



RO-Series

Hardware-Description

2023 September

INDEX

<u>1. Introduction</u>	17
1.1. Foreword	18
1.2. Customer satisfaction	18
1.3. Customer response	18
1.4. Brief description	19
1.5. Scope of delivery	20
<u>2. Commissioning</u>	21
2.1. Step 1 - Safety instructions	22
2.2. Step 2 - Connecting the power supply	23
2.3. Step 3 - Connect to the PC or network	24
2.3.1. Connection via USB	24
2.3.2. Connection via Ethernet	24
2.4. Step 4 - Installing the software and drivers	25
2.5. Step 5 - Connecting the I/O connectors	26
2.6. Step 6 - Function test	28
<u>3. Hardware description</u>	29
3.1. Interfaces	30
3.1.1. RO-ETH	30
3.1.1.1. Hardware description	30
3.1.1.1.1. Overview screen	30
3.1.1.1.2. Technical data	32
3.1.1.1.3. Buttons of the module	33
3.1.1.1.4. Control LEDs	34
3.1.1.1.4.1, Definition of LEDs	35
3.1.1.1.4.2, Blinkingbehavior of the LEDs	36
3.1.1.2. Configuring the module	38
3.1.1.2.1. Configuration via DELIB Configuration utility	38
3.1.1.2.2. Configuration via internal web server	38
3.1.1.2.3. Factory settings	39

INDEX

3.1.1.3. Restore basic configuration	40
3.1.1.3.1. Restore IP address	40
3.1.1.3.2. Restore firmware	41
3.1.2. RO-ETH/LC	42
3.1.2.1. Hardware description	42
3.1.2.1.1. Overview screen	42
3.1.2.1.2. Technical data	43
3.1.2.1.3. DIP-Switches	44
3.1.2.1.4. Control LEDs	48
3.1.2.1.4.1, Definition of LEDs	49
3.1.2.1.4.2, Blinking behavior of the LEDs	50
3.1.2.2. Configuring the module	52
3.1.2.2.1. Configuration via DELIB Configuration utility	52
3.1.2.2.2. Factory settings	52
3.1.3. RO-USB	53
3.1.3.1. Hardware description	53
3.1.3.1.1. Overview screen	53
3.1.3.1.2. Technical data	57
3.1.3.1.3. Control LEDs	58
3.1.3.1.3.1, Definition of LEDs	59
3.1.3.1.4. Configuration via the DELIB Configuration Utility	60
3.1.4. RO-CAN	61
3.1.4.1. Hardware description	61
3.1.4.1.1. Overview screen	61
3.1.4.1.2. Technical data	64
3.1.4.1.3. Control LEDs	65
3.1.4.1.3.1, Definition of LEDs	66
3.1.4.2. Configuring the module	68
3.1.4.2.1. DIP-switches	68
3.1.4.2.2. The "special mode"	70
3.1.4.2.3. Software mode	70
3.1.4.2.4. DIP-switch mode	71
3.1.4.2.4.1, Setting up the transfer rate	71

INDEX

3.1.4.2.4.2, Setting up the CAN module address	72
3.1.5. RO-SER	75
3.1.5.1. Hardware description	75
3.1.5.1.1. Overview screen	75
3.1.5.1.2. Technical data	79
3.1.5.1.3. Control LEDs	80
3.1.5.1.3.1, Definition of LEDs	81
3.1.5.2. Configuring the module	82
3.1.5.2.1. DIP-switches	82
3.1.5.2.2. The "special mode"	84
3.1.5.2.3. DIP-Switch-Mode	84
3.1.5.2.4. Serial text mode	85
3.1.5.2.4.1, Overview of write commands	86
3.1.5.2.4.2, Overview of read commands	88
3.1.5.2.5. Activating echo	90
3.1.5.2.6. Setting up Baud rate	90
3.1.5.2.7. Setting up module address (RS-485 only)	91
3.1.5.2.8. Configuration via the DELIB Configuration Utility	92
3.1.6. Specifications interface connection	93
3.1.6.1. USB	93
3.1.6.2. Ethernet	93
3.1.6.3. CAN	94
3.1.6.4. RS-232	95
3.1.6.5. Retrofit interfaces	96
3.2. Submodule	97
3.2.1. RO-REL16	97
3.2.1.1. Overview image	97
3.2.2. RO-REL8_UM	99
3.2.2.1. Overview image	99
3.2.2.2. Bistable relay outputs	100
3.2.3. RO-OPT8-REL8	101
3.2.3.1. Overview image	101
3.2.4. RO-OPT16	103

INDEX

3.2.4.1. Overview image	103
3.2.4.2. Changing the input voltage	105
3.2.5. RO-MOS16	106
3.2.5.1. Overview image	106
3.2.6. RO-PWM	108
3.2.6.1. Overview image	108
3.2.7. Digital outputs	110
3.2.7.1. Technical data	110
3.2.7.2. Pin assignment of a RO-REL16	111
3.2.7.3. Pin assignment of a RO-REL8_UM	112
3.2.7.4. Pin assignment of a RO-MOS16_P and RO-PWM16_P	113
3.2.7.5. Connection example of a relay module	115
3.2.7.6. Connection example of a changeover relay module	116
3.2.7.7. Connection example of a MOSFET module	117
3.2.7.8. Timeout function	118
3.2.8. Digital inputs	119
3.2.8.1. Technische Daten	119
3.2.8.2. Pinbelegung RO-016	120
3.2.8.3. Pin assignment RO-08-R8	121
3.2.8.4. Connection example RO-016	123
3.2.8.5. Connection example RO-08-R8	124
3.2.8.6. Input filter	126
3.2.8.7. Monitor changes of state	127
3.2.9. Analog input/output modules	128
3.2.9.1. RO-AD16-DA4	128
3.2.9.1.1. Overview image	129
3.2.9.1.2. Technical data	131
3.2.9.1.3. Pin assignment	132
3.2.9.1.3.1. A/D connector (18pin)	132
3.2.9.1.3.2. D/A connector (10pin)	133
3.2.9.2. RO-AD16	134
3.2.9.2.1. Overview image	134
3.2.9.2.2. Technical data	136

INDEX

3.2.9.2.3. Pin assignment	137
3.2.9.2.3.1, A/D connector (18pin)	137
3.2.9.3. RO-AD16_ISO	138
3.2.9.3.1. Overview image	138
3.2.9.3.2. Technical data	140
3.2.9.3.3. Pin assignment	141
3.2.9.3.3.1, A/D connector (18pin)	141
3.2.9.4. RO-DA4/RO-DA8	142
3.2.9.4.1. Overview image	142
3.2.9.4.2. Technical data	143
3.2.9.4.3. Pin assignment	144
3.2.9.4.3.1, Pinout D/A connector (10pin)	144
3.2.9.5. RO-DA2_ISO	145
3.2.9.5.1. Overview image	145
3.2.9.5.2. Technical data	147
3.2.9.5.3. Pin assignment	148
3.2.9.5.3.1, D/A connector (10pin)	148
3.2.10. Stepper Modul	149
3.2.10.1. Overview image	149
3.2.10.2. Technical data	151
3.2.10.3. Control	151
3.2.10.4. Pin assignment	152
3.2.10.4.1. Stepper connector (10pin)	152
3.2.11. RO-CNT8	153
3.2.11.1. Overview image	153
3.2.11.2. Technical data	155
3.2.11.3. Counter Modul Modi	156
3.2.11.4. Block diagram	157
3.2.11.5. Pin assignment	158
3.2.11.5.1. Pin assignment RO-CNT8	158
3.2.12. RO-CNT/IGR	159
3.2.12.1. Overview image	159
3.2.12.2. Technical data	161

INDEX

3.2.12.3. Block diagram	162
3.2.12.4. Evaluation Phi 0/Phi 90° (Graphical)	163
3.2.12.5. LEDs	164
3.2.12.6. Inputs	165
3.2.12.6.1. References	165
3.2.12.6.2. Visual control of the inputs	165
3.2.12.7. Pin assignment	166
3.2.13. RO-PT100-4 / RO-PT100-8	167
3.2.13.1. Overview image	167
3.2.13.2. Introduction	169
3.2.13.3. Technical data	170
3.2.13.4. Block diagram	171
3.2.13.5. Pin assignment	172
3.3. Dimension of RO-Series	174
3.3.1. Top view	174
3.3.2. Side view	175
4. Software description	177
4.1. Using our products	178
4.1.1. Control via our DELIB driver library	178
4.1.2. Control via supplied test programs	178
4.1.3. Control at protocol level	179
4.1.4. DELIB CLI (command-line interface) für Windows	180
4.1.4.1. Configuration of the DELIB CLI	182
4.1.4.2. DELIB CLI Examples	183
4.1.5. Control via graphical applications	187
4.1.5.1. LabVIEW	187
4.1.5.2. ProfiLab	187
4.1.5.3. Licht24 Pro	188
4.1.6. Integration of the DELIB in programming languages	189
4.1.6.1. Embedding the DELIB in Visual-C/C++	189
4.1.6.2. Embedding the DELIB in Visual-C/C++ (Visual Studio 2015)	191

INDEX

4.1.6.3. Embedding the DELIB in Visual-C#	194
4.1.6.4. Embedding the DELIB in Delphi	195
4.1.6.5. Embedding the DELIB in Visual-Basic (VB)	196
4.1.6.6. Embedding the DELIB in Visual-Basic.NET (VB.NET)	197
4.1.6.7. Embedding the DELIB in MS-Office (VBA)	198
4.1.6.8. Embedding the DELIB in LabVIEW	200
4.1.6.8.1. Embedding the DELIB in LabVIEW	200
4.1.6.8.2. Using the VIs in LabVIEW	209
4.1.6.8.3. Setting the module ID in LabVIEW	211
4.1.6.9. Embedding the DELIB in Java	213
4.2. DELIB driver library	214
4.2.1. Overview	215
4.2.1.1. Supported programming languages	216
4.2.1.2. Supported operating systems	218
4.2.1.3. SDK kit for programmers	218
4.2.2. DELIB Setup	219
4.2.3. DELIB Configuration Utility	225
4.2.3.1. Introduction	225
4.2.3.2. Create or edit configuration	226
4.2.3.2.1. Module configuration USB	227
4.2.3.2.1.1, Example for the configuration of identical USB modules	229
4.2.3.2.2. Module configuration Ethernet	234
4.2.3.2.2.1, Automatic search	238
4.2.3.2.2.2, Set up encryption	242
4.2.3.2.2.1, Manual configuration	242
4.2.3.2.2.2, Automatic configuration	246
4.2.3.2.2.1, Authentication	247
4.2.3.2.3. Module Configuration CAN	251
4.2.3.2.4. Module Configuration Serial	254
4.2.3.3. Test module	256
4.2.3.4. Set debug options	260
4.2.4. Using the module selector	261
4.2.4.1. via USB	261

INDEX

4.2.4.2. via Ethernet	262
4.2.4.3. via WiFi / WPS	264
4.2.4.4. Modul Info	265
4.2.5. DELIB Module Config	267
4.2.5.1. Module configurations	267
4.2.5.1.1. Module info page	268
4.2.5.1.2. Module identification	269
4.2.5.1.3. LAN network information	270
4.2.5.1.4. LAN network settings	271
4.2.5.1.5. WiFi network information	273
4.2.5.1.6. WiFi network settings	275
4.2.5.1.7. WiFi WPS connection	277
4.2.5.1.8. NTP configuration	278
4.2.5.1.9. Serial configuration	280
4.2.5.1.10. I/O channel names	282
4.2.5.1.11. CAN configuration	283
4.2.5.1.11.1, CAN status	283
4.2.5.1.11.1, CAN Status Interface	283
4.2.5.1.11.2, CAN Statistics TX/RX	284
4.2.5.1.11.2, CAN Main	285
4.2.5.1.11.1, CAN Main Interface	285
4.2.5.1.11.2, CAN Main I/O Init	287
4.2.5.1.11.3, CAN TX/RX modes	289
4.2.5.1.11.1, CAN TX Mode	289
4.2.5.1.11.2, CAN RX Mode	291
4.2.5.2. I/O-Test	292
4.2.5.2.1. Timeout test function	292
4.2.5.2.2. Digital In	293
4.2.5.2.3. Digital Out	294
4.2.5.2.4. Analog In	296
4.2.5.2.5. Analog Out	297
4.2.6. DELIB Module Demo	299
4.2.6.1. Module selection	300

INDEX

4.2.6.2. General	301
4.2.6.2.1. Module Info	303
4.2.6.3. Digital Input	304
4.2.6.4. Digital Output	305
4.2.6.5. Analog Input	306
4.2.6.6. Analog Output	307
4.2.7. CAN Configuration Utility	309
4.2.7.1. Module selection	310
4.2.7.2. Create new configuration, load, save	312
4.2.7.3. Transfer configuration to the module	314
4.2.7.4. Query statistics from module	315
4.2.7.5. Configuration	317
4.2.7.5.1. Module configuration	318
4.2.7.5.2. I/O configuration	319
4.2.7.5.3. TX configuration	322
4.2.7.5.3.1, Example Interval	324
4.2.7.5.3.2, Trigger example	325
4.2.7.5.4. RX configuration	326
4.2.7.5.4.1, Example RX-DA	328
4.2.7.5.4.2, Example RX-DO	329
4.2.7.6. Structure of the CAN packages	330
4.2.7.6.1. Digital inputs	330
4.2.7.6.2. Digital outputs	331
4.2.7.6.3. Digital input counter (16-bit)	332
4.2.7.6.4. Digital input counter (48-bit) - 32-bit packet	333
4.2.7.6.5. Digital input counters (48-bit) - 64-bit package	334
4.2.7.6.6. Analog inputs / outputs	335
4.2.7.6.6.1, Analog inputs	335
4.2.7.6.6.2, Analog outputs	337
4.2.7.6.6.3, Examples	338
4.2.7.6.7. Temperature inputs	340
4.2.7.6.8. Stepper	342
4.2.7.6.8.1, Command-Liste	342

INDEX

4.2.7.6.8.2, Values for par 1 to command SET_MOTORCHARACTERISTIC	345
4.2.7.6.8.3, Values for par 1 to command GO_REFSWITCH	348
4.2.7.6.8.4, Example	349
4.2.8. DT-Flasher	350
4.2.8.1. About DEDITEC firmware	351
4.2.8.2. Module selection	351
4.2.8.3. Perform firmware update	352
4.2.8.3.1. Update flash files manually	354
4.3. DELIB Sample Sources (Windows program examples)	355
4.3.1. Installation DELIB Sample Sources	356
4.3.2. Use of the DELIB Sample Sources	360
4.3.2.1. Step 1 - Product selection	360
4.3.2.2. Step 2 - Category selection	362
4.3.2.3. Step 3 - Programming language selection	363
4.3.2.4. Step 4 - Source code	364
4.4. DELIB for Linux	367
4.4.1. Using the DELIB driver library for Linux	369
4.4.1.1. Delib USB sample in Linux	369
4.4.1.2. Delib ETH sample in Linux	372
4.4.2. DELIB CLI (command-line interface) for Linux	375
4.4.2.1. Configuration of the DELIB CLI	378
4.4.2.2. DELIB CLI Examples	381
5. DELIB directory structure	383
5.1. Include directory	386
5.2. Library directory	386
5.3. Library directory for Borland	386
5.4. Environment variables	386
6. DELIB API Reference	387
6.1. Available DEDITEC module IDs	388
6.2. Administrative functions	391

INDEX

6.2.1. DapiOpenModule	391
6.2.2. DapiCloseModule	392
6.2.3. DapiGetDELIBVersion	392
6.2.4. DapiSpecialCMDGetModuleConfig	393
6.2.5. DapiOpenModuleEx	396
6.3. Error handling	398
6.3.1. DapiGetLastError	398
6.3.2. DapiGetLastErrorText	399
6.3.3. DapiClearLastError	400
6.3.4. DapiGetLastErrorByHandle	401
6.3.5. DapiClearLastErrorByHandle	402
6.4. A/D converter functions	403
6.4.1. DapiADSetMode	403
6.4.2. DapiADGetMode	405
6.4.3. DapiADGet	405
6.4.4. DapiADGetVolt	406
6.4.5. DapiADGetmA	406
6.4.6. DapiSpecialADReadMultipleAD	407
6.5. Manage D/A outputs	409
6.5.1. DapiDASetMode	409
6.5.2. DapiDAGetMode	411
6.5.3. DapiDASet	412
6.5.4. DapiDASetVolt	413
6.5.5. DapiDASetmA	414
6.5.6. DapiSpecialCmd_DA	415
6.6. Read digital inputs	417
6.6.1. DapiDIGet1	417
6.6.2. DapiDIGet8	417
6.6.3. DapiDIGet16	418
6.6.4. DapiDIGet32	419
6.6.5. DapiDIGet64	420

INDEX

6.6.6. DapiDIGetFF32	421
6.6.7. DapiDIGetCounter	422
6.6.8. DapiSpecialCounterLatchAll	423
6.6.9. DapiSpecialCounterLatchAllWithReset	424
6.6.10. DapiSpecialDIFilterValueSet	425
6.6.11. DapiSpecialDIFilterValueGet	426
6.6.12. Dapi_Special_DI_FF_Filter_Value_Get	427
6.6.13. Dapi_Special_DI_FF_Filter_Value_Set	428
6.7. Manage digital outputs	429
6.7.1. DapiDOSet1	429
6.7.2. DapiDOSet8	429
6.7.3. DapiDOSet16	430
6.7.4. DapiDOSet32	431
6.7.5. DapiDOSet64	432
6.7.6. DapiDOSet1_WithTimer	433
6.7.7. DapiDOReadback32	434
6.7.8. DapiDOReadback64	434
6.7.9. DapiDOSetBit32	435
6.7.10. DapiDOClrBit32	436
6.8. Manage output timeout	437
6.8.1. DapiSpecialCMDTimeout	437
6.8.1.1. DapiSpecialTimeoutSetValueSec	440
6.8.1.2. DapiSpecialTimeoutActivate	441
6.8.1.3. DapiSpecialTimeoutActivateAutoReactivate	442
6.8.1.4. DapiSpecialTimeoutActivateSecureOutputs	443
6.8.1.5. DapiSpecialTimeoutDeactivate	444
6.8.1.6. DapiSpecialTimeoutGetStatus	445
6.8.1.7. DapiSpecialTimeoutDoValueMaskWRSet32	446
6.8.1.8. DapiSpecialTimeoutDoValueMaskRDSet32	447
6.8.1.9. DapiSpecialTimeoutDoValueMaskWRClr32	448
6.8.1.10. DapiSpecialTimeoutDoValueMaskRDClr32	449
6.8.1.11. DapiSpecialTimeoutDoValueLoadDefault	450

INDEX

6.9. Digital counter functions for RO-Counter modules	451
6.9.1. DapiCnt48ModeSet	451
6.9.2. DapiCnt48ModeGet	457
6.9.3. DapiCnt48CounterGet32	458
6.9.4. DapiCnt48CounterGet48	459
6.9.5. DapiSpecialCNT48ResetSingle	460
6.9.6. DapiSpecialCNT48ResetGroup8	461
6.9.7. DapiSpecialCNT48LatchGroup8	462
6.9.8. DapiSpecialCNT48DGet1	463
6.10. PWM Functions	464
6.10.1. DapiPWMOutSet	464
6.10.2. DapiPWMOutReadback	466
6.10.3. DAPI_SPECIAL_PWM_FREQ_SET	467
6.10.4. DAPI_SPECIAL_PWM_FREQ_READBACK	468
6.11. Manage pulse generator outputs	469
6.11.1. DapiPulseGenSet	469
6.12. PT100 Functions	471
6.12.1. DapiTempGet	471
6.13. Stepper motors Functions	472
6.13.1. Commands with DapiStepperCommand	472
6.13.1.1. DAPI_STEPPER_CMD_GO_POSITION	472
6.13.1.2. DAPI_STEPPER_CMD_GO_POSITION_RELATIVE	473
6.13.1.3. DAPI_STEPPER_CMD_SET_POSITION	474
6.13.1.4. DAPI_STEPPER_CMD_SET_FREQUENCY	475
6.13.1.5. DAPI_STEPPER_CMD_GET_FREQUENCY	476
6.13.1.6. DAPI_STEPPER_CMD_SET_FREQUENCY_DIRECTLY	477
6.13.1.7. DAPI_STEPPER_CMD_STOP	478
6.13.1.8. DAPI_STEPPER_CMD_FULLSTOP	479
6.13.1.9. DAPI_STEPPER_CMD_DISABLE	480
6.13.1.10. DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC	481

INDEX

6.13.1.11.	487
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC	
6.13.1.12.	
DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_SAVE	496
6.13.1.13.	
DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_LOAD	497
6.13.1.14.	
DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_LOAD_DEFAULT	498
6.13.1.15. DAPI_STEPPER_CMD_GO_REFSWITCH	499
6.13.1.16. DAPI_STEPPER_CMD_GET_CPU_TEMP	501
6.13.1.17.	
DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE	502
6.13.2. Query status with DapiStepperGetStatus	503
6.13.2.1. DAPI_STEPPER_STATUS_GET_ACTIVITY	503
6.13.2.2. DAPI_STEPPER_STATUS_GET_POSITION	504
6.13.2.3. DAPI_STEPPER_STATUS_GET_SWITCH	505
6.13.3. DapiStepperCommandEx	506
6.14. CAN Runtime Functions	507
6.14.1. RunTimeVarWriteToModule	507
6.15. Manage software FIFO	523
6.15.1. DapiSpecialCMDSWFifo	523
6.15.1.1. DapiSpecialSWFifoSetSubmodule	524
6.15.1.2. DapiSpecialSWFifoGetSubmodule	524
6.15.1.3. DapiSpecialSWFifoActivate	526
6.15.1.4. DapiSpecialSWFifoDeactivate	526
6.15.1.5. DapiSpecialSWFifoGetActivity	527
6.15.1.6. DapiSpecialSWFifoIOActivate	528
6.15.1.7. DapiSpecialSWFifoIODeactivate	528
6.15.1.8. DapiSpecialSWFifoInitAndClear	529
6.15.1.9. DapiSpecialSWFifoSetChannel	530
6.15.1.10. DapiSpecialSWFifoGetChannel	531

INDEX

6.15.1.11. DapiSpecialSWFifoSetFrequencyHz	532
6.15.1.12. DapiSpecialSWFifoGetFrequencyHz	533
6.15.1.13. DapiSpecialSWFifoGetBytesFree	534
6.15.1.14. DapiSpecialSWFifoGetBytesPerSample	535
6.15.1.15. DapiSpecialSWFifoSetMode	536
6.15.1.16. DapiSpecialSWFifoGetMode	537
6.15.1.17. DapiSpecialSWFifoGetStatus	538
6.15.1.18. DapiSpecialSWFifoGetInstanceType	540
6.15.2. DapiWriteFifo	542
6.15.3. DapiReadFifo	543
6.16. Test functions	546
6.16.1. DapiPing	546
6.17. Register write commands	547
6.17.1. DapiWriteByte	547
6.17.2. DapiWriteWord	548
6.17.3. DapiWriteLong	549
6.17.4. DapiWriteLongLong	550
6.18. Register read commands	551
6.18.1. DapiReadByte	551
6.18.2. DapiReadWord	552
6.18.3. DapiReadLong	553
6.18.4. DapiReadLongLong	554
6.19. Programming example	555
6.20. Delib overview table	557
<u>7. Appendix</u>	561
7.1. Contact / Support	562
7.2. Environment and disposal	562
7.3. Revisions	563
7.4. Copyrights and trademarks	564

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 RO series products are versatile. A plug-in system allows submodules to be connected individually, making the use of these modules even more flexible. In the field of PC measurement technology, these modules are ideally suited for setting up extensive automation projects, control tasks or measurement processes.

Both analog and digital data signals can be acquired or output. 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 configuration tool "DELIB-Module Config" included in the scope of delivery additionally enables a fast and uncomplicated start-up during commissioning.

The housing consists of a particularly flat aluminum-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:

- RO module
- 2 pole connector for power connection
- 10, 16 or 18 pole connector with eject mechanism
- Auxiliary tool for connecting the I/O connectors
- USB cable 1,5m
- Installation CD with manuals and drivers

Commissioning



2. Commissioning

2.1. Step 1 - Safety instructions

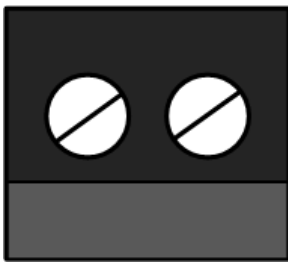
Before commissioning your DEDITEC product, please familiarize yourself with this manual and read the following points carefully:

- 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 - Connecting the power supply

Select a suitable power supply* with sufficient power of at least 5 watts and an output voltage of, for example, +7VDC or +24VDC.

The power supply is connected to the 2-pole pluggable screw terminal. Please note the polarity as shown below. Left V+ and right V-.



* A suitable industrial power supply can be purchased from us as an accessory.

2.3. Step 3 - Connect to the PC or network

2.3.1. Connection via USB

Connect the module, with the included USB cable, to your PC or USB hub.

If you want to connect several USB modules to one PC at the same time, each module must first be assigned its own module number.

See chapter → **Example for configuration of identical USB modules**

2.3.2. Connection via Ethernet

Connection via Ethernet using a switch or hub:

Connect the module with a patch cable to your Ethernet switch or hub. The module is set to DHCP at delivery (DIP switch 1 = ON). The next free IP address is automatically assigned to your module.

Connection via Ethernet directly to the PC:

Connect the module with an Ethernet crossover cable to your PC. Set the DIP switch 1 on the module to "OFF". The module now starts with the IP address stored in the module → **factory IP 192.168.1.1**

2.4. Step 4 - Installing the software and drivers

Installation under Windows:

To operate this product with a Windows based PC, please proceed as follows:

First install the DELIB driver library for Windows by executing file "delib_install.exe" from the DEDITEC driver CD. This is located in the directory "\zip\delib\delib_install.exe".

Alternatively, you can download the latest DELIB version from our homepage. → <http://www.deditec.de/delib>

Installation under Linux:

To be able to run this product with a Linux based PC, please proceed as follows:
Unzip the ZIP file "delib-linux.zip" from the DEDITEC driver CD and copy the delib.dll into your project directory.

Alternatively you can download the latest DELIB version from our homepage. → <https://www.deditec.de/media/zip/delib/delib-linux.zip>

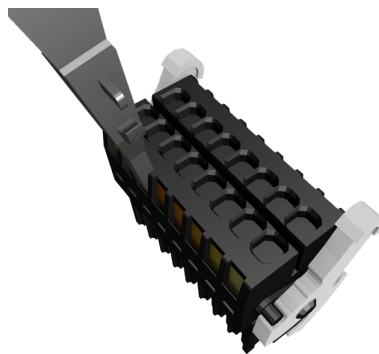
2.5. Step 5 - Connecting the I/O connectors

For the line connection to the I/O connectors you need an auxiliary tool, which is included in the scope of delivery.

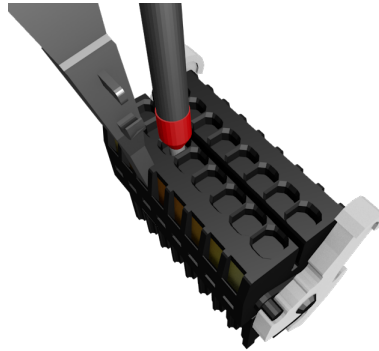


Please proceed as follows for connection:

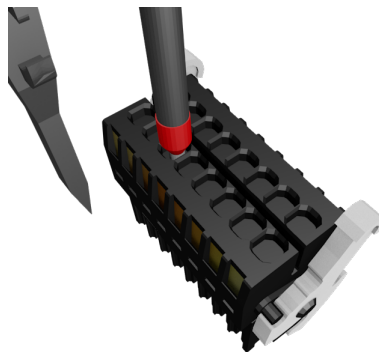
- Insert the operating tool firmly (downwards) into the side opening in the direction of the conductor connection.



- Strip 6-7mm of insulation from the connecting cable and insert it into the open terminal contact.



- Pull out the actuating tool again. Then check whether the line is firmly seated in the terminal!



2.6. Step 6 - Function test

With our tool "DELIB-Module Config" 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 "**DELIB-Module Config**" chapter.

Hardware description



3. Hardware description

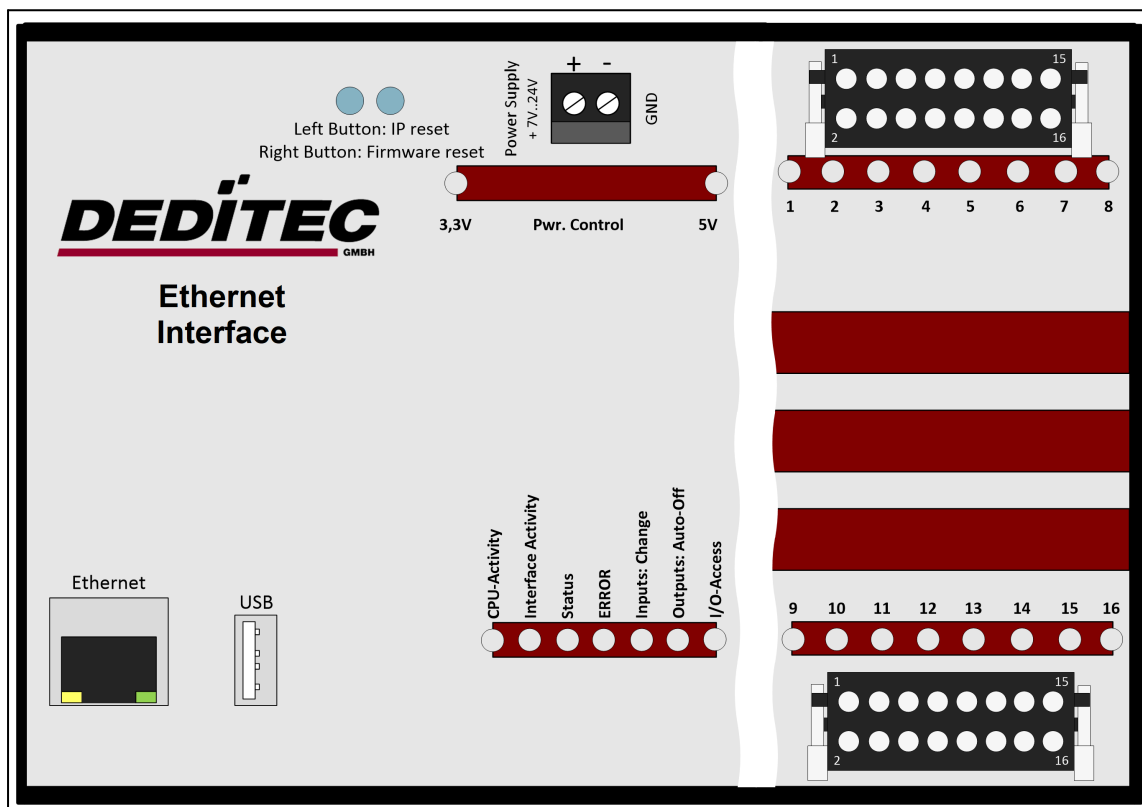
3.1. Interfaces

3.1.1. RO-ETH

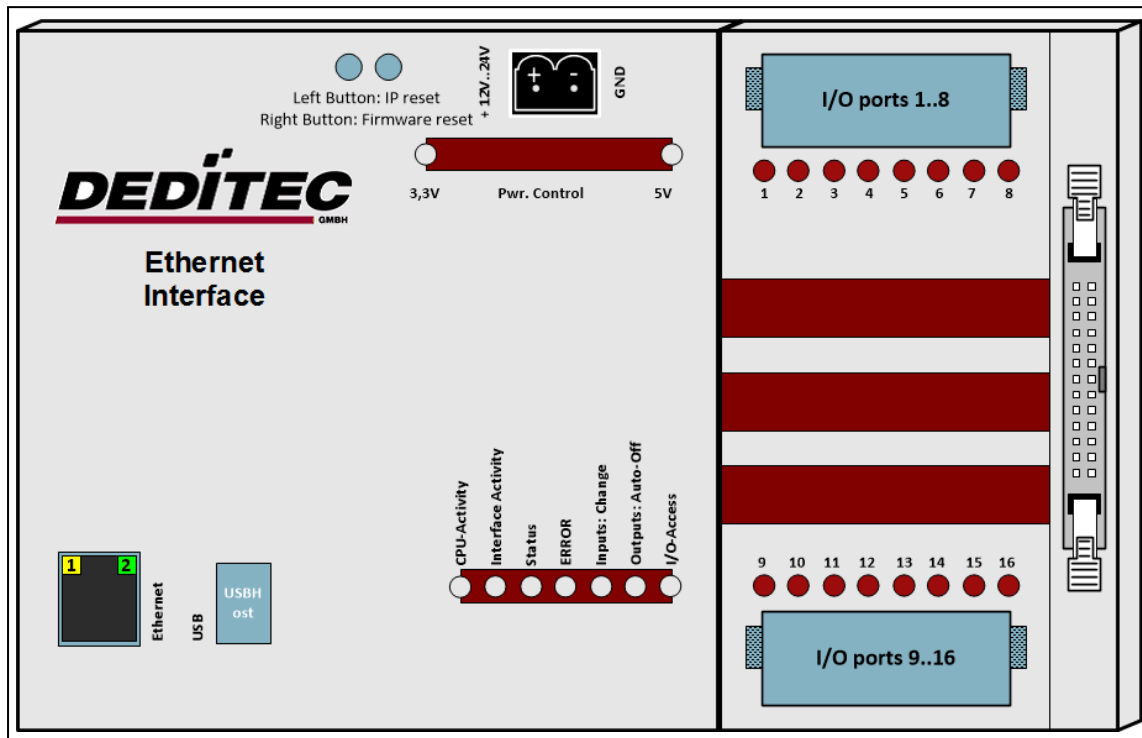
3.1.1.1. Hardware description

3.1.1.1.1. Overview screen

The illustration shows the control module with ethernet interface (left side) combined with an input/output module (right side).



The illustration shows the control module with ethernet interface (left side) combined with a flexible connector input/output module (right side).



3.1.1.1.2. Technical data

Electrical data:

Supply voltage: 7V DC .. 24V DC

Power consumption: max. 5W

Environment:

Ambient temperature: +10..+50 °C

Humidity: 90 %

Condensation: Not allowed

Mechanics:

Dimensions in mm (LxWxH): L* x 122 x 51,5

*the length varies with the type and
number of connected submodules

Fastening: Top-hat rail TS 35 x 7.5 mm

Optional:

galv. separation: Available as accessory

3.1.1.1.3. Buttons of the module

Left Button:

Reset IP address to default
(see chapter "Restore IP Address")

Right Button:

Reset firmware to factory settings.
(see chapter "Restore Firmware")

3.1.1.1.4. Control LEDs

The Ethernet module has a series of control LEDs. They are used for easy visual indication of various state functions.

While switching the module on, in normal operating mode, the module should signalize the following sequence:

- approx. 20 sec after switching the module on, LED 1 and 2 are flashing briefly.
-> Operating system has been loaded successfully.
- Then LED 3 is on permanently and LED 1 is flashing. -> Module is ready.

3.1.1.1.4.1. Definition of LEDs

LED	Label	Description
above	3,3V	Internal 3,3V power supply
above	5V	Internal 5V power supply
1	CPU Activity	2x flashing + long break. Operating system reports: Status OK
2	Interface Activity	Active communication over Ethernet
3	Status	LED is on -> Module is ready
4	ERROR	Error during ethernet-transfer (for details see document "Serial protocol")
5	Inputs: Change	"State change" between 2 read-out cycles detected
6	Outputs: Auto-Off	Due to timeout, all outputs are switched-off for safety reasons
7	I/O Access	CPU-access to the connected I/O modules.

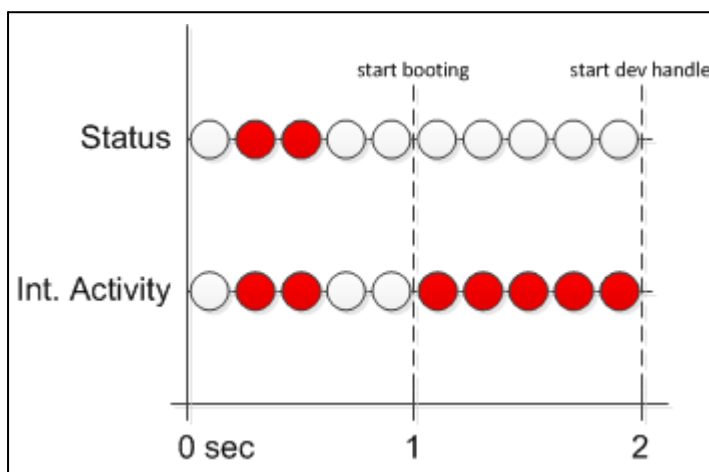
3.1.1.1.4.2. Blinking behavior of the LEDs

The blinking sequences of the status LEDs are shown below.



1. Boot process

The boot process starts immediately after the power supply is switched on.

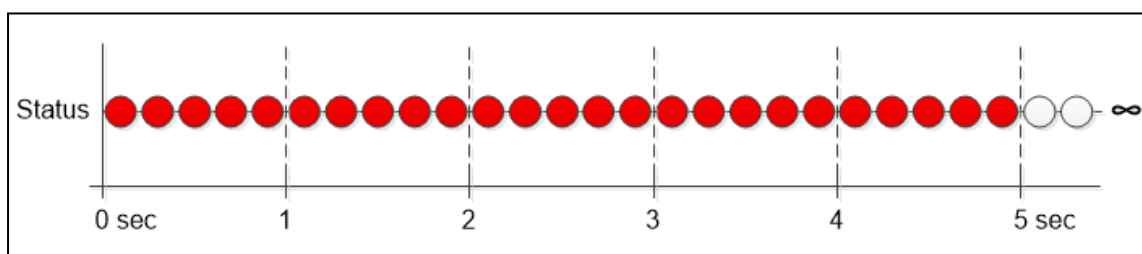


The boot process sequence is run through once.

2. Application or bootloader

2.1 Application

The boot process has been successfully completed and the product is now in the application. The product is ready for operation.



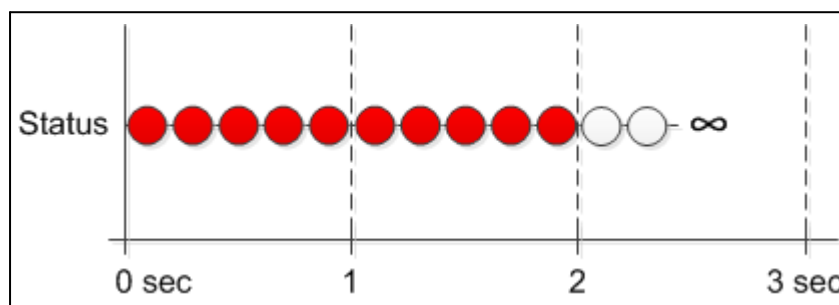
The status LED lights up for 5 seconds and goes out for about 300ms. The application sequence repeats itself.

2.2. Bootloader

The product is in the boot loader after booting. The application has not been loaded. This indicates an error in the firmware.

Updating the firmware can usually fix the problem → **Perform firmware update**

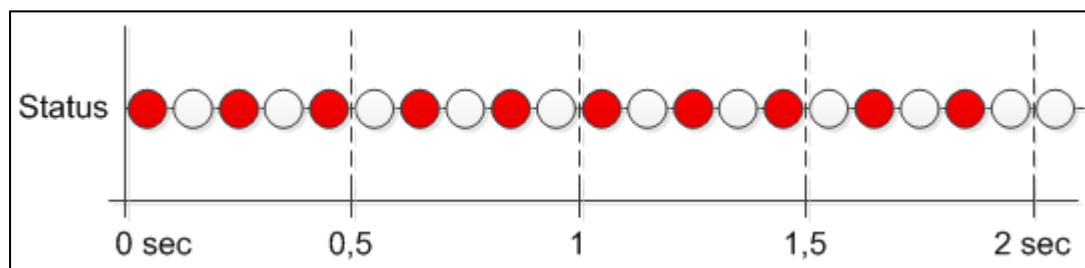
The status LED lights up for 2 seconds and goes out for about 300ms. The bootloader sequence repeats.



3. Bootloader forced

The product can be forced into the bootloader with DIP switch 4. → **see chapter "DIP switches"**

This function is for service purposes only.



The status LED flashes alternately for 2 seconds. This sequence is only run through once.

3.1.1.2. Configuring the module

3.1.1.2.1. Configuration via DELIB Configuration utility

The configuration is described in chapter Configuration Ethernet interface.

3.1.1.2.2. Configuration via internal web server

The RO-ETH module has its own web server via which the configuration can also be made.

The screenshot shows the DEDITEC web configuration interface. At the top, there is a header with the company name "DEDITEC GMBH" and an "Infohotline" number: "+(49) 0 22 32 50 40 80". Below the header, there are navigation tabs: "Configuration", "In-/Outputs", "Custom", and "Logout". The "Configuration" tab is selected. Below the tabs, there is a status bar showing "Product name: RO-ETH (DEDITEC RO-ETH Module)" and "User: gast / Session ends in: 684". On the left side, there is a sidebar menu with the following items: "General", "Network configuration", "Network time (NTP)", "HTTP-Server", "Mail-Server", "User-Manager", "FW-Update", "Log's", "Status / Reboot", and "Security". The "Network configuration" item is selected. The main content area is titled "Network configuration" and contains the following fields: "IP-address:" with value "192.168.1.1", "Netmask:" with value "255.255.255.0", "Std-GW:" with value "192.168.1.254", "DNS:" with value "192.168.1.254", and "TCP port:" with value "9912". Below these fields, there is a notice: "Notice: Changing TCP port requires board reboot". At the bottom right of the main content area, there is a circular icon with a "G" and the text "Update parameter".

3.1.1.2.3. Factory settings

The factory settings of the ethernet module include following settings:

IP address: 192.168.1.1

The delivery status can be restored at any time by pressing the left button.

see chapter: Reset IP address

IP Adresse	192.168.1.1
Port	9912
Subnetz Maske	255.255.255.0
Standard Gateway	192.168.1.254

3.1.1.3. Restore basic configuration

3.1.1.3.1. Restore IP address

The default value of the IP address is: 192.168.1.1

Left Button:

Restore IP address to default (192.168.1.1):

To restore the IP address proceed as follows:

- Push the button at least 5 sec.
- After that, the left LEDs "CPU Activity" and "Interface Activity" should be flashing 4 times (confirmation of receipt)
- After this, the module has following settings:

IP Adresse	192.168.1.1
Port	9912
Subnetz Maske	255.255.255.0
Standard Gateway	192.168.1.254

3.1.1.3.2. Restore firmware

To restore the firmware to default value proceed as follows:

Right Button:

Restore firmware to factory settings

To restore the firmware to factory settings proceed as follow:

- Press the button at least 10sec.
- After this, the three LED's "CPU Activity", "Interface Activity" and "Status" should be flashing 4 times (confirmation of receipt).
- After this, the module restarts.

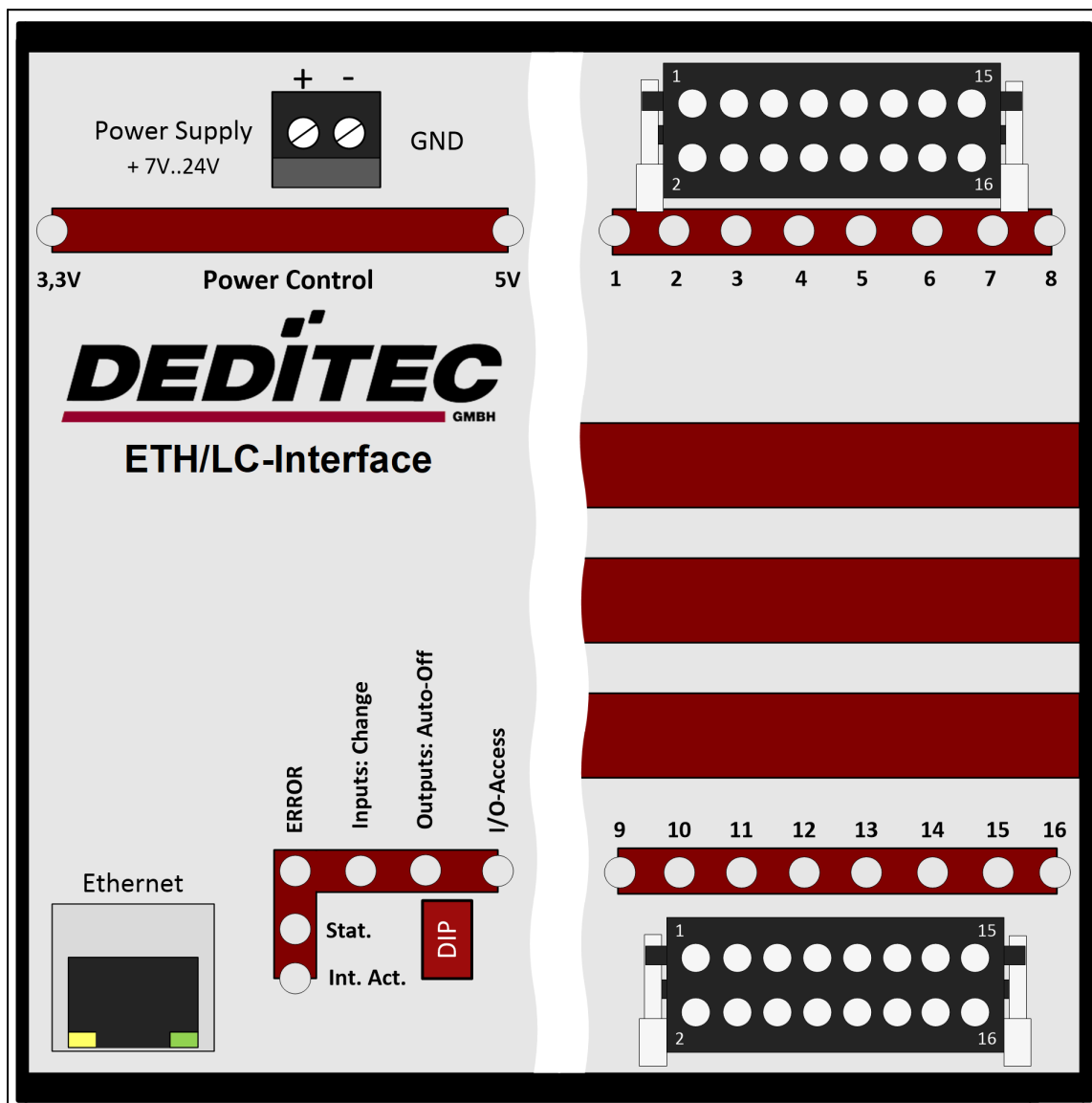
The firmware and configuration of the factory settings are now active again!

3.1.2. RO-ETH/LC

3.1.2.1. Hardware description

3.1.2.1.1. Overview screen

The figure shows the control module with Ethernet interface (left) in combination with an input/output module (right).



