USB module
handle = DapiOpenModule(RO_USB1, 0);
printf("handle = %x\n", handle);
if (handle==0)
{
// USB module was not found
printf("Module could not be opened\n");
return;
}
```

6.3.2. DapiCloseModule

Description

This command closes an open module.

Definition

ULONG DapiCloseModule(ULONG handle);

Parameter

handle=This is the handle of an open module.

Return value

None

Programming example

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

6.3.3. DapiGetDELIBVersion

Description

This function returns the installed DELIB version.

Definition

ULONG DapiGetDELIBVersion(ULONG mode, ULONG par);

Parameters

mode=Mode used to read the version (must be 0).

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

Return value

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

Programming example

```
version = DapiGetDELIBVersion(0, 0);
//With version 1.32 installed, version = 132(hex)
```

6.3.4. DapiSpecialCMDGetModuleConfig

Description

Diese Funktion gibt die Hardwareausstattung (Anzahl der Ein- und Ausgangskanäle) des Moduls zurück.

Definition

ULONG DapiSpecialCommand(ULONG handle,

DAPI_SPECIAL_CMD_GET_MODULE_CONFIG, par, 0, 0);

Parameter

handle=Dies ist der handle eines offenen Moduls

Querying the number of digital input channels

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI

Query number of digital input flip-flops

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI_FF

Query number of digital input counters (16-bit counter)

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI_COUNTER

Query number of digital input counters (48-bit counter)

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_CNT48

Querying the number of digital output channels

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO

Querying the number of digital pulse generator outputs

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_PULSE_GEN

Querying the number of digital PWM outputs

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_PWM_OUT

Querying the number of digital input/output channels

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DX

Querying the number of analog input channels
par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_AD

Querying the number of analog output channels
par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DA

Query number of temperature channels
par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_TEMP

Query number of stepper channels
par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_STEPPER

Return value

Querying the number of digital input channels
return=number of digital input channels

Query number of digital input flip-flops
return=number of digital input flip-flops

Query number of digital input counters (16-bit counter)
return=number of digital input counters (16-bit counter)

Query number of digital input counters (48-bit counter)
return=number of digital input counters (48-bit counter)

Querying the number of digital output channels
return=number of digital output channels

Querying the number of digital pulse generator outputs
return=number of digital pulse generator outputs

Querying the number of digital PWM outputs
return=number of digital PWM outputs

Querying the number of digital input/output channels
return=number of digital input/output channels

Querying the number of analog input channels
return=number of analog input channels

Querying the number of analog output channels
return=number of analog output channels

Query number of temperature channels
return=number of temperature channels

Query number of stepper channels
return=number of stepper channels

Programmierbeispiele

```
ret=DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI, 0, 0);
//Returns the number of digital input channels
ret=DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO, 0, 0);
//Returns the number of digital output channels
ret=DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DX, 0, 0);
//Returns the number of digital input/output channels
ret=DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_AD, 0, 0);
//Returns the number of analog input channels
ret=DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DA, 0, 0);
//Returns the number of analog output channels
ret=DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_STEPPER, 0, 0);
//Returns the number of stepper channels
```

6.3.5. DapiOpenModuleEx

Description

This function opens specifically a module with Ethernet interface. The parameters IP address, port number, the duration of the timeout and the encryption type can be determined.

The module is opened independently of the settings made in the ICT-Tool.

Definition

ULONG DapiOpenModuleEx(ULONG moduleID, ULONG nr, unsigned char exbuffer, 0);*

Parameter

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

nr = Specifies which module (in case of multiple modules) should be opened.

nr = 0 → 1. Module

nr = 1 → 2. Module

exbuffer = Buffer for IP address, port number, duration of timeout and encryption type

Return value

handle = Corresponding handle for the module

handle = 0 → Module was not found

Comment

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

This command is supported by all modules with Ethernet interface.

Universelle Ethernet moduleID

The moduleID:

ETHERNET_MODULE = 29

is a universal Ethernet moduleID and can be used to address any Ethernet product.

Encryption Type

The following encryption types are available:

DAPI_OPEN_MODULE_ENCRYPTION_TYPE_NONE = 0

DAPI_OPEN_MODULE_ENCRYPTION_TYPE_NORMAL = 1

DAPI_OPEN_MODULE_ENCRYPTION_TYPE_ADMIN = 2

Programming example