3.1.2.1.2. Technical data

Electrical data:

Supply voltage: 7V DC .. 24V DC

Power consumption: max. 5W

Environment:

Ambient temperature: +10..+50 °C

Humidity: 90 %

Condensation: Not allowed

Mechanics:

Dimensions in mm (LxWxH): L* x 122 x 51,5

*the length varies with the type and
number of connected submodules

Fastening: Top-hat rail TS 35 x 7.5 mm

Optional:

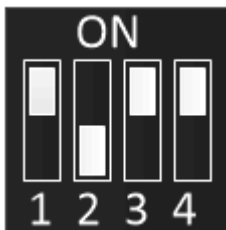
galv. separation: Available as accessory

3.1.2.1.3. DIP-Switches

The following functions can be activated or deactivated using the DIP switches:

- DIP 1: ON = DHCP on, OFF = DHCP off
- DIP 2: ON = Write protection on, OFF = Write protection off
- DIP 3: ON = Factory settings off, OFF = Factory settings on
- DIP 4: Bootloader Mode (for service purposes only)

DIP switch positions in delivery state:



Notes:

Changes to the DIP switches are only applied after a restart of the module !

The DIP switch settings are always prioritized before the software settings !

Explanation of the DIP switch functions:

DIP 1	Mode / Explanation
ON	DHCP is enabled The network settings IP, subnet mask, DNS domain and gateway are obtained from your network via a DHCP server. DHCP is enforced even if DHCP has been disabled by software.
OFF	DHCP disabled The network settings stored in Module-Configuration-Memory are used. These settings can be edited and saved using the DELIB software.

DIP 2	Mode / Explanation
ON	Write protection enabled No configurations of the network settings can be made via the DELIB Configuration Utility or the Module Config.
OFF	Write protection disabled Configurations of the network settings can be made via the DELIB Configuration Utility or the Module Config.

DIP 3	Mode / Explanation
ON	<p>Factory settings disabled</p> <p>The module starts with the settings configured in the Module-Configuration-Memory.</p>
OFF	<p>Factory settings activated</p> <p>The values stored in the Module-Configuration-Memory (IP address, gateway, subnet mask, DHCP) are ignored when the module starts.</p> <p>The module starts with the following factory settings.</p> <p>Ethernet/LAN Boardname: Product dependent (e.g. BS-ETH) Write Protection: OFF DHCP: OFF IP: 192.168.1.1 Subnetmask: 255.255.255.0 Gateway: 192.168.1.254 Port: 9912</p> <p>The values stored in the Module-Configuration-Memory are not changed.</p> <p>Important!</p> <p>If the DHCP DIP switch is also activated, the network settings of the DHCP server are used.</p>

DIP 4	Mode / Explanation
ON	Bootloader disabled The module starts normally.
OFF	Bootloader enabled The module remains forcibly in the boot loader. The firmware is not started.

3.1.2.1.4. Control LEDs

The Ethernet module has a series of control LEDs. They are used for easy visual indication of various state functions.

While switching the module on, in normal operating mode, the module should signalize the following sequence:

- approx. 20 sec after switching the module on, LED 1 and 2 are flashing briefly.
-> Operating system has been loaded successfully.
- Then LED 3 is on permanently and LED 1 is flashing. -> Module is ready.

3.1.2.1.4.1. Definition of LEDs

LED	Label	Description
above	3,3V	Internal 3,3V power supply
above	5V	Internal 5V power supply
1	CPU Activity	2x flashing + long break. Operating system reports: Status OK
2	Interface Activity	Active communication over Ethernet
3	Status	LED is on -> Module is ready
4	ERROR	Error during ethernet-transfer (for details see document "Serial protocol")
5	Inputs: Change	"State change" between 2 read-out cycles detected
6	Outputs: Auto-Off	Due to timeout, all outputs are switched-off for safety reasons
7	I/O Access	CPU-access to the connected I/O modules.

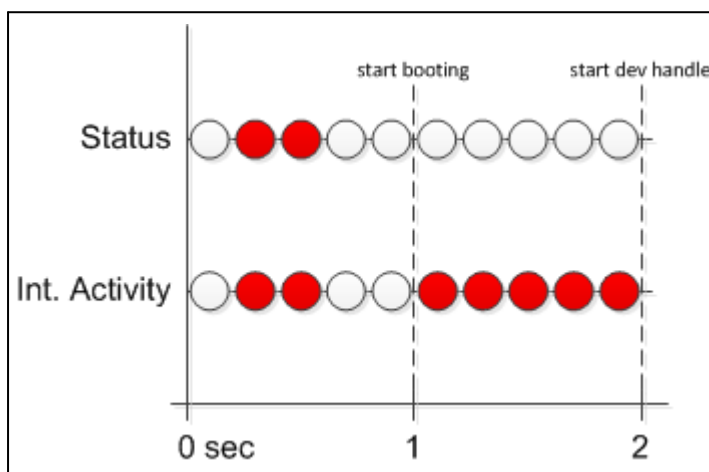
3.1.2.1.4.2. Blinking behavior of the LEDs

The blinking sequences of the status LEDs are shown below.



1. Boot process

The boot process starts immediately after the power supply is switched on.

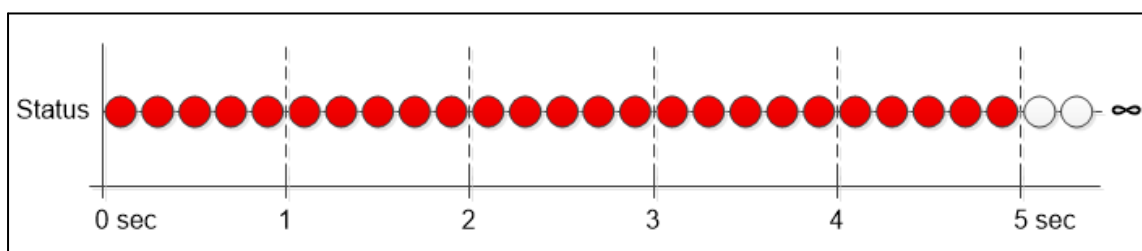


The boot process sequence is run through once.

2. Application or bootloader

2.1 Application

The boot process has been successfully completed and the product is now in the application. The product is ready for operation.



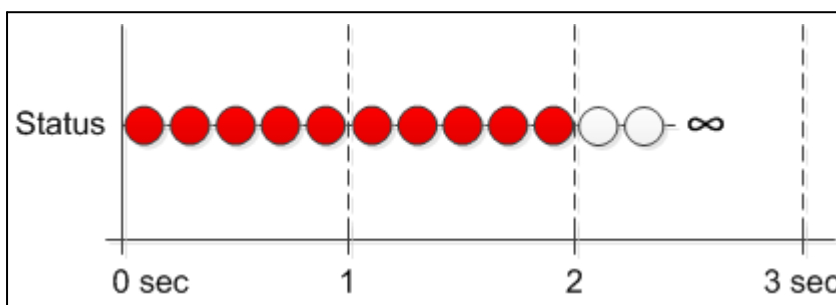
The status LED lights up for 5 seconds and goes out for about 300ms. The application sequence repeats itself.

2.2. Bootloader

The product is in the boot loader after booting. The application has not been loaded. This indicates an error in the firmware.

Updating the firmware can usually fix the problem → **Perform firmware update**

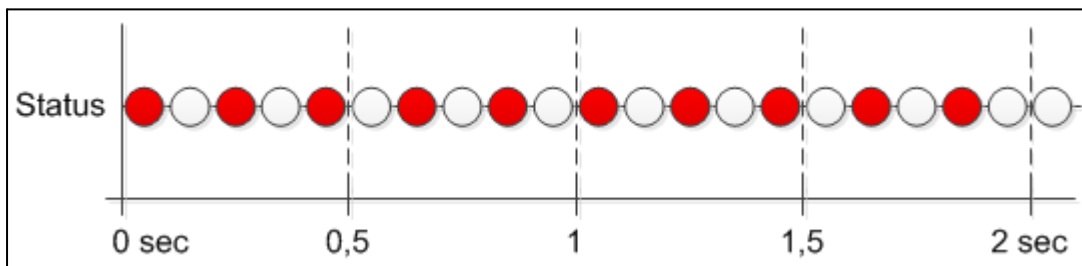
The status LED lights up for 2 seconds and goes out for about 300ms. The bootloader sequence repeats.



3. Bootloader forced

The product can be forced into the bootloader with DIP switch 4. → **see chapter "DIP switches"**

This function is for service purposes only.



The status LED flashes alternately for 2 seconds. This sequence is only run through once.

3.1.2.2. Configuring the module

3.1.2.2.1. Configuration via DELIB Configuration utility

The configuration is described in chapter Configuration Ethernet interface.

3.1.2.2.2. Factory settings

The RO-ETH/LC interface supports the DHCP protocol, which enables automatic assignment of the network configuration.

The DHCP protocol is activated by default in the delivery state.

If a manual network configuration is desired, DIP switch 1 (DHCP) and DIP switch 2 (write protection) must be set to OFF.

The configuration can then be performed via the DELIB Configuration Utility.

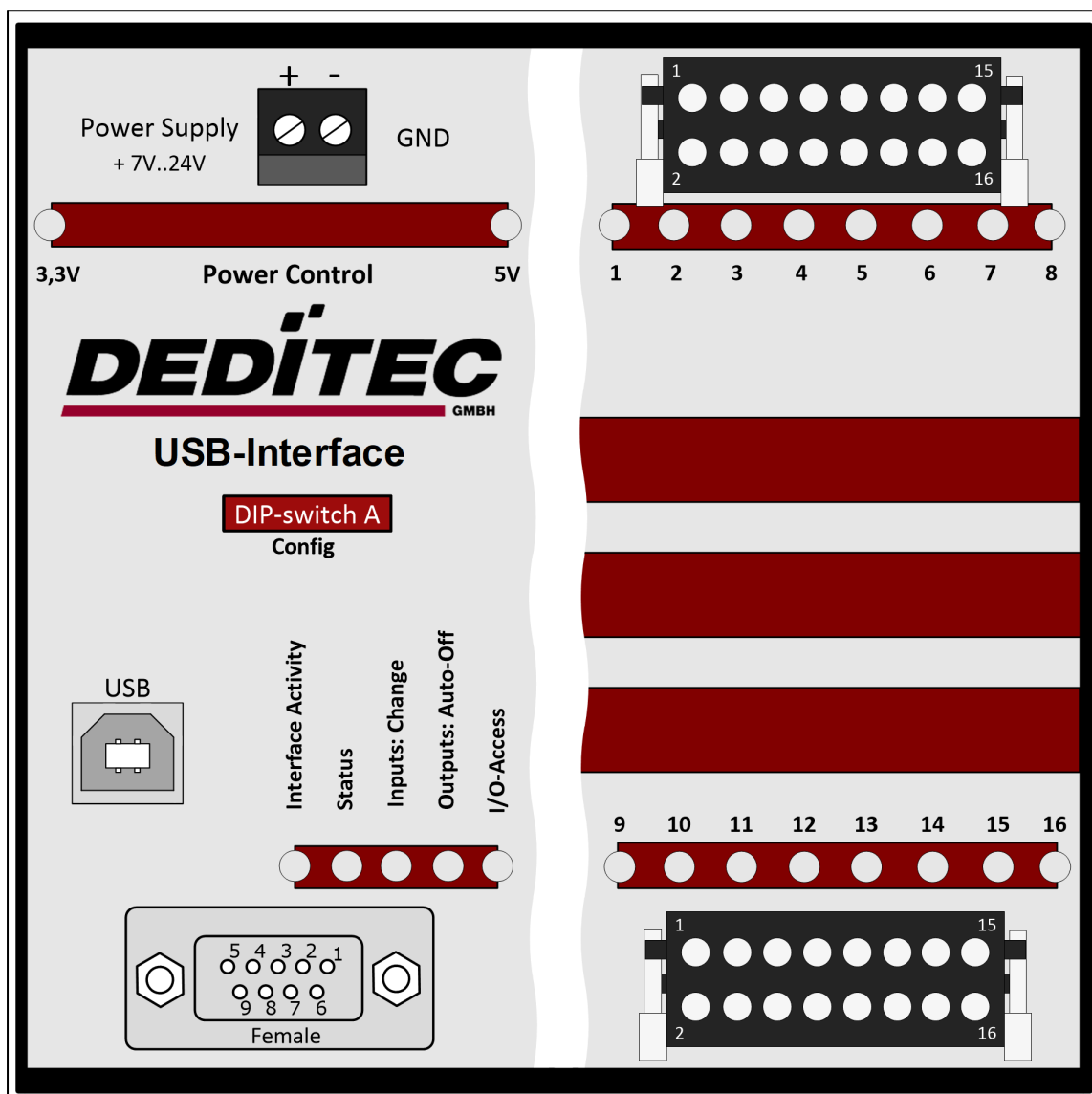
3.1.3. RO-USB

3.1.3.1. Hardware description

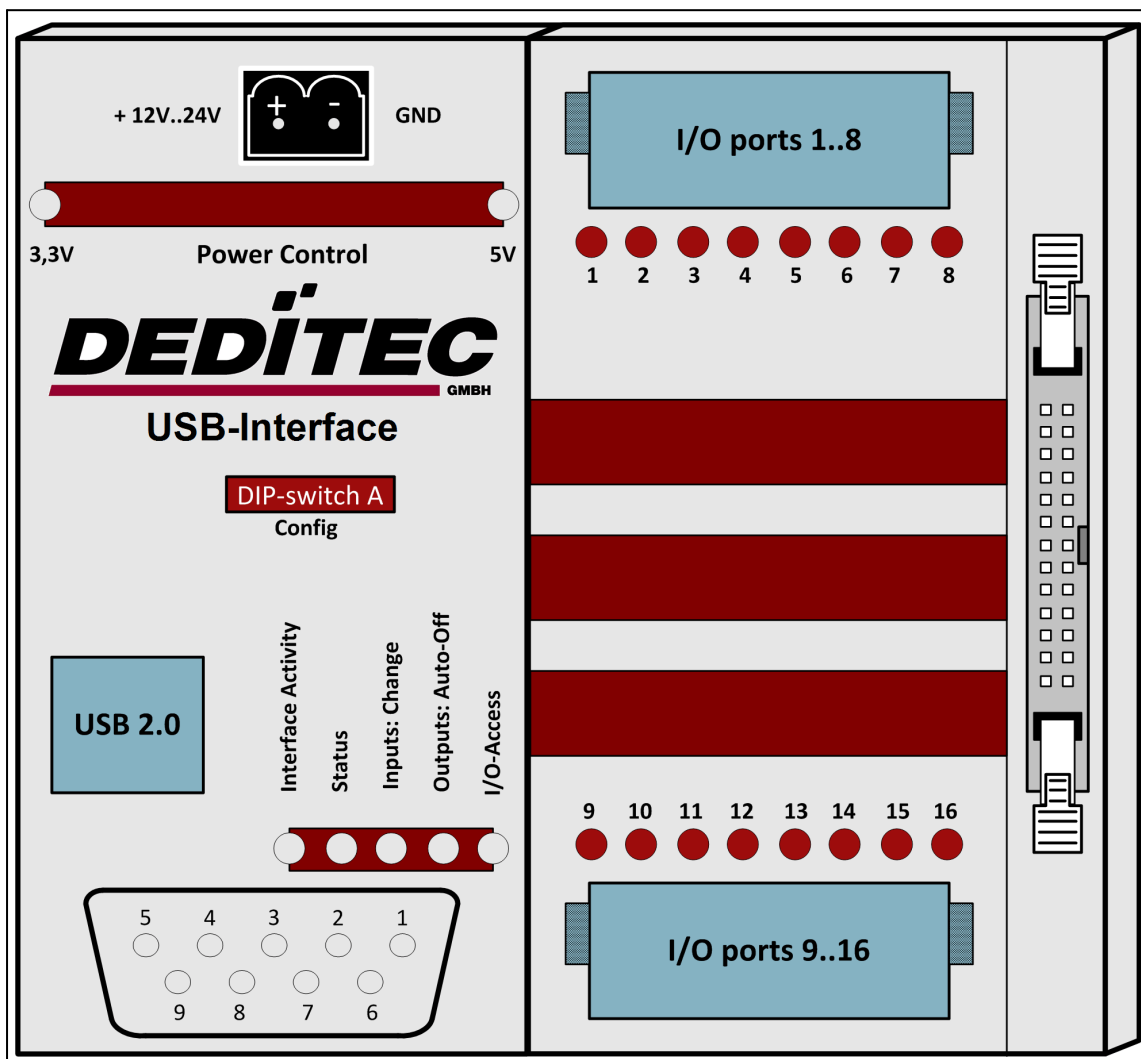
3.1.3.1.1. Overview screen

Ser Interface (Revision 2.0)

The picture shows the control module with USB interface (left) in combination with an input/output module (right). A USB A/B cable is used for the connection to the USB bus. A corresponding cable is included in the scope of delivery.

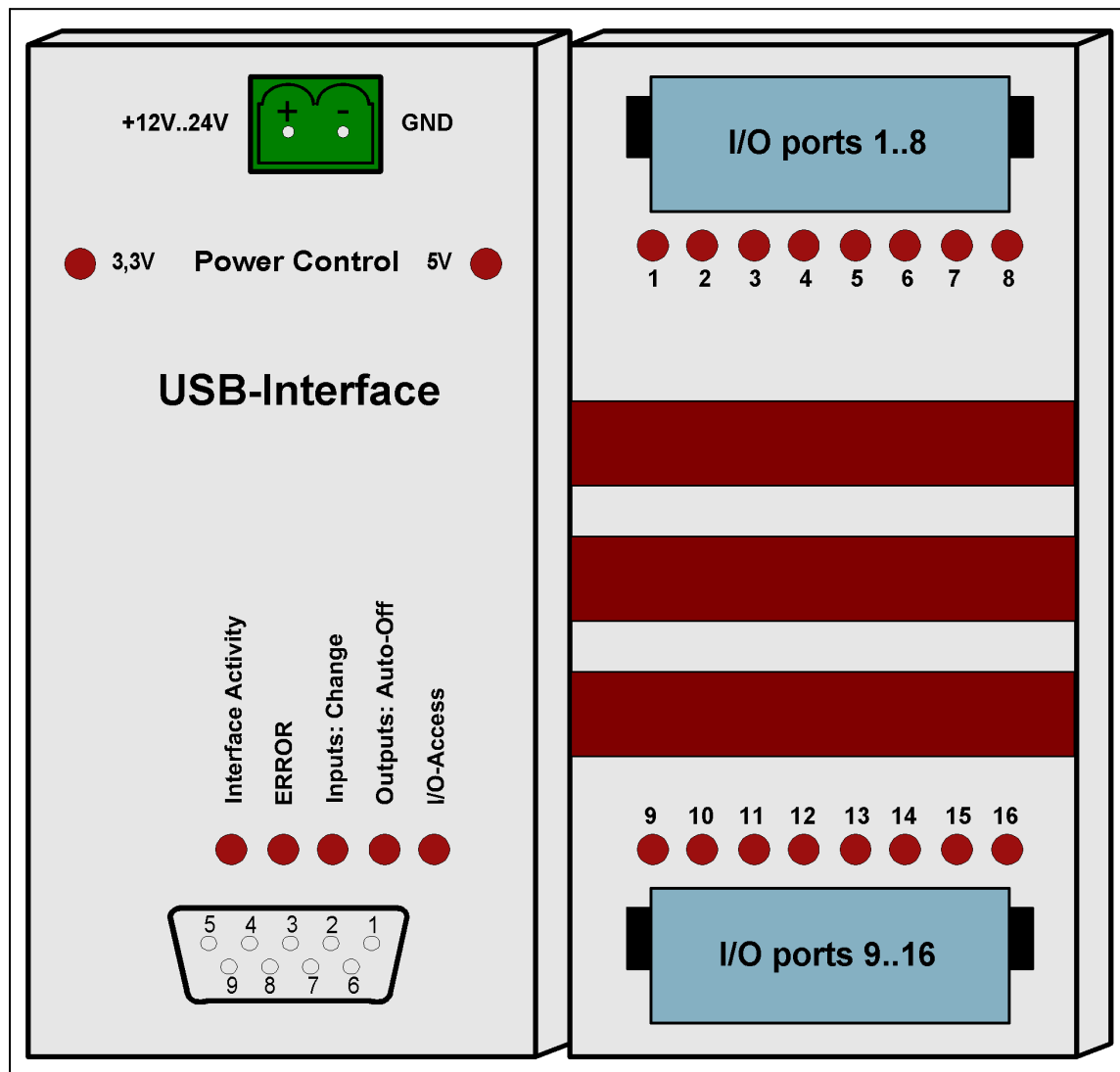


The picture below shows the control module with USB interface (left) in combination with an I/O module of the flexible connector system (right). A USB A/B cable is used for the connection to the USB bus. A corresponding cable is included in the scope of delivery.

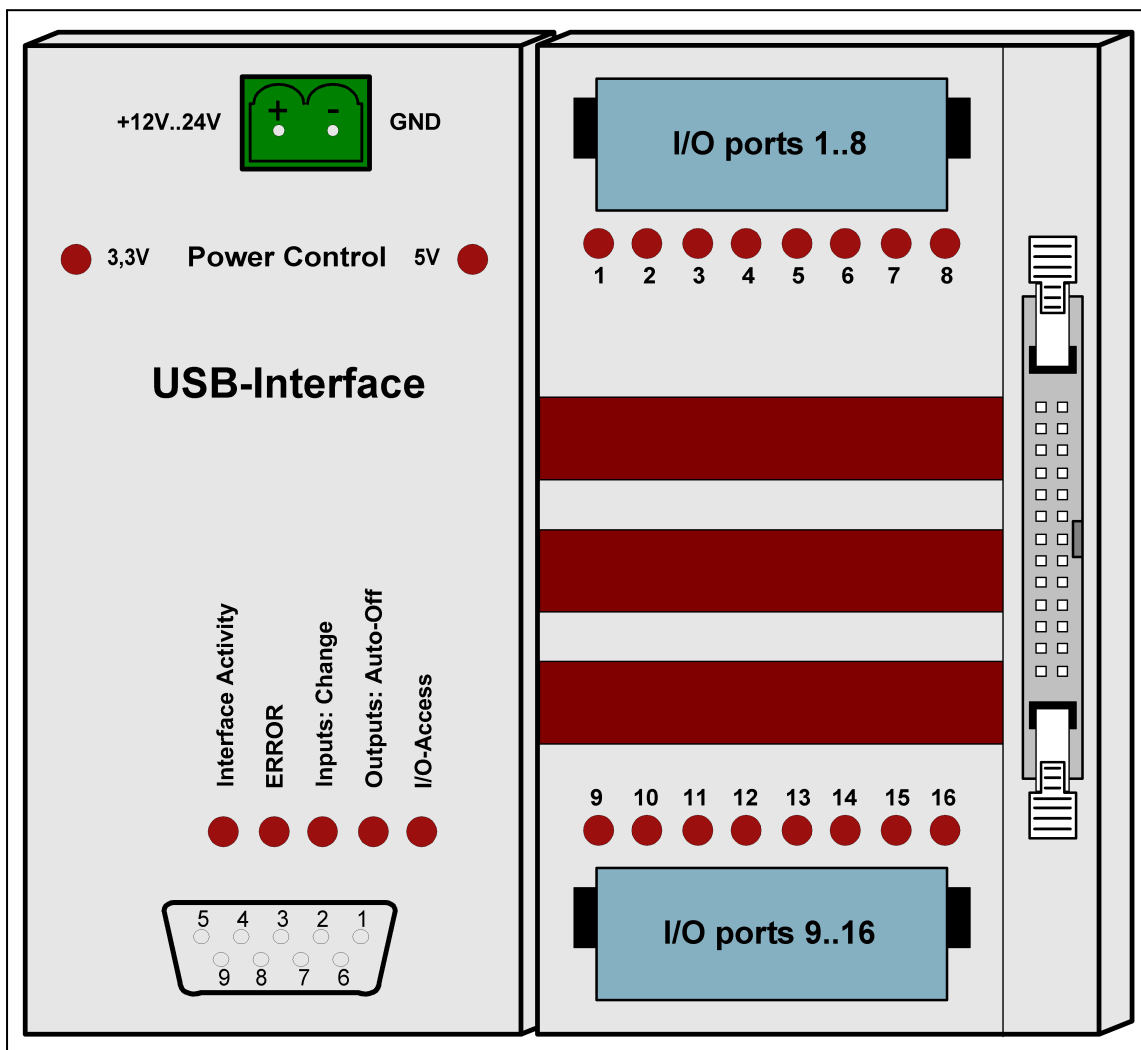


Seriell Interface (Revision 1.0)

The picture shows the control module with USB interface (left) in combination with an input/output module (right). For the connection to the USB bus, a suitable adapter module in the form of a USB stick is supplied.



The picture below shows the control module with USB interface (left) in combination with an I/O module of the flexible connector system (right). For the connection to the USB bus, a suitable adapter module in the form of a USB stick is supplied.



3.1.3.1.2. Technical data

Electrical data:

Supply voltage: 7V DC .. 24V DC

Power consumption: max. 5W

Environment:

Ambient temperature: +10..+50 °C

Humidity: 90 %

Condensation: Not allowed

Mechanics:

Dimensions in mm (LxWxH): L* x 122 x 51,5

*the length varies with the type and
number of connected submodules

Fastening: Top-hat rail TS 35 x 7.5 mm

Optional:

galv. separation: Available as accessory

3.1.3.1.3. Control LEDs

The USB module has a series of control LEDs. They are used for easy visual indication of various state functions.

While switching-on the module, it should signalize the following sequence:

- all 5 LEDs flashing briefly
- right LED (I/O access) flashing briefly
- all 5 LEDs flashing briefly

3.1.3.1.3.1. Definition of LEDs

LED	Description
3,3V	Internal 3,3V power supply
5V	Internal 5V power supply
Interface Activity	Active communication- over the USB bus
ERROR	Error during USB-transfer (for details see document "USB protocol")
Inputs: Change	State change between 2 read-out cycles detected
Outputs: Auto-Off	Due to timeout, all outputs are switched-off for safety reasons
I/O Access	CPU-access on the inputs and outputs of the connected modules

3.1.3.1.4. Configuration via the DELIB Configuration Utility

The configuration is described in the chapter Configuration **USB interface**.

3.1.4. RO-CAN

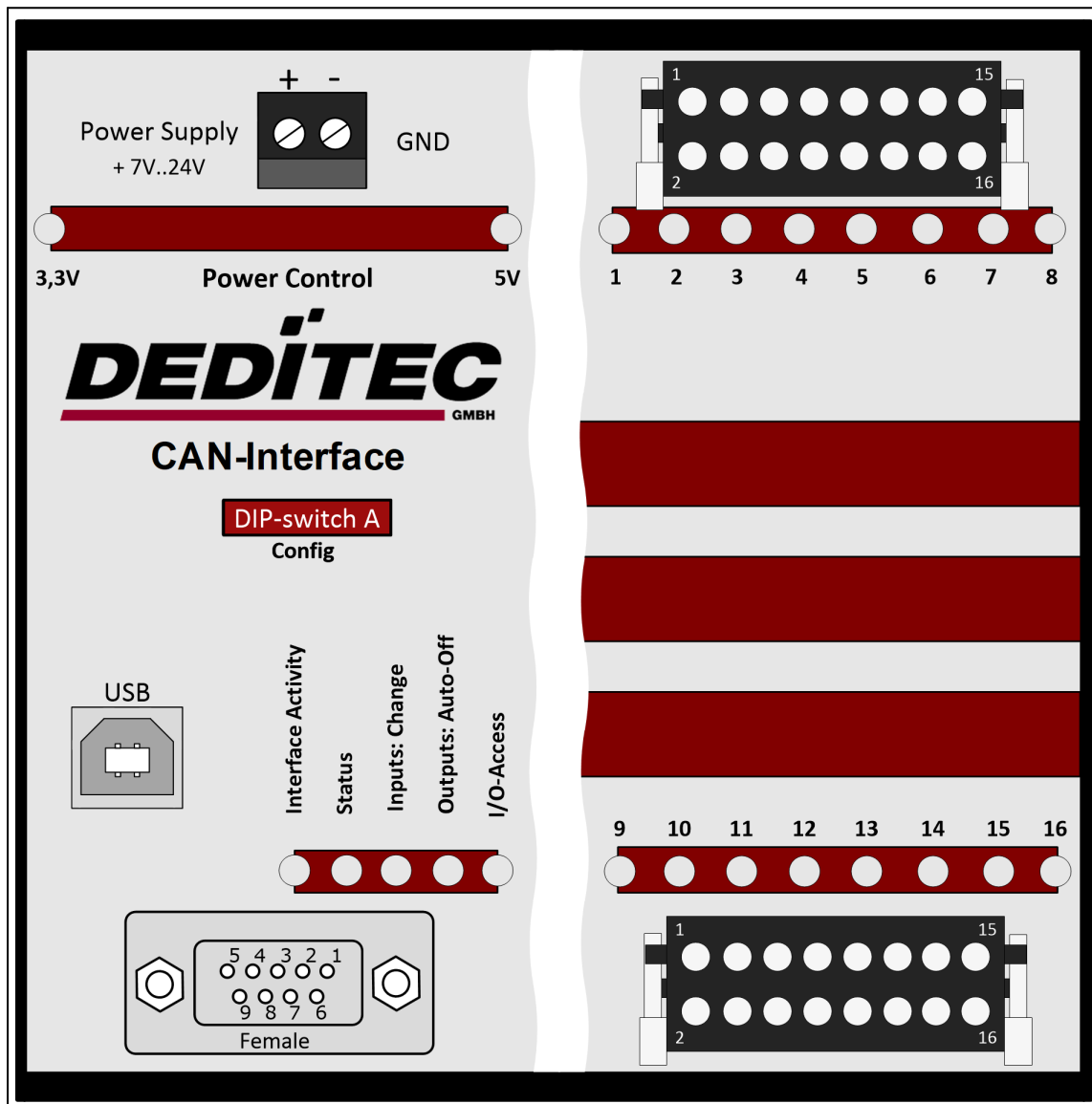
3.1.4.1. Hardware description

3.1.4.1.1. Overview screen

RO-CAN Interface (Revision 2.0)

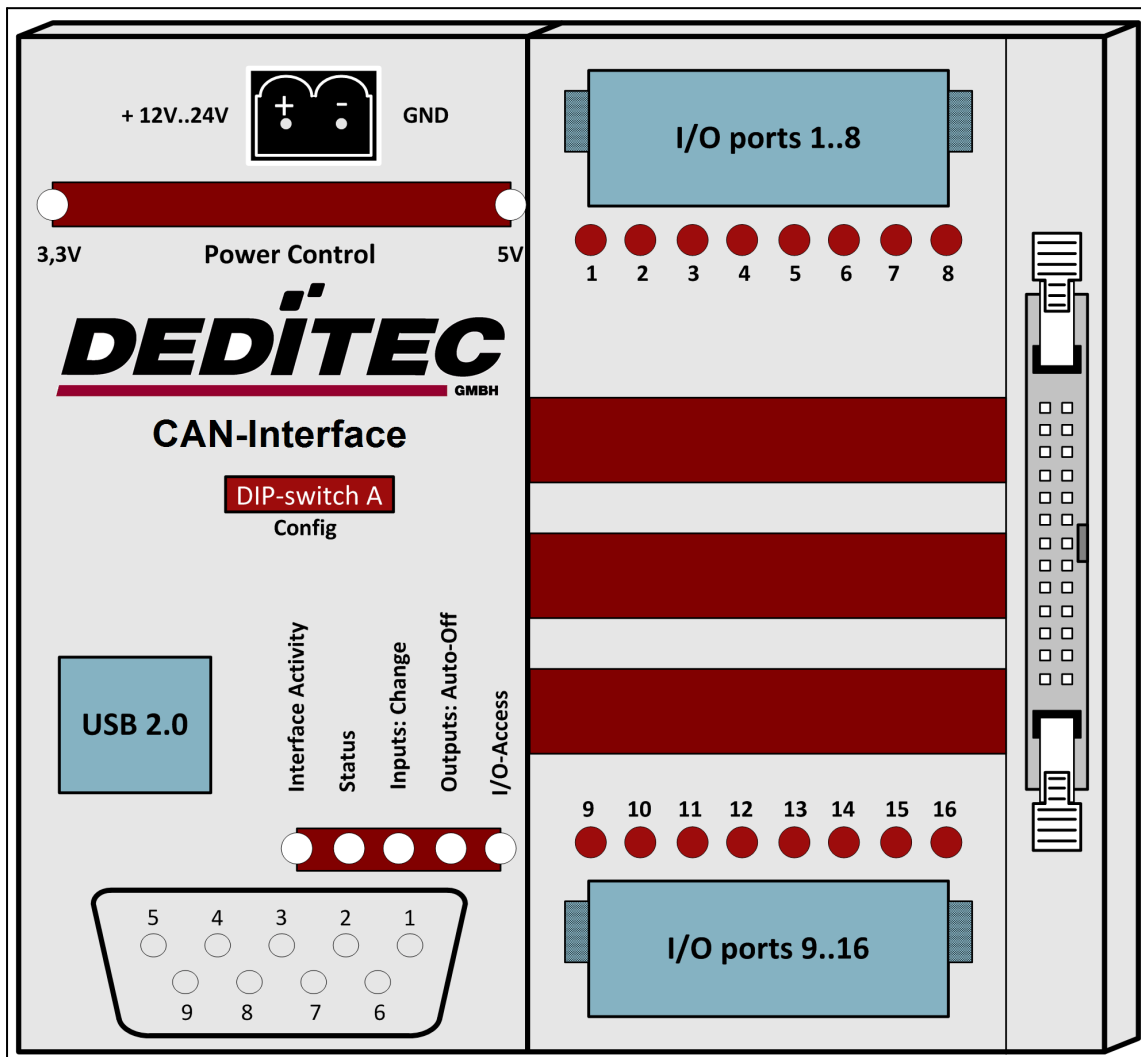
The picture shows the control module with CAN interface (left) in combination with an input/output module (right).

The module has an additional USB interface for configuration.



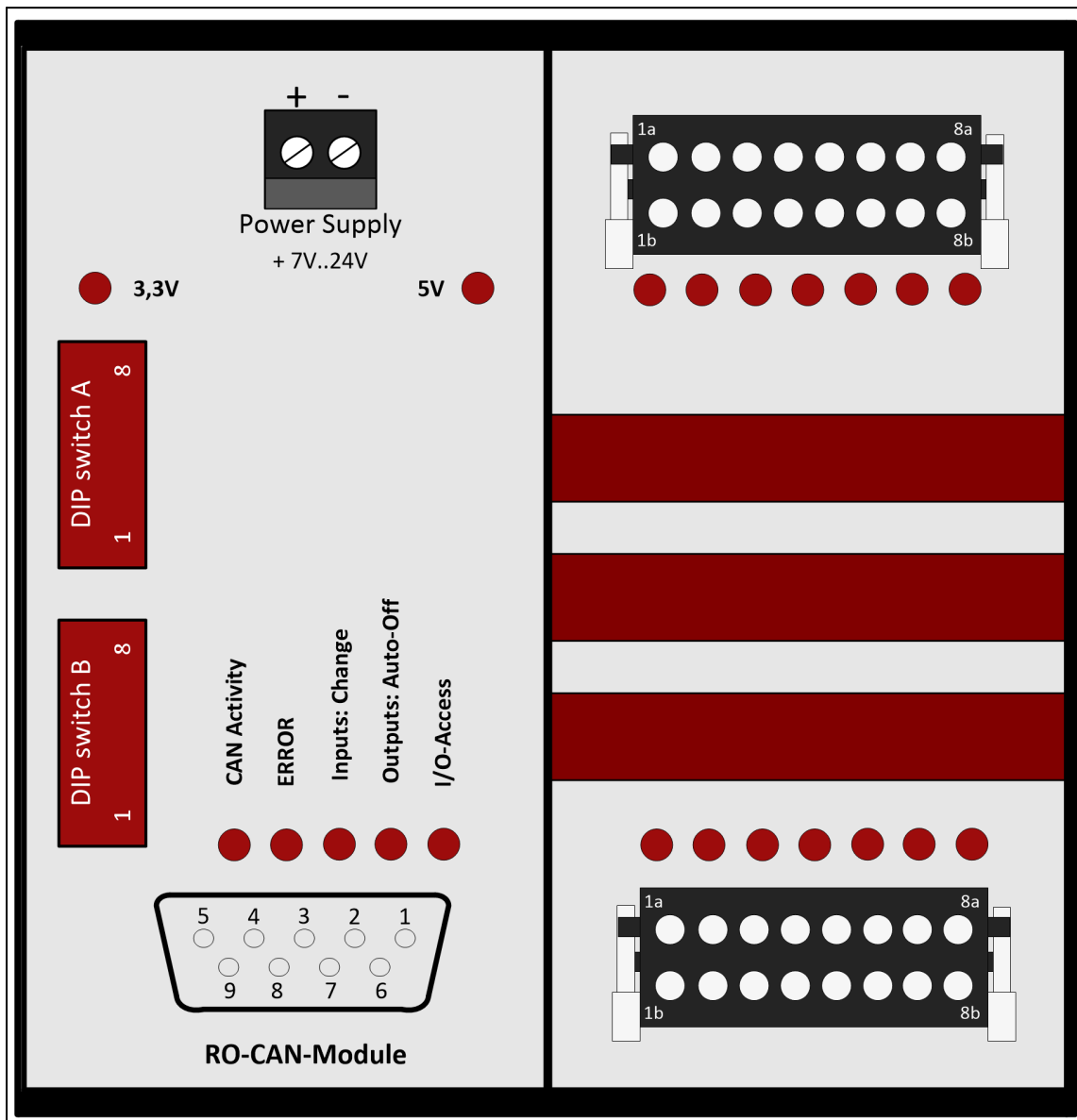
The picture shows the control module with CAN interface (left) in combination with an I/O module of the flexible connector system (right).

The module has an additional USB interface for configuration.



RO-CAN Interface (Revision 1.0)

The picture shows the control module with CAN interface (left) in combination with an input/output module (right).



3.1.4.1.2. Technical data

Electrical data:

Supply voltage: 7V DC .. 24V DC

Power consumption: max. 5W

Environment:

Ambient temperature: +10..+50 °C

Humidity: 90 %

Condensation: Not allowed

Mechanics:

Dimensions in mm (LxWxH): L* x 122 x 51,5

*the length varies with the type and
number of connected submodules

Fastening: Top-hat rail TS 35 x 7.5 mm

Optional:

galv. separation: Available as accessory

3.1.4.1.3. Control LEDs

The CAN module has a series of control LEDs. They are used for easy visual indication of various state functions.

While switching-on the module in DIP-switch mode or software mode, the module should signalize the following sequence:

- all 5 LEDs flashing briefly
- right LED (I/O access) flashing briefly

In "special mode", the following signal sequence should appear:

- all 5 LEDs flashing briefly
- right LED (I/O access) flashing briefly
- all 5 LEDs flashing briefly

3.1.4.1.3.1. Definition of LEDs

RO-CAN Interface (Revision 2.0)

Designation	Declaration
3,3V	Internal 3.3V supply voltage available.
5V	Internal 5V supply voltage available.
Interface Activity	Communication via CAN bus active.
Status	Module is in the application or bootloader
Inputs: Change	Change of state between 2 readout files was recorded.
Outputs: Auto-Off	All outputs were switched off for safety reasons due to the timeout.
I/O Access	Access of the CPU to inputs and outputs of the connected modules.

RO-CAN Interface (Revision 1.0)

Designation	Declaration
3,3V	Internal 3.3V supply voltage available.
5V	Internal 5V supply voltage available.
CAN Activity	Communication via CAN bus active.
ERROR	Error in CAN transmission (for more details see document "CAN protocol").
Inputs: Change	Change of state between 2 readout files was recorded.
Outputs: Auto-Off	All outputs were switched off for safety reasons due to the timeout.
I/O Access	Access of the CPU to inputs and outputs of the connected modules.

3.1.4.2. Configuring the module

In order to integrate a module into an existing bus system, at first it is necessary to assign a free module address and the appropriate bit rate. The "special mode" may be alternatively used to operate the system quickly.

3.1.4.2.1. DIP-switches

Some settings can be easily configured using DIP switches. "Special mode", software mode or DIP switch mode can be configured

DIP Switch A8	DIP Switch A7	Declaration
ON	ON	"Special Mode": flashing sequence at start, 100KHz, CAN-ID=0x100, Response-Module-Addr=1, no extended ID's
ON	OFF	For SERVICE purposes only: Application will not start. Forced in bootloader
OFF	ON	Software mode: Configuration via the CAN configuration utility
OFF	OFF	DIP switch mode: Configuration via DIP switch, response module addr=1, CAN 2.0A

Only in DIP switch mode

DIP Switch	Declaration
A6 bis A4	Setting the transmission rate
A3 bis B1	Setting the CAN address (Revision 1.0)
A3 bis A1	Setting the CAN address (Revision 2.0)

Hint:

The positions of the DIP switches are queried once when the module is started.

3.1.4.2.2. The "special mode"

The "special mode" is for setting the device easily to the default values. This is helpful for a quick and easy setup and simplify an error analysis or an initial operation.

This mode is active, if switching the DIP-switches A7 and A8 to "ON". The remaining DIP-switches are deactivated. The module will work with the following settings:

11 bit-addressing

100 kbit/s bitrate

CAN-address = 0x100

Response-Modul-Addr = 1 (responses are returned to this address)

3.1.4.2.3. Software mode

The configuration is described in the chapter **Configuration CAN interface**.

3.1.4.2.4. DIP-switch mode

In this mode, the interface module must be configured via the DIP switches. For this, the DIP switches A7=OFF and A8=OFF. The address space is 11 bits wide (CAN 2.0A). Depending on the revision of the CAN interface, the module address can be set via DIP A3..A1 (revision 2.0), or DIP A3..A1 and B8..B1 (revision 1.0). The response module addr is 1 (the responses are sent back to this address).

3.1.4.2.4.1. Setting up the transfer rate

Depending on the range of the CAN bus, different transmission speeds are achieved. The following bit rates can be set via DIP switches. Other bit rates may only be possible via software.

DIP Switch	1Mbit	500K	250K	125K	100K	50K	20K	10K
A6	On	On	On	On	Off	Off	Off	Off
A5	On	On	Off	Off	On	On	Off	Off
A4	On	Off	On	Off	On	Off	On	Off

3.1.4.2.4.2. Setting up the CAN module address

Each device in the CAN network requires a fixed address in order to be directly addressed. Depending on the revision of the CAN interface, up to 8 (revision 2.0) or 2047 (revision 1.0) different addresses can be set via DIP switches.

RO-CAN Interface (Revision 2.0)

DIP Switch	Bit	Value ON	Value OFF
A3	Bit 10	1024	0
A2	Bit 9	512	0
A1	Bit 8	256	0

Available addresses

Address	DIP A3	DIP A2	DIP A1
0	Off	Off	Off
256	Off	Off	On
512	Off	On	Off
768	Off	On	On
1024	On	Off	Off
1280	On	Off	On
1536	On	On	Off
1792	On	On	On

RO-CAN Interface (Revision 1.0)

DIP Switch	Bit	Value ON	Value OFF
A3	Bit 10	1024	0
A2	Bit 9	512	0
A1	Bit 8	256	0
B8	Bit 7	128	0
B7	Bit 6	64	0
B6	Bit 5	32	0
B5	Bit 4	16	0
B4	Bit 3	8	0
B3	Bit 2	4	0
B2	Bit 1	2	0
B1	Bit 0	1	0

Examples

DIP Switch	Address 0	Address 117	Address 588
A3	Off	Off	Off
A2	Off	Off	On
A1	Off	Off	Off
B8	Off	Off	Off
B7	Off	On	On
B6	Off	On	Off
B5	Off	On	Off
B4	Off	Off	On
B3	Off	On	On
B2	Off	Off	Off
B1	Off	On	Off

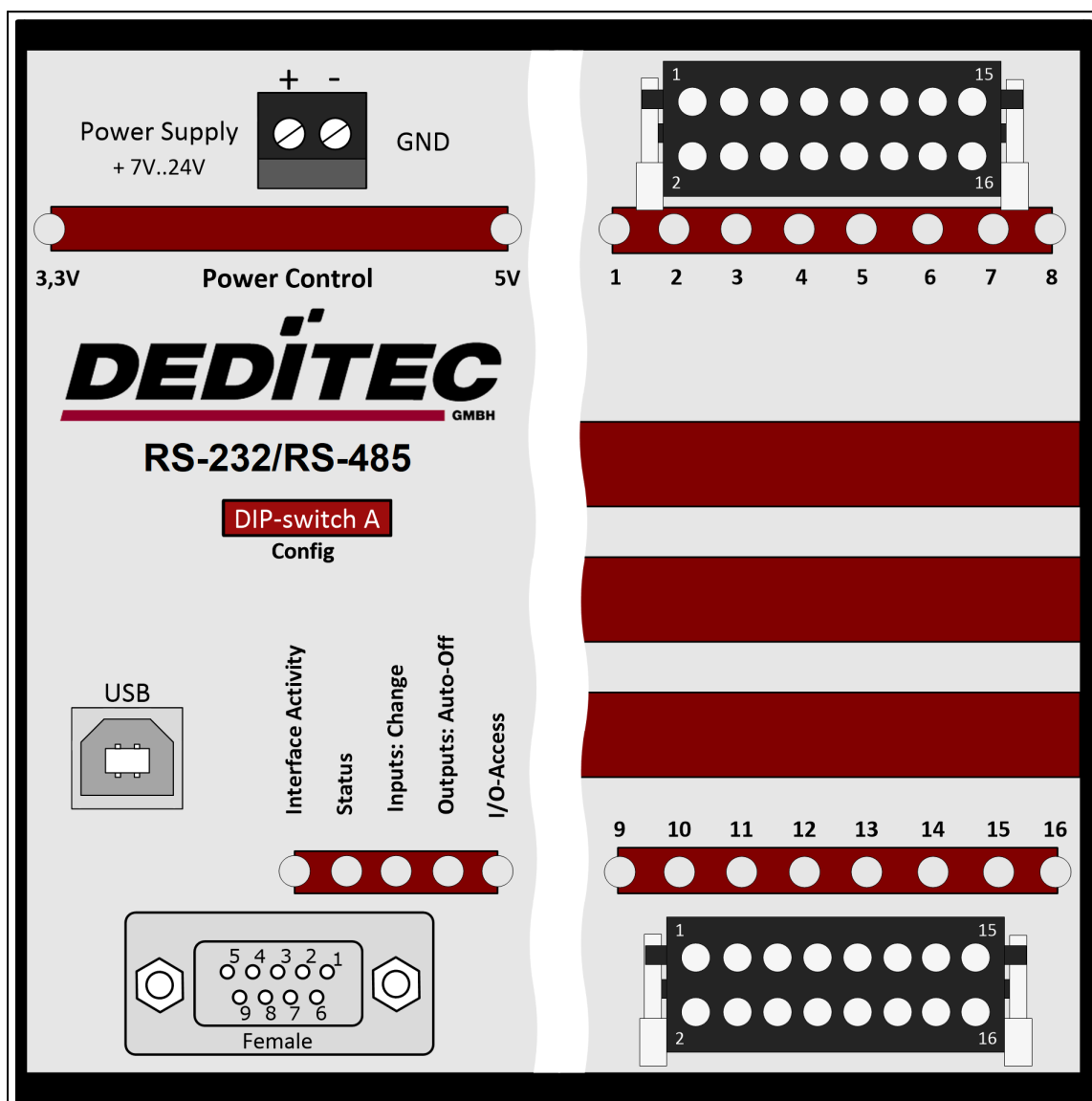
3.1.5. RO-SER

3.1.5.1. Hardware description

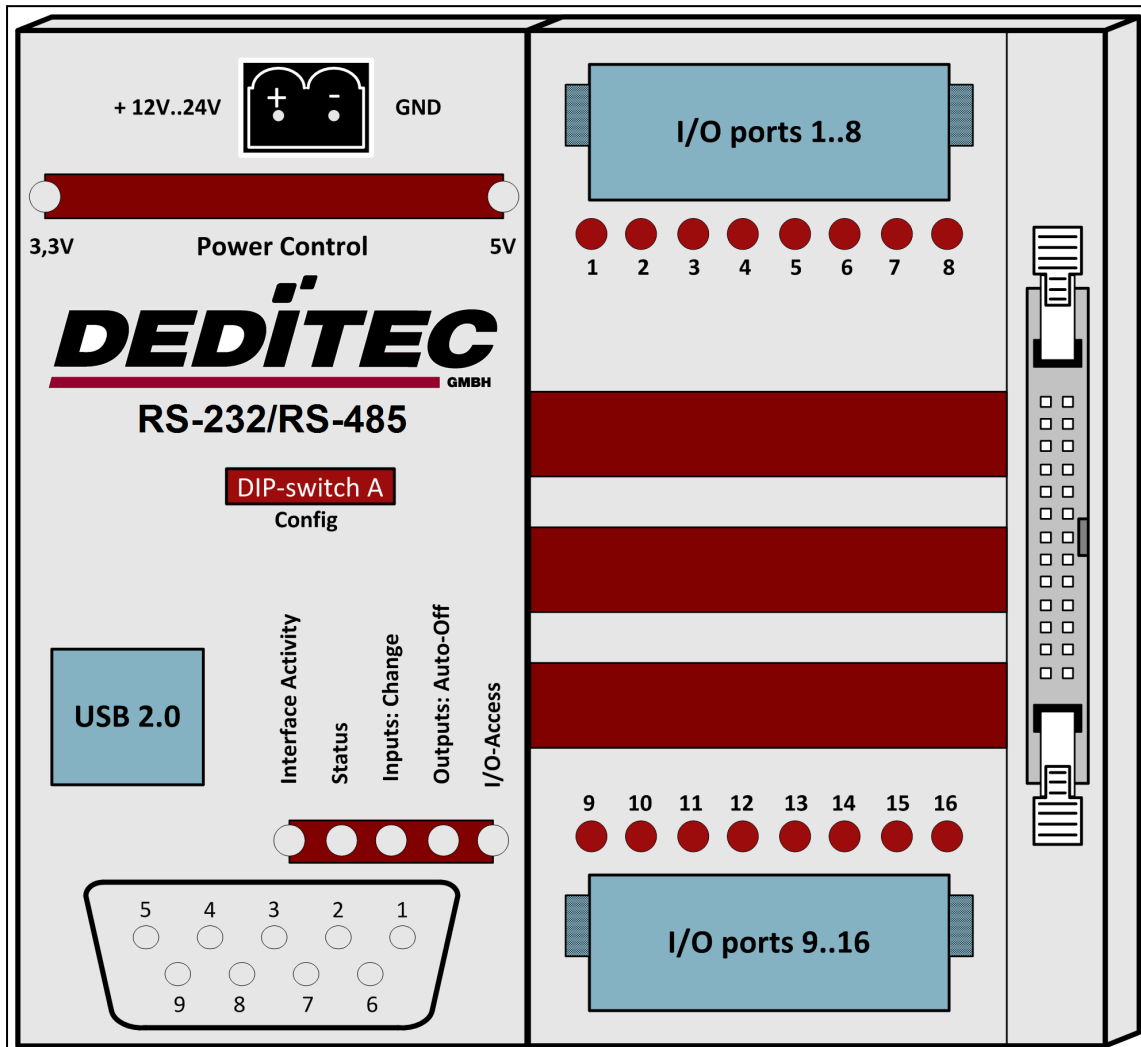
3.1.5.1.1. Overview screen

RO-SER Interface (Revision 2.0)

The illustration shows the control module with RS-232/RS-485 interface (left) in combination with an input/output module (right).

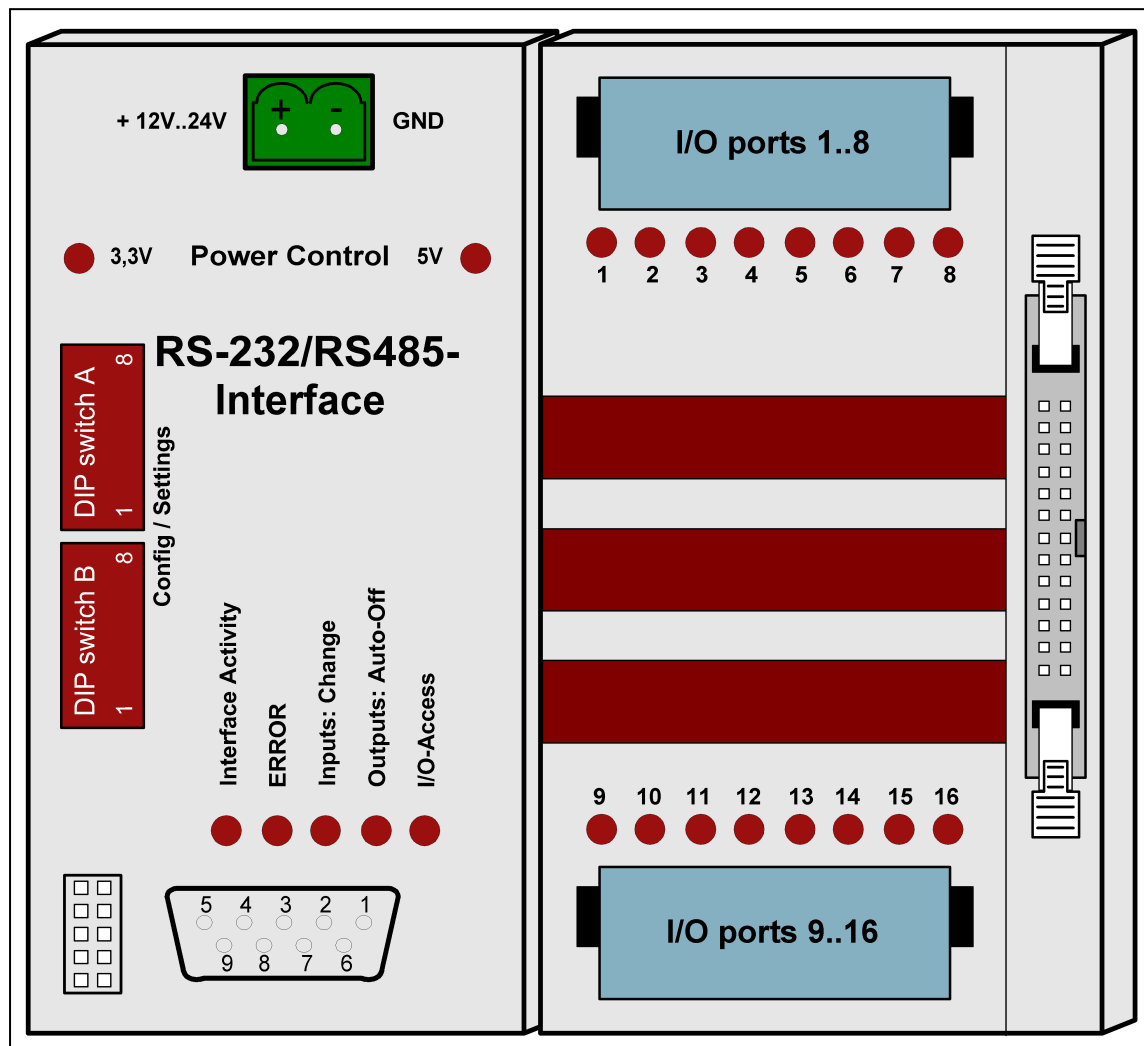


The illustration shows the control module with RS-232/RS-485 interface (left) in combination with an I/O module of the flexible connector system (right).

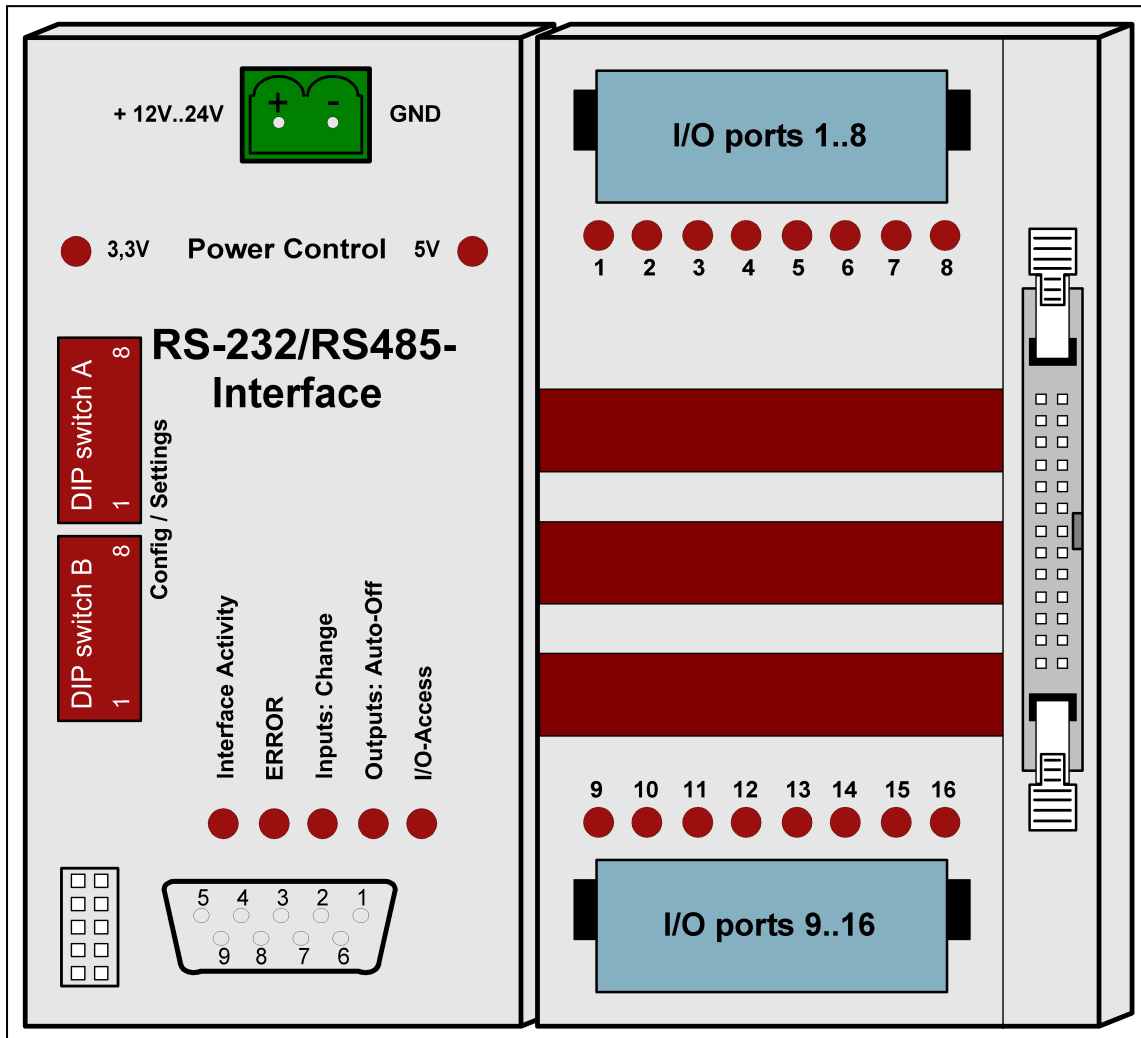


RO-SER Interface (Revision 1.0)

The illustration shows the control module with RS-232/RS-485 interface (left) in combination with an input/output module (right).



The illustration shows the control module with RS-232/RS-485 interface (left) in combination with an I/O module of the flexible connector system (right).



3.1.5.1.2. Technical data

Electrical data:

Supply voltage: 7V DC .. 24V DC

Power consumption: max. 5W

Environment:

Ambient temperature: +10..+50 °C

Humidity: 90 %

Condensation: Not allowed

Mechanics:

Dimensions in mm (LxWxH): L* x 122 x 51,5

*the length varies with the type and
number of connected submodules

Fastening: Top-hat rail TS 35 x 7.5 mm

Optional:

galv. separation: Available as accessory

3.1.5.1.3. Control LEDs

On the RS-232/RS-485 module there are a number of control LEDs. They are used for simple visual status indication of various functions.

In normal operation, the following flashing sequence should occur after switching on the module:

- all five LEDs flash briefly
- right LED (I/O Access) flashes briefly

In "special mode", the following flashing sequence should occur after switching on the module:

- all five LEDs flash briefly
- right LED (I/O Access) flashes briefly
- all five LEDs flash briefly

3.1.5.1.3.1. Definition of LEDs

LED	Description
3,3V	Internal 3,3V power supply
5V	Internal 5V power supply
Interface Activity	Active communication over the RS-232/RS-485 bus
ERROR	Error during serial-transfer (for details see document "Serial protocol")
Inputs: Change	State change between 2 read-out cycles detected
Outputs: Auto-Off	Due to timeout, all outputs are switched-off for safety reasons
I/O Access	CPU-access on the inputs and outputs of the connected modules

3.1.5.2. Configuring the module

In order to integrate a module into an existing bus system, it is at first necessary to assign a free module address and the appropriate bit rate. The "special mode" may be alternatively used to operate the system quickly.

3.1.5.2.1. DIP-switches

Some settings can be easily configured using DIP switches. "Special Mode", text mode or DIP switch mode can be configured

DIP Switch A8	DIP Switch A7	Declaration
ON	ON	"Special Mode": flashing sequence at start, 115k baud rate, module address=0
ON	OFF	For SERVICE purposes only: Application will not start. Forced in bootloader
OFF	ON	DIP-Switch-Mode
OFF	OFF	Serial text mode

Only in serial text or DIP switch mode

DIP Switch A6	Declaration
ON	Echo = ON, serial received characters are returned
OFF	Echo = OFF

DIP Switch	Declaration
A5	Reserved
A4 bis A1	Setting the baud rate

Hint:

The positions of the DIP switches are queried once when the module is started.

3.1.5.2.2. The "special mode"

"Special mode" is used to quickly and easily set the unit to fixed default values. This is helpful for a quick and easy commissioning of the module. An error analysis or initial commissioning is thus facilitated.

If switches A7 and A8 are set to "on", this mode is entered. All other DIP switches are now deactivated.

The module then works with the following settings:

115K baud

Module address = 0 (deactivated)

Echo = Off

3.1.5.2.3. DIP-Switch-Mode

In this mode, the configuration of the serial interface (baud rate and echo) is set directly by DIP switch.

To enter the DIP switch mode, DIP switch A7 must be set to "on" and A8 to "off". Communication with the module must take place via the serial protocol (for further information, please refer to the Manual "[DEDITEC Communication protocols](#)").

Hint:

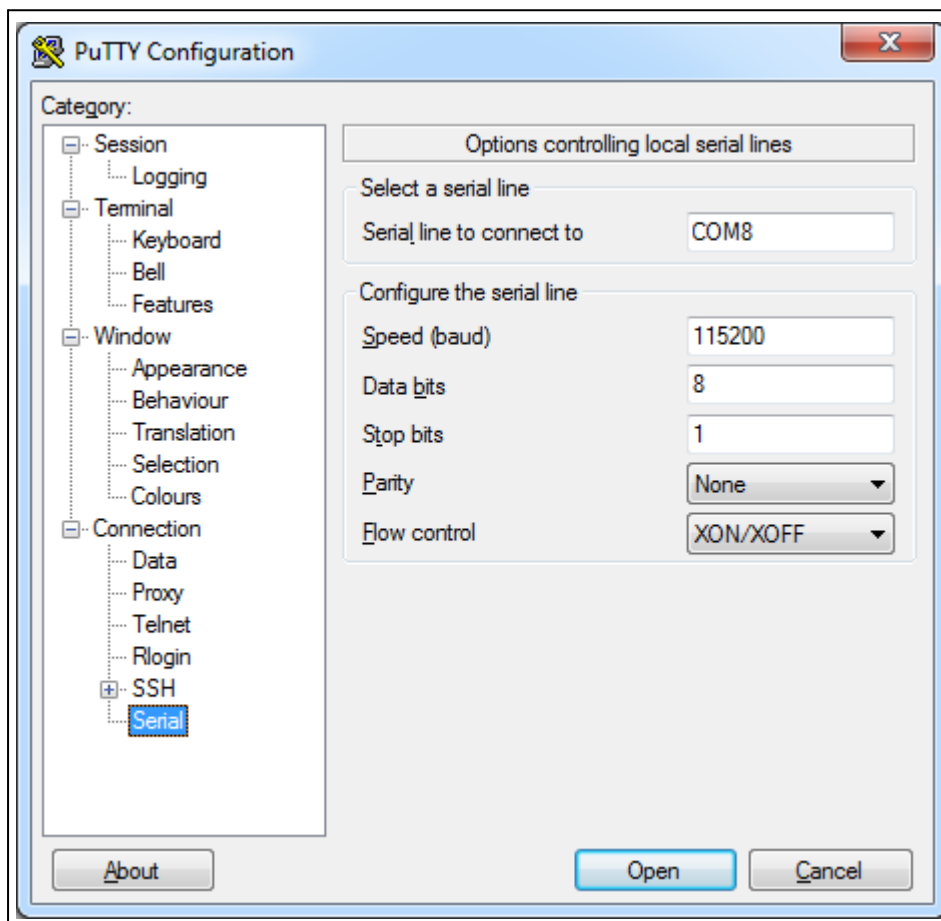
In DIP switch mode, the module cannot be addressed via the DELIB driver library.

3.1.5.2.4. Serial text mode

The serial text mode allows the control of digital and analogue inputs/outputs, as well as read and write access to registers, directly by "clear text" commands via any terminal program (e.g. PuTTY).

For this purpose, DIP switches A7 and A8 must be set to "off". Since the text mode is an extended DIP switch mode, the baud rate (DIP A1..A4) and the echo (DIP A6) must also be configured via DIP switch.

Example configuration PuTTY



Hint:

In text mode, the module cannot be addressed via the DELIB driver library or the serial protocol.

3.1.5.2.4.1. Overview of write commands

Definition

cmd1 + cmd2 + par1 + "=" + par2

Overview

Description	cmd1	cmd2	par1 [dec]	par2 [dec]
Set digital output	S (Set)	D (Digital)	Channel number	1 Bit Value (0=Aus, 1=An)
Set analogue output		A (Analog)		2 Byte Value (0 .. 65535)
Write register (1 byte)	W (Write)	B (Byte)	Register	1 Byte Value (0 .. 255)
Write register (2 byte)		W (Word)		2 Byte Value (0 .. 65535)

Return value

ERROR or OK

Examples

```
sd5=1
OK
// switches digital output 5 on (corresponds to DapiDOSet1(handle, 5, 1))

sd2=0
OK
// Switches digital output 2 off (corresponds to DapiDOSet1(handle, 2, 0))

sa3=4096
OK
// sets the data value 0x1000 at analogue output 3 (corresponds to
DapiDASet(handle, 3, 0x1000))

wb8194=64
OK
// writes the data value 0x40 into register 0x2002 (corresponds to
DapiWriteByte(handle, 0x2002, 0x40))

ww8192=12288
OK
// writes the data value 0x3000 into register 0x2000 (corresponds to
DapiWriteWord(handle, 0x2000, 0x3000))
```

3.1.5.2.4.2. Overview of read commands

Definition

cmd1 + cmd2 + par1

Overview

Description	cmd1	cmd2	par1 [dec]	Return-Value
Digital input read	G (Get)	D (Digital)	Channel number	1 Bit Value (0=Aus, 1=An)
Analogue input read		A (Analog)		2 Byte Value (0 .. 65535)
Read register (1 byte)	R (Read)	B (Byte)	Register	1 Byte Value (0 .. 255)
Read register (2 byte)		W (Word)		2 Byte Value (0 .. 65535)

Return value

ERROR or Value

Examples

```
gd3
1
// reads digital input 3 (corresponds to DapiDIGet1(handle, 3))

ga2
4711
// Reads analogue output 2 (corresponds to DapiADGet(handle, 2))

rb8194
64
// reads register 0x2002 (corresponds to DapiReadByte(handle, 0x2002))

rw8192
12288
// reads register 0x2000 (corresponds to DapiReadWord(handle, 0x2000))
```

3.1.5.2.5. Activating echo

Received serial characters are returned to the display (ON = yes, OFF = no).

3.1.5.2.6. Setting up Baud rate

The table lists which Baud rates are adjustable by using the 4 DIP-switches (A1 to A4).

Baud rate	DIP-switch A4	DIP-switch A3	DIP-switch A2	DIP-switch A1
1,25 Mbit	On	On	On	On
625 Kbit	On	On	On	Off
250 Kbit	On	On	Off	On
125 Kbit	On	On	Off	Off
115200 Bit	On	Off	On	On
57600 Bit	On	Off	On	Off
50000 Bit	On	Off	Off	On
38400 Bit	On	Off	Off	Off
19200 Bit	Off	On	On	On
9600 Bit	Off	On	On	Off
4800 Bit	Off	On	Off	On
2400 Bit	Off	On	Off	Off
1200 Bit	Off	Off	On	On
600 Bit	Off	Off	On	Off
300 Bit	Off	Off	Off	On

3.1.5.2.7. Setting up module address (RS-485 only)

Since several modules can be connected to a BUS system in RS-485 operation, it is necessary to assign each module its own address. This can be set between 0 and 255 with the DIP switches B1 to B8. If the module number is 0, the module number is ignored.

Baud rate	Bit	Value ON	Value OFF
DIP Switch B8	Bit 7	128	0
DIP Switch B7	Bit 6	64	0
DIP Switch B6	Bit 5	32	0
DIP Switch B5	Bit 4	16	0
DIP Switch B4	Bit 3	8	0
DIP Switch B3	Bit 2	4	0
DIP Switch B2	Bit 1	2	0
DIP Switch B1	Bit 0	1	0

Example:

Baud rate	Address 0	Address 25	Address 237
DIP Switch B8	Off	Off	On
DIP Switch B7	Off	Off	On
DIP Switch B6	Off	Off	On
DIP Switch B5	Off	On	Off
DIP Switch B4	Off	On	On
DIP Switch B3	Off	Off	On
DIP Switch B2	Off	Off	Off
DIP Switch B1	Off	On	On

Hint:

RS-485 mode is only available in revision 1.0.

3.1.5.2.8. Configuration via the DELIB Configuration Utility

The configuration is described in the chapter **Configuration RS-232/RS-485 Interface**.

3.1.6. Specifications interface connection

3.1.6.1. USB

Technical data:

Standard:	USB 1.1 / USB 2.0
Connection establishment:	USB cable type A to 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

3.1.6.2. Ethernet

Technical data:

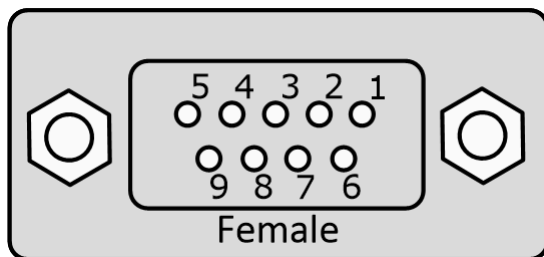
Standard:	Ethernet 100/10Mbit
Connection setup:	Ethernet / LAN cabel / WiFi
Access time PC to module*:	Wire: 1,56 ms / Wireless: 4,06 ms

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

3.1.6.3. CAN

Pin assignment D-SUB socket:

CAN-L	Pin 2
CAN-H	Pin 7
GND	Pin 5



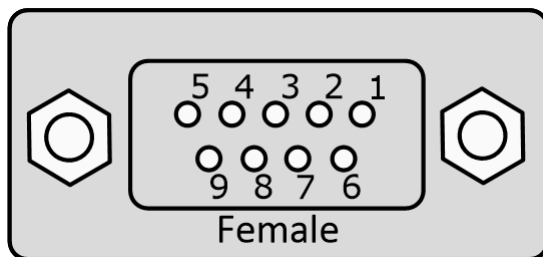
Technical data:

Standard:	ISO 11898
Speed:	1 Mbit/s, 500 Kbit/s, 250 Kbit/s, 125 Kbit/s, 100 Kbit/s, 50 Kbit/s, 20 Kbit/s, 10 Kbit/s
Connection establishment:	Open CAN protocol
Galvanic isolation:	up to 1kV rms
Special features:	Automatic processing of CAN packets (Auto RX/TX mode)

3.1.6.4. RS-232

Pin assignment D-SUB socket:

TX	Pin 2
RX	Pin 3
GND	Pin 5



Technical data:

Standard:	RS-232
Speed:	up to 115200 Baud

3.1.6.5. Retrofit interfaces

The unit can be retrofitted with a CAN or RS-232 interface adapter. Please observe the chapter → **Safety instructions**

For retrofitting, please proceed as follows:

- Disconnect the module from the power supply.
- Remove one of the lateral covers of the housing by loosening the four cross screws.
- Now carefully pull the printed circuit board out of the housing and place it on a solid and dry surface.
- Then plug the interface adapter onto the 10-pin header so that the two mounting blocks are located above the mounting holes.
- Now screw the adapter onto the bottom side of the PCB and then tighten the screws of the two mounting blocks again.
- The installation is done in reverse order. Make sure to push the PCB into the uppermost guide rail.

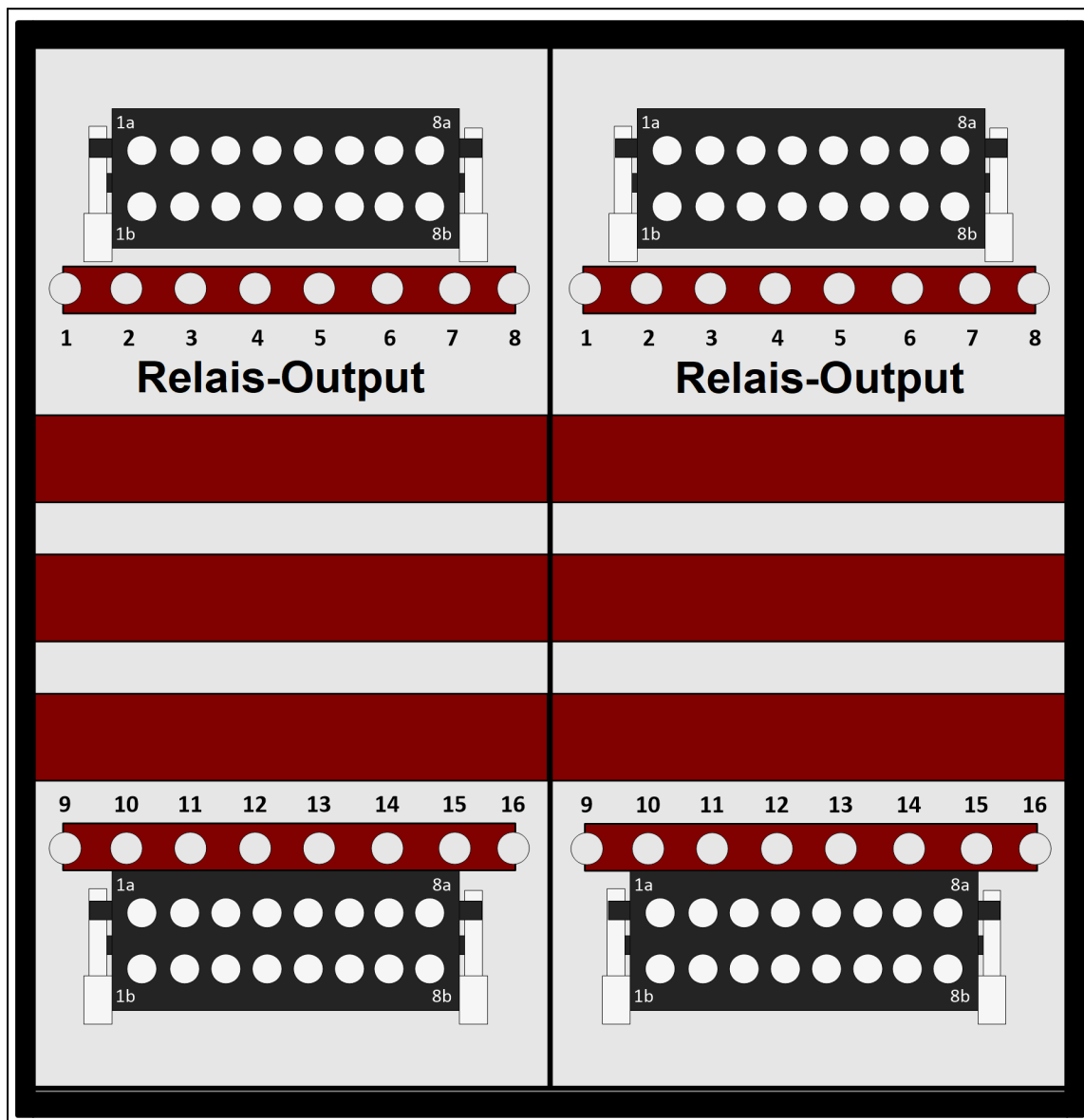
After the installation the module can be put into operation directly. The interface adapters are initialized automatically.

3.2. Submodule

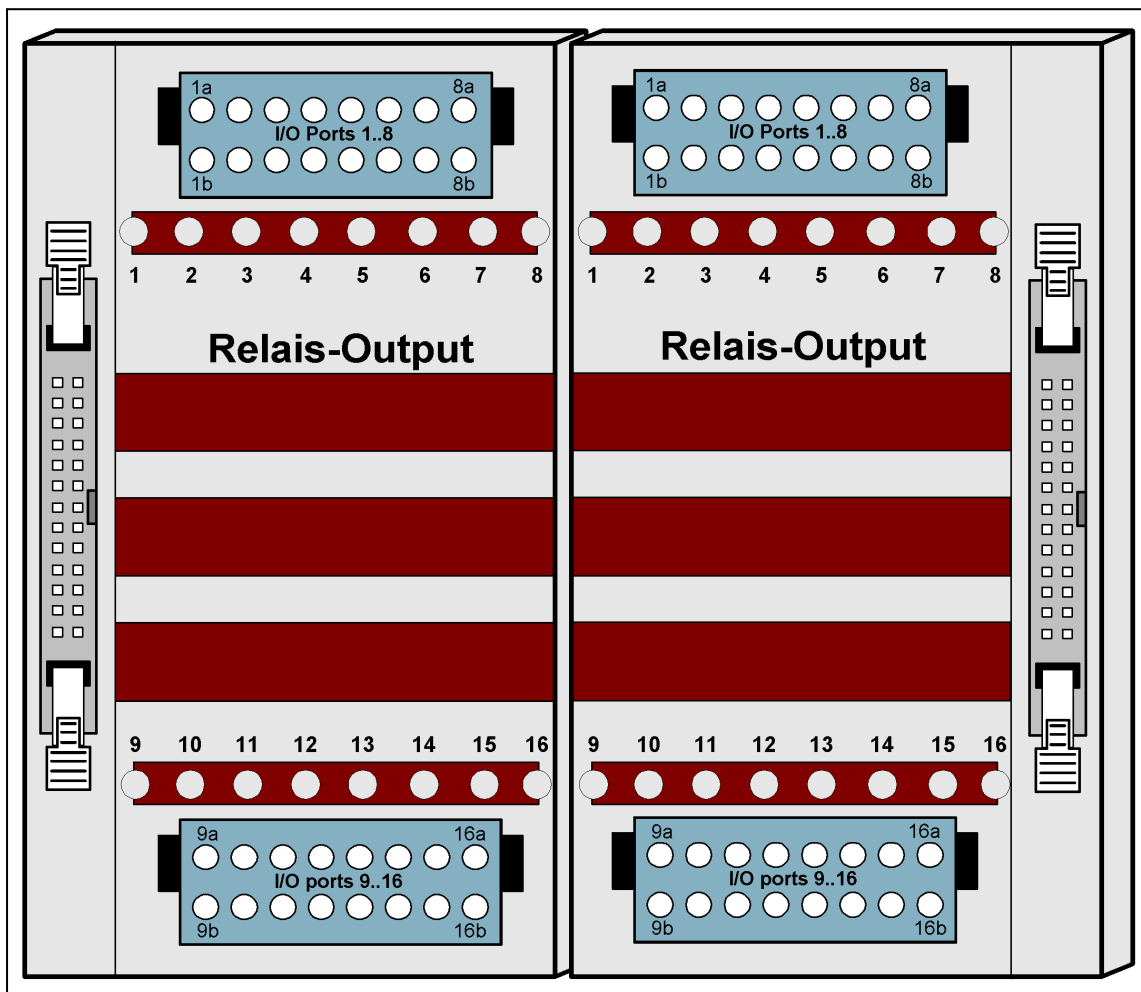
3.2.1. RO-REL16

3.2.1.1. Overview image

The figure shows two modules side by side with corresponding numbering of the connection terminals.



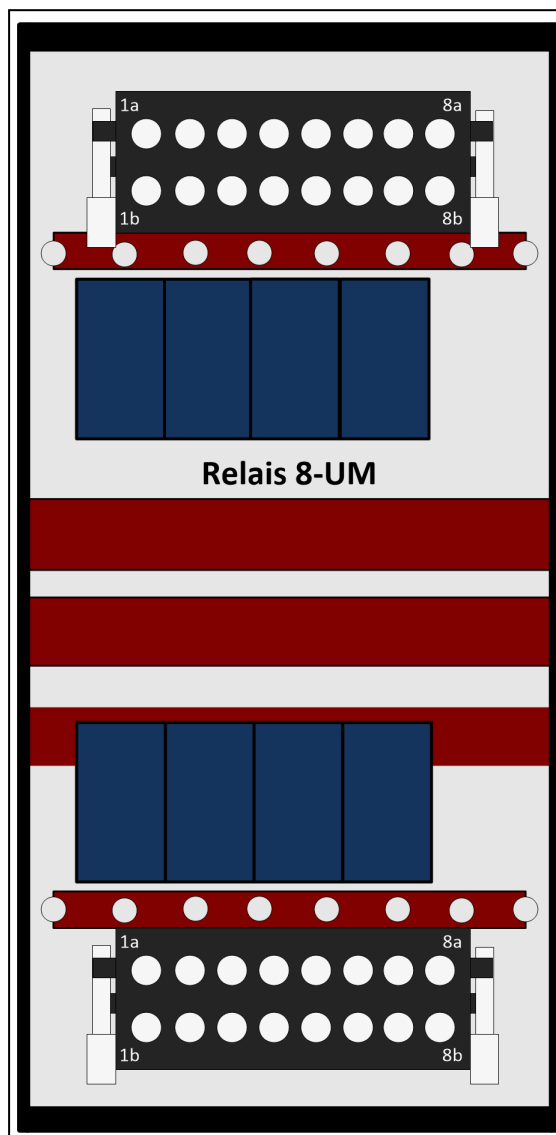
The figure below shows an I/O module of the flexible connector system with 32 outputs and corresponding numbering of the connection terminals. At the outer ends of the module there is a 26pin tub header with locking levers each. Thus several module units can be connected with each other by means of a ribbon cable.



3.2.2. RO-REL8_UM

3.2.2.1. Overview image

The figure shows a RO-REL8_UM module with corresponding numbering of the connection terminals.



3.2.2.2. Bistable relay outputs

Due to the bistable relays installed in this module, the switching positions of the relays are maintained even if the power supply is lost or the module is restarted.

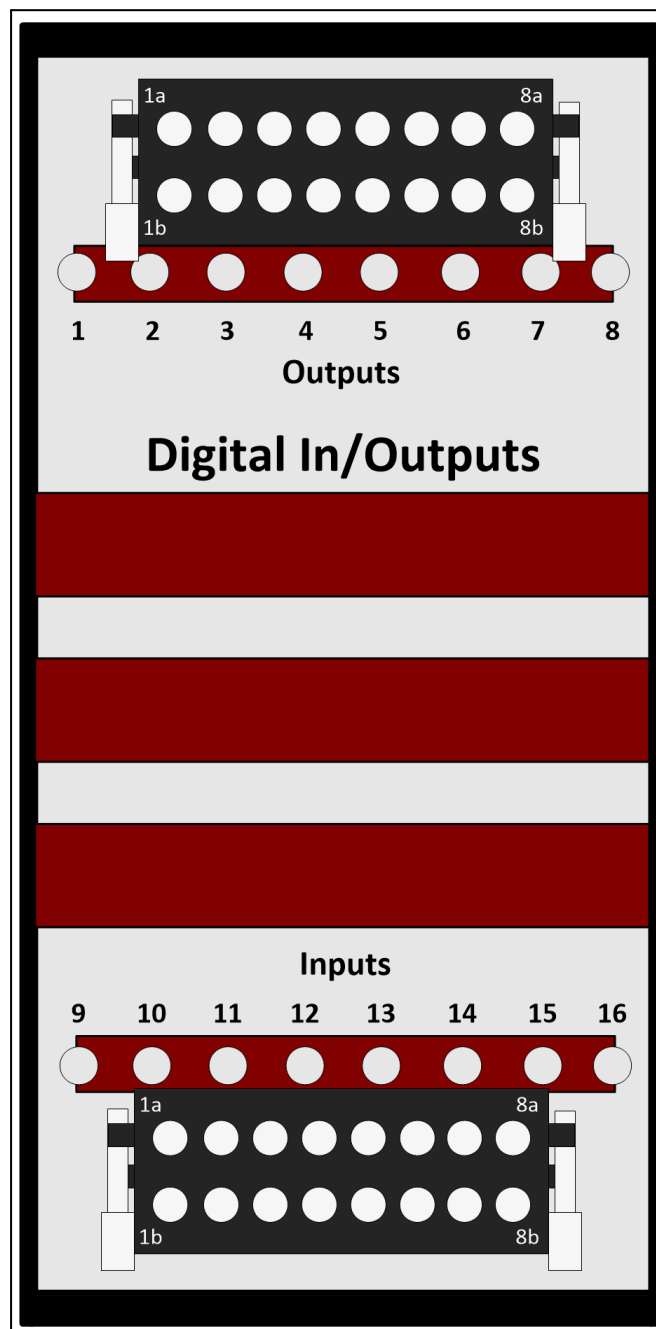
Each of these bistable relays has a changeover contact with two LEDs as status indication.

The bistable relays can be used in many ways due to their special properties. Examples of application can be found, for example, in the automotive industry, but they are also very popular in network technology. In addition, the safety aspect of these relays is enormous, by maintaining the switching positions, faulty switching is prevented.

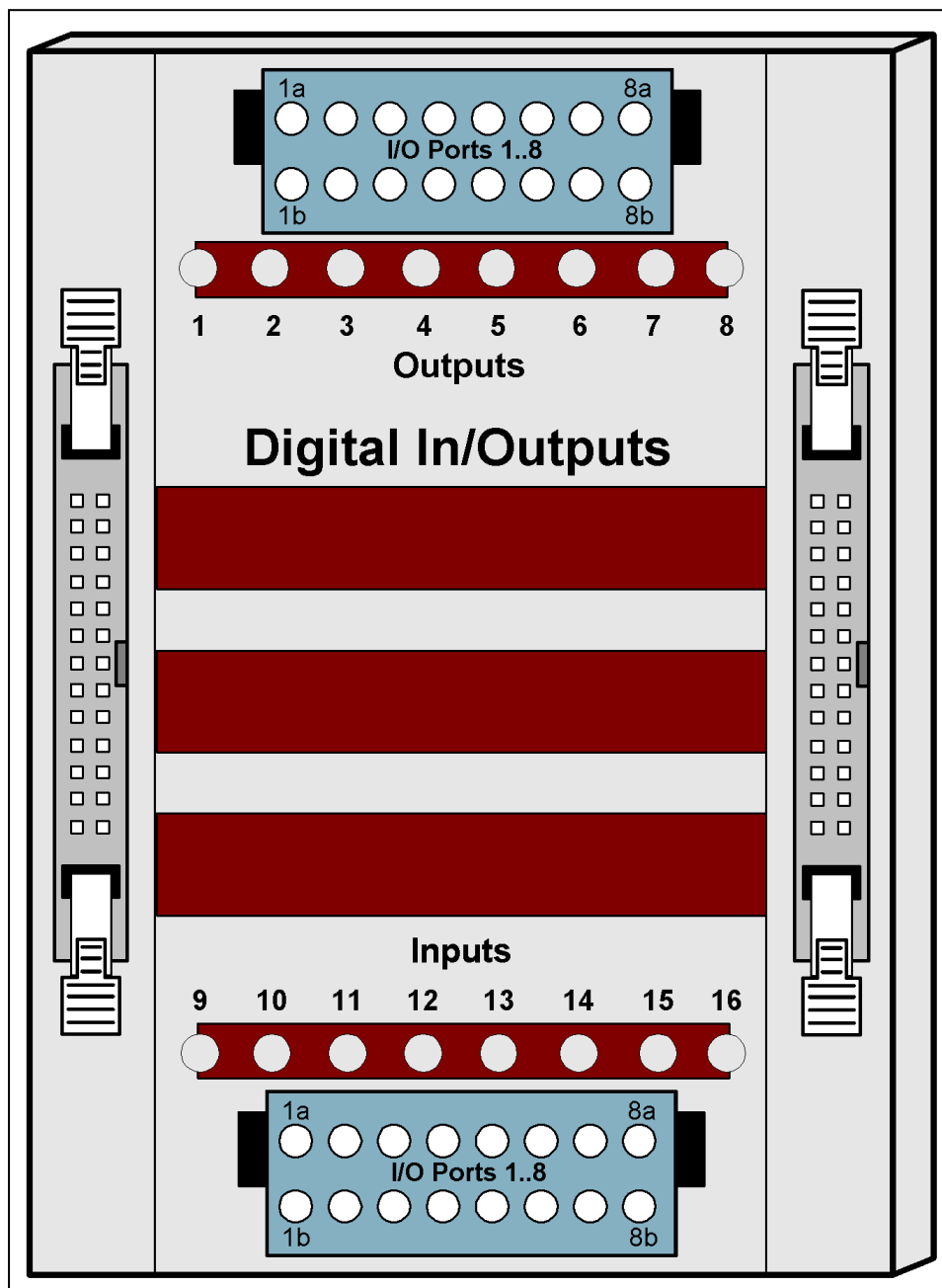
3.2.3. RO-OPT8-REL8

3.2.3.1. Overview image

The figure shows a RO-O8-R8 module with corresponding numbering of the connection terminals.



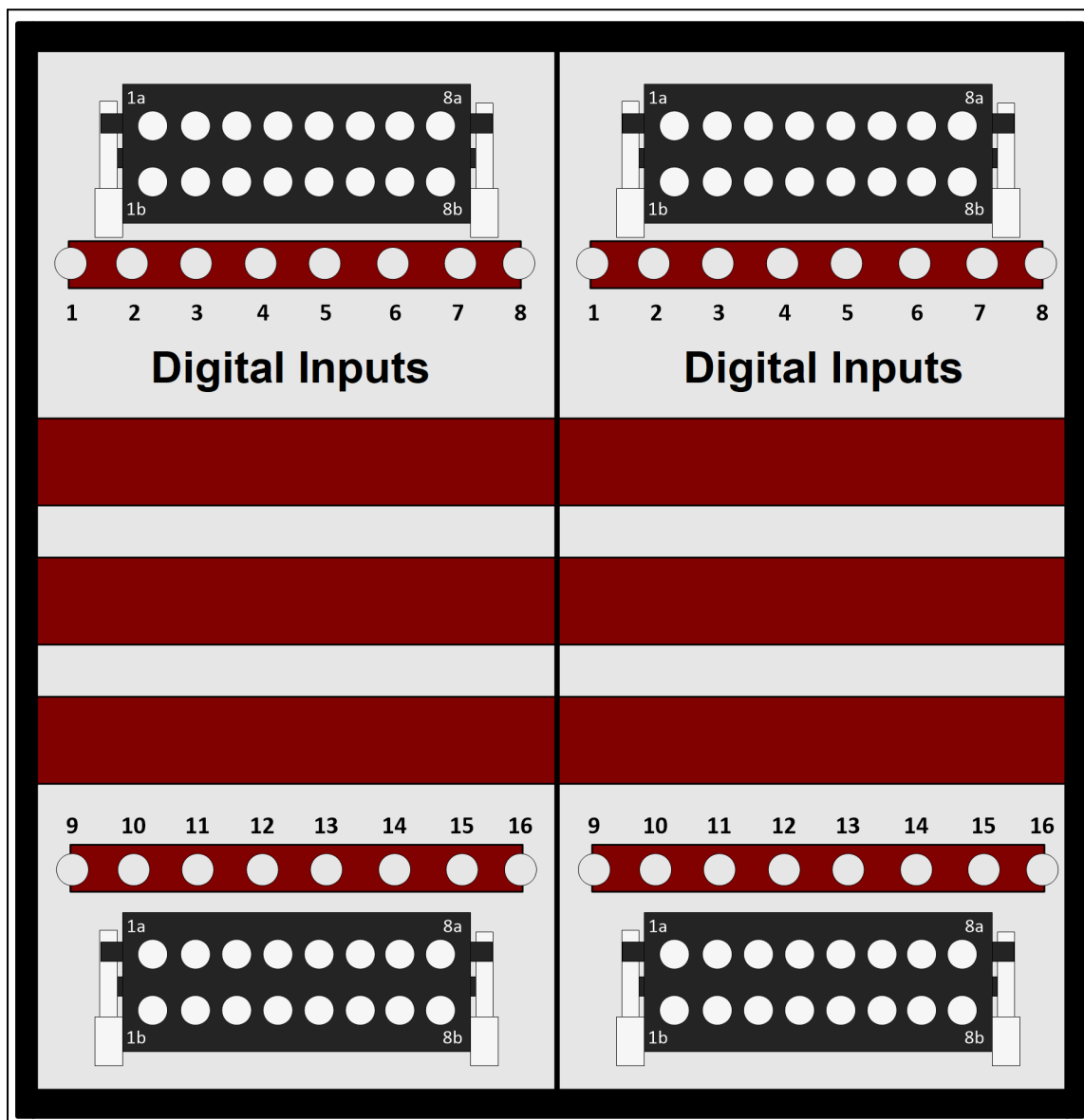
Lower figure shows a RO-08-R8 module of the flexible connector system with corresponding numbering of the connection terminals. At the outer ends of the module there is a 26-pin tub header with locking levers. Thus several module units can be connected with each other by means of a ribbon cable.