```
// Open ETH-Module with parameter
DAPI_OPENMODULEEX_STRUCT open_buffer;

strcpy((char*) open_buffer.address, "192.168.1.10");
open_buffer.portno = 0;
open_buffer.timeout = 5000;
open_buffer.encryption_type = 0;

handle = DapiOpenModuleEx(RO_ETH, 0, (unsigned char*)
&open_buffer, 0);
printf("Module handle = %x\n", handle);
```

6.4. Error handling

6.4.1. DapiGetLastError

Description

This function returns the last captured error. If an error has occurred, it must be cleared with **DapiClearLastError()** otherwise every call to DapiGetLastError() will return the "old" error.

If several modules are to be used, it is recommended to use **DapiGetLastErrorByHandle()**.

Definition

ULONG DapiGetLastError();

Parameter

None

Return value

Error code

0=no error. (see delib_error_codes.h)

Programming example

```
BOOL IsError()
{
    unsigned char msg[500];
    unsigned long error_code = DapiGetLastError();
    if (error_code != DAPI_ERR_NONE)
    {
        DapiGetLastErrorText((unsigned char*) msg,
sizeof(msg));
        printf("Error Code = 0x%x * Message = %s\n",
error_code, msg);
        DapiClearLastError();
        return TRUE;
    }
    return FALSE;
}
```

6.4.2. DapiGetLastErrorText

Description

This function reads the text of the last captured error. If an error has occurred, it must be cleared with **DapiClearLastError()** otherwise every call to DapiGetLastErrorText() will return the "old" error.

Definition

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

Parameter

msg = Buffer for the text to be received

msg_length = Length of the text buffer

Programmierbeispiel

```
BOOL IsError()
{
    unsigned char msg[500];
    unsigned long error_code = DapiGetLastError();

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

        DapiClearLastError();

        return TRUE;
    }

    return FALSE;
}
```

6.4.3. DapiClearLastError

Description

This function clears the last error captured with **DapiGetLastError()**.

Definition

void DapiClearLastError();

Parameter

None

Return value

None

Programming example

```
BOOL IsError()
{
    unsigned char msg[500];
    unsigned long error_code = DapiGetLastError();

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

        DapiClearLastError();

        return TRUE;
    }

    return FALSE;
}
```

6.4.4. DapiGetLastErrorByHandle

Description

This function returns the last captured error of a specific module (handle). If an error has occurred, it must be cleared with **DapiClearLastErrorByHandle()** otherwise every call to DapiGetLastErrorByHandle() returns the "old" error.

Definition

ULONG DapiGetLastErrorByHandle(ULONG handle);

Parameter

handle=This is the handle of an open module.

Return value

Error code

0=no error. (see delib_error_codes.h)

Programmierbeispiel

```
BOOL IsError(ULONG handle)
{
    unsigned long error_code =
DapiGetLastErrorByHandle(handle);

    if (error_code != DAPI_ERR_NONE)
    {
        printf("Error detected on handle 0x%x - Error
Code = 0x%x\n", handle, error_code);

        DapiClearLastErrorByHandle(handle);

        return TRUE;
    }

    return FALSE;
}
```

6.4.5. DapiClearLastErrorByHandle

Description

This function clears the last error of a specific module (handle) captured with **DapiGetLastErrorByHandle().**

Definition

void DapiClearLastErrorByHandle();

Parameter

handle=This is the handle of an open module.

Return value

None

Programming example

```
BOOL IsError(ULONG handle)
{
    unsigned long error_code =
DapiGetLastErrorByHandle(handle);

    if (error_code != DAPI_ERR_NONE)
    {
        printf("Error detected on handle 0x%x - Error
Code = 0x%x\n", handle, error_code);

        DapiClearLastErrorByHandle(handle);

        return TRUE;
    }

    return FALSE;
}
```

6.5. Read digital inputs

6.5.1. DapiDIGet1

Description

This command reads a single digital input.

Definition

ULONG DapiDIGet1(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module.

ch=Indicates the number of the input to be read (0, 1, 2, 3, ..)

Return value

State of the input (0/1)

6.5.2. DapiDIGet8

Description

This command reads 8 digital inputs simultaneously.

Definition

ULONG DapiDIGet8(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Gives the number of the input to read from (0, 8, 16, 24, ..)

Return value

State of the read inputs

6.5.3. DapiDIGet16

Description

This command reads 16 digital inputs simultaneously.

Definition

ULONG DapiDIGet16(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an opened module.

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

Return value

State of the read inputs

6.5.4. DapiDIGet32

Description

This command reads 32 digital inputs simultaneously.

Definition

ULONG DapiDIGet32(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an opened module.

ch=Gives the number of the input to read from (0, 32, 64, ..)

Return value

State of the read inputs

Programming example