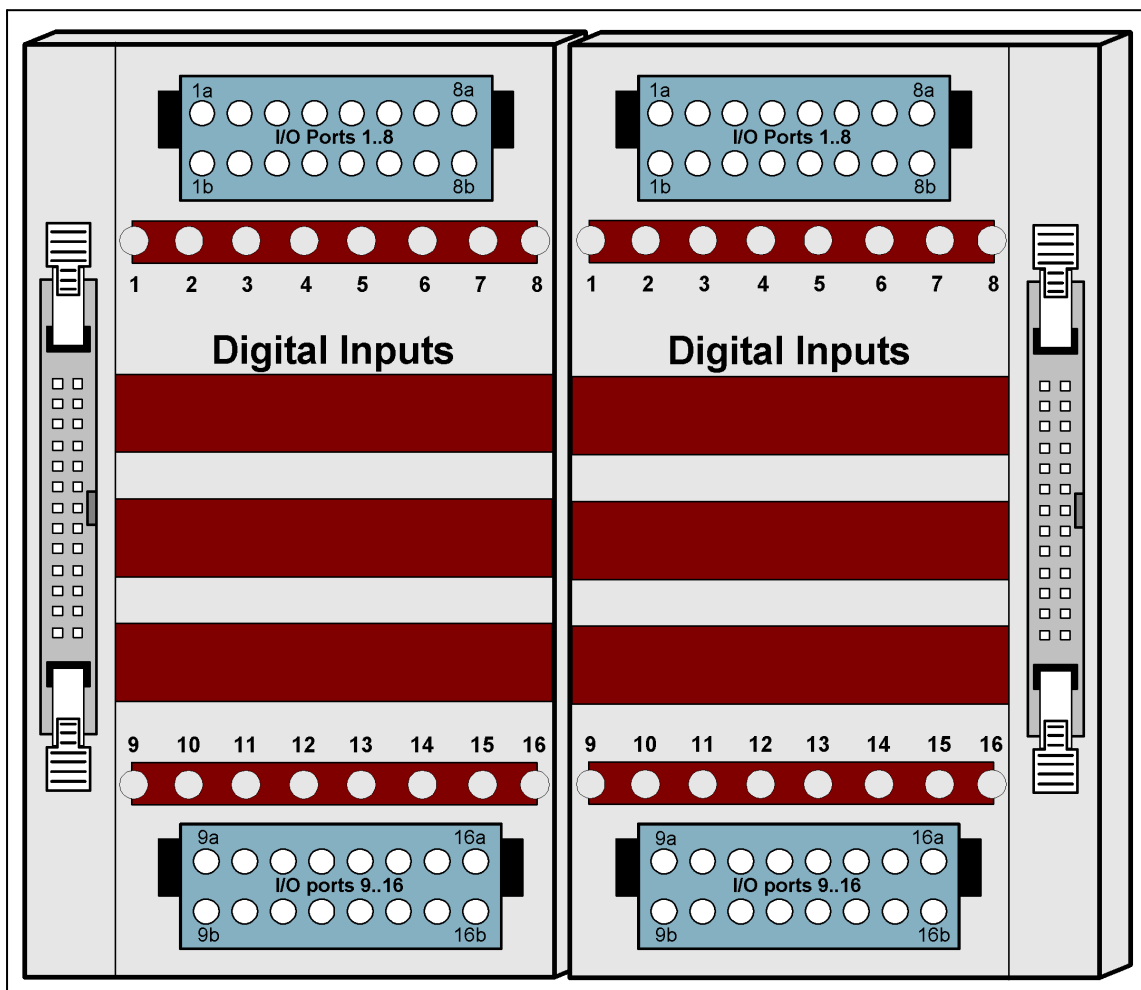
3.2.4. RO-OPT16

3.2.4.1. Overview image

The figure shows two modules side by side with corresponding numbering of the connection terminals.



The figure below shows an I/O module of the flexible connector system with 32 inputs and corresponding numbering of the connection terminals. At the outer ends of the module there is a 26-pin tub header with locking levers. Thus several module units can be connected with each other by means of a ribbon cable.



3.2.4.2. Changing the input voltage

Input voltage range	5V – 15V	15V – 30V
Resistance value	1K	2K2

In order to adapt the input voltage range, 2 series resistors must be exchanged per connector. I.e. you can feed in 4 input channels each (1-4, 5-8, 9-12 and 13-16) with different voltage. The individual steps for this are explained below.

Notice!

Before opening the device, please note the following:

Pull out the mains plug!

Do not touch electronic components, as they may be destroyed by electrostatic discharge! If necessary, touch grounded housings or radiators before working.

Remove the side element of the module by loosening the three cross screws.

Pull the printed circuit board together with the front panel sideways out of the housing.

Lift off front panel

The input modules all have single-row socket strips in which the input resistors for the voltage range are inserted

Now carefully pull the resistors out of the socket strips

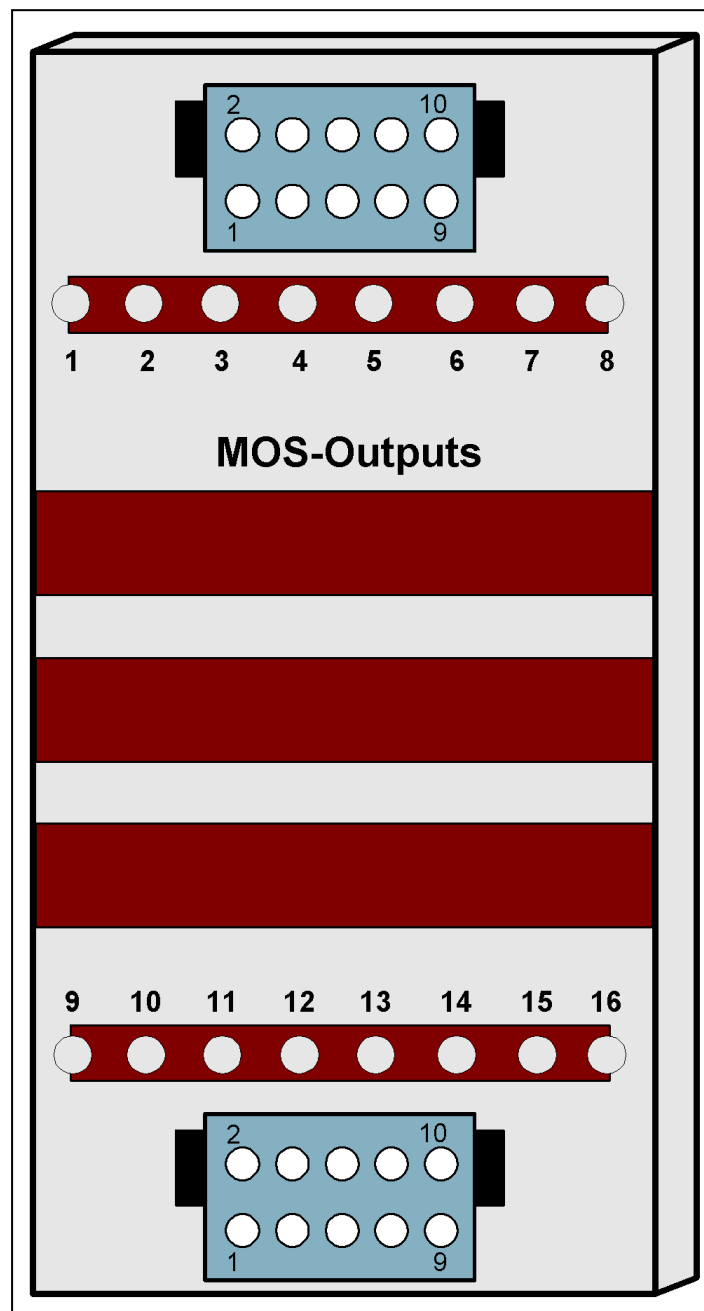
Insert new resistors

Installation takes place in reverse order

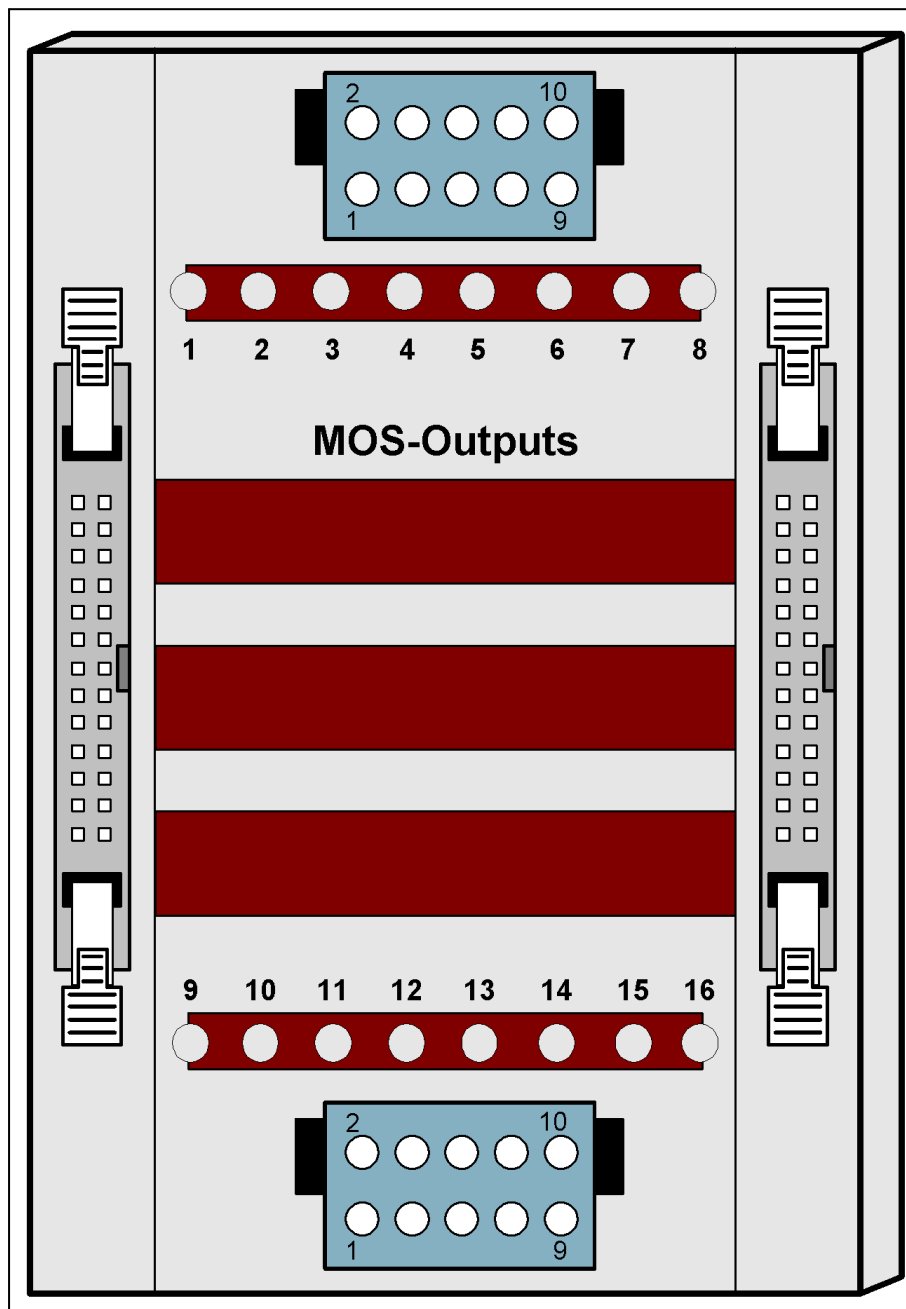
3.2.5. RO-MOS16

3.2.5.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals.



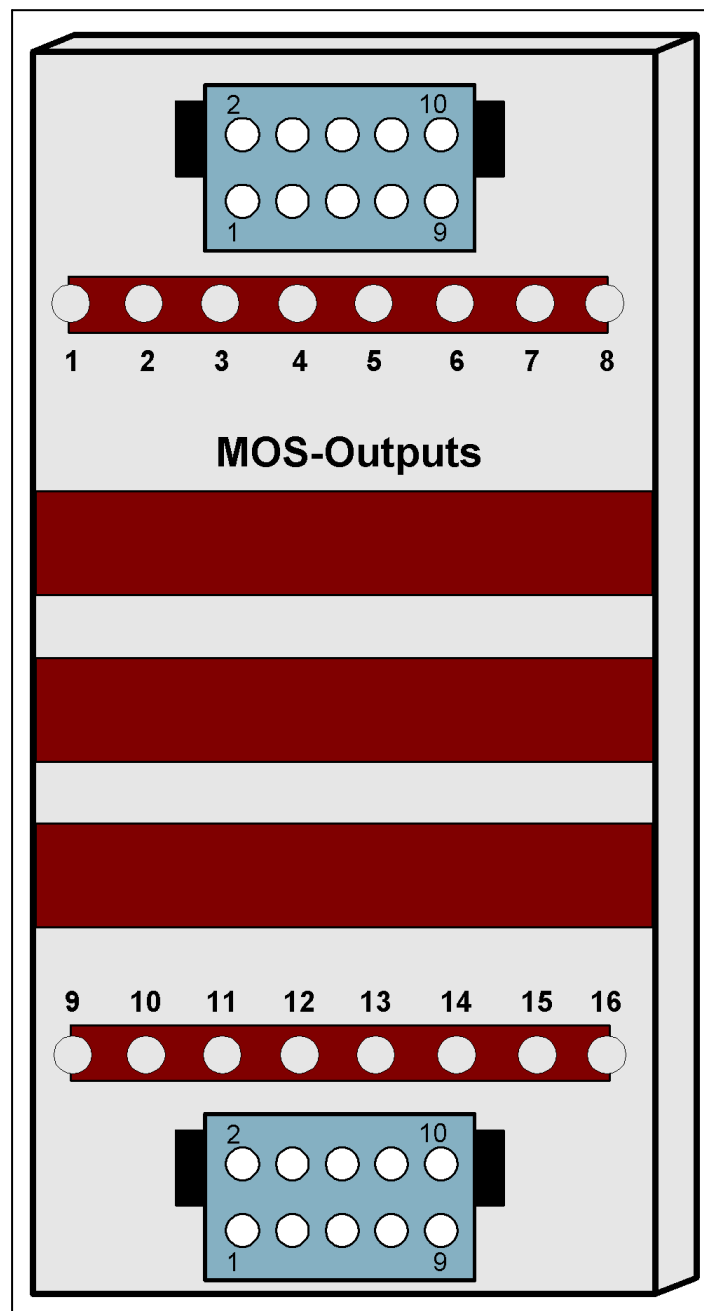
The figure below shows an I/O module of the flexible connector system with 16 outputs and corresponding numbering of the connection terminals. At the outer ends of the module there is a 26pin tub header with locking levers each. Thus several module units can be connected with each other by means of a ribbon cable.



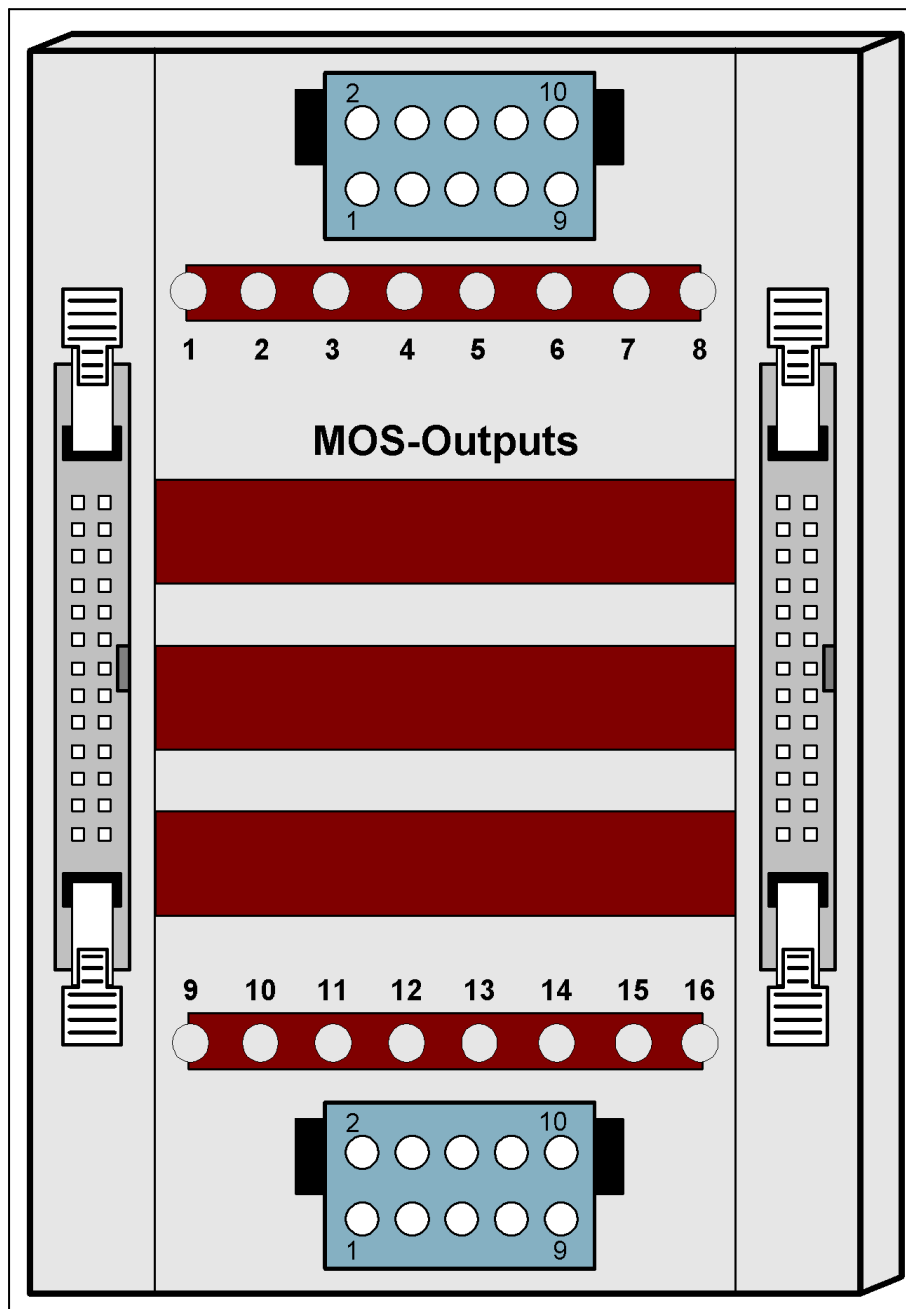
3.2.6. RO-PWM

3.2.6.1. Overview image

The figure shows a module with PWM outputs side by side with corresponding numbering of the connection terminals.



The figure below shows an I/O module of the flexible connector system with 16 MOSFET outputs with pulse width modulation and corresponding numbering of the connection terminals. At the outer ends of the module there is a 26pin tub header with locking levers each. Thus several module units can be connected with each other by means of a ribbon cable.



3.2.7. Digital outputs

3.2.7.1. 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

Technical data bistable relay

Number of channels per block:	8
Type:	Schließer (CO)
Max. Switching voltage:	30V DC
Max. Switching current:	2A DC
Max. Switching capacity:	60W DC
Insulation resistance:	1000 MOhm @500V DC
Switching and fall time:	max. 4ms

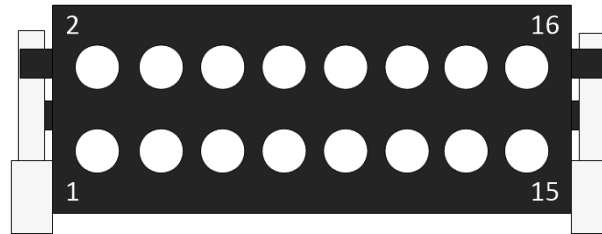
Technical data MOSFET 2A

Number of channels per block:	8
Typ:	P-Ch.
Max. Switching voltage:	48V DC
Min. Switching voltage::	2.8V
Max. Switching current:	2A DC
Max. Switching power:	60W

Additional functions

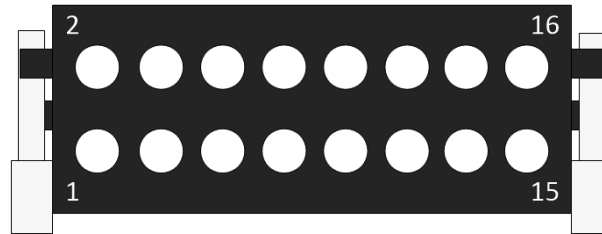
Timeout protection

3.2.7.2. Pin assignment of a RO-REL16



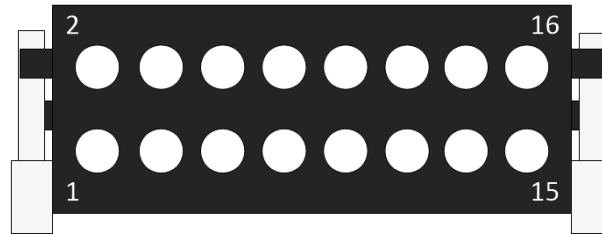
Digital Output Channel	Pin	
Output Channel 1	1	2
Output Channel 2	3	4
Output Channel 3	5	6
Output Channel 4	7	8
Output Channel 5	9	10
Output Channel 6	11	12
Output Channel 7	13	14
Output Channel 8	15	16

3.2.7.3. Pin assignment of a RO-REL8_UM



Digital Output Channel	Pin	
Output Channel 1	1	2
Output Channel 2	3	4
Output Channel 3	5	6
Output Channel 4	7	8
Output Channel 5	9	10
Output Channel 6	11	12
Output Channel 7	13	14
Output Channel 8	15	16

3.2.7.4. Pin assignment of a RO-MOS16_P and RO-PWM16_P



Digital Output Channel	Pin	Digital Output Channel	Pin
Output Channel 1	1	VCC+	2
Output Channel 2	3	VCC+	4
Output Channel 3	5	-	6
Output Channel 4	7	GND	8
Output Channel 5	9	GND	10
Output Channel 6	11	-	12
Output Channel 7	13	VCC+	14
Output Channel 8	15	VCC+	16

For the optocoupler outputs, the correct polarity must be observed when connecting (**see picture below**)

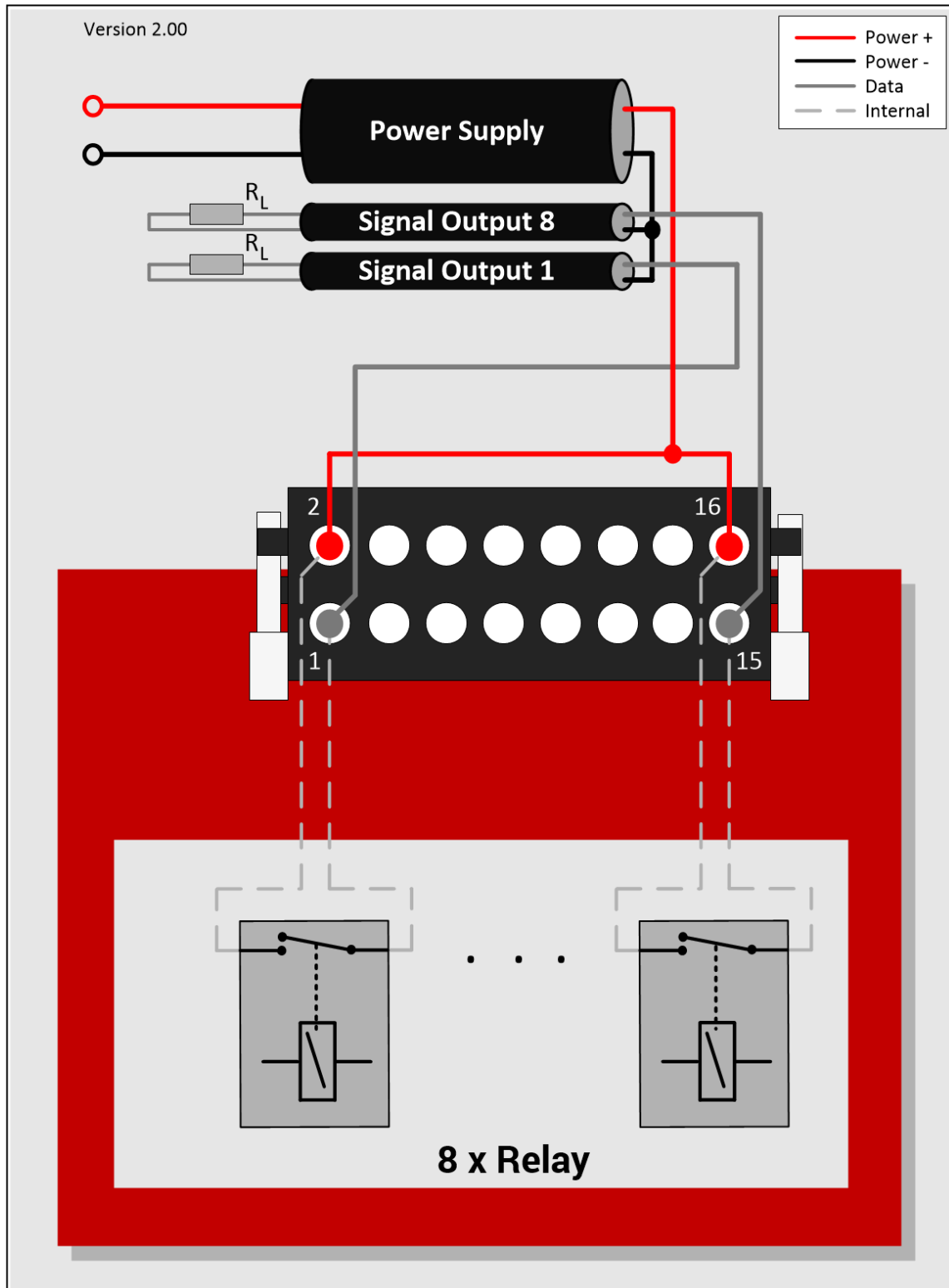
Attention:

Each connector has 4 VCC+ inputs, each of which is designed for a maximum load of 10A.

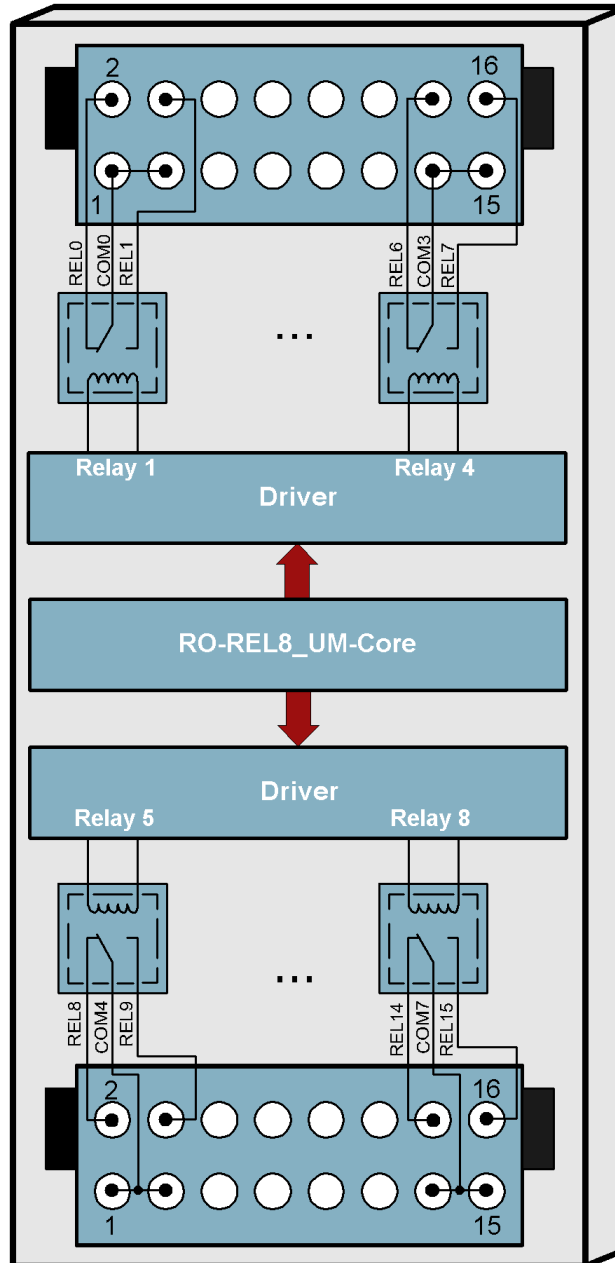
For example, if the total load of the connector is 30A, 3 VCC+ inputs must be used.

Total load	Required VCC+ inputs
<= 10A	1
<= 20A	2
<= 30A	3
<= 40A	4

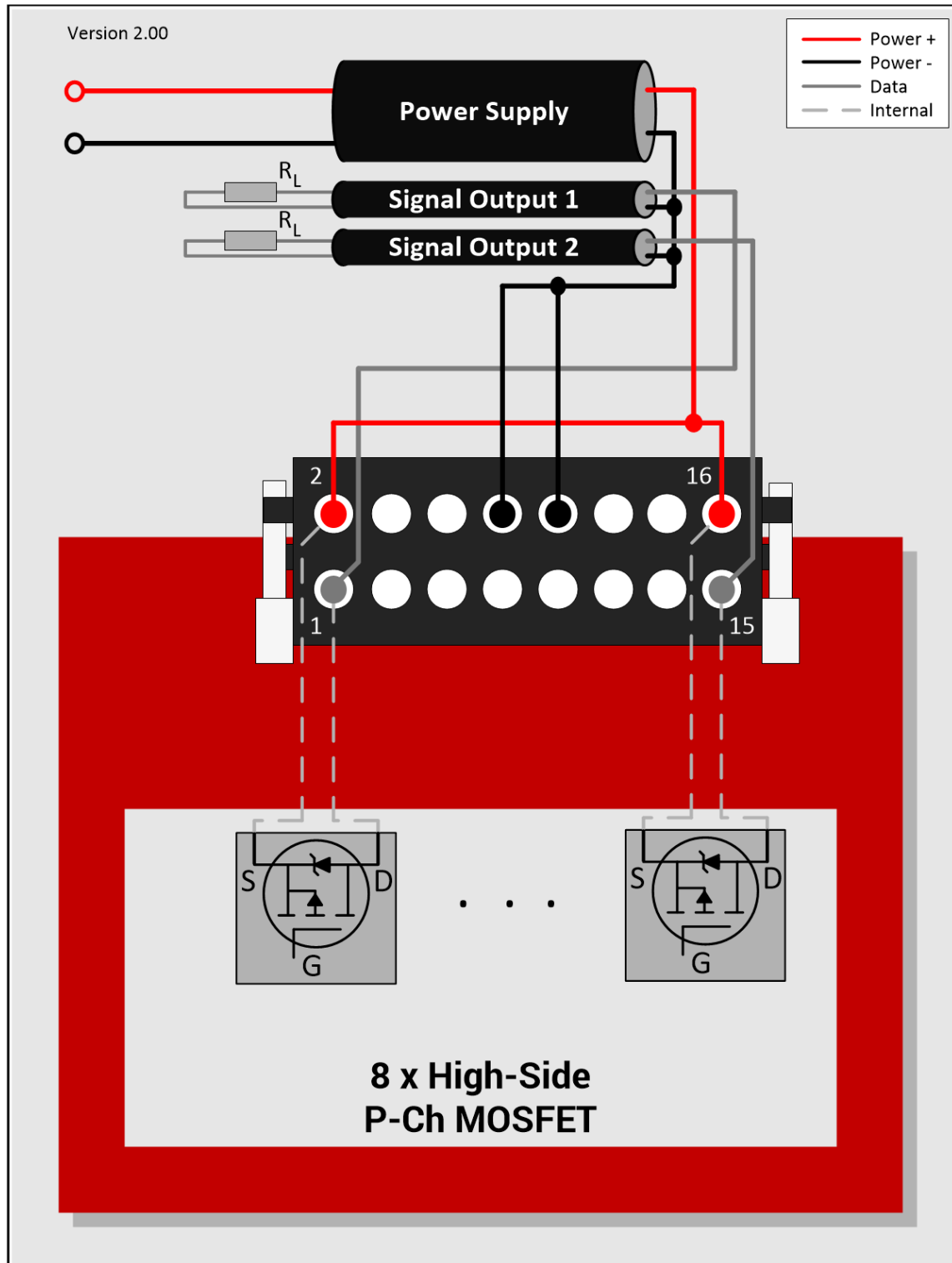
3.2.7.5. Connection example of a relay module



3.2.7.6. Connection example of a changeover relay module



3.2.7.7. Connection example of a MOSFET module



J:\Produkte\90-Anschlussbeispiele Produkte\MOS-PWM\wago-16_NET_COS\vsd\

3.2.7.8. 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.**

3.2.8. Digital inputs

3.2.8.1. Technische Daten

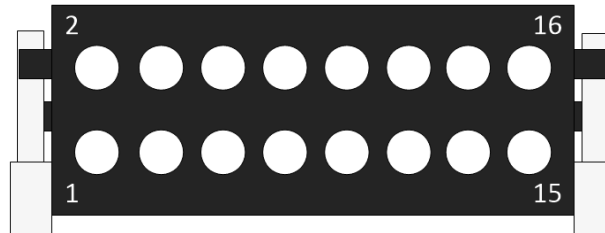
Technical data optocoupler

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

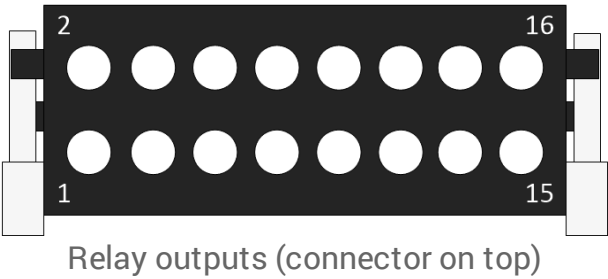
- 16 bit counter per channel.
- Maximum possible counts: 65535 / channel. Reset to zero after memory overflow.
- Internal counting logic up to 10kHz with latch function.
- Programmable filter for input channels (flip-flop and counter): Minimum low or high pulse duration: 5ms...255ms.
- Detects change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

3.2.8.2. Pinbelegung R0-016

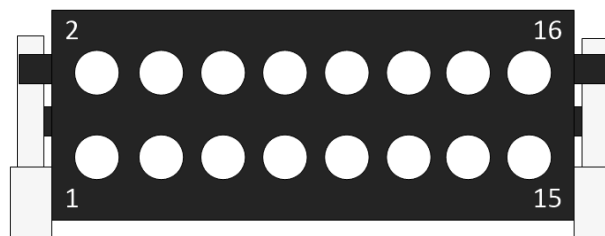


Digital Input Channel	Pin	
Input Channel 1	1	2
Input Channel 2	3	4
Input Channel 3	5	6
Input Channel 4	7	8
Input Channel 5	9	10
Input Channel 6	11	12
Input Channel 7	13	14
Input Channel 8	15	16

3.2.8.3. Pin assignment R0-08-R8



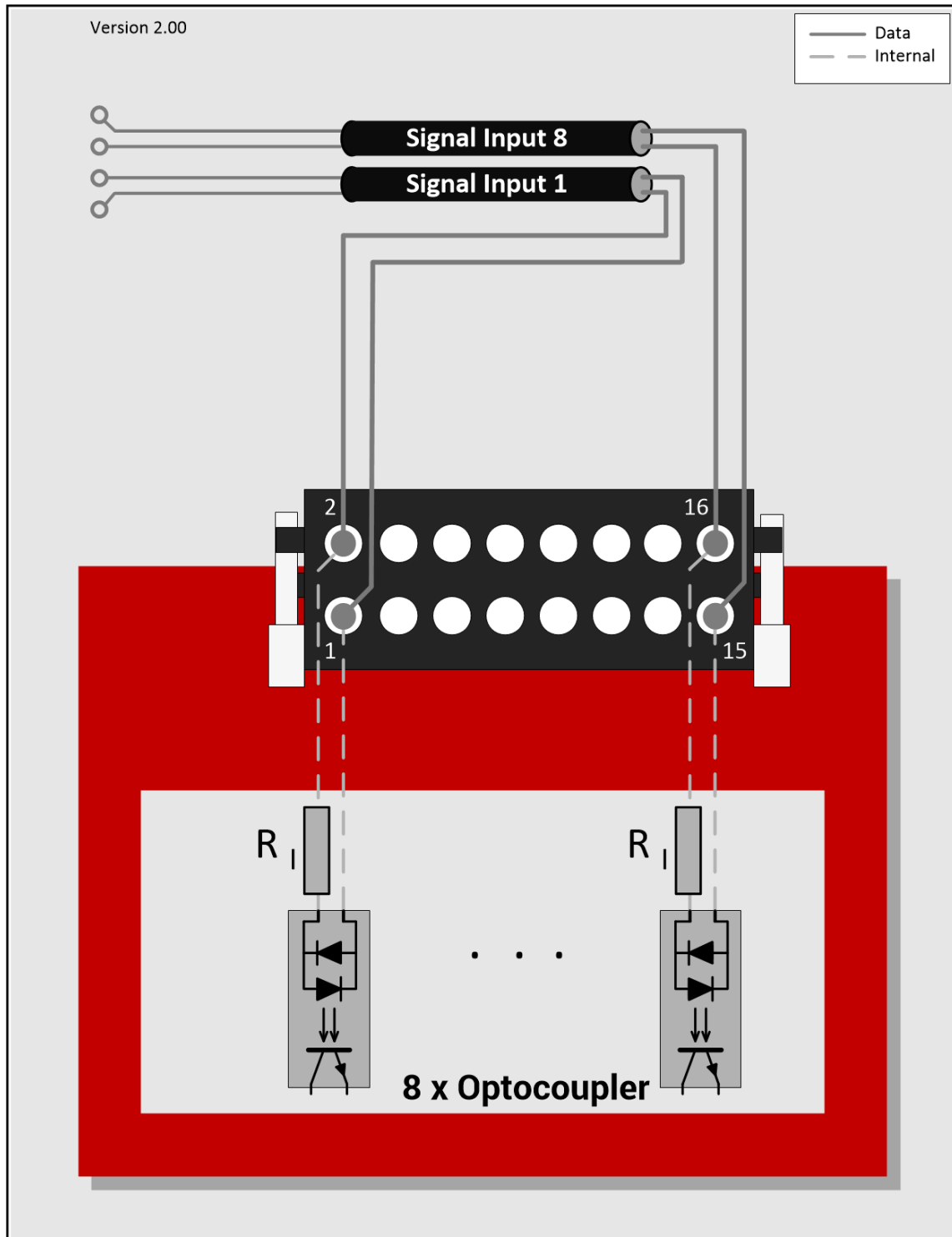
Digital Output Channel	Pin	
Output Channel 1	1	2
Output Channel 2	3	4
Output Channel 3	5	6
Output Channel 4	7	8
Output Channel 5	9	10
Output Channel 6	11	12
Output Channel 7	13	14
Output Channel 8	15	16



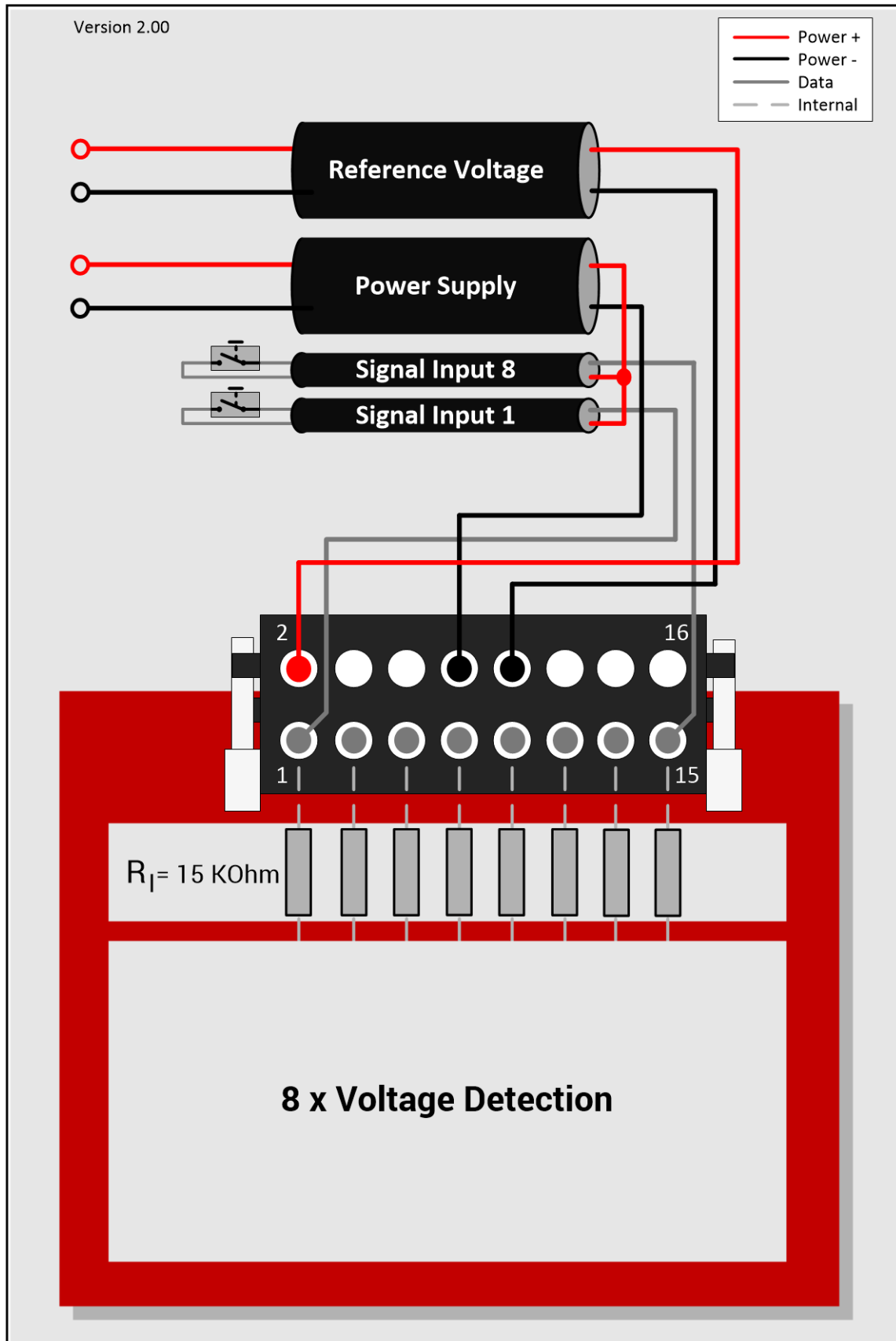
Optocoupler inputs (connector at the bottom)

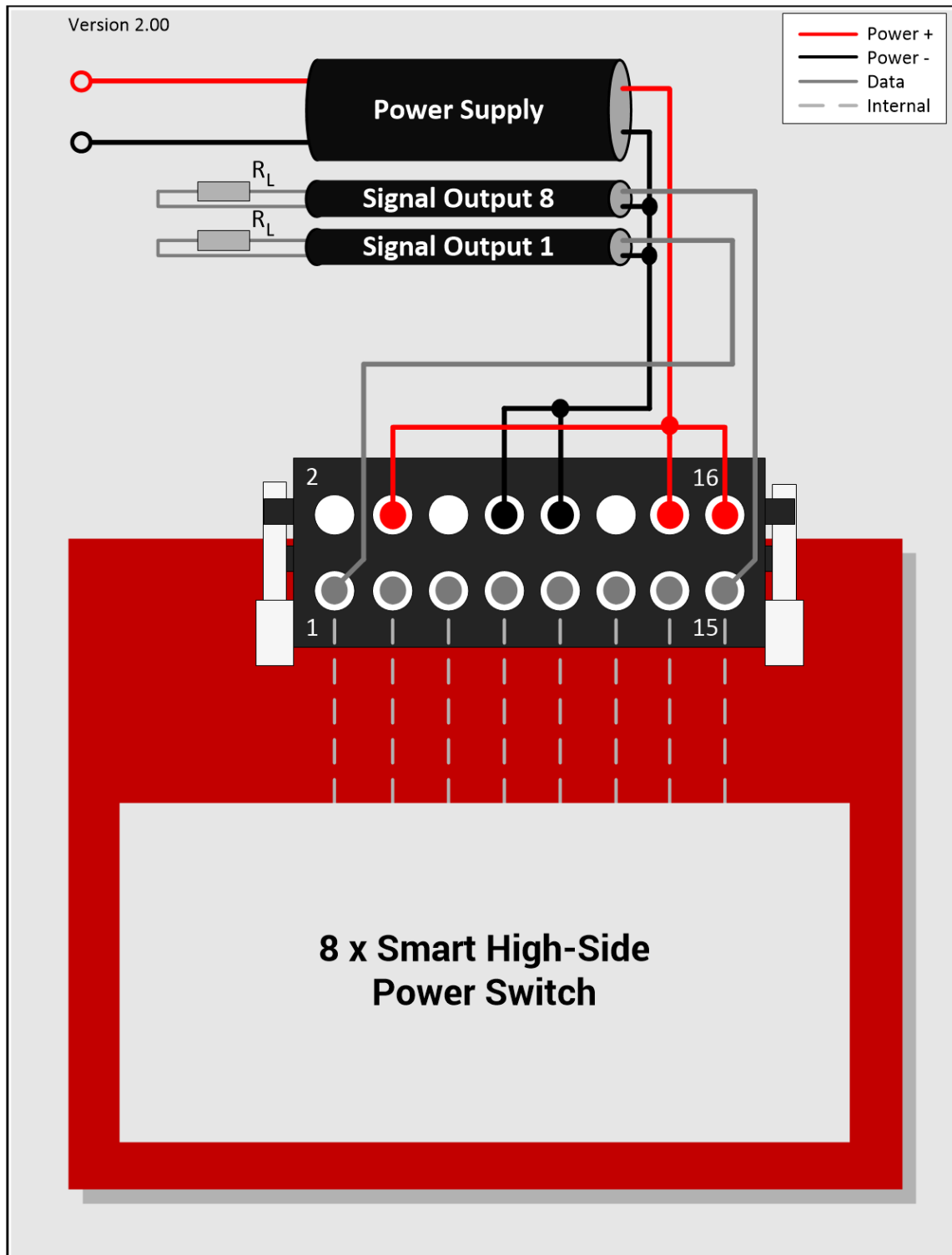
Digital Input Channel	Pin	
Input Channel 1	1	2
Input Channel 2	3	4
Input Channel 3	5	6
Input Channel 4	7	8
Input Channel 5	9	10
Input Channel 6	11	12
Input Channel 7	13	14
Input Channel 8	15	16