```
unsigned long data;
// ------------------------------------
// Read a value from the inputs (input 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();
// ------------------------------------
// Read a value from the inputs (input 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();
```

6.5.5. DapiDIGet64

Description

This command reads 64 digital inputs simultaneously.

Definition

ULONG DapiDIGet64(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Gives the number of the input to read from (0, 64, ..)

Return value

State of the read inputs

6.5.6. DapiDIGetFF32

Description

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

Definition

ULONG DapiDIGetFF32(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

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

Return value

State of 32 input state changes

6.5.7. DapiDIGetCounter

Description

This command reads the input counter of a digital input.

Definition

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

Parameter

handle=This is the handle of an open module

ch=Gives the number of the input to read from

mode=0 (normal counting function)

mode=DAPI_CNT_MODE_READ_WITH_RESET (Read counter and reset counter directly)

mode=DAPI_CNT_MODE_READ_LATCHED (Read the stored counter value)

Return value

Specification of the counter value

Programming example

```
value = DapiDIGetCounter(handle, 0 , 0);
// Counter of DI Chan 0 is read

value = DapiDIGetCounter(handle, 1 , 0);
// Counter of DI Chan 1 is read


value = DapiDIGetCounter(handle, 8 ,0);
// Counter of DI Chan 8 is read

value = DapiDIGetCounter(handle, 0 ,
DAPI_CNT_MODE_READ_WITH_RESET);
// Counter of DI Chan 0 is read AND reset

value = DapiDIGetCounter(handle, 1 ,
DAPI_CNT_MODE_READ_LATCHED);
// Readout of the stored meter reading from DI Chan 1
```

6.5.8. DapiSpecialCounterLatchAll

Description

This command stores the counter readings of all input counters simultaneously in a buffer (latch).

This way, all counter readings of the latch can subsequently be read out one after the other.

A special feature here is that a simultaneous "freezing" of the counter readings is possible and the frozen states (latch) can then be read out individually one after the other.

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_COUNTER, DAPI_SPECIAL_COUNTER_LATCH_ALL, 0, 0);

Parameter

None

Return value

None

Comment

Modules that are supported by these commands can be found in our **DELIB overview table**.

Programming example

```
DapiSpecialCommand(ULONG handle,
DAPI_SPECIAL_CMD_COUNTER,
DAPI_SPECIAL_COUNTER_LATCH_ALL, 0, 0);
```

6.5.9. DapiSpecialCounterLatchAllWithReset

Description

This command stores the counter readings of all input counters simultaneously in a buffer (latch).

In addition, the counter readings of the input counters are reset afterwards.

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_COUNTER, DAPI_SPECIAL_COUNTER_LATCH_ALL_WITH_RESET, 0, 0);

Parameter

None

Return value

None

Comment

Modules supported by these commands can be found in our **DELIB overview table**.

Programming example

```
DapiSpecialCommand(ULONG handle,
DAPI_SPECIAL_CMD_COUNTER,
DAPI_SPECIAL_COUNTER_LATCH_ALL_WITH_RESET, 0, 0);
```

6.5.10. DapiSpecialDIFilterValueSet

Description

This command sets an input filter in [ms], in which time interval interference pulses at digital input channels are filtered.

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,

DAPI_SPECIAL_DI_FILTER_VALUE_SET, ULONG time_ms, 0);

Parameter

handle=This is the handle of an open module

time_ms=time interval [ms], when digital input channels are read.

Remark

Default value: 0ms

Value range: 0(=off) , 1(ms) - 254(ms)

Modules that are supported by these commands can be found in our **DELIB overview table**.

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FILTER_VALUE_SET, 5, 0);
// Sets the time interval to 5ms

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FILTER_VALUE_SET, 150, 0);
// Sets the time interval to 150ms
```

6.5.11. DapiSpecialDIFilterValueGet

Description

This command returns the previously set value of the time interval for filtering noise pulses at digital input channels in [ms].

Definition

ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI, DAPI_SPECIAL_DI_FILTER_VALUE_GET, 0, 0);

Parameter

handle=This is the handle of an open module

Return value

Time [ms]

Comment

Modules that are supported by these commands can be found in our **DELIB overview table**.

Programming example

```
value = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FILTER_VALUE_GET, 0, 0);
//Returns the time interval for reading out the digital input channels.
```

6.5.12. Dapi_Special_DI_FF_Filter_Value_Get

Description

This command returns the predefined value of the time interval for sampling the input flip-flops and the input counters in [ms].