3.2.8.4. Connection example RO-016



3.2.8.5. Connection example RO-08-R8



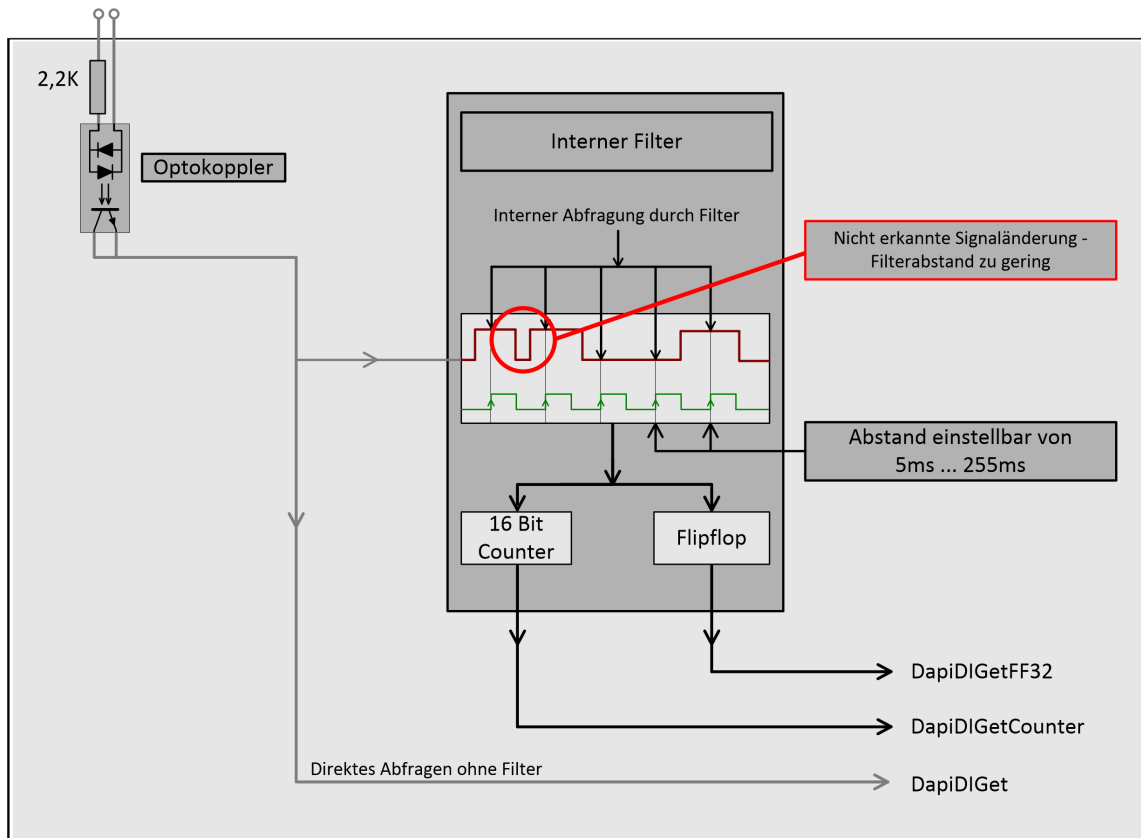


3.2.8.6. Input filter

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 → **Read digital inputs**

Schematic view of the filter:



3.2.8.7. 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**

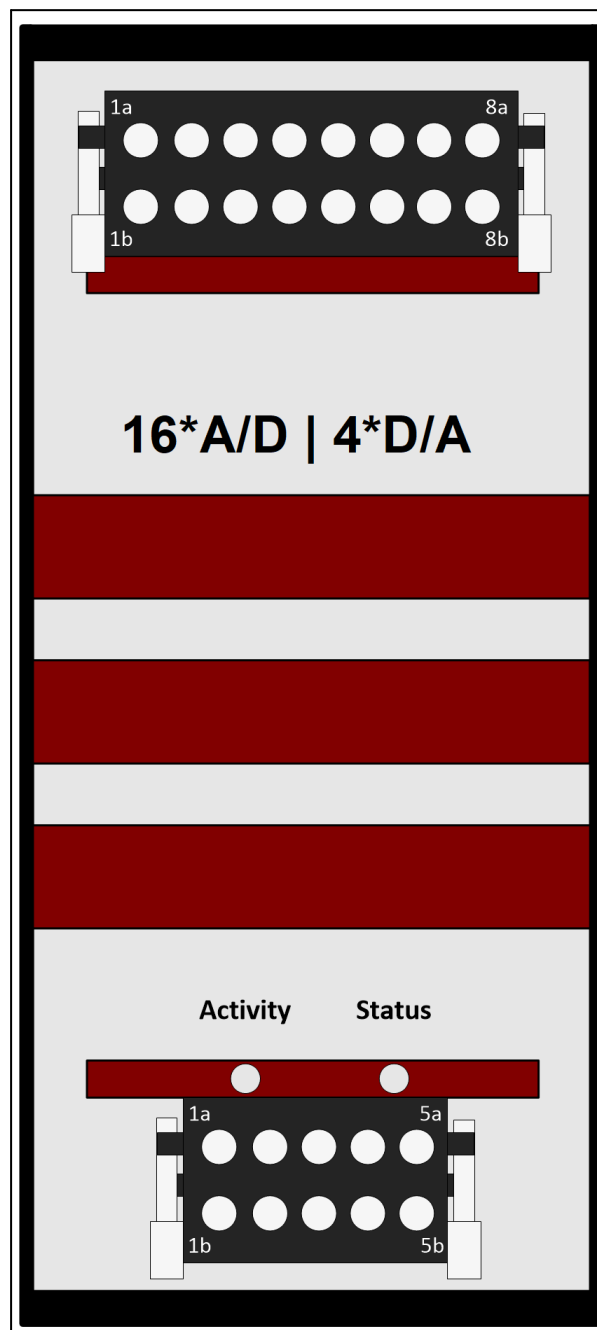
3.2.9. Analog input/output modules

3.2.9.1. RO-AD16-DA4

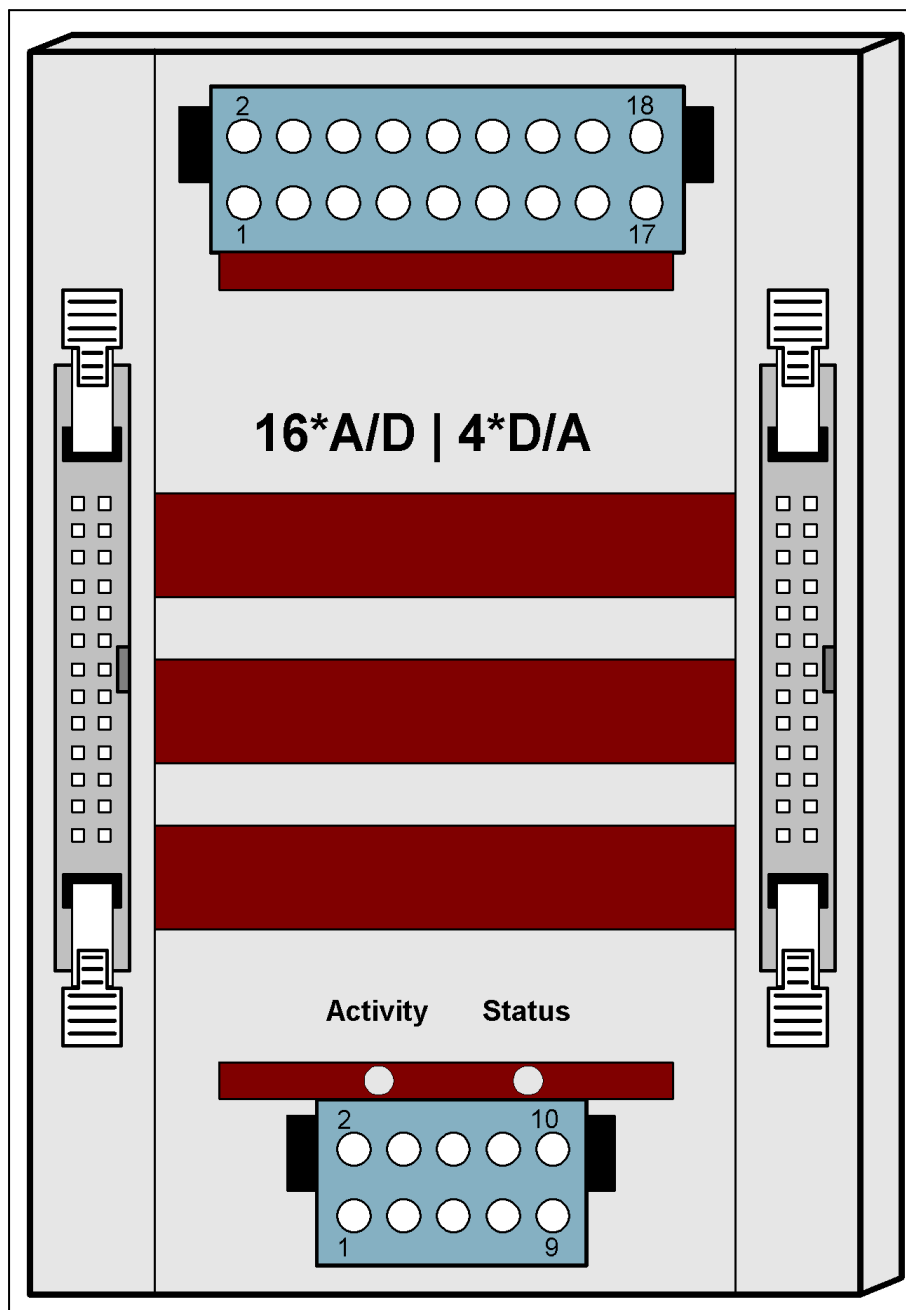
With 16 A/D inputs this module is a good basis for converting voltages into digital values. Additionally thanks to the 4 D/A outputs it is possible to convert digital values into voltages.

3.2.9.1.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals.



The figure shows an A/D module of the flexible connector system with corresponding numbering of the connection terminals.



3.2.9.1.2. Technical data

A/D inputs

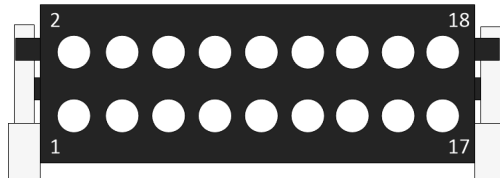
Mode U:	(voltage)
Unipolar:	0-5V, 0-10V
Bipolar:	+5V, +10V
Mode I:	(current)
Area:	0-20mA (optional)

D/A outputs

Mode U:	(voltage)
Unipolar:	0-5V, 0-10V
Bipolar:	+5V, +10V

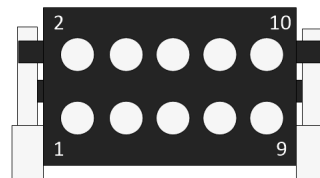
3.2.9.1.3. Pin assignment

3.2.9.1.3.1. A/D connector (18pin)



Pin	Analog Channel	Input	Pin	Analog Channel	Input
1	AGND		2	AGND	
3	AD1		4	AD0	
5	AD3		6	AD2	
7	AD5		8	AD4	
9	AD7		10	AD6	
11	AD9		12	AD8	
13	AD11		14	AD10	
15	AD13		16	AD12	
17	AD15		18	AD14	

3.2.9.1.3.2. D/A connector (10pin)



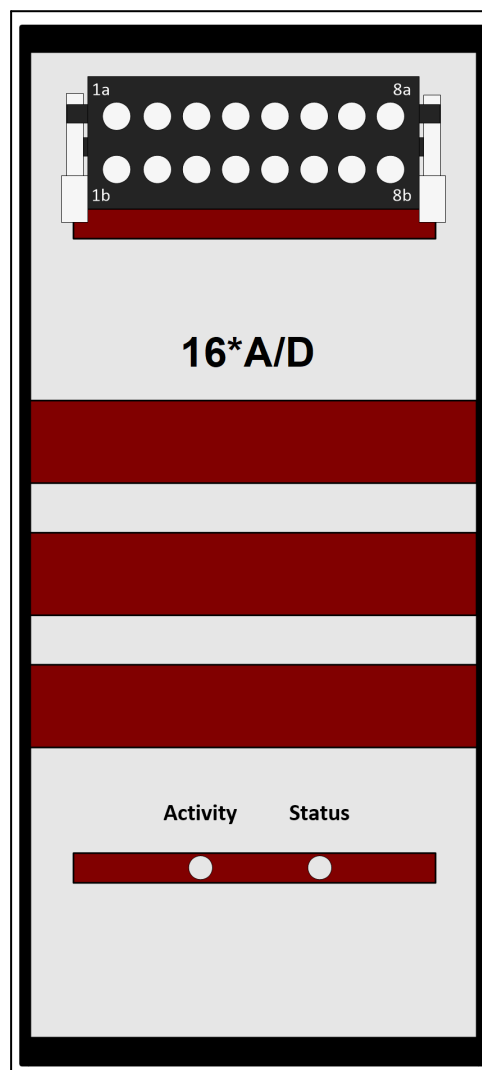
Pin	Analog Channel	Output	Pin	Analog Channel	Output
1	AGND		2	DA0	
3	AGND		4	DA1	
5	AGND		6	DA2	
7	AGND		8	DA3	
9	AGND		10	AGND	

3.2.9.2. RO-AD16

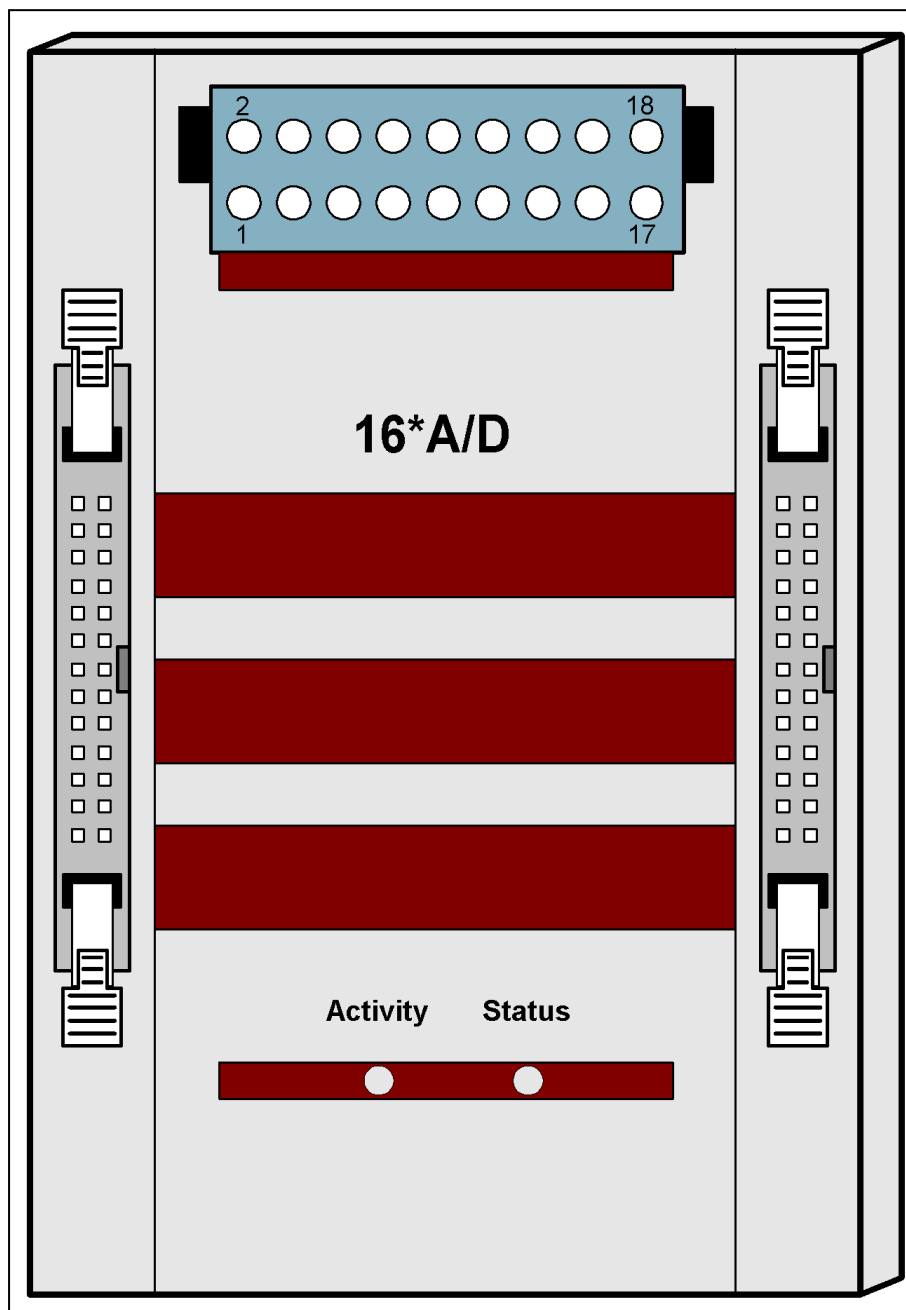
With 16 A/D input channels, this module provides a good basis for converting voltages into digital values.

3.2.9.2.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals.



The figure shows an I/O module of the flexible connector system with corresponding numbering of the connection terminals.



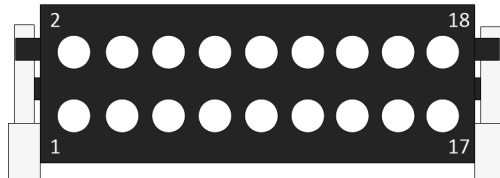
3.2.9.2.2. Technical data

A/D input channels

Mode U:	(voltage)
Unipolar:	0-5V, 0-10V
Bipolar:	+5V, +10V
Mode I:	(current)
Area:	0-20mA (optional)

3.2.9.2.3. Pin assignment

3.2.9.2.3.1. A/D connector (18pin)



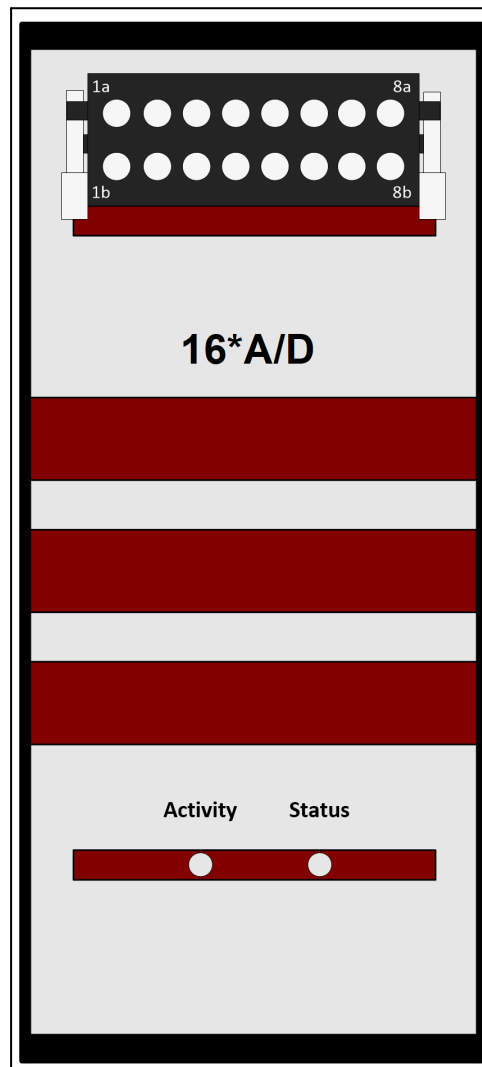
Pin	Analog Channel	Input	Pin	Analog Channel	Input
1	AGND		2	AGND	
3	AD1		4	AD0	
5	AD3		6	AD2	
7	AD5		8	AD4	
9	AD7		10	AD6	
11	AD9		12	AD8	
13	AD11		14	AD10	
15	AD13		16	AD12	
17	AD15		18	AD14	

3.2.9.3. RO-AD16_ISO

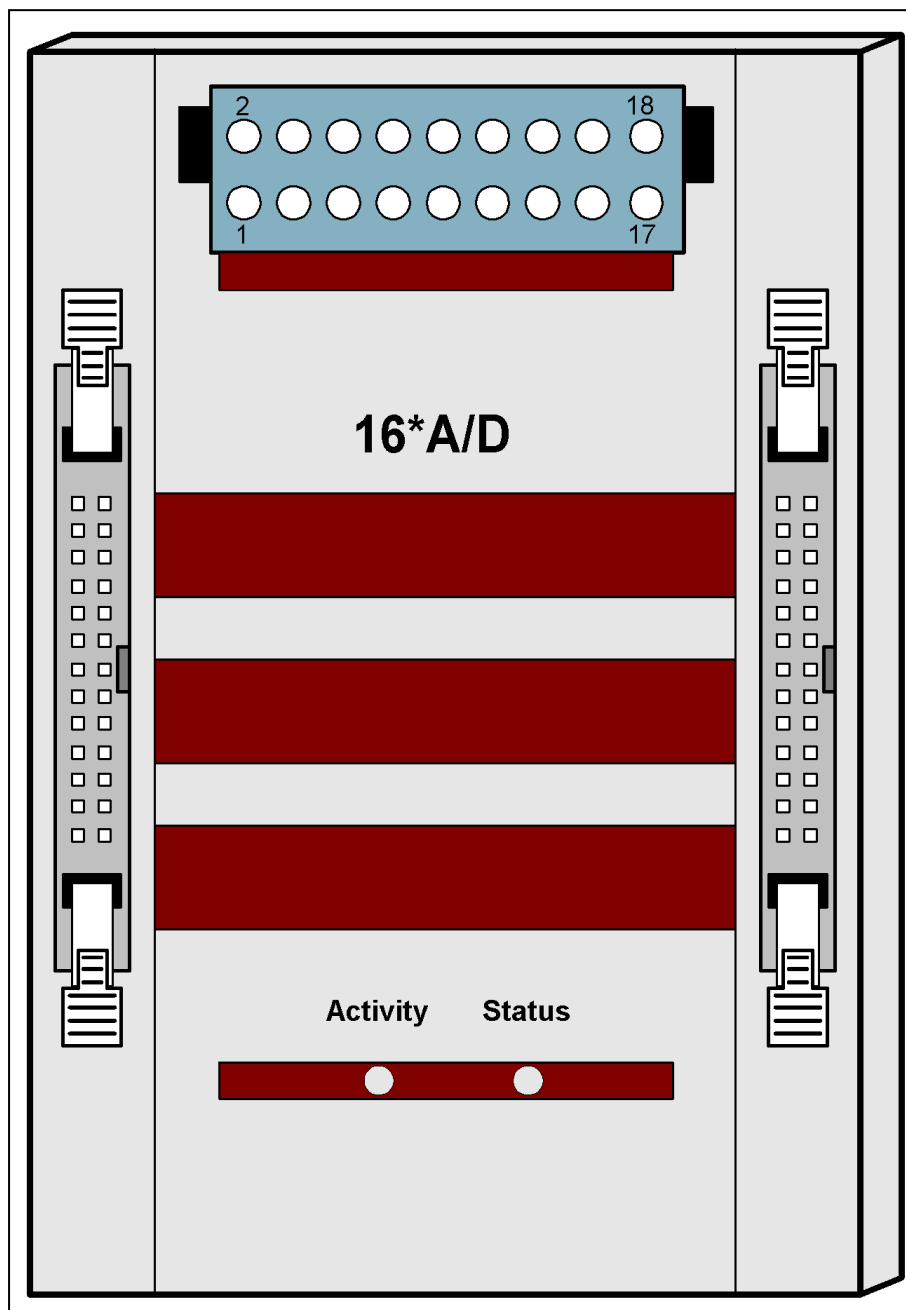
With 16 A/D input channels (galvanically isolated), this module provides a good basis for converting voltages into digital values.

3.2.9.3.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals.



The figure shows an I/O module of the flexible connector system with corresponding numbering of the connection terminals.



3.2.9.3.2. Technical data

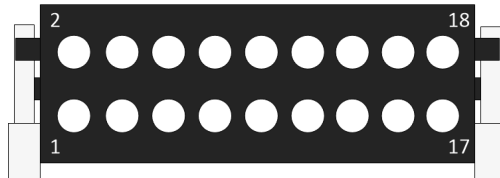
A/D input channels

Galvanically isolated

Mode U:	(voltage)
Unipolar:	0-5V, 0-10V
Bipolar:	+5V, +10V
Mode I:	(current)
Area:	0-20mA (optional)

3.2.9.3.3. Pin assignment

3.2.9.3.3.1. A/D connector (18pin)



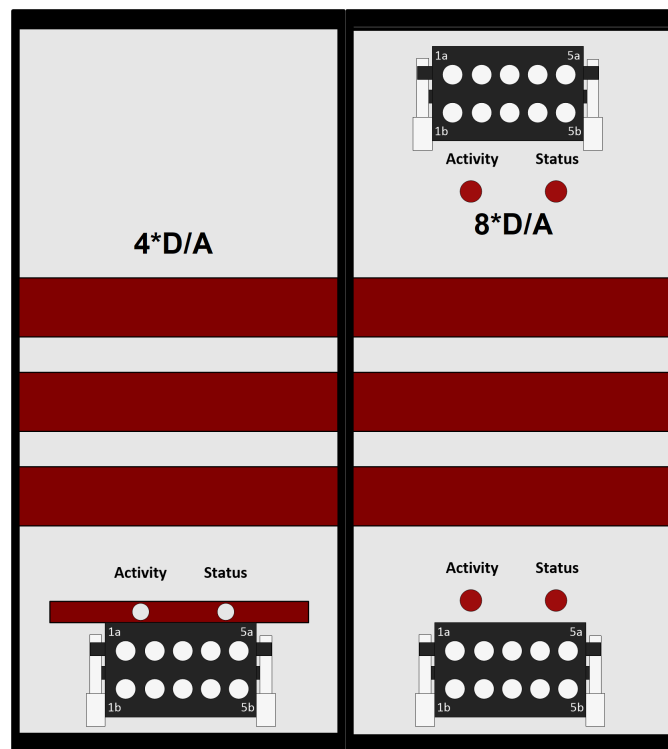
Pin	Analog Channel	Input	Pin	Analog Channel	Input
1	AGND		2	AGND	
3	AD1		4	AD0	
5	AD3		6	AD2	
7	AD5		8	AD4	
9	AD7		10	AD6	
11	AD9		12	AD8	
13	AD11		14	AD10	
15	AD13		16	AD12	
17	AD15		18	AD14	

3.2.9.4. RO-DA4/RO-DA8

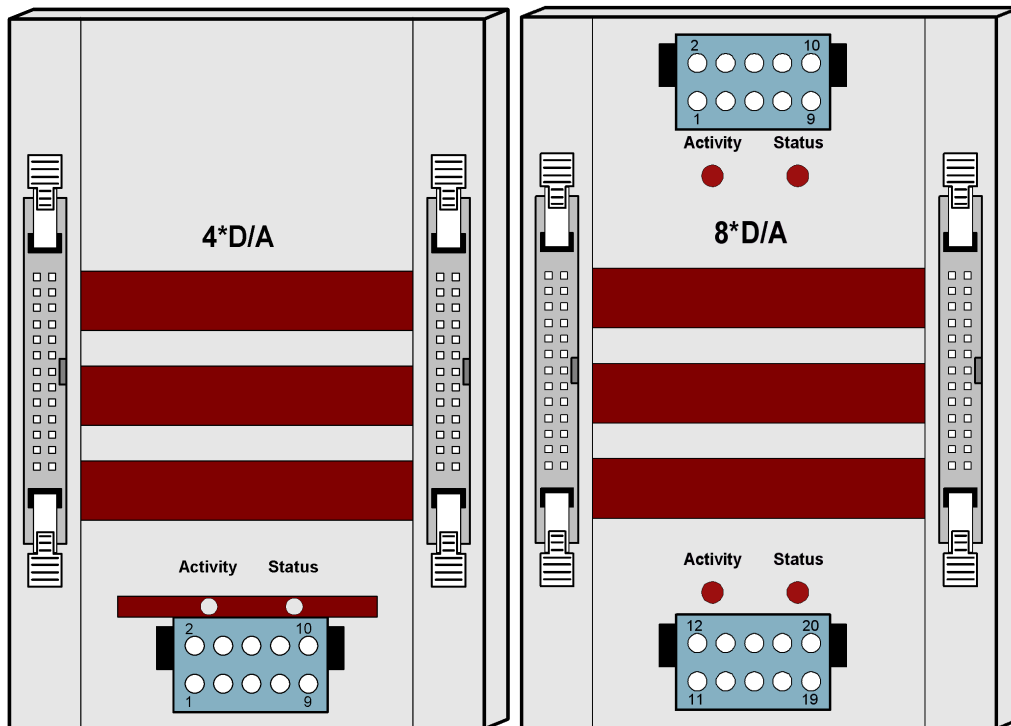
With 4/8 D/A outputs, these modules provide a good basis for converting digital values into voltages.

3.2.9.4.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals at a D/A 4.



The figure shows an I/O module of the flexible connector system with corresponding numbering of the connection terminals.



3.2.9.4.2. Technical data

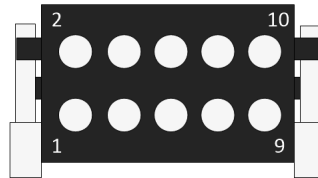
D/A outputs

Unipolar: 0-5V, 0-10V

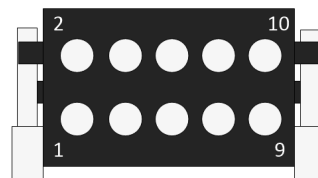
Bipolar: +5V, +10V

3.2.9.4.3. Pin assignment

3.2.9.4.3.1. Pinout D/A connector (10pin)



Pin		Pin	
1	AGND	2	DA0
3	AGND	4	DA1
5	AGND	6	DA2
7	AGND	8	DA3
9	AGND	10	AGND



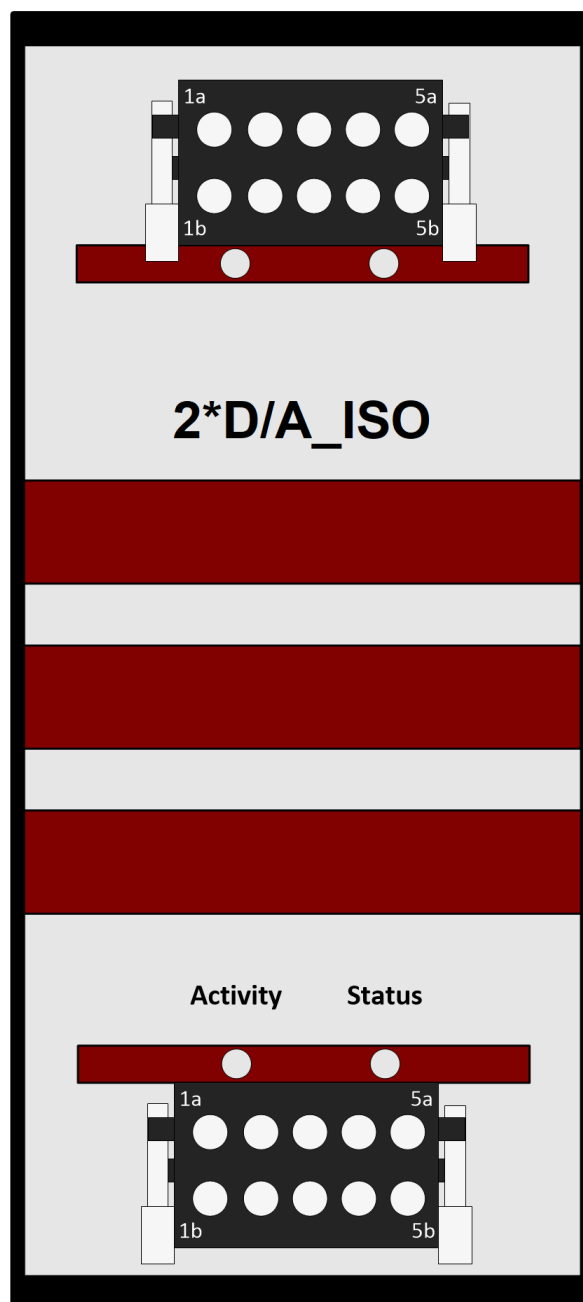
Pin		Pin	
11	AGND	12	DA4
13	AGND	14	DA5
15	AGND	16	DA6
17	AGND	18	DA7
19	AGND	20	AGND

3.2.9.5. RO-DA2_ISO

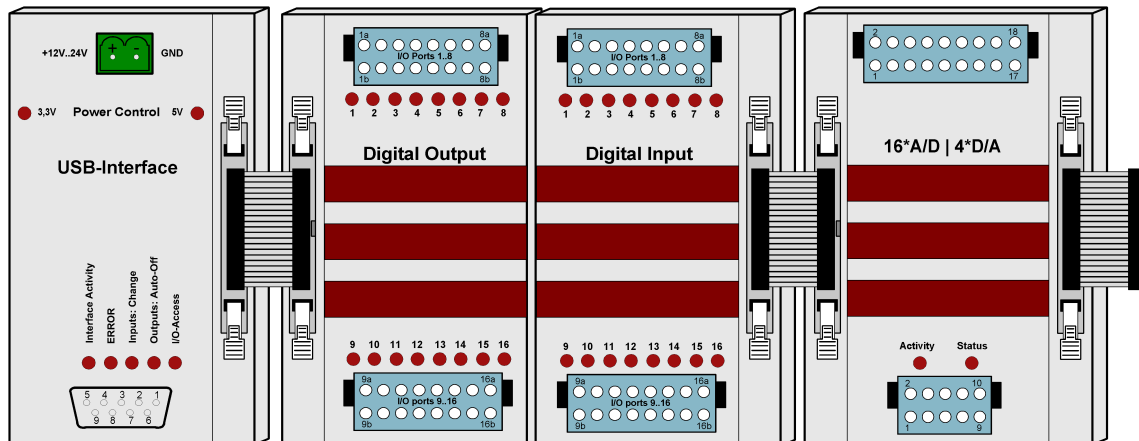
With 2 galvanically isolated D/A outputs, this module provides a good basis for converting digital values into voltages.

3.2.9.5.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals.



The figure shows an I/O module of the flexible connector system with corresponding numbering of the connection terminals.



3.2.9.5.2. Technical data

D/A outputs

Timeout protection

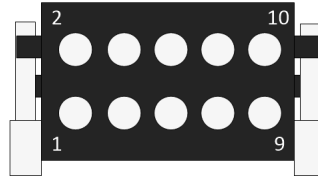
Mode U:	(Voltage)
Unipolar:	0V-5V, 0V-10V
Bipolar:	+5V, +10V

Mode I:	(current)
	0-20mA, 4-20mA, 0-24mA

3.2.9.5.3. Pin assignment

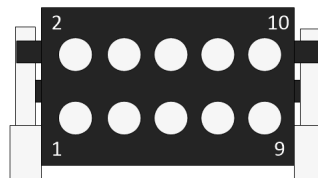
3.2.9.5.3.1. D/A connector (10pin)

Connector top:



Pin		Pin	
1	VOUT_A	2	+Vsense_A
3	VOUT_A	4	+Vsense_A
5	AGND	6	-Vsense_A
7	AGND	8	-Vsense_A
9	AGND	10	IOUT_A

Connector below:

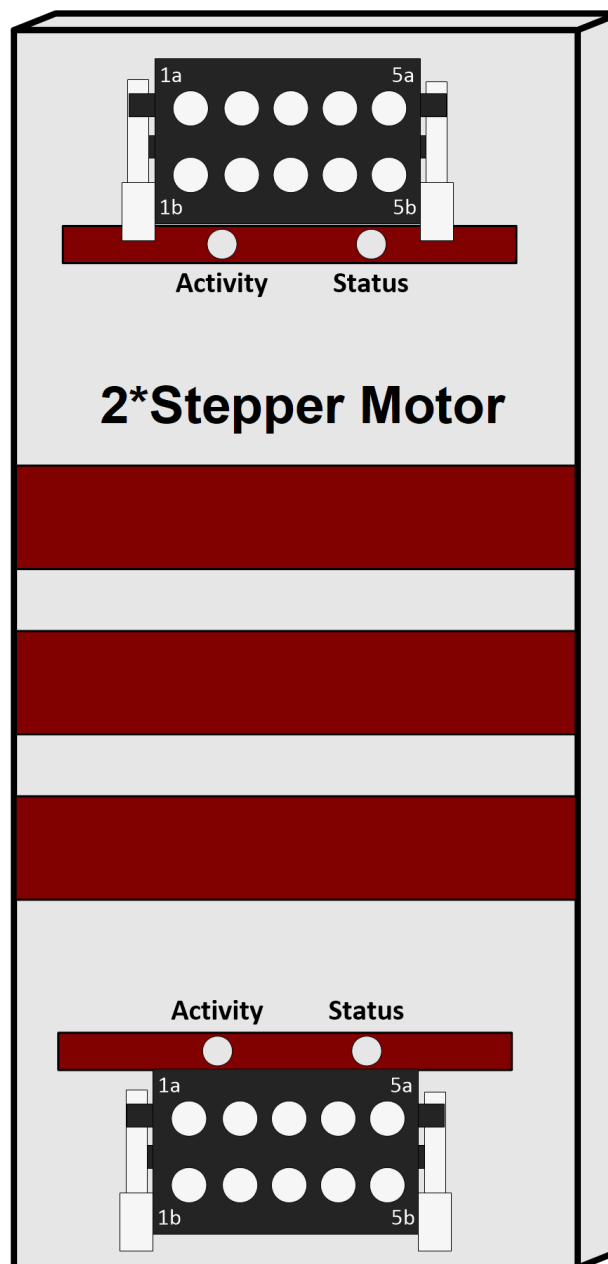


Pin		Pin	
1	VOUT_B	2	+Vsense_B
3	VOUT_B	4	+Vsense_B
5	AGND	6	-Vsense_B
7	AGND	8	-Vsense_B
9	AGND	10	IOUT_B

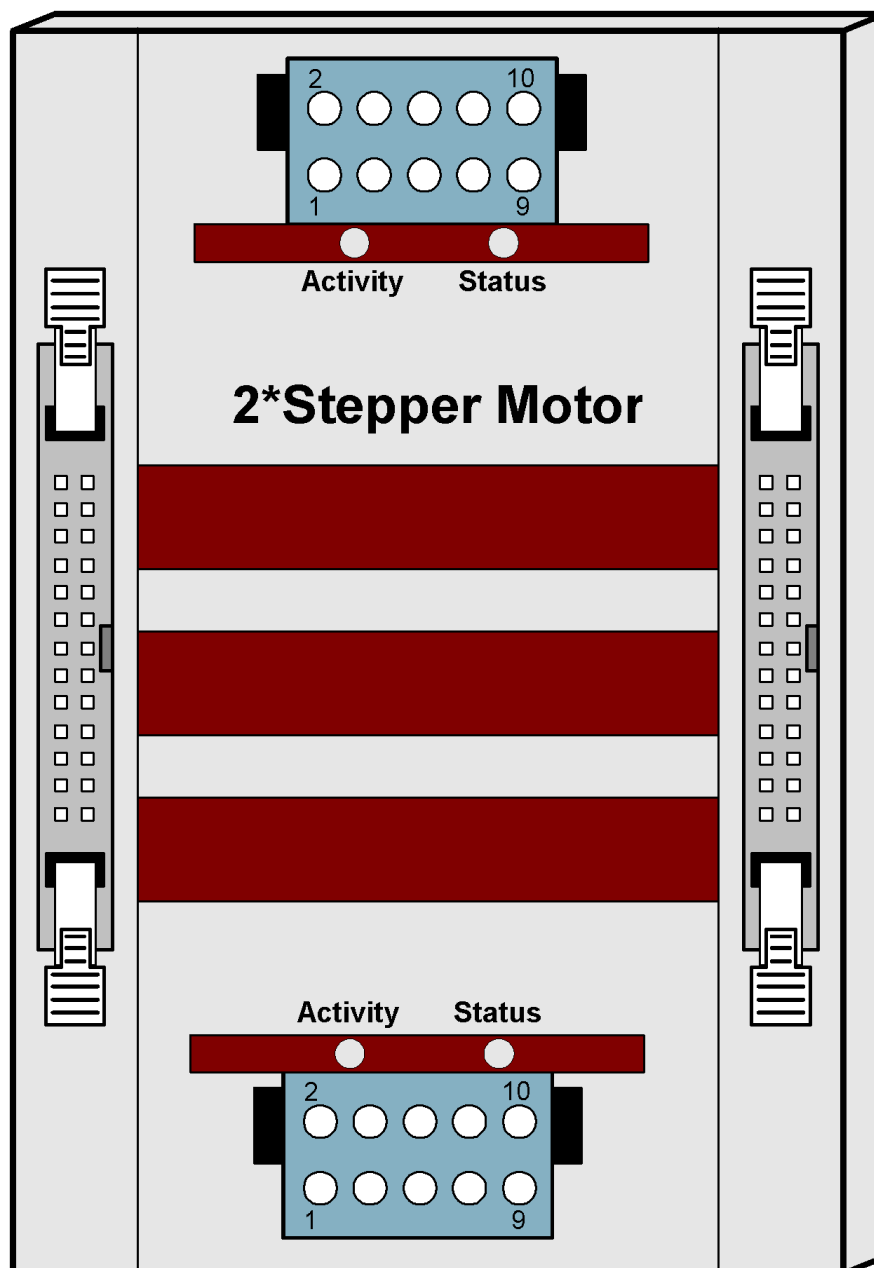
3.2.10. Stepper Modul

3.2.10.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals.



The figure shows a stepper module of the flexible connector system with corresponding numbering of the connection terminals.



3.2.10.2. Technical data

General:

- Module for two stepper motors
- Convenient connector system with ejector mechanism
- Can be easily combined with other modules of the RO series

Adjustable parameters:

- Start/stop frequency
- Maximum cadence
- Acceleration ramp
- Brake ramp
- Phase current
- Holding current
- Holding time

3.2.10.3. Control

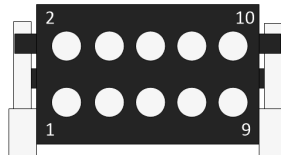
All parameters can be conveniently set via the DELIB Library.

Two reference switches are used to move to reference positions. A further two limit switches ensure safe stopping and exclusive travel in the opposite direction when actuated.

3.2.10.4. Pin assignment

3.2.10.4.1. Stepper connector (10pin)

Pin assignment of a connector and at the same time of a stepper motor:



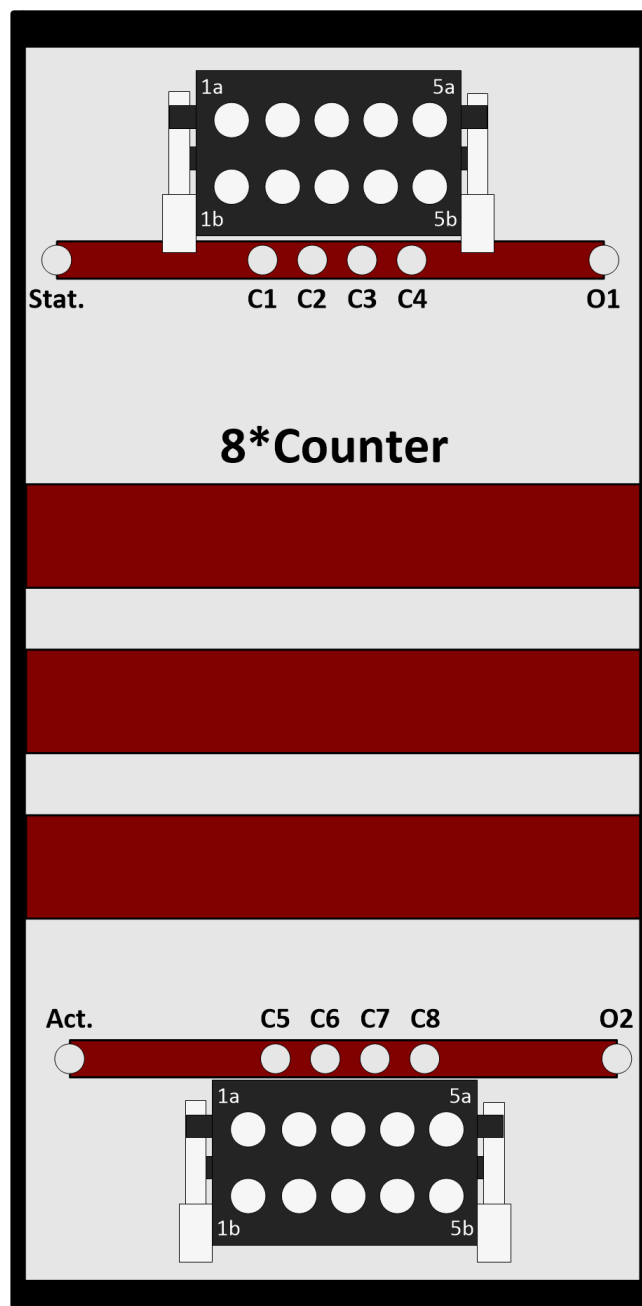
Pin		Pin	
1	24 V (Motor supply)	2	0 V (Motor supply)
3	Phase 1 (+)	4	Reference switch 2*)
5	Phase 1 (-)	6	Reference switch 1*)
7	Phase 2 (+)	8	Limit switch 2*)
9	Phase 2 (-)	10	Limit switch 1*)

*) The switches are to be wired against 24 V.

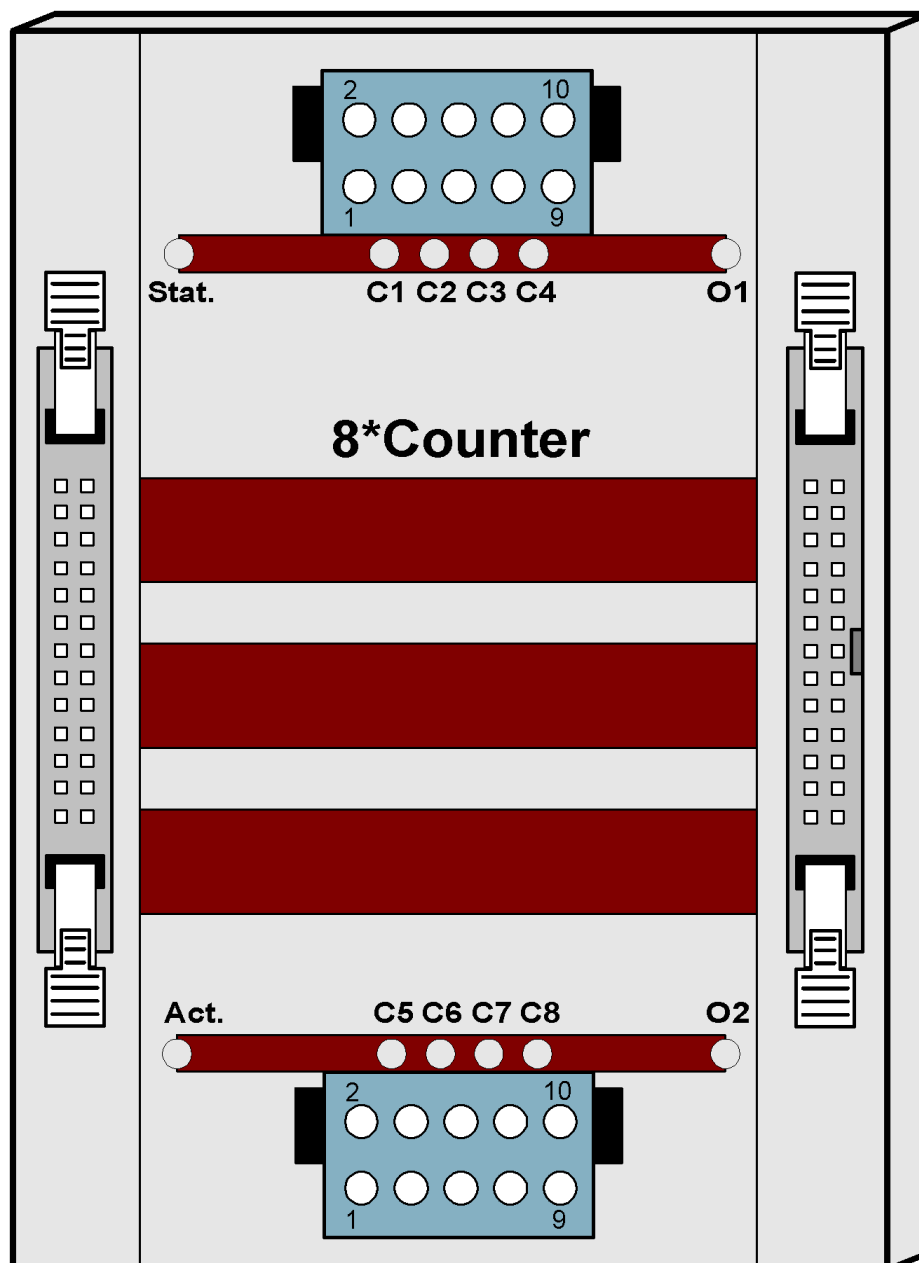
3.2.11. RO-CNT8

3.2.11.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals.