Definition

ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI, DAPI_SPECIAL_DI_FF_FILTER_VALUE_GET, 0, 0);

Parameter

handle=This is the handle of an open module

Return value

Time [ms]

Comment

Modules that are supported by these commands can be found in our **DELIB overview table**.

Programming example

```
value = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FF_FILTER_VALUE_GET, 0, 0);
//Returns the time interval for sampling the digital input channels.
```

6.5.13. Dapi_Special_DI_FF_Filter_Value_Set

Description

This command sets a filter [ms], in which time interval the input flip-flops and the input counters, are polled.

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,

DAPI_SPECIAL_DI_FF_FILTER_VALUE_SET, ULONG time_ms, 0);

Parameter

handle=This is the handle of an open module

time_ms=Time interval [ms] by which digital input channels are sampled.

Comment

This command supports only pulse times between 5ms and 255ms.

If no time is set, the default value is 100ms.

Modules that are supported by these commands can be found in our **DELIB overview table.**

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FF_FILTER_VALUE_SET, 5, 0);
// Sets the time interval to 5ms

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DI,
DAPI_SPECIAL_DI_FF_FILTER_VALUE_SET, 150, 0);
// Sets the time interval to 150ms
```

6.6. Manage digital outputs

6.6.1. DapiDOSet1

Description

This command sets a single output.

Definition

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

Parameter

handle=This is the handle of an open module

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

data=Indicates the data value that will be written (0 / 1)

Return value

None

6.6.2. DapiDOSet8

Description

This command sets 8 digital outputs simultaneously.

Definition

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

Parameter

handle=This is the handle of an open module.

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

data=Indicates the data values that will be written

Return value

None

6.6.3. DapiDOSet16

Description

This command sets 16 digital outputs simultaneously.

Definition

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

Parameter

handle=This is the handle of an open module

ch=Gives the number of the output, from which should be written (0, 16, 32, ..)

data=Gives the data values that will be written

Return value

None

6.6.4. DapiDOSet32

Description

This command sets 32 digital outputs simultaneously.

Definition

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

Parameter

handle=This is the handle of an open module

ch=Gives the number of the output, from which should be written (0, 32, 64, ..)

data=Gives the data values that will be written

Return value

None

Programming example

```
// Write a value to the outputs
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();
// ------------------------------------
// Write a value to the outputs
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();
// ------------------------------------
// Write a value to the outputs
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();
```

6.6.5. DapiDOSet64

Description

This command sets 64 digital outputs simultaneously.

Definition

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

Parameter

handle=This is the handle of an open module

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

data=Gives the data values that will be written

Return value

None

6.6.6. DapiDOSet1_WithTimer

Description

This function sets a digital output (ch) to a value (data - 0 or 1) for a specified time in ms.

Definition

void DapiDOSet1_WithTimer(ULONG handle, ULONG ch, ULONG data, ULONG time_ms);

Parameter

handle=This is the handle of an open module

ch=Gives the number of the output to write from (0, 32, 64, ..)

data=Gives the data values that will be written

time_ms=Gives the time in which the output is set [ms].

Return value

None

Comment:

This command is only supported by our RO-O8-R8 module.

This command loses its validity if it is overwritten with other values.

If you want to deactivate the command, you have to overwrite it with time_ms=0.

Modules that are supported by these commands can be found in our **DELIB overview table**.

Programming example

```
DapiDOSet1_WithTimer(handle, 2, 1, 1000);
//Setting channel 2 for 1000msec to 1
```

6.6.7. DapiDOReadback32

Description

This command reads back the 32 digital outputs.

Definition

ULONG DapiDOReadback32(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Gives the number of the output to read back from (0, 32, 64, ..)

Return value

State of 32 outputs.

6.6.8. DapiDOReadback64

Description

This command reads back the 64 digital outputs.

Definition

ULONG DapiDOReadback64(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an opened module

ch=Indicates the number of the output from which to read back (0, 64, ..)

Return value

State of 64 outputs.

6.6.9. DapiDOSetBit32

Description

This command can be used to switch outputs specifically to 1 without changing the states of the neighboring outputs.

Definition

void DapiDOSetBit32(uint handle, uint ch, uint data);

Parameter

handle = This is the handle of an opened module

ch = Indicates the number of the output, from which is to be written

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

Return value

None

Comment:

Only the bits with a valence of 1 in the data parameter are considered by the command..

Programming example

```
data = 0x1; // Output 0 is set to 1, the state of output 1-31 remains unaffected
DapiDOSetBit32(handle, 0, data);
data = 0xf; // Output 0-3 is set to 1, the state of output 4-31 remains unaffected
DapiDOSetBit32(handle, 0, data);
data = 0xff; // Output 0-7 is set to 1, the state of output 8-31 remains unaffected
DapiDOSetBit32(handle, 0, data);
data = 0xff000000; // Output 23-31 is set to 1, the state of output 0-22 remains unaffected
DapiDOSetBit32(handle, 0, data);
```

6.6.10. DapiDOClrBit32

Description

This command can be used to switch outputs specifically to 0 without changing the states of the neighboring outputs.

Definition

void DapiDOClrBit32(uint handle, uint ch, uint data);

Parameter

handle = This is the handle of an open module

ch = Specifies the number of the output from which to write

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

Return value

None

Comment:

Only the bits with a valence of 1 in the data parameter are taken into account by the command.

Programming example

```
data = 0x1; // Output 0 is set to 0, the state of output 1-31 remains unaffected
DapiDOSetBit32(handle, 0, data);
data = 0xf; // Output 0-3 is set to 0, the state of output 4-31 remains unaffected
DapiDOSetBit32(handle, 0, data);
data = 0xff; // Output 0-7 is set to 0, the state of output 8-31 remains unaffected
DapiDOSetBit32(handle, 0, data);
data = 0xff000000; // Output 23-31 is set to 0, the state of output 0-22 remains unaffected
DapiDOSetBit32(handle, 0, data);
```

6.7. Manage output timeout

6.7.1. DapiSpecialCMDTimeout

Description

This command is used to set the timeout protection function.

There are three different timeout methods since 2021.

"normal" timeout

This is the timeout that our modules have had since 2009.

Procedure for the timeout command:

The timeout is activated by command.

If then a so-called timeout event takes place (pause between two accesses to the module is greater than the allowed timeout time) the following happens:

- All outputs are switched off

- The timeout status goes to "2

- The timeout LED turns on (for modules that have such a status)

Further accesses to the outputs are then still possible, but the timeout is no longer active. Not again until it has been reactivated.

"auto reactivate" Timeout

This is a timeout mode implemented since 2021 that automatically re-enables the timeout after the timeout event occurs.

Procedure for the timeout command:

The timeout is activated by command.

If then a so-called timeout event takes place (pause between two accesses to the module is greater than the allowed timeout time) the following happens:

- All outputs are switched off
- The timeout status goes to "4
- The timeout LED turns on (for modules that have such a status)

Further accesses to the outputs are then still possible. AND the timeout is still active. If the timeout time is exceeded again, the outputs are switched off again.

"secure outputs" Timeout

This is a timeout mode implemented since 2021 that prevents write access to the outputs after the timeout event has occurred.This ensures that the software must first restore the outputs to a "safe" state because the module's timeout mechanism has changed the outputs to predefined values.

Procedure for the timeout command:

The timeout is activated by command.

If then a so-called timeout event takes place (pause between two accesses to the module is greater than the allowed timeout time) the following happens:

- All outputs are switched off
- The timeout status goes to "6
- The timeout LED turns on (for modules that have such a status)

Further access to the outputs is NOT possible. Only after reactivating the timeout or deactivating the timeout the outputs can be written.

Definition

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

Parameter

handle=This is the handle of an open module

cmd = function to be executed

par1 = value passed to the function

par2 = value passed to the function

6.7.1.1. DapiSpecialTimeoutSetValueSec

Description

This command is used to set the timeout time.

Definition

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

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_SET_VALUE_SEC

par1 = Seconds [s]

par2 = Milliseconds [100ms] (Value 6 = 600ms)

Comment

The permissible value range of the time specification is between 0.1 seconds and 6553 seconds

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_SET_VALUE_SEC, 3, 7);
//Timeout time is set to 3.7sec.
```

6.7.1.2. DapiSpecialTimeoutActivate

Description

This command activates the "normal" timeout.

After the timeout event,.

- ..all outputs are switched off

- ..the timeout status is set to "2

- ..the timeout LED is switched on (for modules that have such a status)

Further accesses to the outputs are then still possible, but the timeout is no longer active.

Not again until it has been reactivated.