The figure shows a Counter-8 module of the flexible connector system with corresponding numbering of the terminals.



3.2.11.2. Technical data



- 8 Input counter
 - max 20 MHz input frequency
 - 32-Bit counter
 - 15V-30V galvanically isolated inputs / Reconfigurable to 5V-12V
 - Each input channel can be provided with a filter
 - Each input channel indicates with an LED that an input state has been detected
-
- 2 Output counter
 - Max 5 MHz output frequency
 - 32-Bit counter
 - The number of pulses to be generated can be set (0 = infinite)
 - Setting the low time (10ns steps)
 - Setting the high time (10ns steps)
 - PWM outputs with e.g. the following modes:
 - 1kHz - frequency PWM ratio with 0.01% resolution
 - 10kHz - frequency PWM ratio with 0.1% resolution
 - 100kHz - frequency PWM ratio with 1% resolution
-
- Indicator LEDs Activity LED (short flashing on interface communication)
 - Status LED (short flashing when D/A converter is accessed)
 - Connectors 10 pin pluggable terminal strips for one D/A output each

3.2.11.3. Counter Modul Modi

The counter module has the following 5 modes:

Mode 0: Counting pulses With this mode, pulses can be counted on the basis of a rising edge. A reset is possible via software.

Mode 1: Counting pulses with "reset on read".

With this mode, pulses can be counted on the basis of a rising edge. When the counter is read out, the counter is reset to zero at the same time.

Mode 2: Period duration determination

In this mode the period "T" of the applied input signal is measured. An internal 100MHz counter serves as time base. At a rising edge of the input signal the 32 bit counter is latched and can be evaluated while the next signal is counted again.

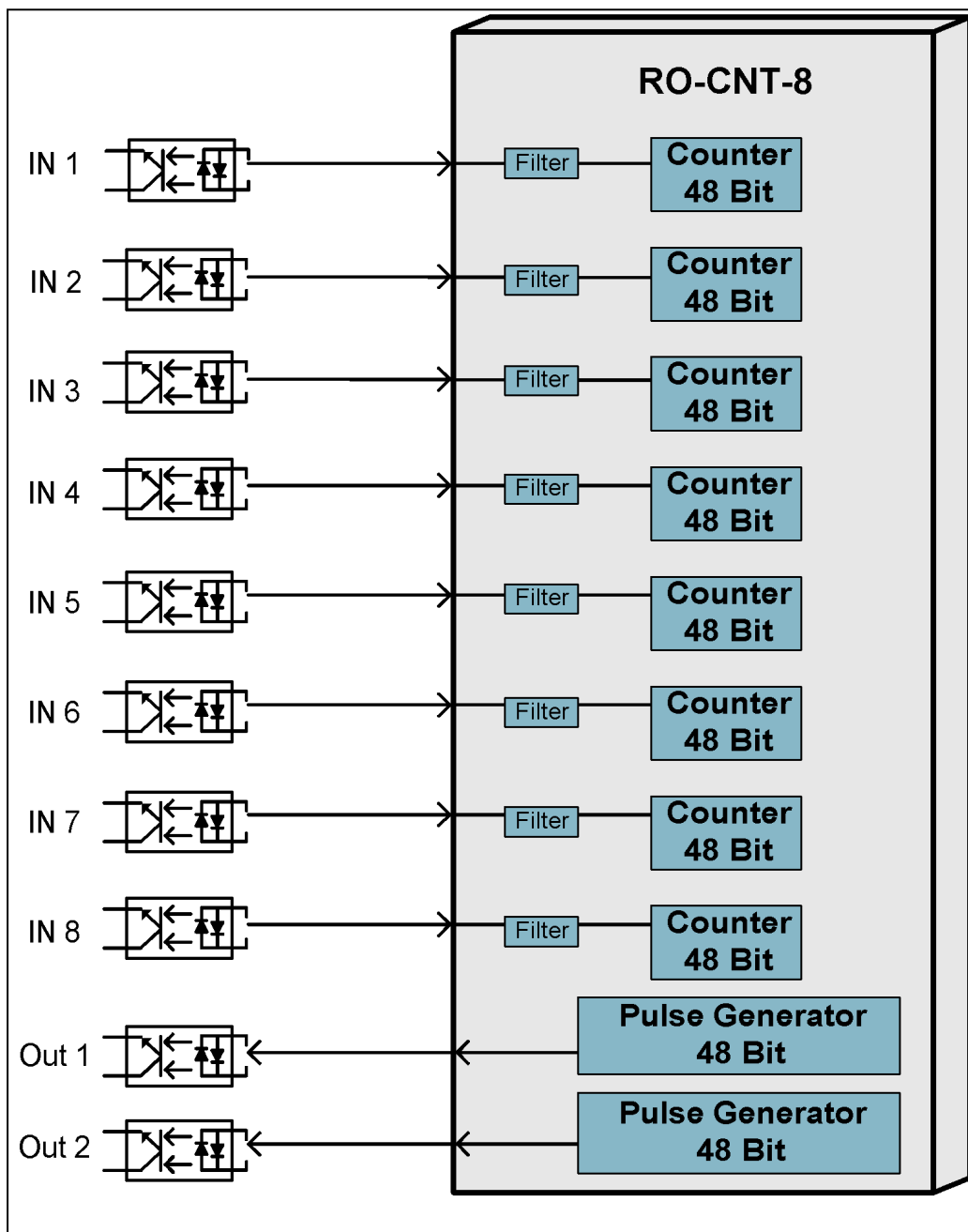
Mode 3: Number measurement/frequency measurement

With this mode the number of rising edges of the input signal within a certain time can be measured. Here the times 1ms, 10ms, 100ms, 1sec. can be configured by software. In this way, e.g. frequency measurements can be carried out.

Mode 4: PWM measurements

With this mode, the "high" time and the "low" time of a signal are measured with two different counters. This allows the high / low ratio to be determined (PWM). By using an internal 100MHz counter, extremely accurate PWM ratios can be determined here.

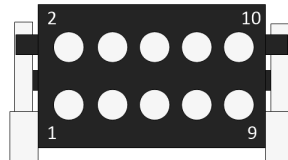
3.2.11.4. Block diagram



3.2.11.5. Pin assignment

3.2.11.5.1. Pin assignment RO-CNT8

Connector 10 pin pluggable terminal strip for 4 * counter inputs each

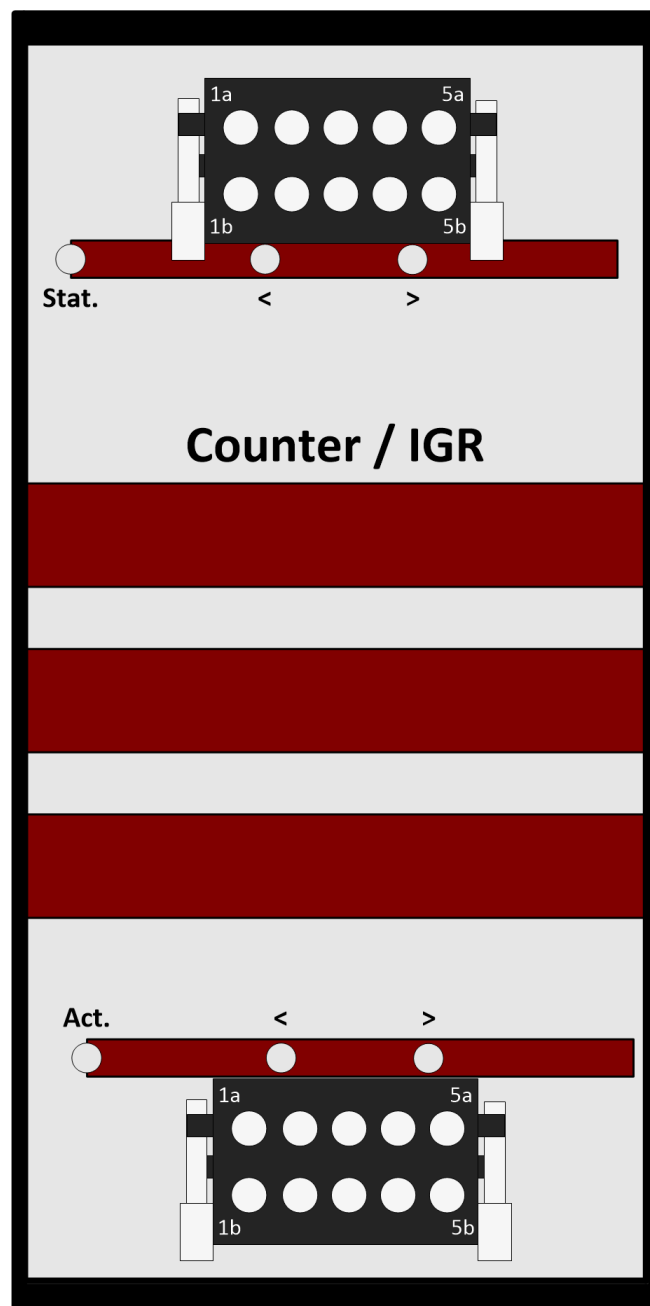


Pin	Belegung
1	Inputchannel 1-
2	Inputchannel 1+
3	Inputchannel 2-
4	Inputchannel 2+
5	Inputchannel 3-
6	Inputchannel 3+
7	Inputchannel 4-
8	Inputchannel 4+
9	GND
10	CNT-OUT +5V

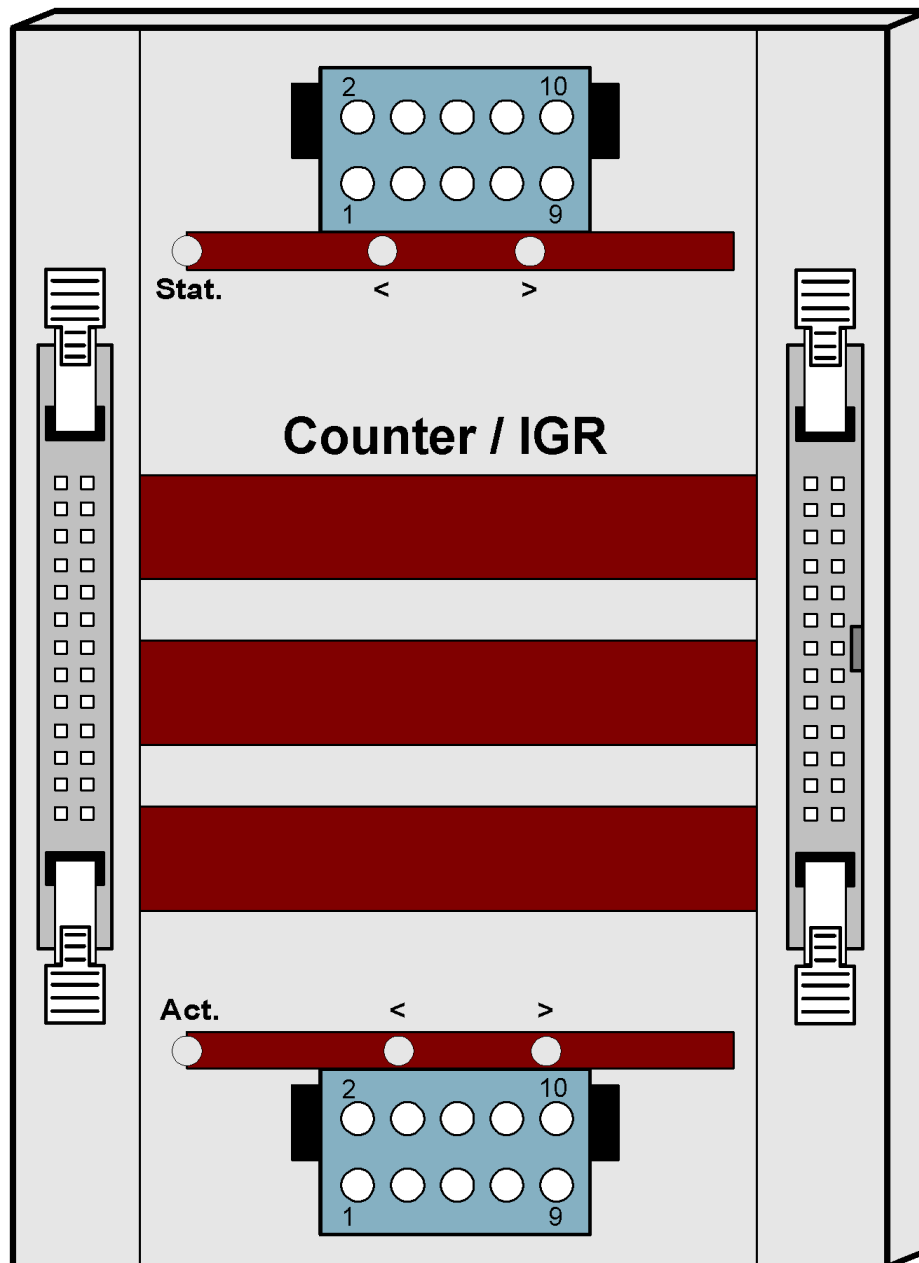
3.2.12. RO-CNT/IGR

3.2.12.1. Overview image

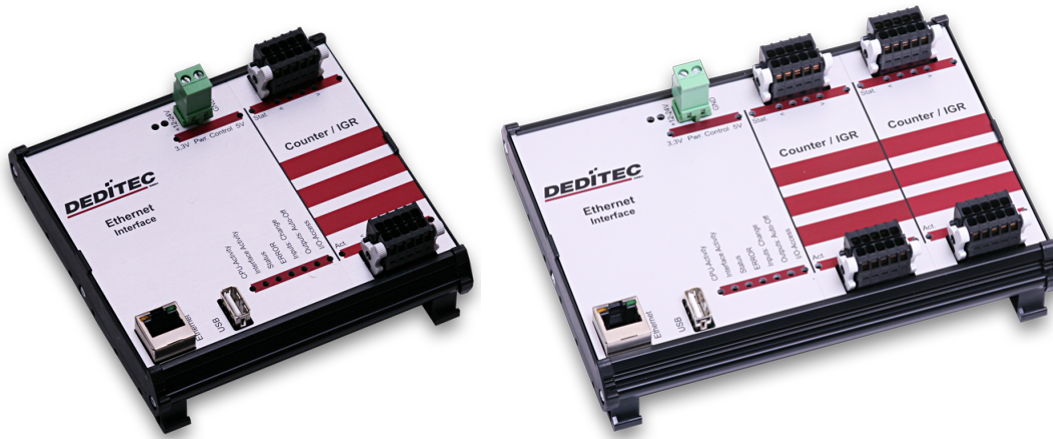
The figure shows a module with corresponding numbering of the connection terminals.



The figure shows a counter/IGR module of the flexible connector system with corresponding numbering of the terminals.



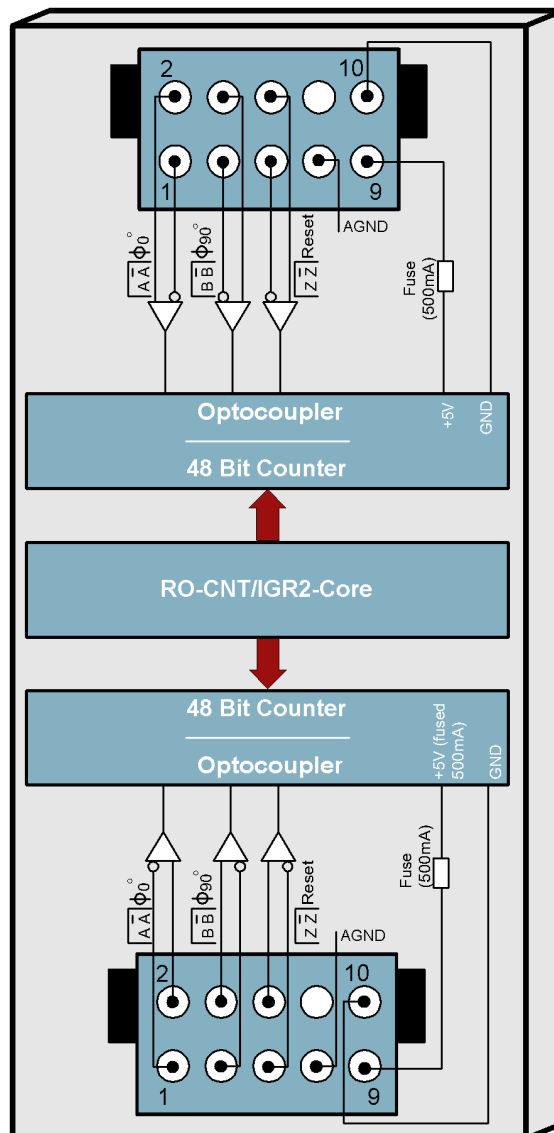
3.2.12.2. Technical data



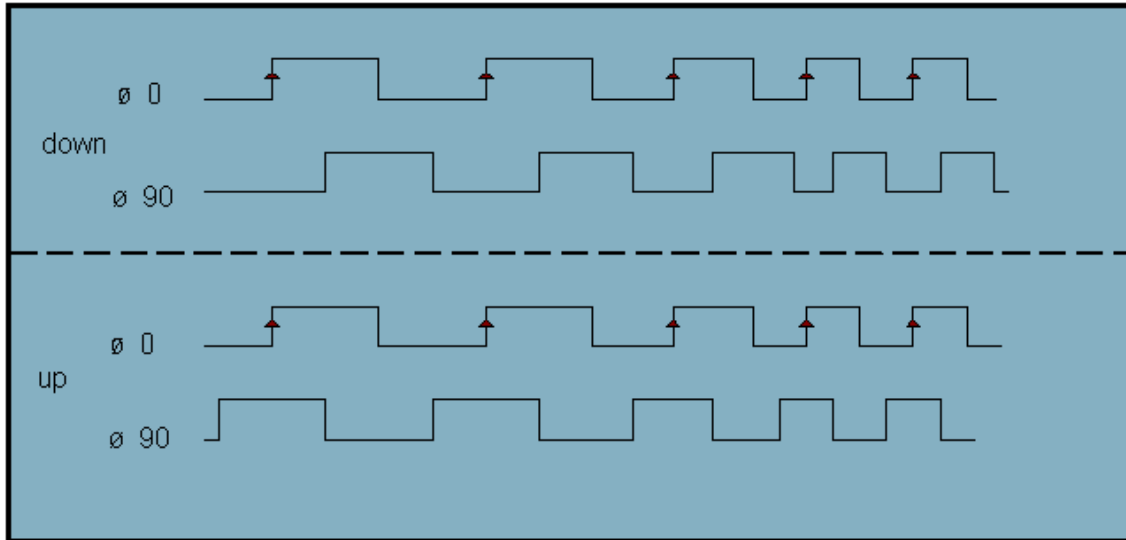
- 48Bit Counter
- Max 10MHz Phi 0/Phi 90° Frequency
- Galvanically isolated (A, B, Z of the inputs of both incremental encoder inputs among each other and also to the 24V module supply)
- Indicator LEDs Activity LED (short flashing on interface communication)
- Status LED (flashes briefly when the incremental encoders are accessed)
- Direction LEDs (light up depending on the selected direction of rotation of the incremental encoder)
- Connectors 10 pin pluggable terminal strips for one incremental encoder each
- Operating temperature 10°C...+50°C

3.2.12.3. Block diagram

The block diagram clearly shows how the module is constructed.



3.2.12.4. Evaluation Phi 0/Pi 90° (Graphical)



This graphic illustrates when the counting module counts up/down.

If the level is low at Phi 90°, it is counted down. If the level is high, it is counted up.

3.2.12.5. LEDs

Activity LED:

- Short flashing on interface communication

Status LED:

- Short flashing on access from the interface to the incremental encoders

Direction LEDs:

- left direction LED flashes briefly when the counter module counts down.
- right direction LED flashes briefly when the counter module counts up.
- both direction LEDs flash briefly when the counter reading is reset.

Counter LED:

- flashes when a counting process is taking place.

3.2.12.6. Inputs

3.2.12.6.1. References

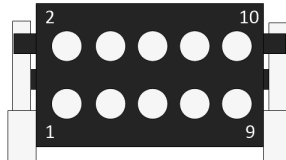
By restarting the counter module, the absolute position is not recognized and the current position is set to 0. This may result in incorrect measurements. Limit switches can be used to set reference runs in which a specific position is approached. This leads to the exact calibration of the encoder. Incremental encoders still output a reference pulse (index) on a third channel per full revolution. Some incremental encoders also have a distance-coded reference mark.

3.2.12.6.2. Visual control of the inputs

The status as well as the direction of rotation of each incremental encoder is displayed directly via LED. Errors in the wiring can thus be detected more quickly.

3.2.12.7. Pin assignment

Connector 10 pin pluggable terminal strip for 1 * incremental encoder input each

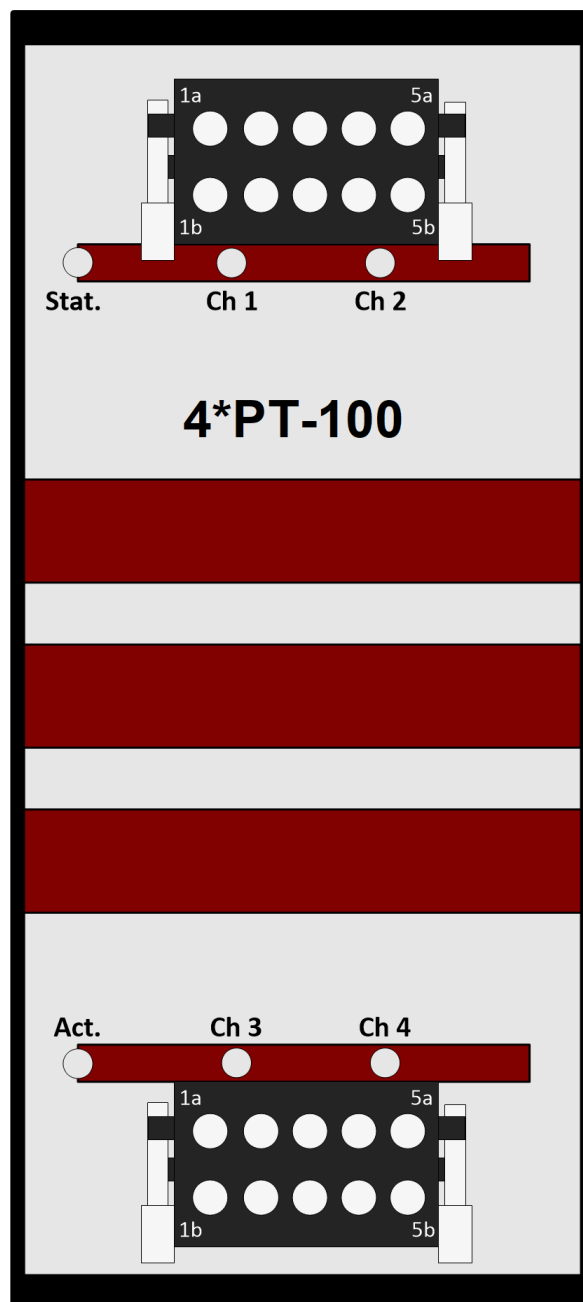


Pin	Occupancy	Funktion	Galvanic isolation
1	Inputchannel A-	Phi 0°	Galvanically isolated
2	Inputchannel A+		
3	Inputchannel B-	Phi 90°	
4	Inputchannel B+		
5	Inputchannel Z-	Reset	
6	Inputchannel Z+		
7	GND-Analog	GND to Pin 1-6	
8	N.C.		
9	GND	GND to Pin 10	No galvanic isolation
10	+5V	Fused (500mA)	

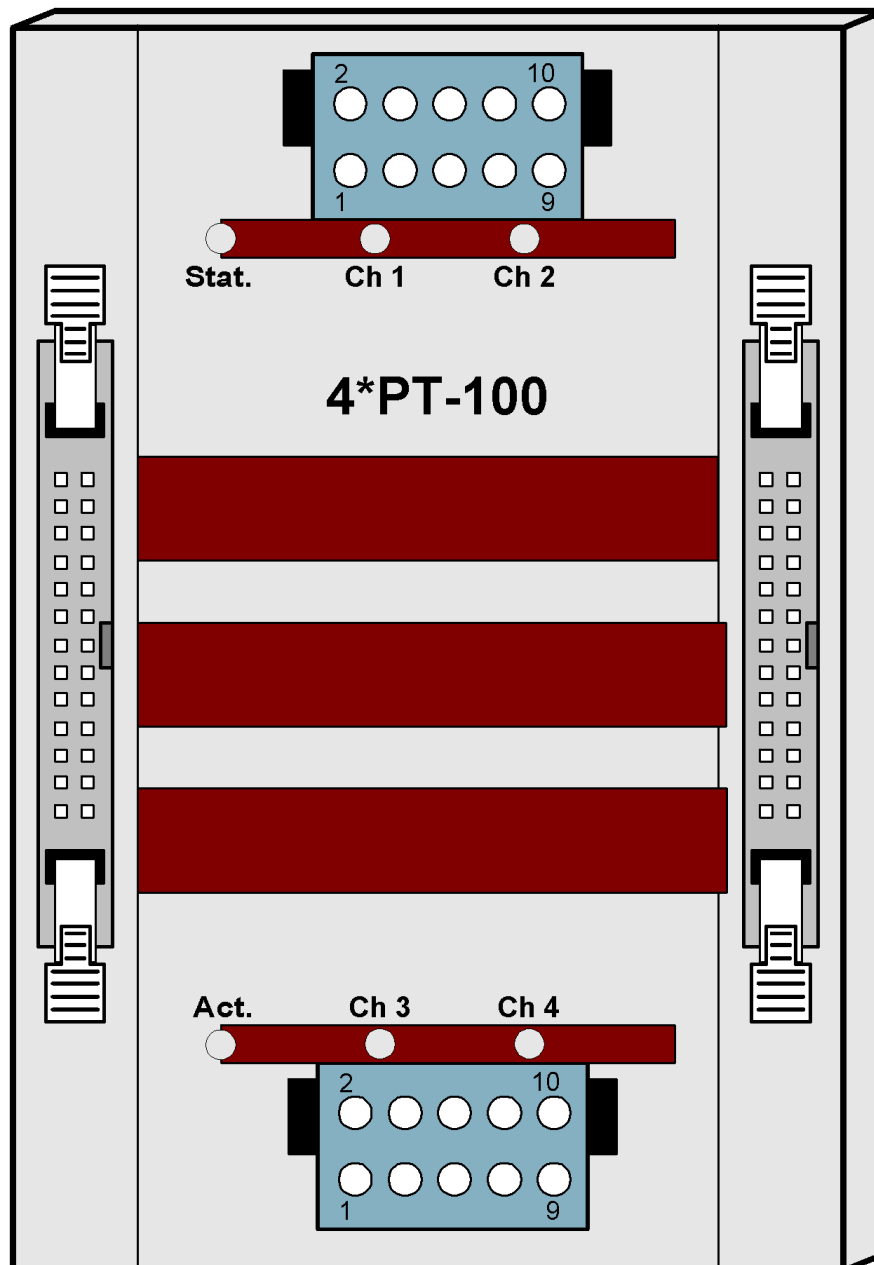
3.2.13. RO-PT100-4 / RO-PT100-8

3.2.13.1. Overview image

The figure shows a module with corresponding numbering of the connection terminals.



The figure shows a Counter-8 module of the flexible connector system with corresponding numbering of the terminals.



3.2.13.2. Introduction

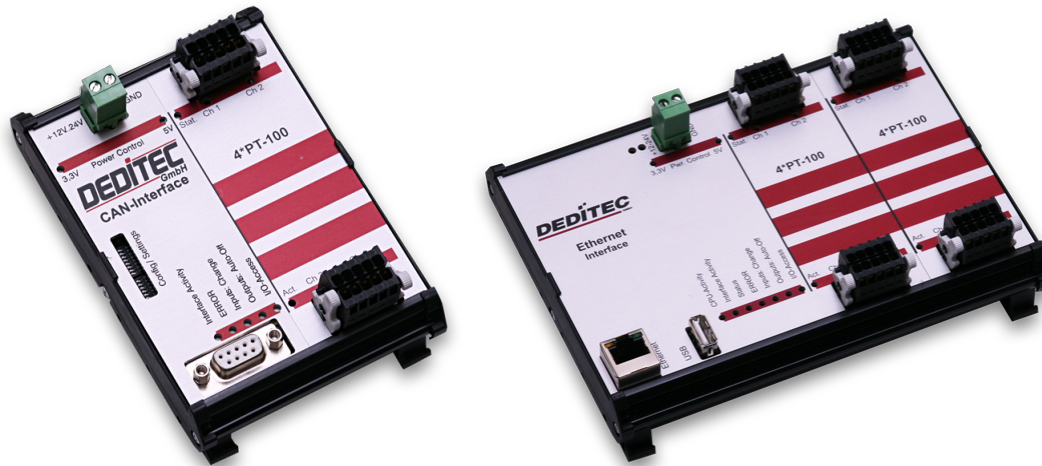
A PT100 sensor can be connected via the two- or three-wire connection, with which a very precise measurement is possible thanks to the integrated 24 bit A/D converter is possible.

Depending on the sensor, the temperature measurement range is between -200° C and

850°C. The line resistance is determined and compensated by an additional sensor line, thus the use of a much longer connection line is possible.

Service-friendly plug-in connectors with locking protection and ejector mechanism are used as terminal blocks. These allow fast, subsequent reconnection of the connected systems. The cable connection itself is made via a screwless plug-in system.

3.2.13.3. Technical data



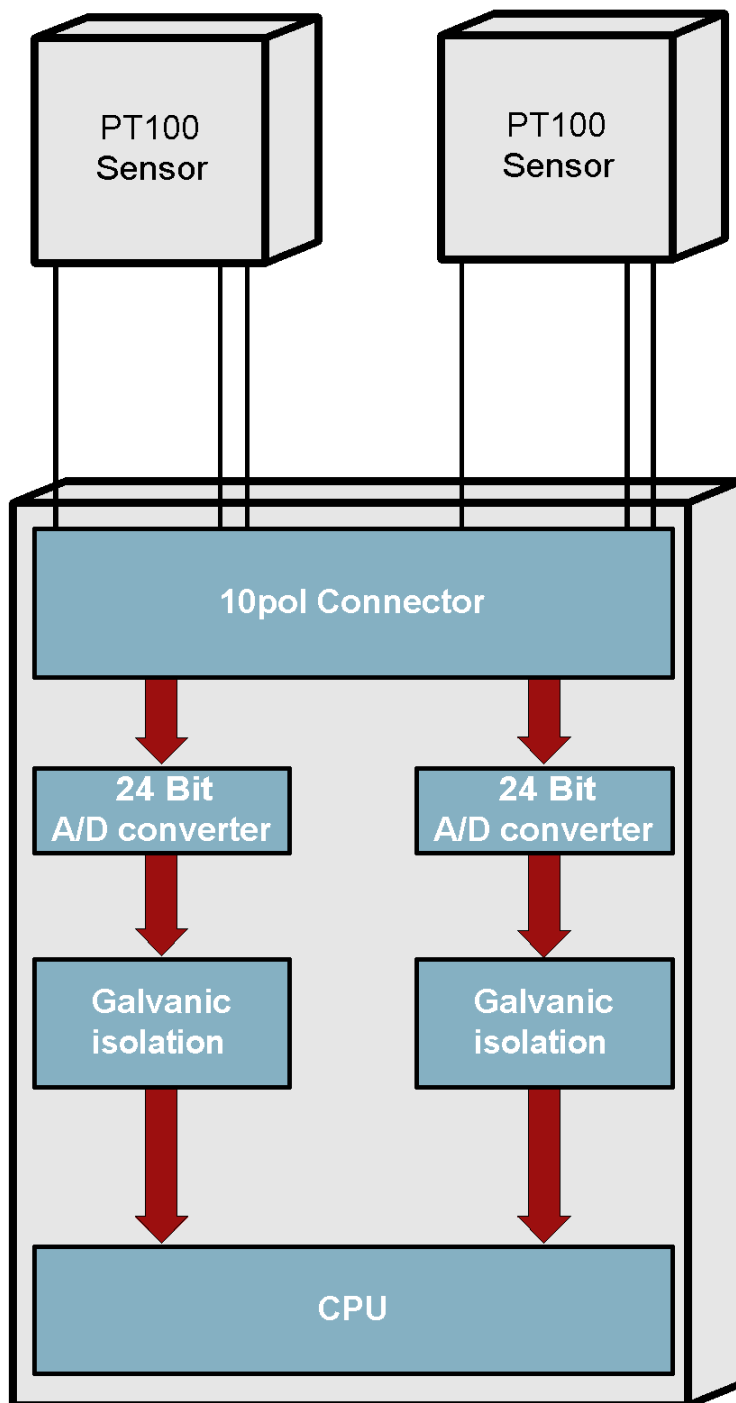
- 24 bit A/D converter for accurate measurement
- The temperature measurement range is between -200°C to 850°C.
- Galvanically isolated inputs.
- Indicator LEDs Activity LED (short flashing on interface communication)
- Status LED (short flashing when PT100 is accessed)
- Connection LED 1 per input (flashes briefly when a PT100 sensor is connected. Lights up when inputs are not occupied or when a defective PT100 sensor is connected).
- Connector 10 pin pluggable terminal strip for 2 * PT100 inputs each
- Can be easily combined with other modules of the RO series

Product specific data:

Product	Inputs
RO-PT100-4	4 * PT100 Inputs
RO-PT100-8	8 * PT100 Inputs

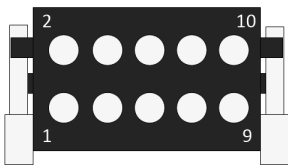
3.2.13.4. Block diagram

The block diagram clearly shows how the module is constructed.



3.2.13.5. Pin assignment

Connector 10 pin pluggable terminal strip for 2 * PT100 inputs each



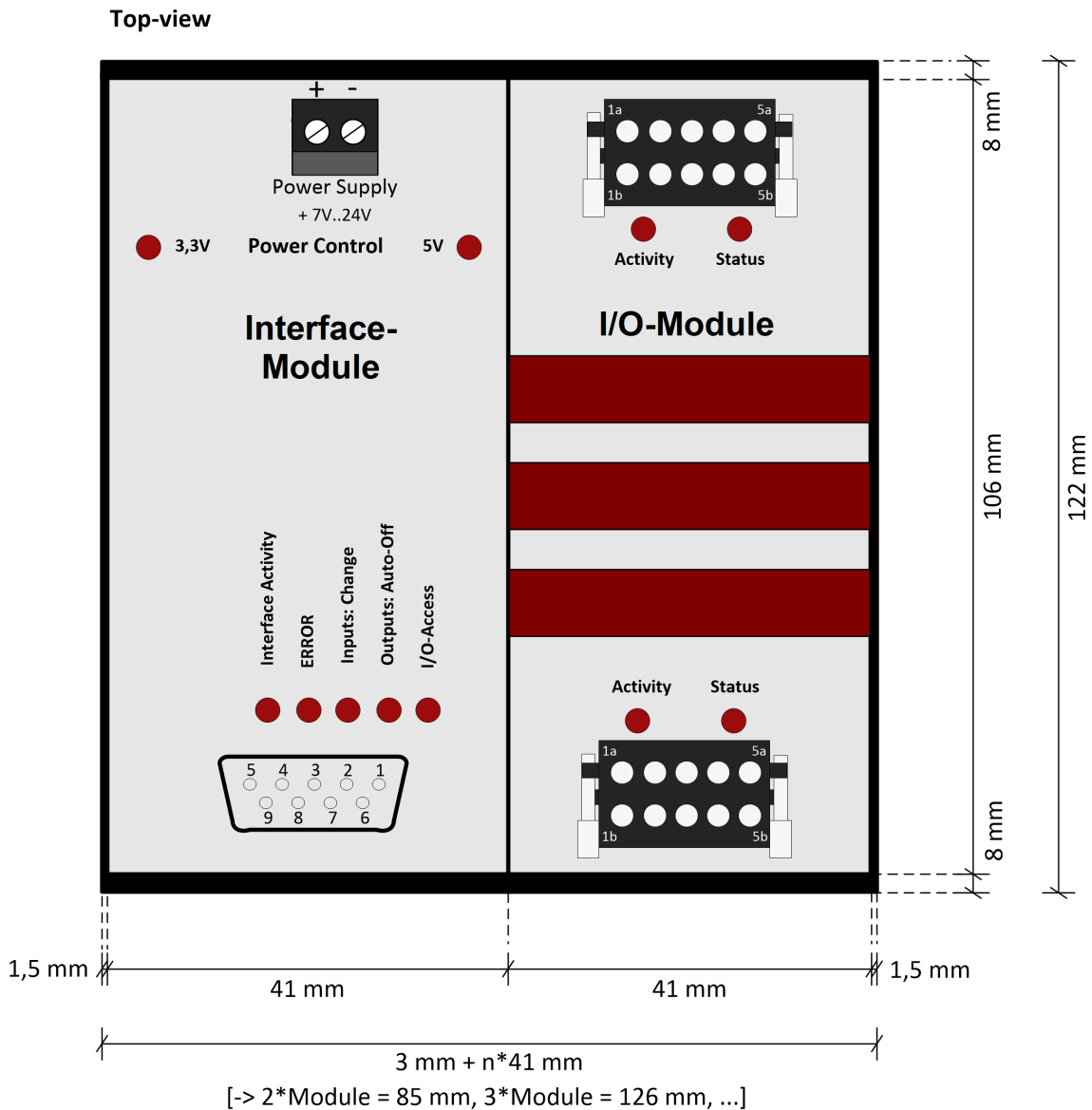
Pin	Occupancy
1	AIN-_A
2	AIN+_A
3	AIN-_A
4	REFIN+_A
5	-
6	-
7	AIN-_B
8	AIN+_A
9	AIN-_B
10	REFIN+_B

Dual-channel configuration: The temperature sensor is connected to pin 1 (AIN-A) and pin 2 (AIN+ A). Pin 3 (AIN-A) and pin 4 (REFIN + A) must be bridged here.

Three-channel configuration: The temperature sensor is connected to pin 1 (AIN-A) and pin 2 (AIN+ A). Pin 4 (REFIN+ A) serves as the sensor line and must be connected between the temperature sensor and pin 1 (AIN- A).

3.3. Dimension of RO-Series

3.3.1. Top view



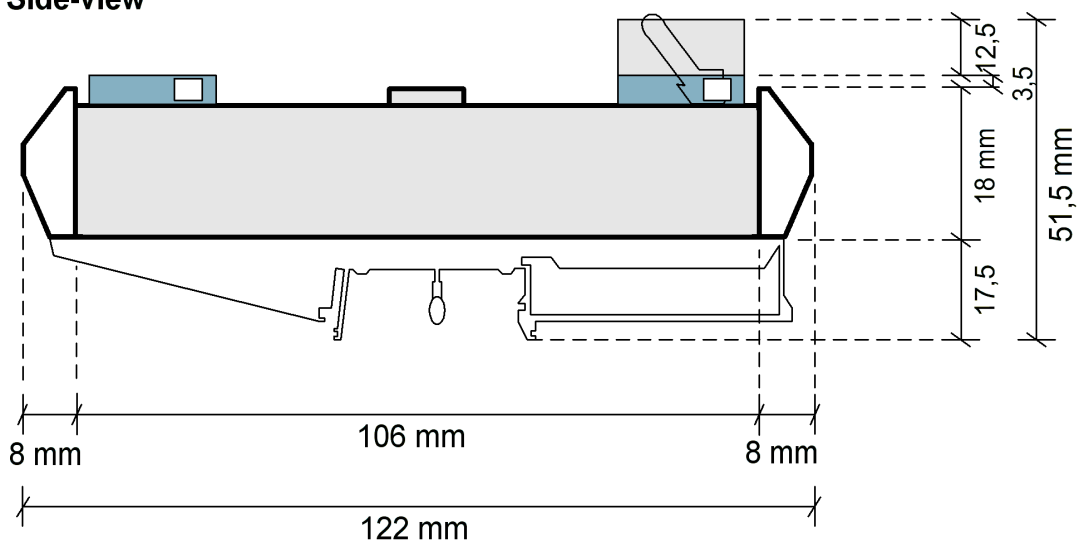
Anmerkung:

Die Breite des ETH-Interface-Moduls entspricht dem Doppelten eines CAN-, SER- oder USB-Interface.

Die Breite beträgt also $3 \text{ mm} + 82 \text{ mm} + n \cdot 41 \text{ mm}$.

3.3.2. Side view

Side-view



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 DELIB Module Config you can test the functionality of our control & regulation products without much configuration effort.

For detailed information see chapter **DELIB_Module_Config**.

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 ["nunit"]]

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

Upper and lower case are not considered here.

Parameter

Command	Channel	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			

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 DELIB Configuration Utility.

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 DELIB Configuration Utility 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 "nunit" argument can be appended to all output requests to be formatted as follows:

```
delib_cli DI8 0 hex nunit
```

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 "nunit" can also be appended to all output requests to be formatted as follows:

```
delib_cli AI 3 hex nunit
```

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

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

```
delib_cli AI 3 V nunit
```

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++

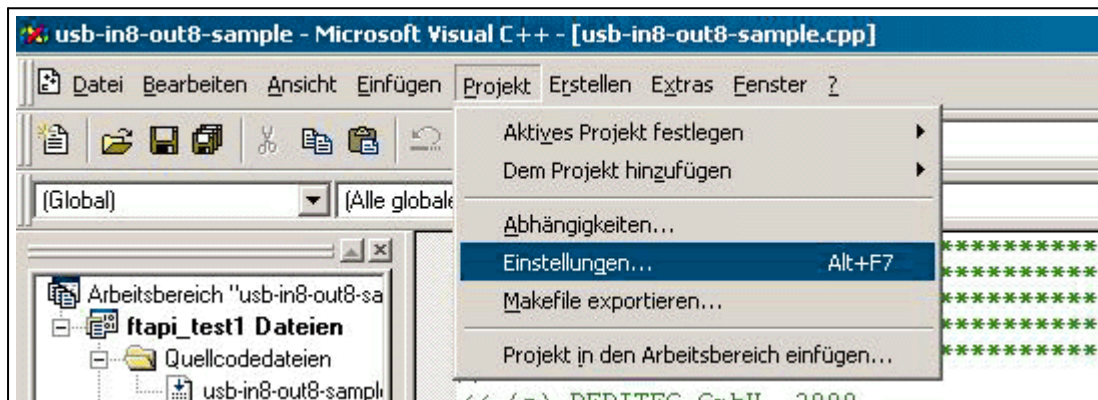
To make it easier to reference the DELIB include and the DELIB lib directory, environment variables are defined when the DELIB is installed.

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

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

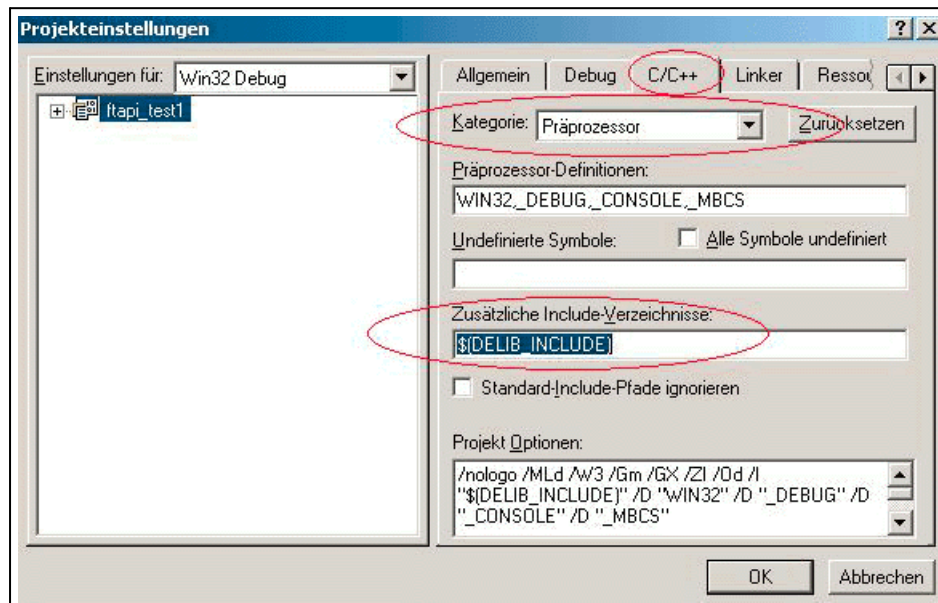
These are entered below in the project settings of the compiler.