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, 0, 0);

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_ACTIVATE

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_ACTIVATE, 0, 0);
//The "normal" timeout is activated.
```

6.7.1.3. DapiSpecialTimeoutActivateAutoReactivate

Description

This command activates the "auto reactivate" timeout.

In this mode, the timeout is automatically reactivated after the timeout event.

After the timeout event.

- ..all outputs are switched off
- ..the timeout status is set to "4
- ..the timeout LED is switched on (for modules that have such a status)

Further accesses to the outputs are then still possible AND the timeout is still active.

If the timeout time is exceeded again, the outputs are switched off again.

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, 0, 0);

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_ACTIVATE_AUTO_REACTIVATE

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_ACTIVATE_AUTO_REACTIVATE, 0, 0);
//The "auto reactivate" timeout is activated.
```

6.7.1.4. DapiSpecialTimeoutActivateSecureOutputs

Description

This command activates the "secure" timeout.

In this mode, write access to the outputs after a timeout event is prevented.

This ensures that the software first has to restore a "secure" state of the outputs,

because the timeout mechanism of the module has changed the outputs to predefined values.

After the timeout event.

- ..all outputs are switched off
- ..the timeout status is set to "6
- ..the timeout LED is switched on (for modules that have such a status)

Further accesses to the outputs are NOT possible. Only after reactivating the timeout or deactivating the timeout the outputs can be written.

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, 0, 0);

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_ACTIVATE_SECURE_OUTPUTS

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_ACTIVATE_SECURE_OUTPUTS, 0, 0);
//The "secure" timeout is activated.
```

6.7.1.5. DapiSpecialTimeoutDeactivate

Description

This command disables the timeout.

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, 0, 0);

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_DEACTIVATE

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_DEACTIVATE, 0, 0);
//Disables the timeout.
```

6.7.1.6. DapiSpecialTimeoutGetStatus

Description

Dieser Befehl dient dem Auslesen des Timeout-Status.

Definition

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

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_GET_STATUS

Return value

Return = 0 (timeout is disabled)

Values for the "normal" timeout

Return = 1 (Timeout "normal" is activated)

Return = 2 (Timeout "normal" has occurred)

Values for the "auto reactivate" timeout

Return = 3 (Timeout "auto reactivate" is activated)

Return = 4 (Timeout "auto reactivate" has occurred one or more times)

Values for the "secure" timeout

Return = 5 (Timeout "secure" is activated).

Return = 6 (Timout "secure" has taken place. In this status, writing to the outputs is prevented).

Programming example

```
unsigned long status = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_GET_STATUS, 0, 0);
printf("Status = %lu\n", status);
//Query the timeout status with output.
```

6.7.1.7. DapiSpecialTimeoutDoValueMaskWRSet32

Description

This command determines the outputs to be set in case of a timeout.

Definition

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

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_WR_SET32

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

par2 = [32 Bit] Specifies the outputs which are to be activated in case of a timeout.

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_WR_SET32, 0, 0xff);
//The first 8 relays are switched on in the timeout case.
```

6.7.1.8. DapiSpecialTimeoutDoValueMaskRDSet32

Description

This command is used to read out the transferred values.

Definition

ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, 0, 0);

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_RD_SET32

Return value

[32 bit] Value passed to the SET command

Programmierbeispiel

```
long value = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_RD_SET32, 0, 0);
printf("%0x\n", value);
//The value that was passed to the SET command is read out and displayed.
```

6.7.1.9. DapiSpecialTimeoutDoValueMaskWRClr32

Description

This command determines the outputs that are to be switched off in the event of a timeout.

Definition

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

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_WR_CLR32

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

par2 = [32 Bit] Specifies the outputs which are to be deactivated in case of a timeout.

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_WR_CLR32, 0, 0xff);
//The first 8 relays are switched off in the timeout case.
```

6.7.1.10. DapiSpecialTimeoutDoValueMaskRDClr32

Description

This command is used to read out the transferred values.

Definition

ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, 0, 0);

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_RD_CLR32

Return value

[32 bit] Value that is passed to the CLR command

Programming example

```
long value = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_RD_CLR32, 0, 0);
printf("%0x\n", value);
//The value that was passed to the CLR command is read out and displayed.
```

6.7.1.11. DapiSpecialTimeoutDoValueLoadDefault

Description

Resets the SET and CLR values to the default value.