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



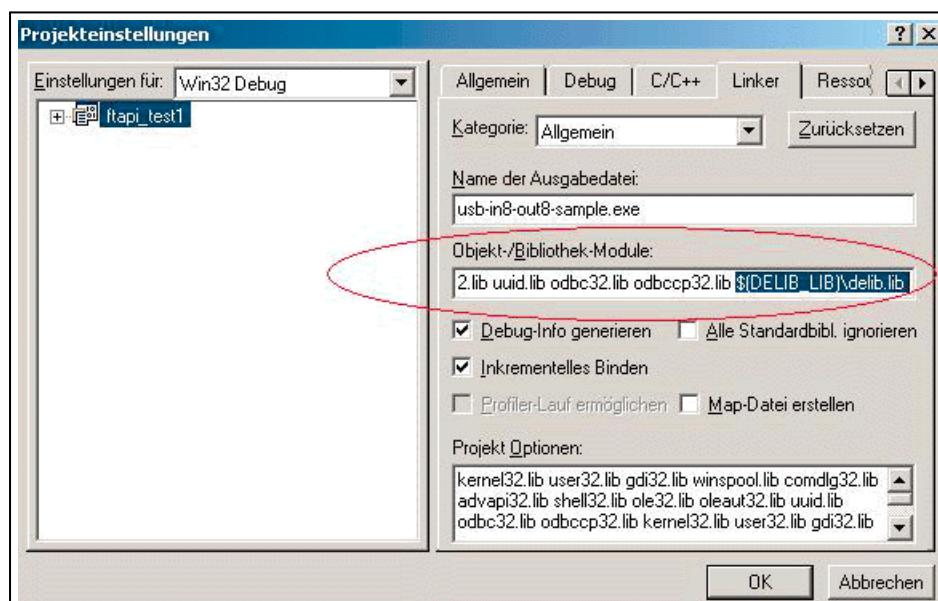
DELIB.H entry in the Visual-C/C++ project settings

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

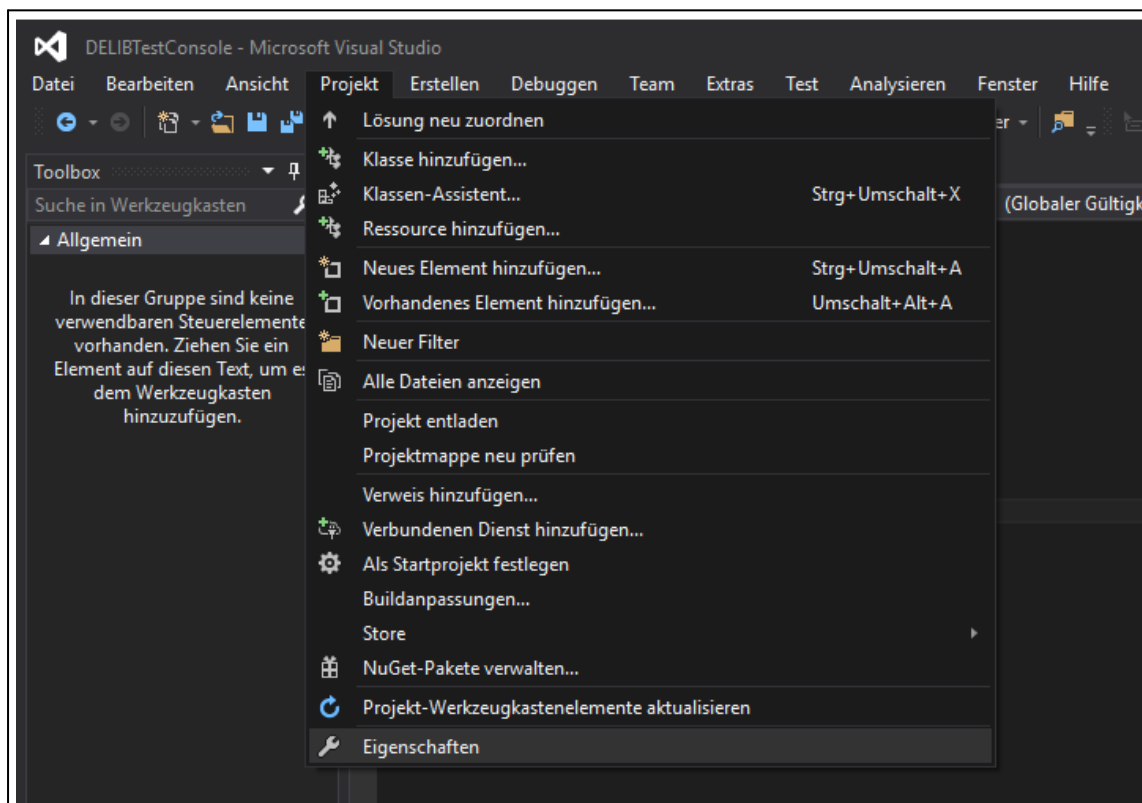
DELIB_INCLUDE = C:\Programs\DEDITEC\DELIB\include

64 Bit DELIB Installation

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

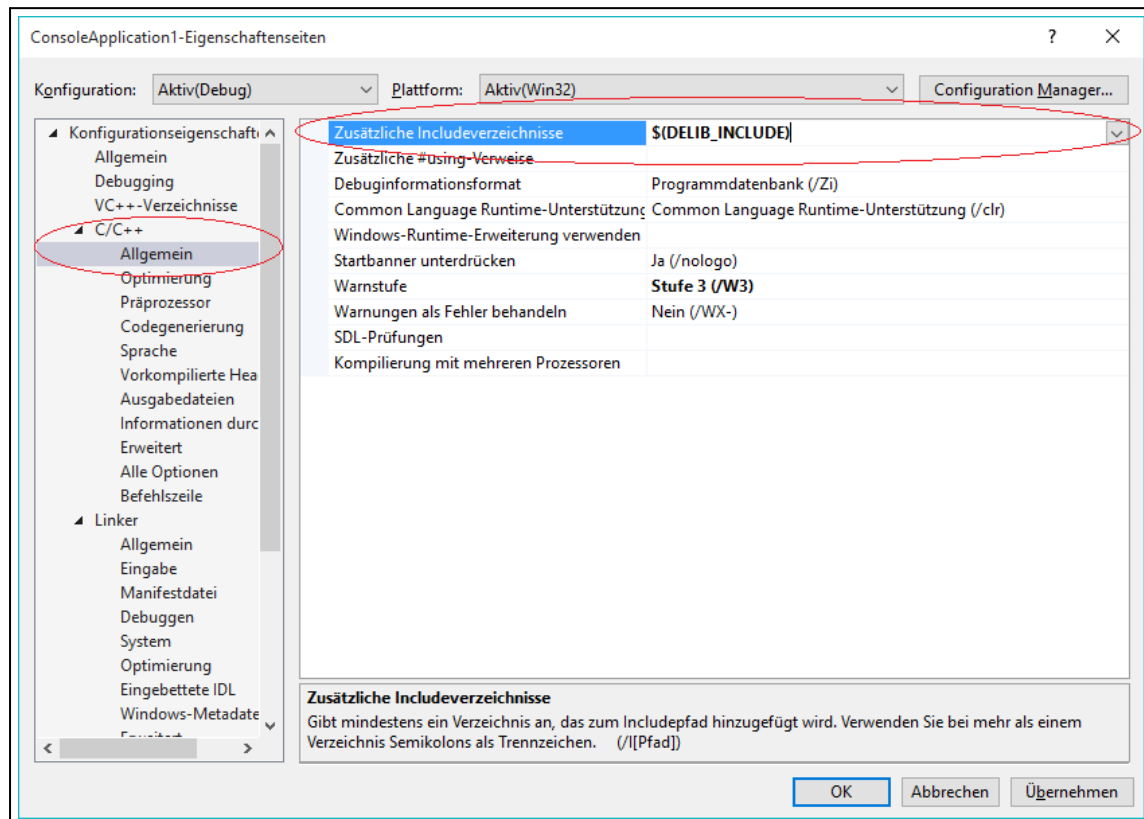
DELIB64_INCLUDE = C:\Programs\DEDITEC\DELIB64\include

These are entered in the project settings of the compiler in the following.
Visual-C/C++ Start and open in the menu "Project → Properties.



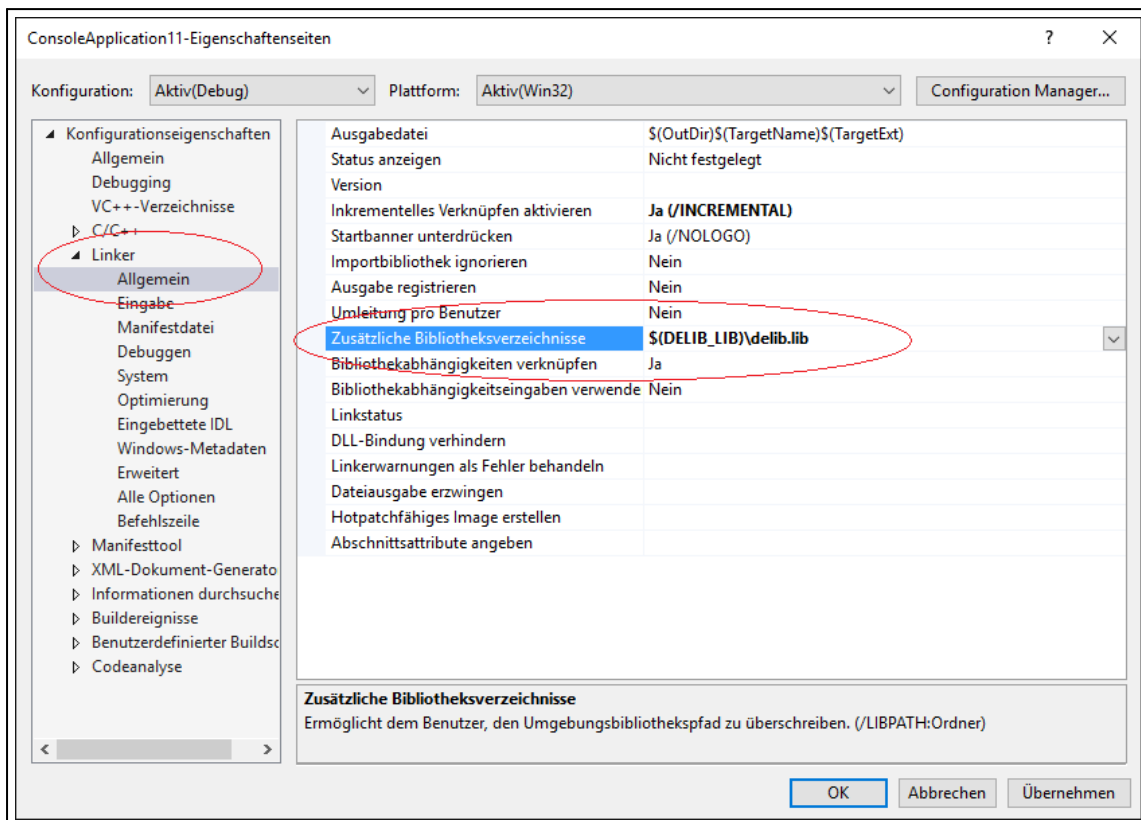
DELIB.H entry in the Visual-C/C++ project settings

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

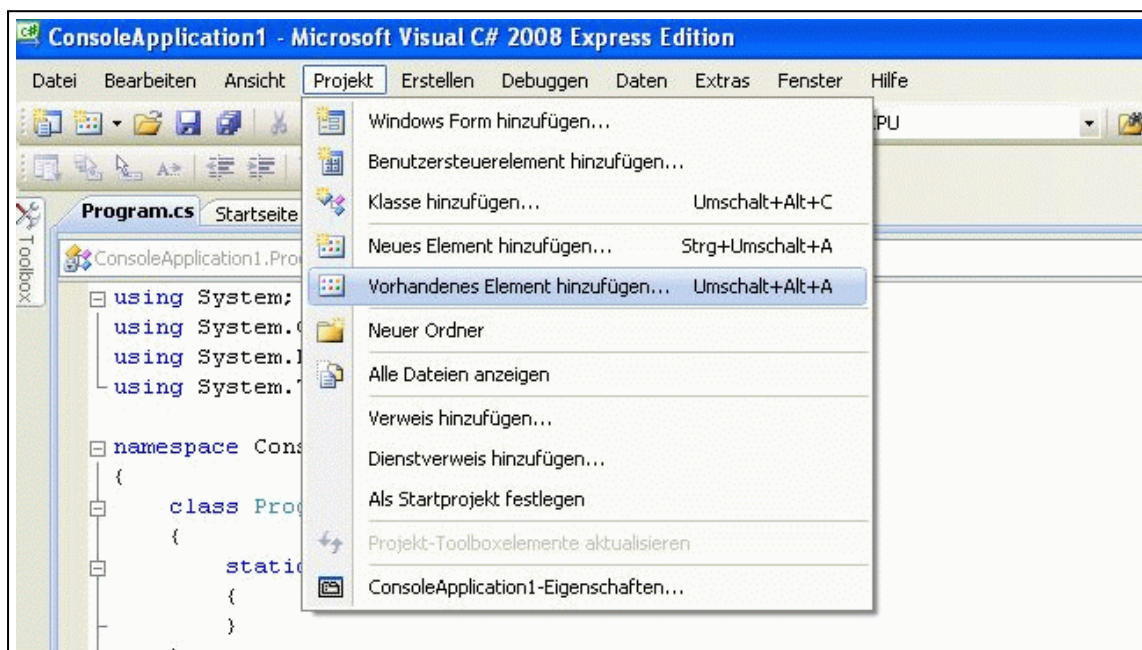
Under the "Linker" tab, enter "\$ (DELIB_LIB)\delib.lib" for "General".



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.



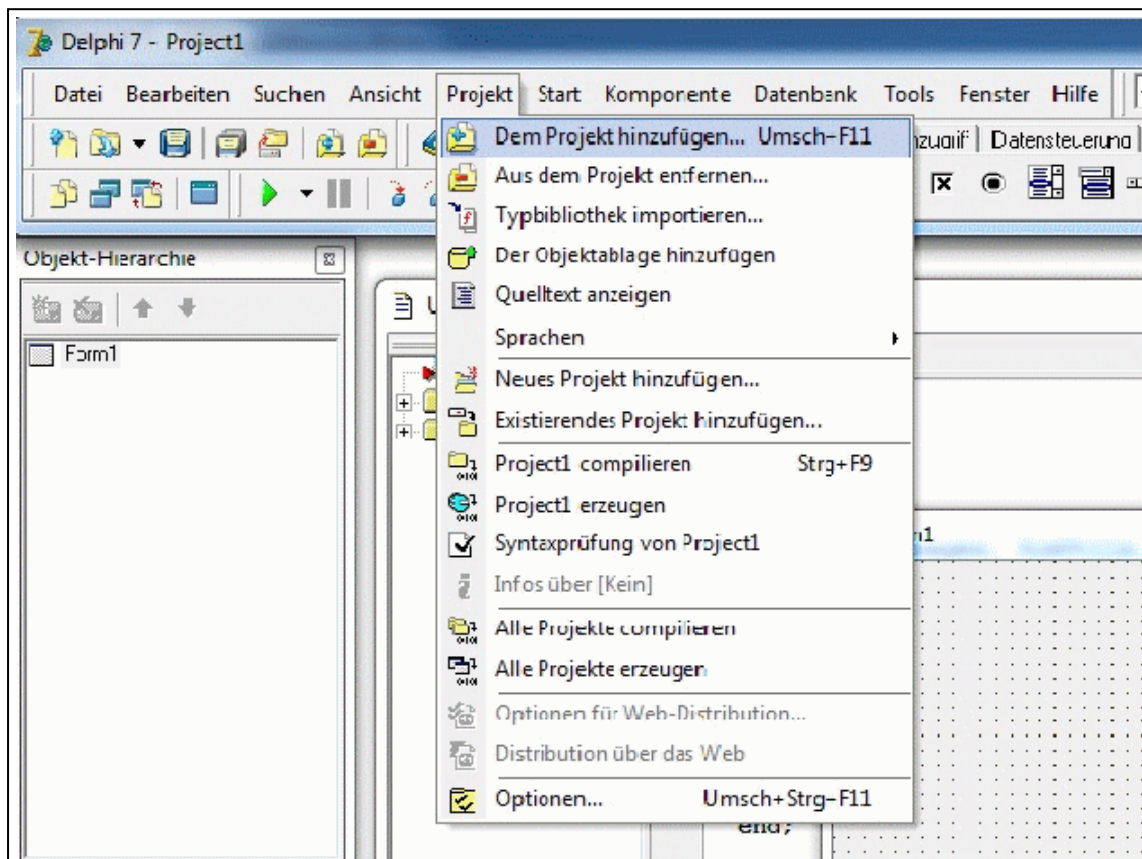
Add the following reference in your program:

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.

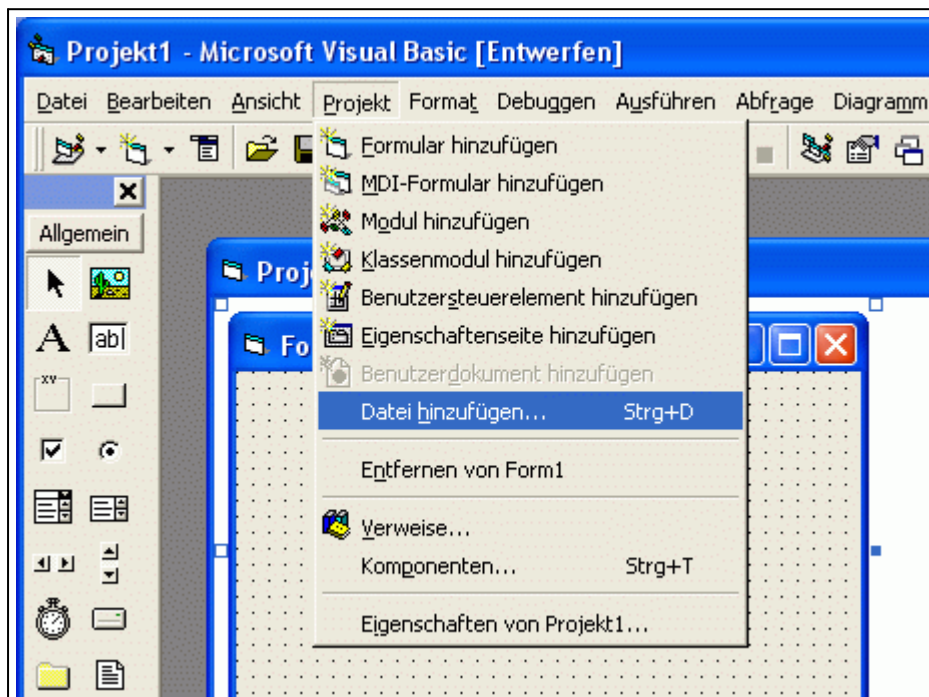
Start Delphi and use the menu "Project → Add to project" in the directory C:\Programs\DEDITEC\DELIB\include\ to open the file delib.pas for import.



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.

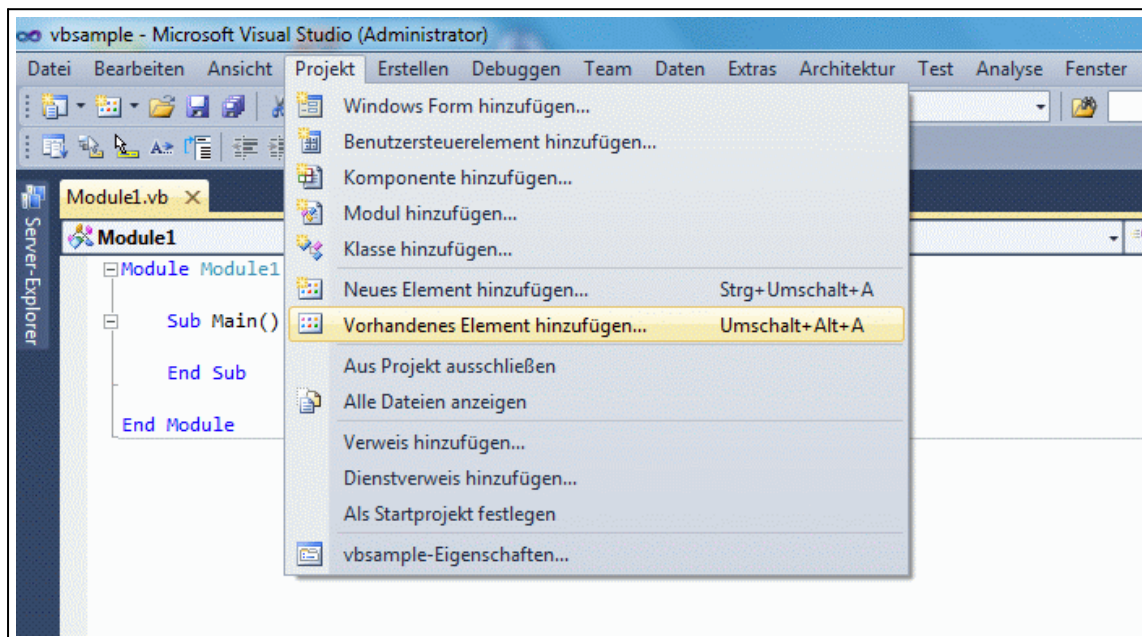


4.1.6.6. Embedding the DELIB in Visual-Basic.NET (VB.NET)

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

C:\Programme\DEDITEC\DELIB\include.

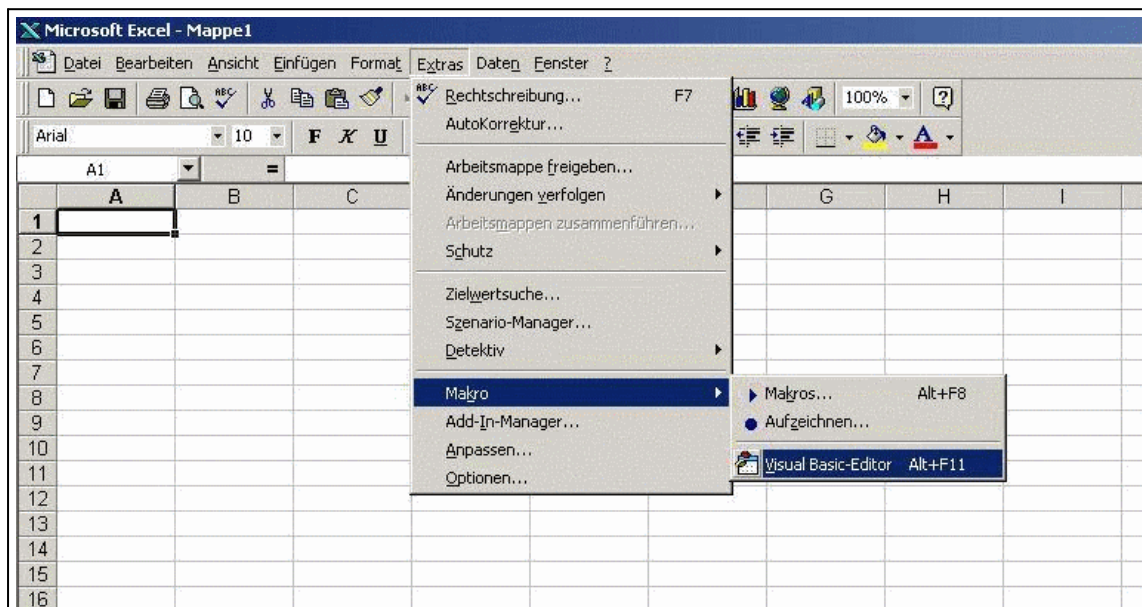
Start VB.NET and use the menu "Project → Add existing element" in the directory C:\Programme\DEDITEC\DELIB\include\ to open the file delib.vb for import.



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

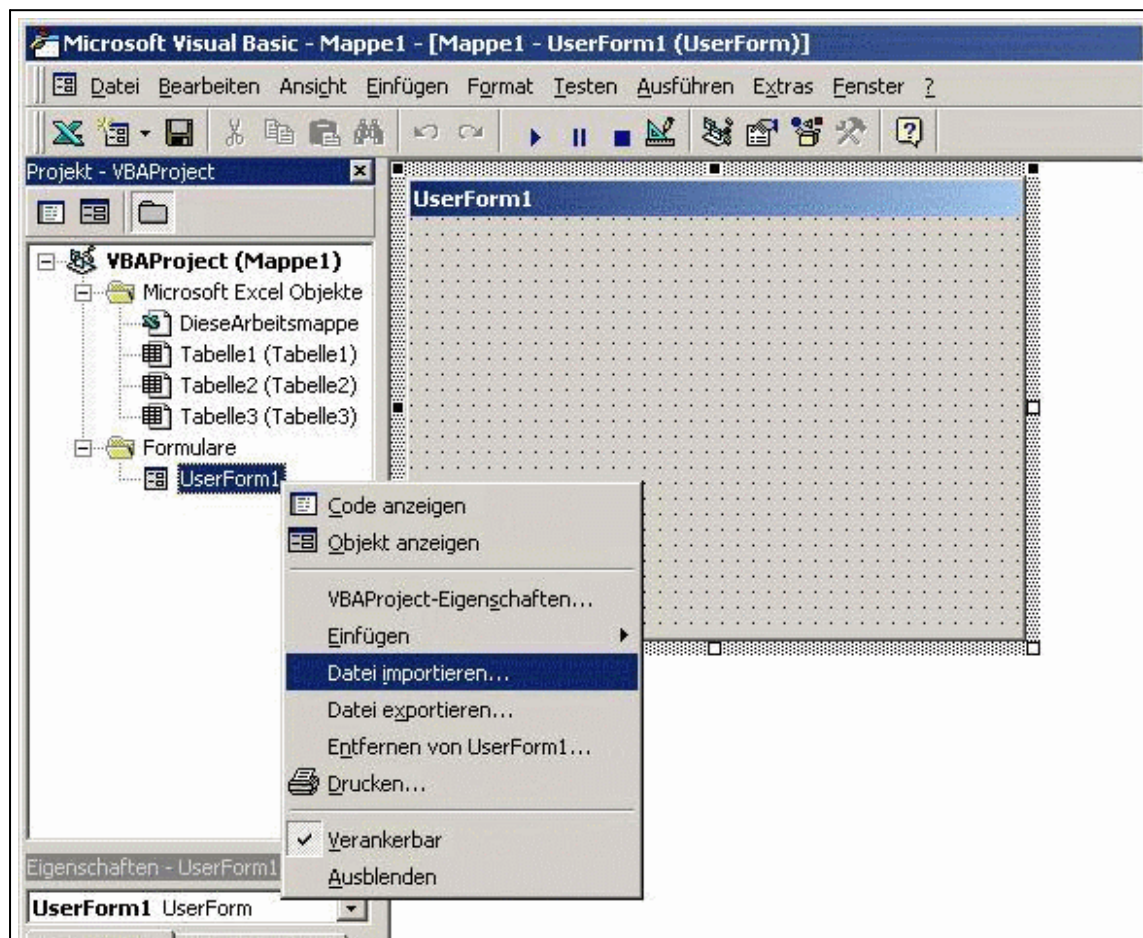
The required file for VBA is located in the directory
C:\Programs\DEDITEC\DELIB\include.

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.

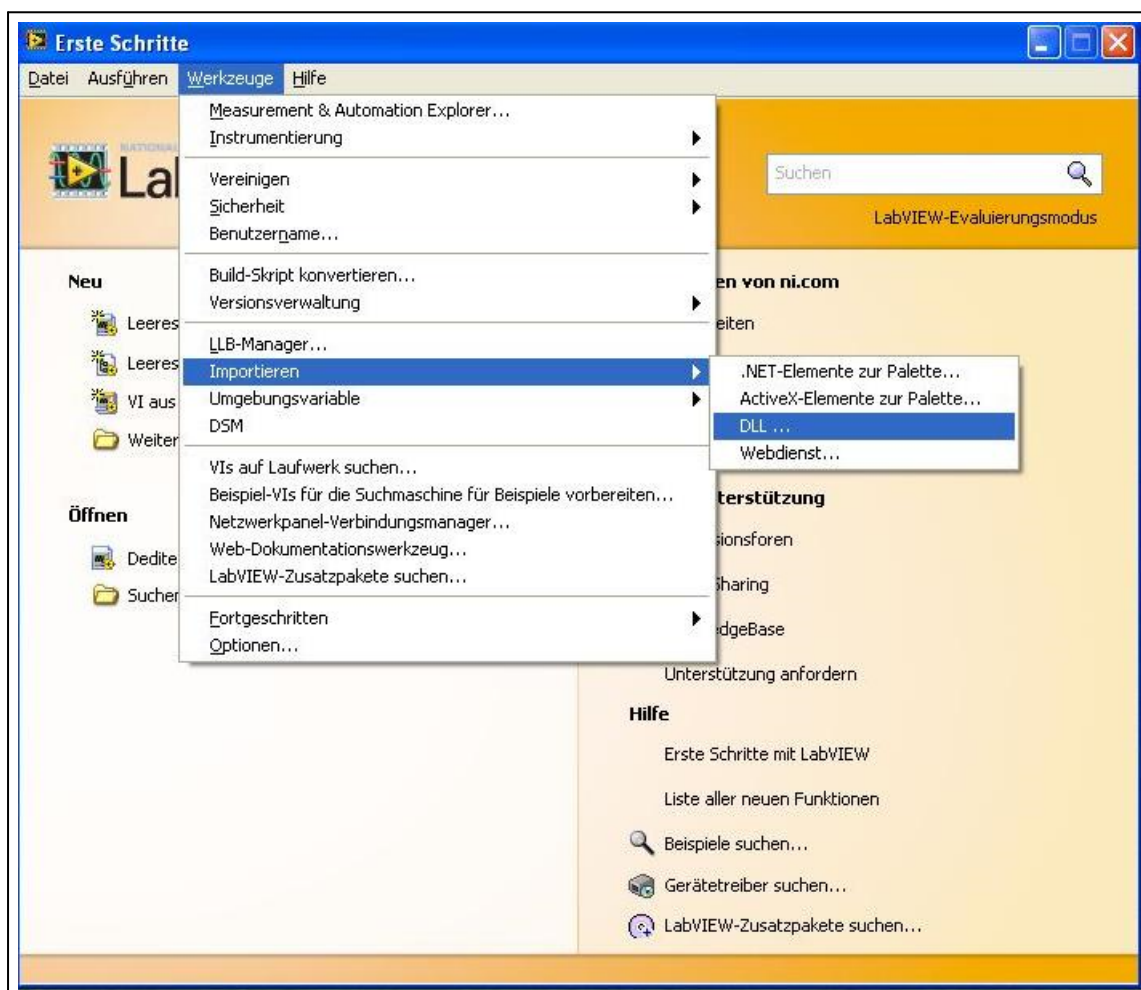
Description of the integration of the "delib.dll" in LabVIEW version 11

- 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".

DLL importieren

Erstellungs- oder Aktualisierungsmodus angeben

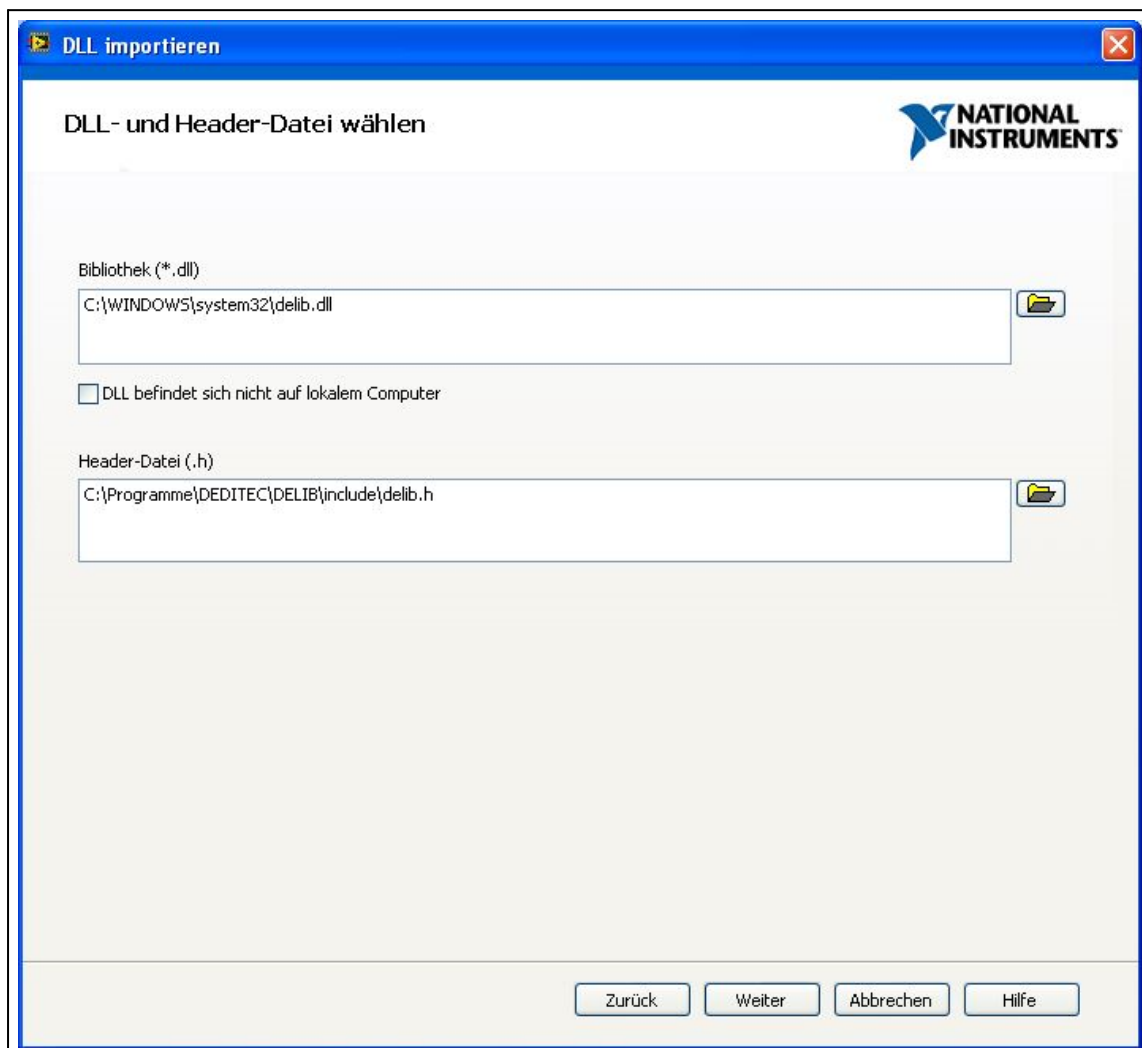
☒ VIs für DLL erstellen
Erzeugt VIs basierend auf der vorliegenden DLL- und Header-Datei.

☐ VIs für DLL ändern
Aktualisiert zuvor importierte VIs für die folgenden Projektbibliotheken

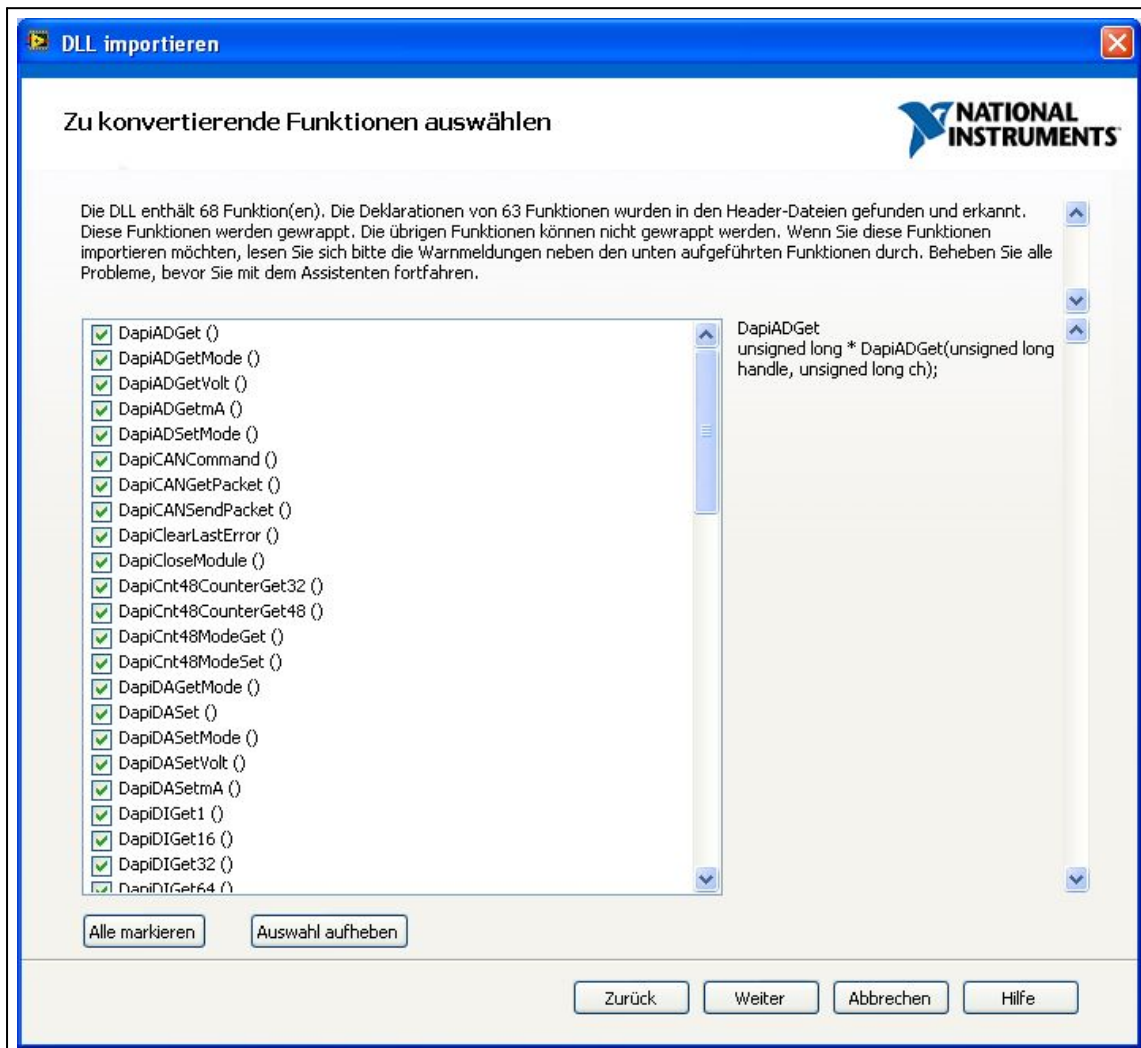
Projekt	DLL-Datei	Datum

Zurück Weiter Abbrechen Hilfe

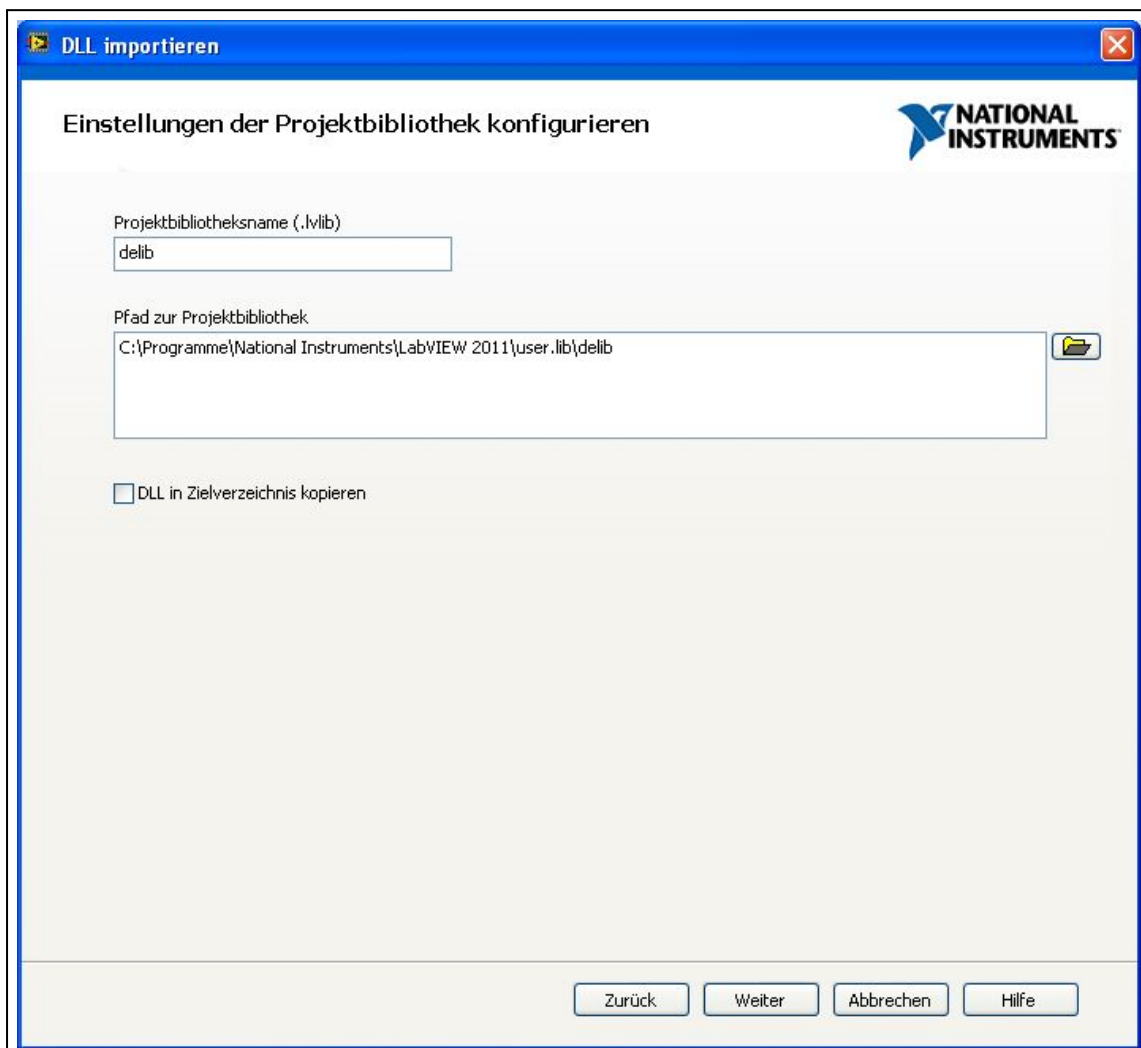
- In the next window, use the browser buttons to specify the location of the delib.dll and delib.h files and continue with "Next".



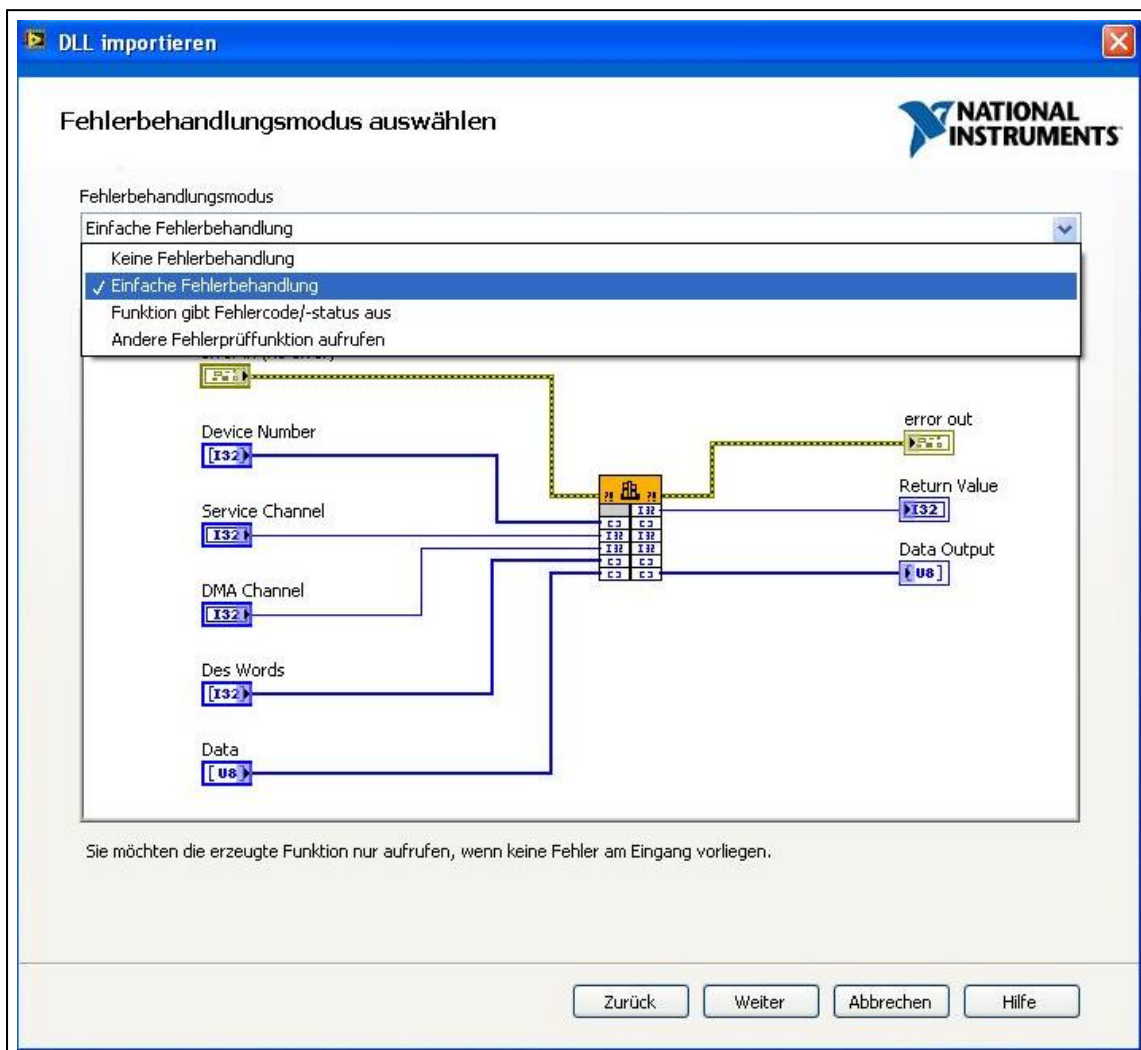
- Click on "Next" again to continue.
- The header file will now be analyzed. Then click "Next" again in the following window to continue.



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



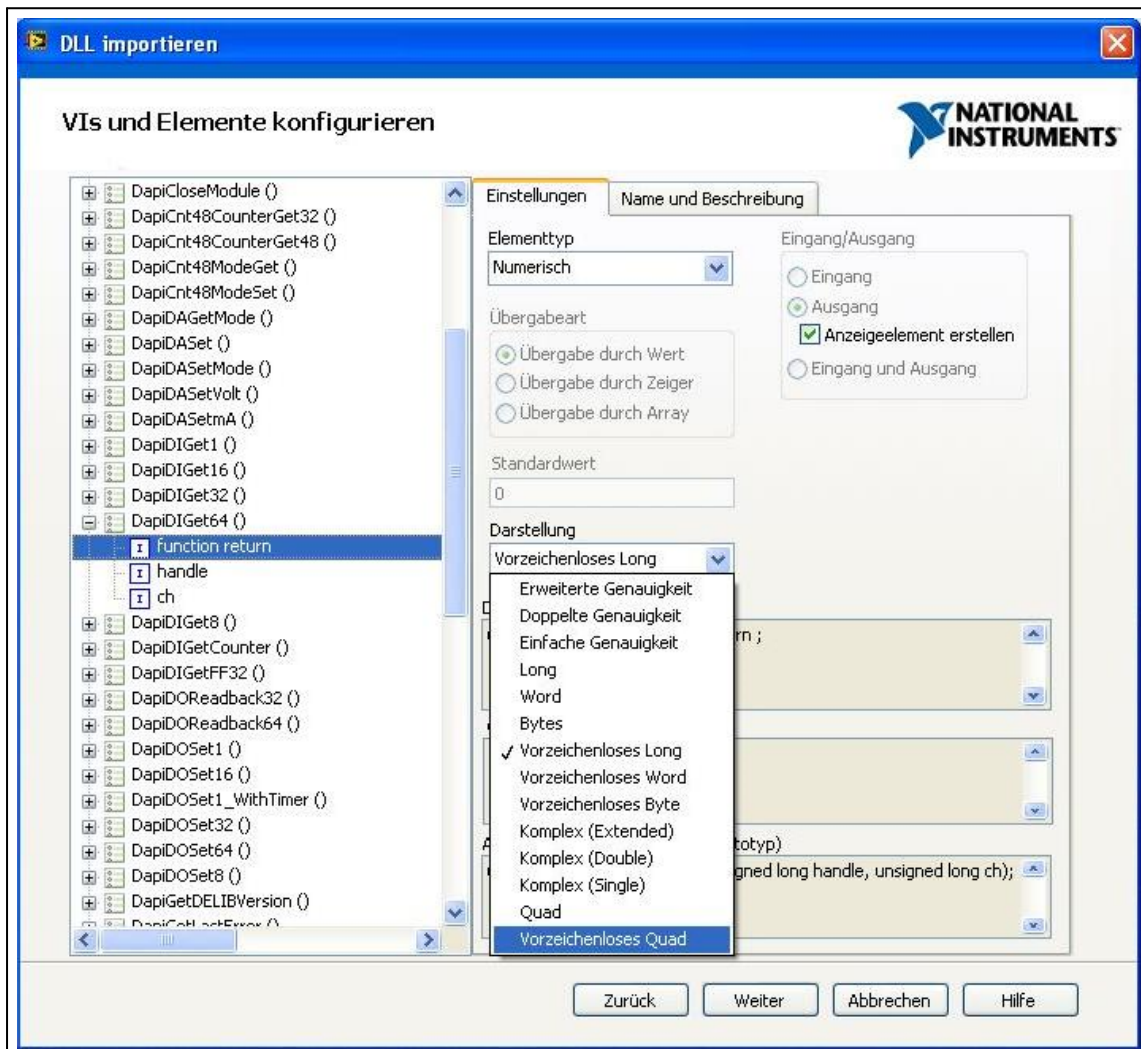
- In the following window, select the "Simple error handling" option from the drop-down menu and continue with "Next".