(SET value = 0, CLR value = FFFFFFFF)

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT, cmd, 0, 0);

Parameter

cmd = DAPI_SPECIAL_TIMEOUT_DO_VALUE_LOAD_DEFAULT

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_TIMEOUT,
DAPI_SPECIAL_TIMEOUT_DO_VALUE_LOAD_DEFAULT, 0, 0);
//SET and CRL values are set to the default value.
```

6.8. Test functions

6.8.1. DapiPing

Description

This command checks the connection to an open module.

Definition

ULONG DapiPing(ULONG handle, ULONG value);

Parameter

handle=This is the handle of an opened module.

value=Passed test value, in the value range of 0-255 (8-bit), to the module.

Return value

Here the test value passed with "value" must return

6.9. Register write commands

6.9.1. DapiWriteByte

Description

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

Definition

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

Parameter

handle=This is the handle of an open module

address=Address to be accessed

value=Gives the data value that will be written (8 bits)

Return value

None

Comment

This should only be used by experienced programmers. This way all available registers can be accessed directly.

6.9.2. DapiWriteWord

Description

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

Definition

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

Parameter

handle=This is the handle of an open module

address=Address to be accessed

value=Gives the data value that will be written (16 bit)

Return value

None

Comment

This should only be used by experienced programmers. This way all available registers can be accessed directly.

6.9.3. DapiWriteLong

Description

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

Definition

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

Parameter

handle=This is the handle of an open module

address=Address to be accessed

value=Gives the data value that will be written (32 bit)

Return value

None

Comment

This should only be used by experienced programmers. This way all available registers can be accessed directly.

6.9.4. DapiWriteLongLong

Description

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

Definition

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

Parameter

handle=This is the handle of an open module

address=Address to be accessed

value=Gives the data value that will be written (64 bit)

Return value

None

Comment

This should only be used by experienced programmers. This way all available registers can be accessed directly.

6.10. Register read commands

6.10.1. DapiReadByte

Description

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

Definition

ULONG DapiReadByte(ULONG handle, ULONG adress);

Parameter

handle=This is the handle of an open module

address=address to be accessed

Return value

Content of the register to be read (8 bit)

Comment

This should only be used by experienced programmers. This way all available registers can be accessed directly.

6.10.2. DapiReadWord

Description

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

Definition

ULONG DapiReadWord(ULONG handle, ULONG adress);

Parameter

handle=This is the handle of an open module

address=address to be accessed

Return value

Content of the register to be read (16 bit)

Comment

This should only be used by experienced programmers. This way all available registers can be accessed directly.

6.10.3. DapiReadLong

Description

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

Definition

ULONG DapiReadLong(ULONG handle, ULONG adress);

Parameter

handle=This is the handle of an open module

address=address to be accessed

Return value

Contents of the register to be read (32 bit)

Comment

This should only be used by experienced programmers. This way all available registers can be accessed directly.

Program example

```
char v0, v1, v2, v3;
uint ver;
float fw_ver;

ver = (uint)DapiReadLong(handle, 0xfff4);

v3 = (char)((ver >> 24) & 0xff);
v2 = (char)((ver >> 16) & 0xff);
v1 = (char)((ver >> 8) & 0xff);
v0 = (char)((ver >> 0) & 0xff);

fw_ver = (((float)v0) - '0') * 10 + (((float)v1) - '0')
+ (((float)v2) - '0') / 10 + (((float)v3) - '0') / 100;
// Here the firmware version of the module is read out.
```

6.10.4. DapiReadLongLong

Description

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

Definition

ULONGLONG DapiReadLongLong(ULONG handle, ULONG adress);

Parameter

handle=This is the handle of an open module

address=address to be accessed

Return value

Contents of the register to be read (64 bit)

Comment

This should only be used by experienced programmers. This way all available registers can be accessed directly.

6.11. Programming example

```
// ********************************************************************
// ********************************************************************
//
// (c) DEDITEC GmbH, 2009
//
// web: http://www.deditec.de
//
// mail: vertrieb@deditec.de
//
// dtapi_prog_beispiel_input_output.cpp
//
// ********************************************************************
// ********************************************************************
//
// Include the following libraries when linking: delib.lib
// Please do this in the project settings (Project/Settings/Linker(Object-
Library modules) ... configure last entry
#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 ;
}
```

6.12. Delib overview table

Commands	Available for
DAPI_SPECIAL_CMD_SET_DIR_DX_1	USB-MINI-TTL8
DAPI_SPECIAL_CMD_SET_DIR_DX_8	USB-MINI-TTL8 USB-TTL32 USB-TTL64 ETH-TTL64
DAPI_SPECIAL_CMD_GET_DIR_DX_1	is not supported
DAPI_SPECIAL_CMD_GET_DIR_DX_8	is not supported

Commands	Available for	Does not work with
DAPI_SPECIAL_CMD_TIMEOUT DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_WR_SET32 DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_RD_SET32 DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_WR_CLR32 DAPI_SPECIAL_TIMEOUT_DO_VALUE_MASK_RD_CLR32 DAPI_SPECIAL_TIMEOUT_DO_VALUE_LOAD_DEFAULT	ETH-TTL64 ETH-RELAIS8 USB-RELAIS8 RO-SERIE BS-SERIE NET-SERIE USB-TTL-64	USB-Mini-Stick
DAPI_SPECIAL_TIMEOUT_SET_VALUE_SEC DAPI_SPECIAL_TIMEOUT_ACTIVATE DAPI_SPECIAL_TIMEOUT_DEACTIVATE	All modules	

Commands	Available for	Does not work with
DAPI_SPECIAL_TIMEOUT_GET_STATUS		

Commands	Starter USB*	Starter ETH**	RO Serie	BS Serie	NET Serie	Other
DAPI_SPECIAL_COUNTER_ LATCH_ALL			x			
DAPI_SPECIAL_COUNTER_ LATCH_ALL_WITH_RESET			x			
DapiDOSet1_WithTimer			x			
DAPI_SPECIAL_CMD_SW_FIFO DAPI_SPECIAL_SW_FIFO_INIT_ AND_CLEAR ... DAPI_SPECIAL_SW_FIFO_ IO_DEACTIVATE					x	
DAPI_SPECIAL_CMD_AD DAPI_SPECIAL_RO_AD_ FIFO_ACTIVATE ... DAPI_SPECIAL_RO_AD_ FIFO_INIT			x			

*: USB-OPTOIN8, USB-Mini-Stick, USB-TTL-64

**: ETH-TTL64, ETH-OPTOIN8, ETH-RELAIS8

Commands	Starter USB*	Starter ETH**	RO Serie	BS Serie	NET Serie	Other
DAPI_SPECIAL_DI_FF_FILTER DAPI_SPECIAL_DI_FF_FILTER_VALUE_SET DAPI_SPECIAL_DI_FF_FILTER_VALUE_GET	5-255	1-255	1-255	1-255	1-255	
DAPI_SPECIAL_DI_FILTER DAPI_SPECIAL_DI_FILTER_VALUE_SET DAPI_SPECIAL_DI_FILTER_VALUE_GET	x	0, 1-254	0, 1-254	0, 1-254	0, 1-254	
DAPI_SPECIAL_CMD_GET_INTERNAL_STATISTIC	x	x	x	x		

*: USB-OPTOIN8, USB-Mini-Stick, USB-TTL-64

**: ETH-TTL64, ETH-OPTOIN8, ETH-RELAIS8

Commands	Available for
DAPI_SPECIAL_CMDEXT_CAN_WR_RUNTIME_VALUE DAPI_SPECIAL_CMDEXT_CAN_RD_RUNTIME_VALUE	NET-CPU-PRO, BS-WEU, RO-ETH-LC

Security measures



7. Security measures

There are several security measures to protect your module from unauthorized access from outside.

7.1. Write protection

With DIP switch 2, you can deactivate access via the Ethernet or USB interface to the **Module-Configuration-Memory** with write protection.

DIP-2 on = write protection active

DIP-2 off = write protection deactivated

7.2. TCP encryption

You can use the ICT tool to encrypt communication via TCP.

Partial or full access to the **Module-Configuration-Memory** can also be set using the admin and user protocols.

You can find more information on this in the chapter : **TCP encryption**

7.3. Web interface

Access to the module via the web interface can be activated/deactivated using the ICT tool.

Further information on this can be found in the chapter: **TCP encryption**

Appendix



8. Appendix

8.1. Contact / Support

If you have any questions about the product or need assistance with commissioning, you can reach us at the following numbers:

Support Software
Tel. +49 (0) 22 32 / 50 40 8 – 20

Support Hardware
Tel. +49 (0) 22 32 / 50 40 8 – 30

Support via E-mail
support@deditec.de

8.2. Environment and disposal

You can return the defective or obsolete product to us at the end of its service life. As a manufacturer and distributor of electronic assemblies, we will take care of the proper disposal for you in accordance with the applicable legal regulations. For this purpose, it is best to use our return form on the homepage:

Return form

8.3. Revisions

Rev 3.02	Pin assignment of the digital inputs and outputs revised Overview screen revised
Rev 3.01	DEDITEC Design Update 2022
Rev 3.00	DEDITEC Design Update 2021
Rev 2.05	"Software" and "DELIB API reference" chapters revised
Rev 2.04	"DELIB driver library" chapter revised.
Rev 2.03	Index added
Rev 2.02	"Firmware update" chapter added, Chapter "Software" revised
Rev 2.01	Manual extended by USB-RELAIS-8_A & USB-OPTOIN-8_A
Rev 2.00	First DEDITEC manual

8.4. 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.