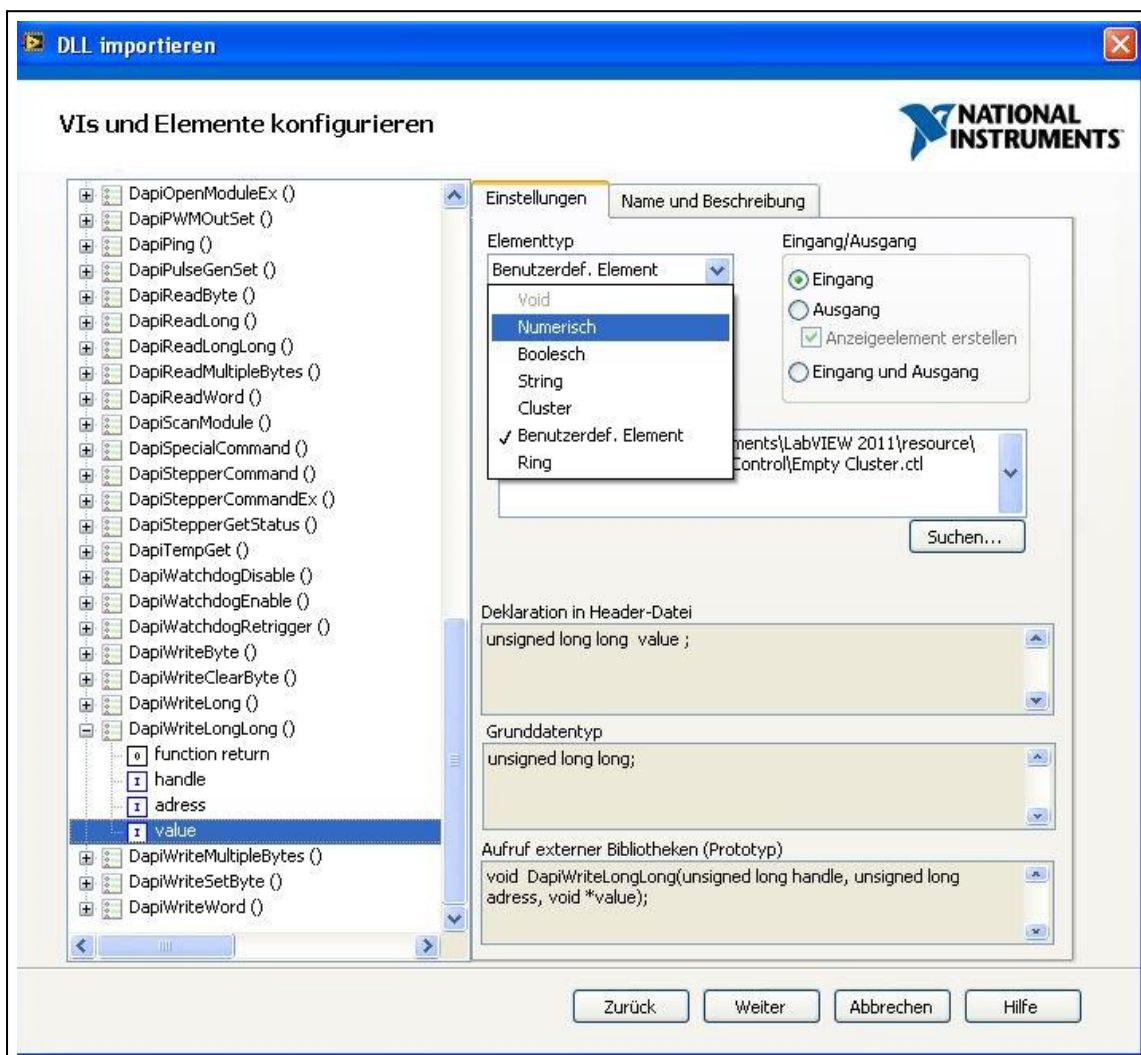
- For VIs that work with 64-bit values, the representation must be changed from "Unsigned Long" to "Unsigned Quad".

- The following VIs must be edited:

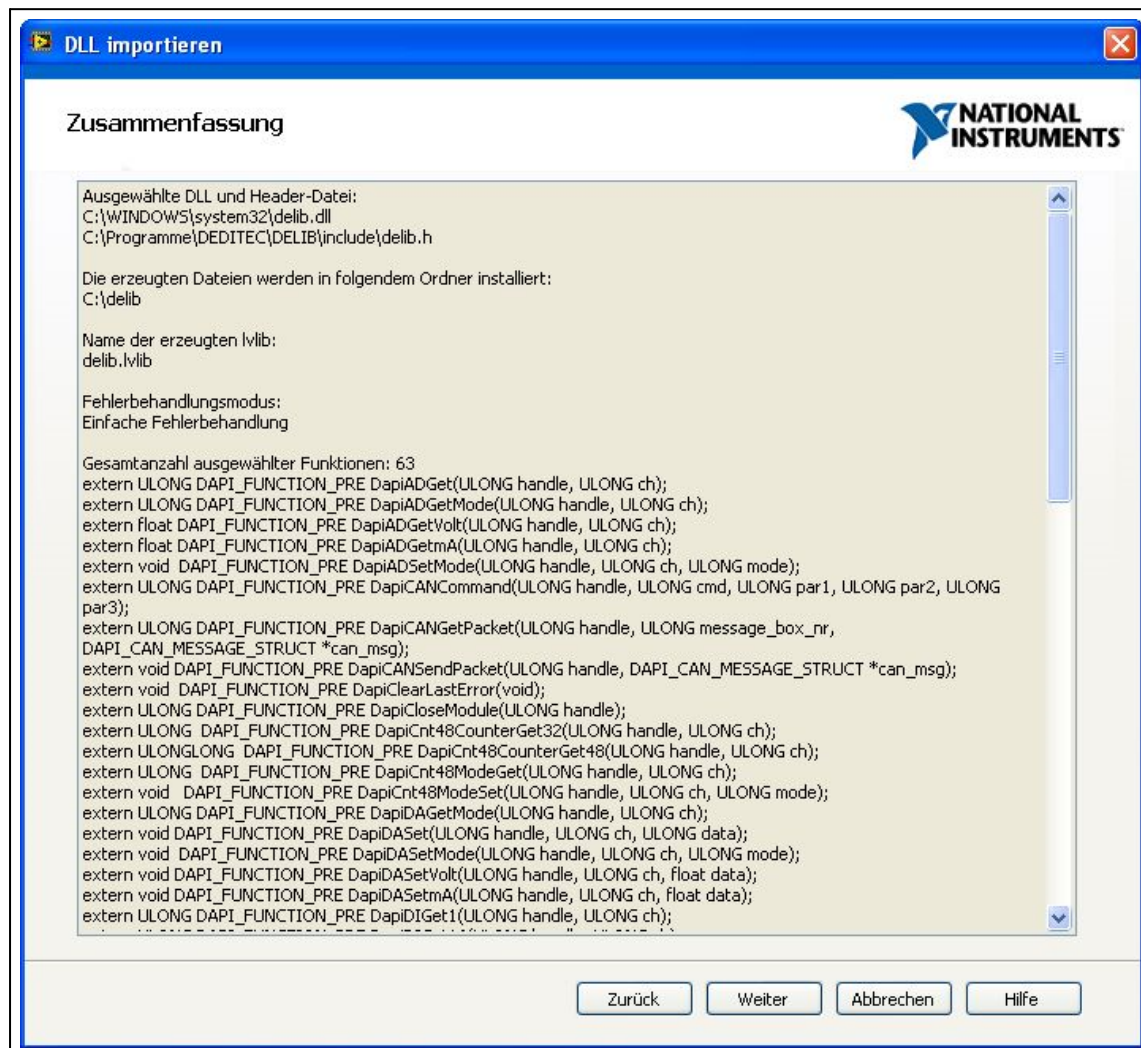
- DapiCNT48CounterGet48 (function return)
- DapiDIGet64 (function return)
- DapiDOSet64 (data)
- DapiDOReadBack64 (function return)



- 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);
```

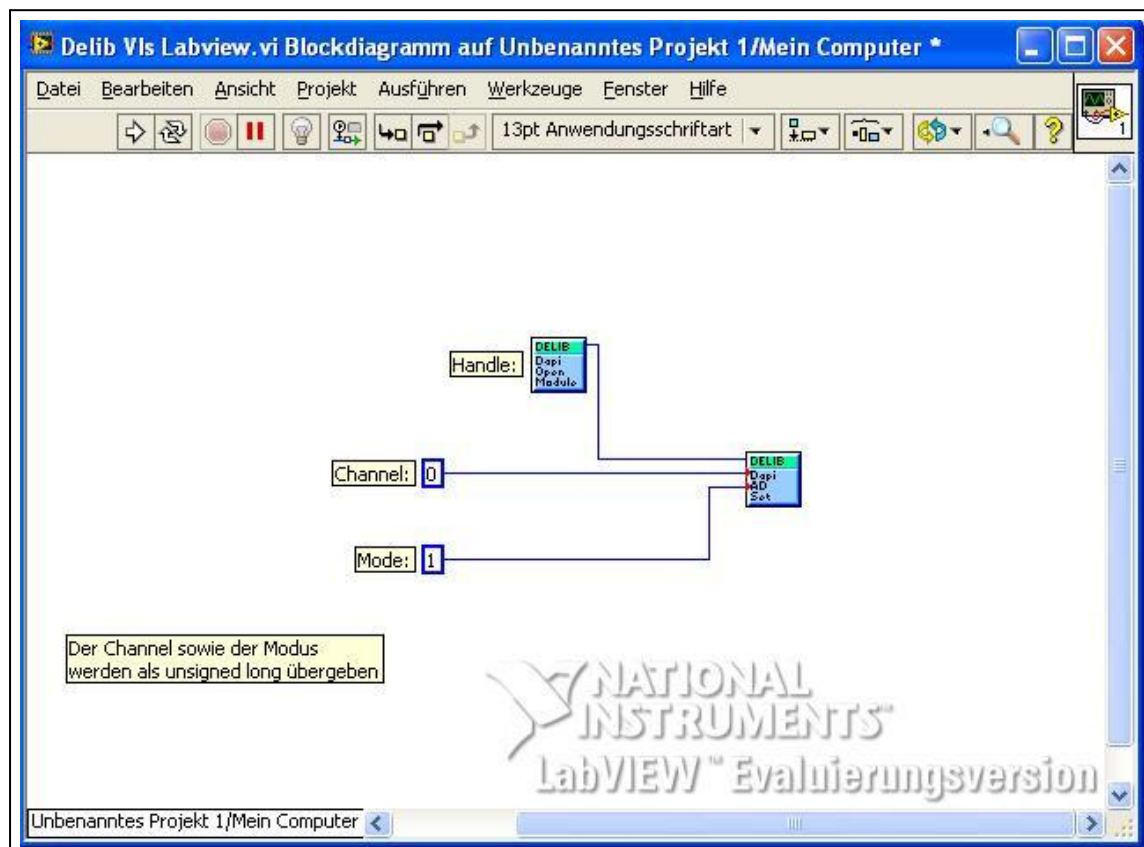
Alternatively, you can use the following notation:

```
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
```

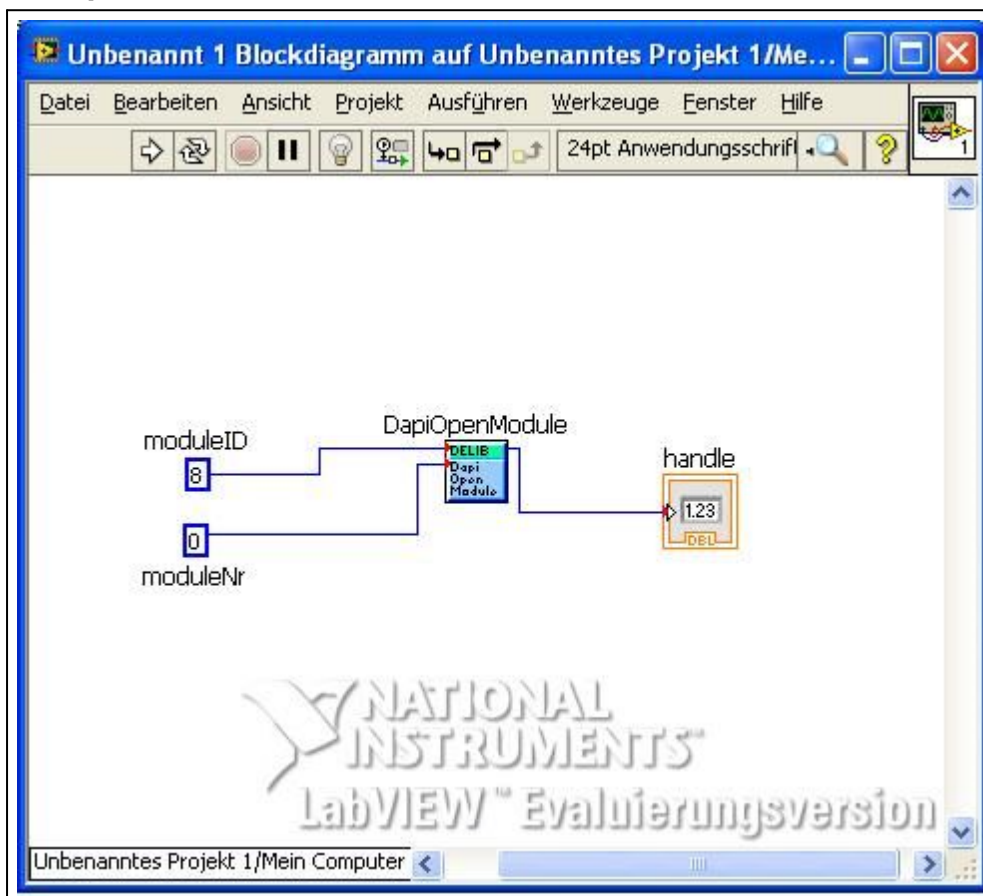
no. 0.

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

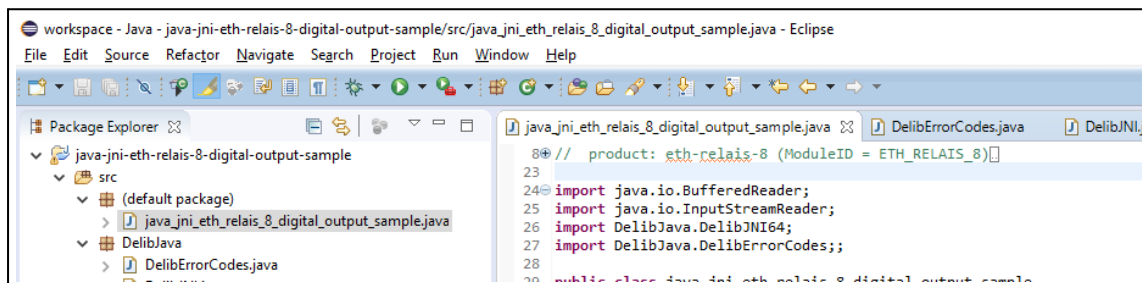
The required files for Java are located in the following directory, depending on the DELIB installation

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

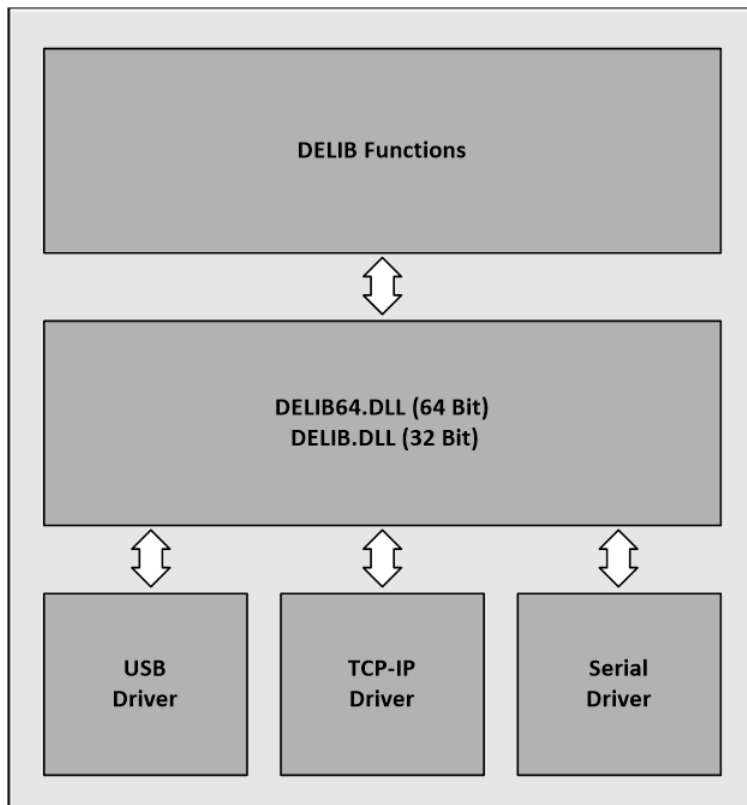
The DELIB driver library contains the DELIB API and various programs for the configuration test of our products.

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

In the chapter DELIB API Reference 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

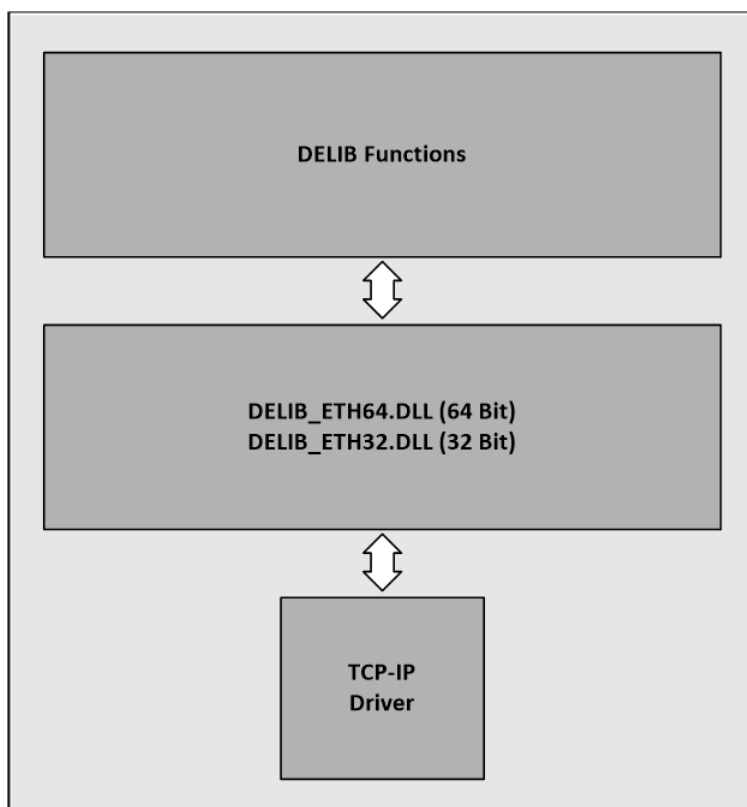
While the DELIB is available for ALL products, the DELIB-ETH does not access any other drivers (like USB).

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.

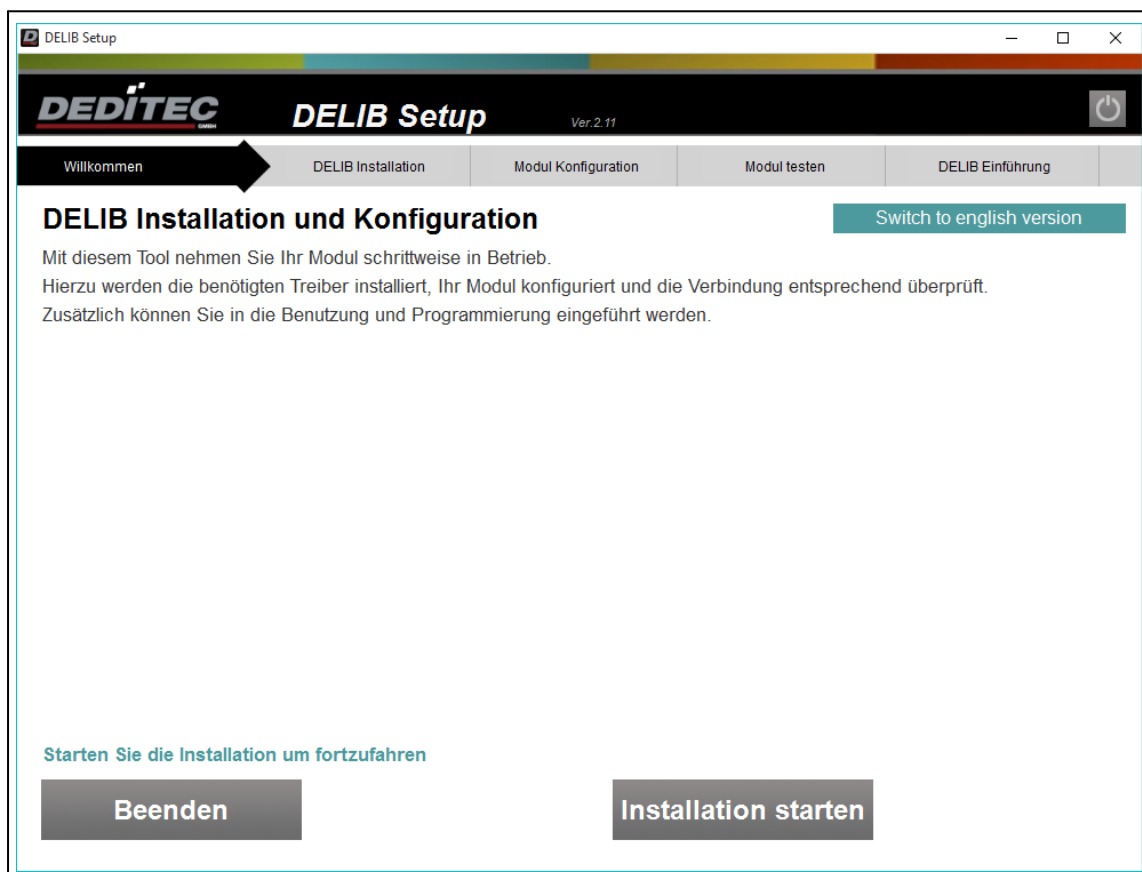
4.2.2. DELIB Setup

The DELIB Setup guides you through the installation of our DELIB driver library. Afterwards you will be guided through the configuration process as well as functional tests for our different products.

The current version of the DELIB Setup is available for download on our homepage.

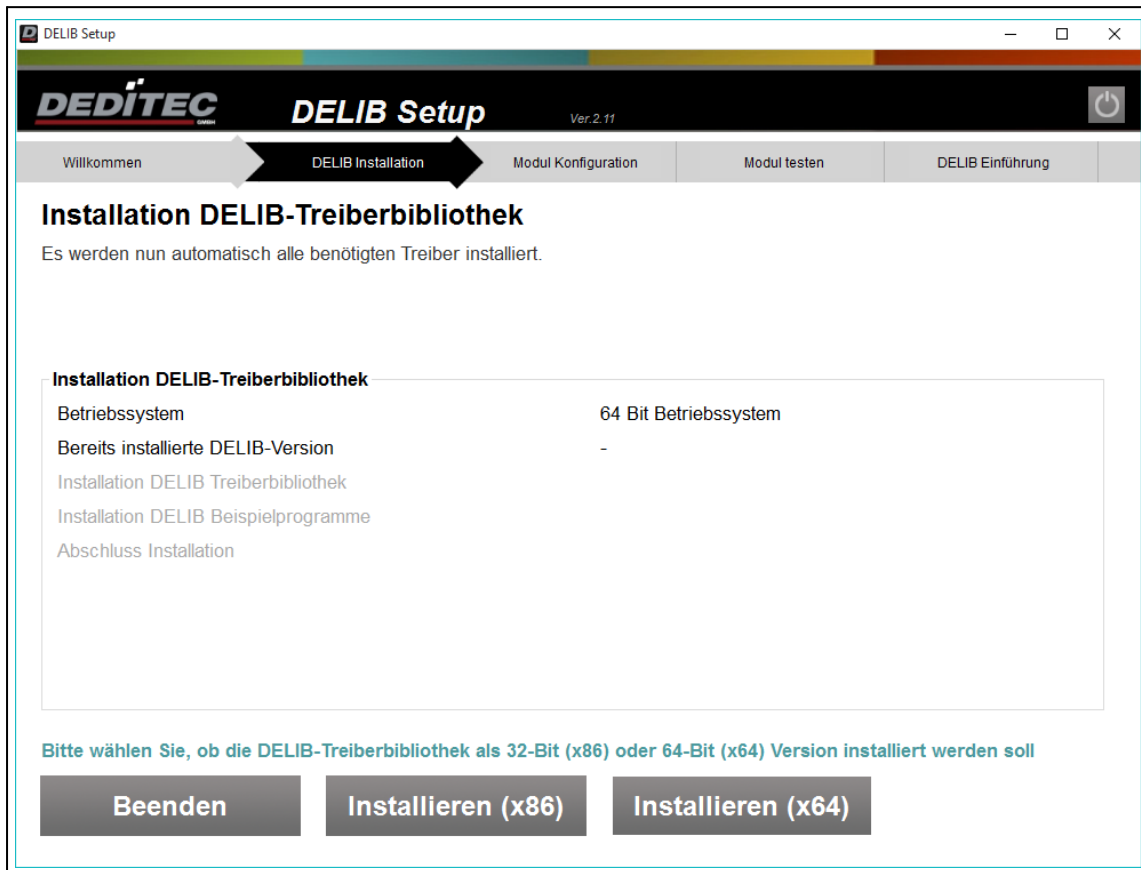
Link: <https://www.deditec.de/delib>

The DELIB Setup guides you step by step through the installation of the DELIB driver library including configuration and commissioning of the products.

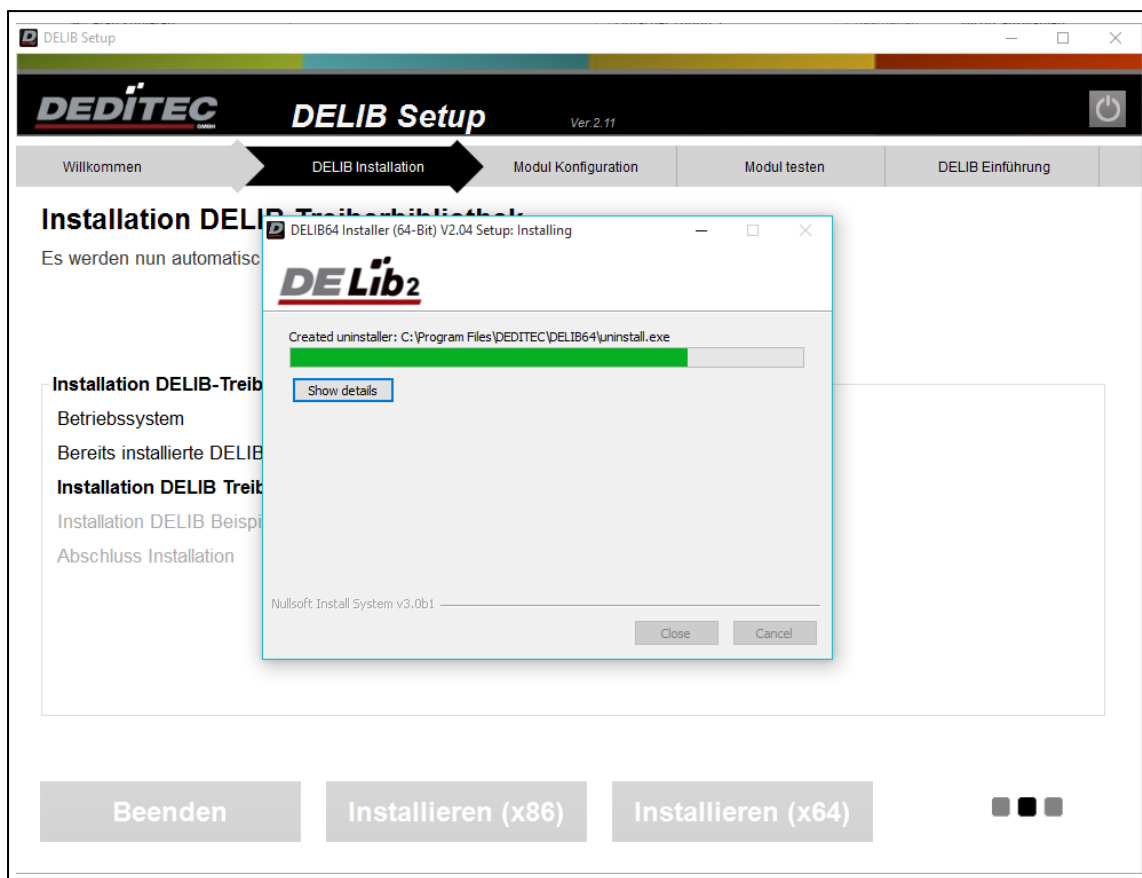


The DELIB setup checks the operating system and whether versions of the DELIB driver library are already installed.

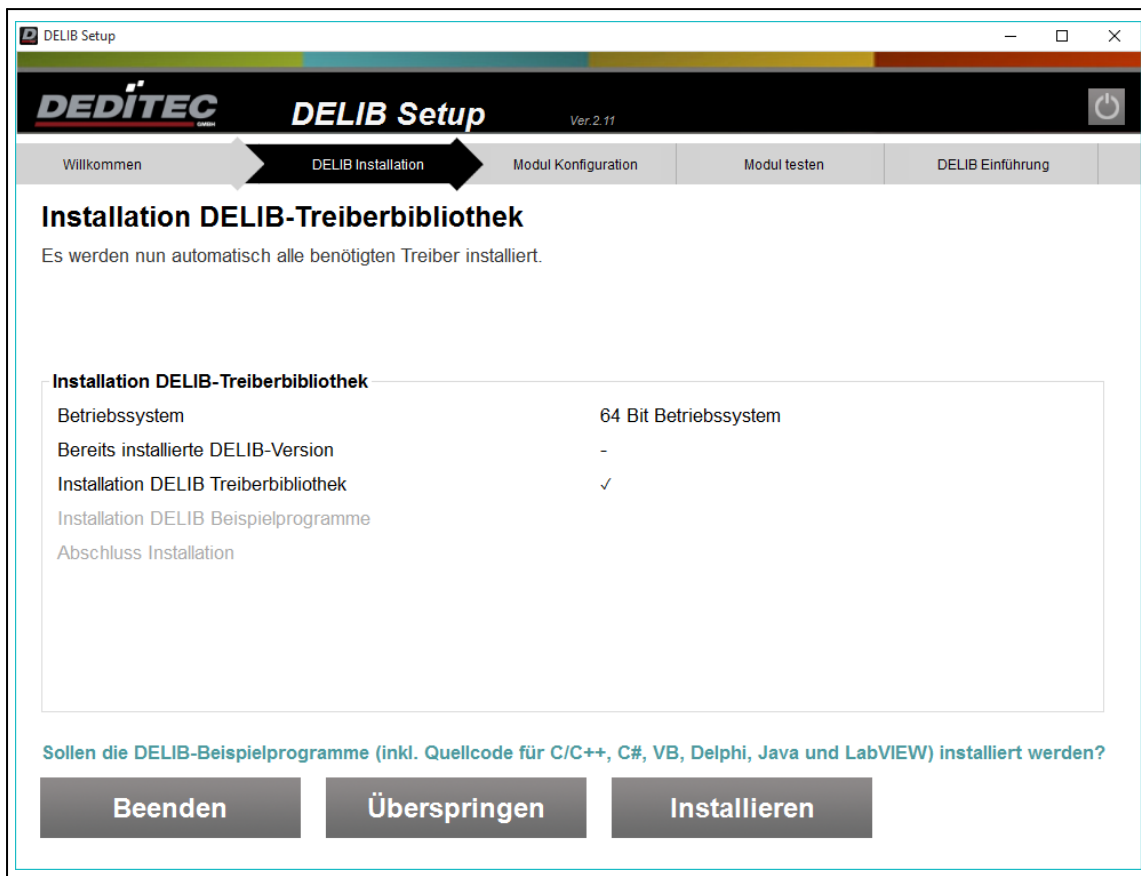
You can then choose whether to install the 32-bit or 64-bit version of the DELIB driver library.



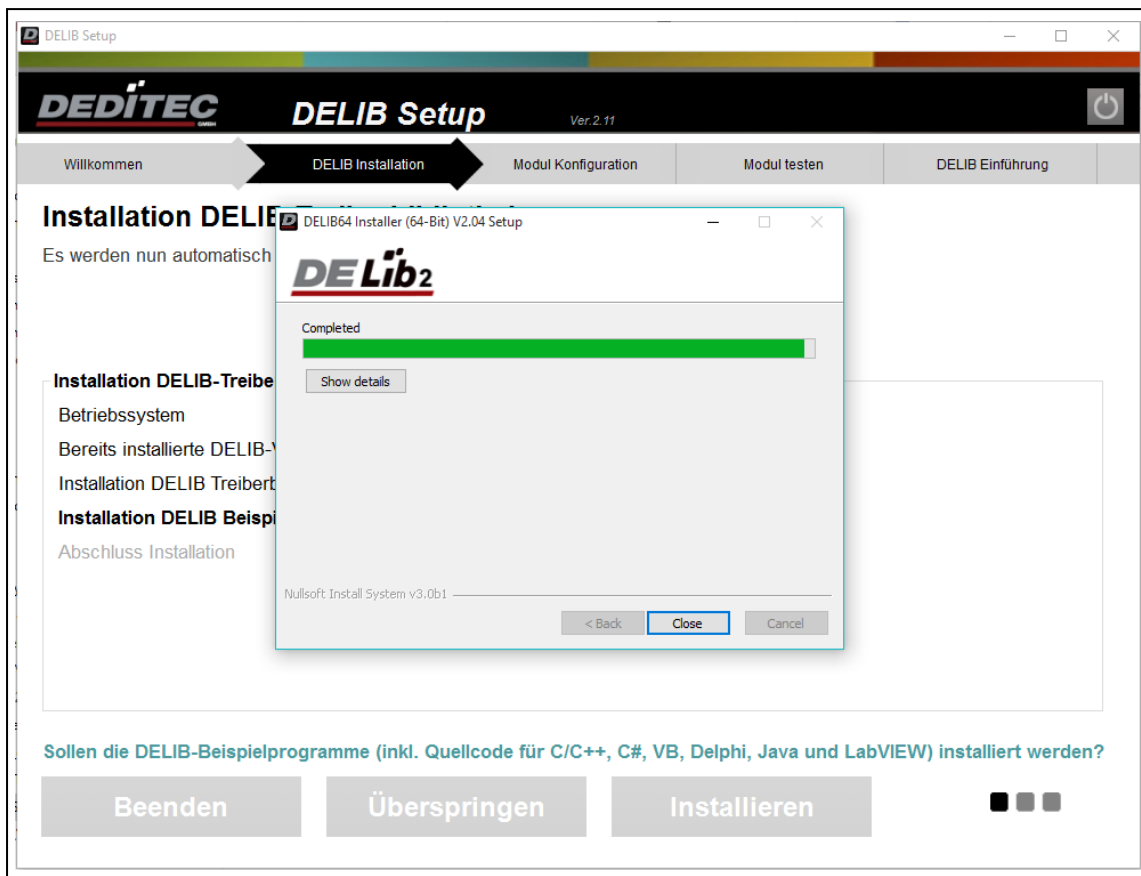
Installation progress of the driver library.



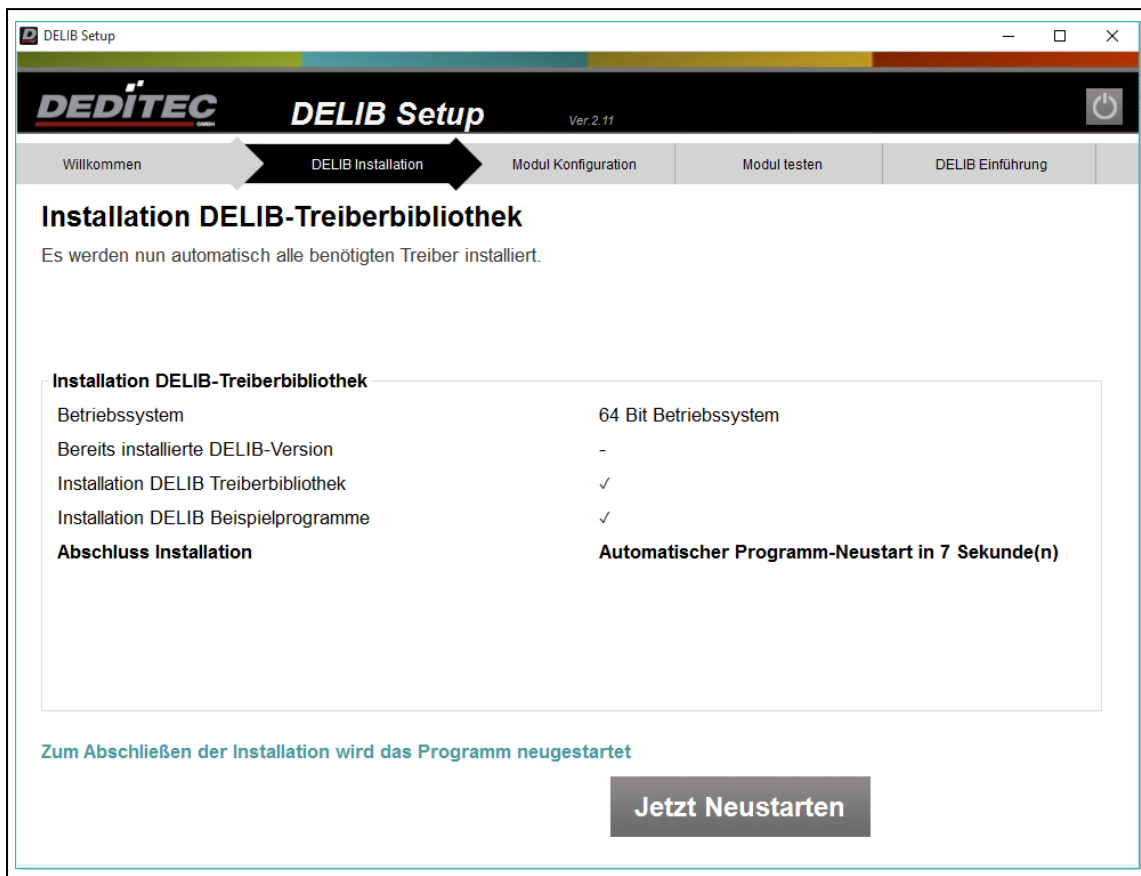
Additionally, you can choose to install the DELIB sample programs.



Installation progress of the DELIB sample programs.



The program is restarted to complete the installation. In the next step, the DELIB Configuration Utility is used to configure and test the product.



4.2.3. DELIB Configuration Utility

4.2.3.1. Introduction

The DELIB Configuration Utility allows the configuration of the Ethernet, CAN or serial interface of a product.

The configuration is required for the first commissioning.

Products with USB interface are excluded. These only have to be configured if you want to operate several identical USB products on one PC.

The DELIB Configuration Utility is included in the installation of the DELIB driver library.

Default path:

32-bit: C:\Program Files (x86)\DEDITEC\DELIB\programs\delib-configuration-utility.exe

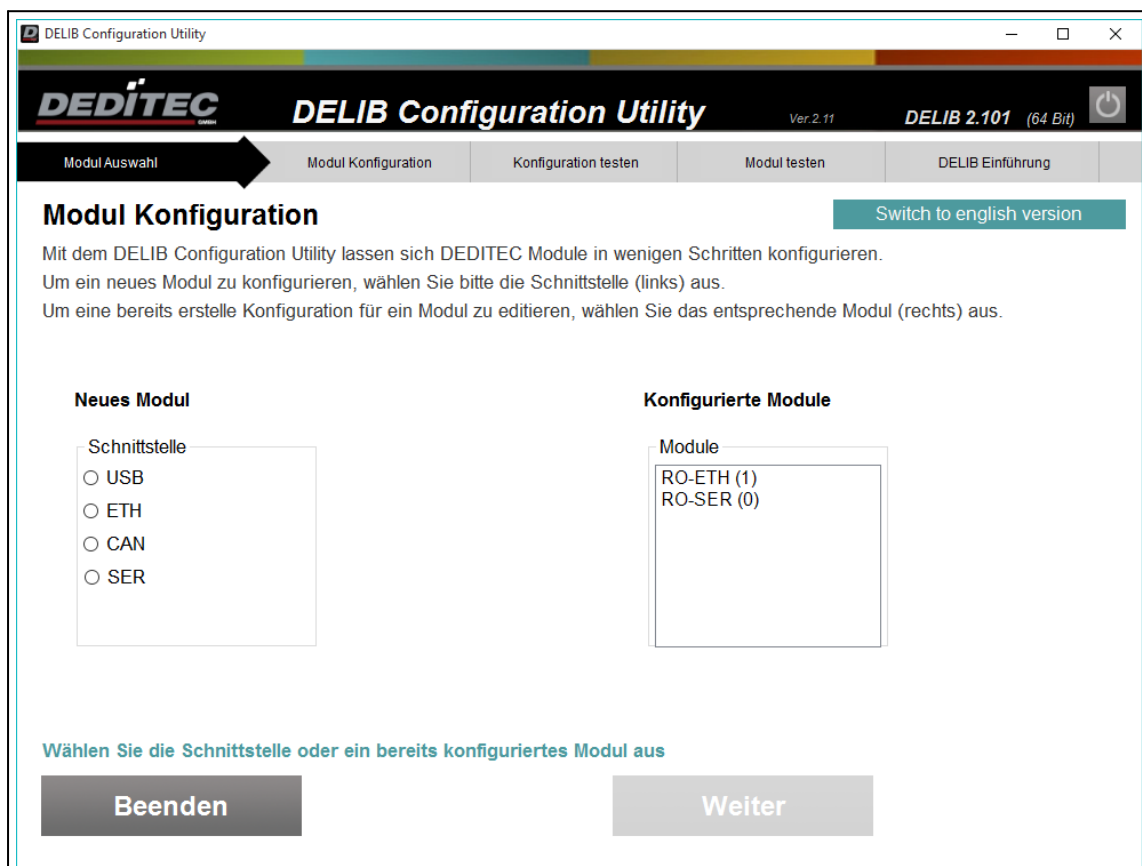
64-bit: C:\Program Files\DEDITEC\DELIB64\programs\delib-configuration-utility_x64.exe.

You can also open the DELIB Configuration Utility from the Start menu under "All Programs" → "DEDITEC" → "DELIB Configuration Utility".

4.2.3.2. Create or edit configuration

For a new configuration, select the desired interface in the left selection box under "New module".

If you want to edit an existing configuration, you will find the selection box of the existing configurations on the right.

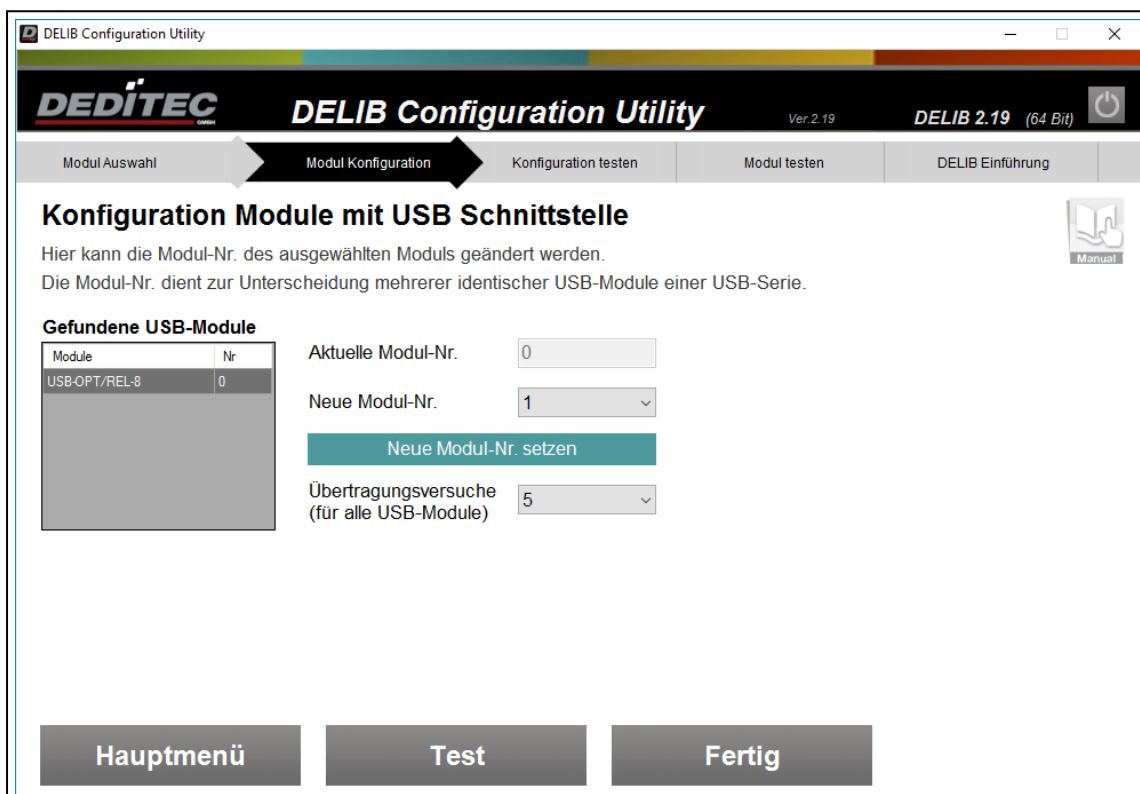


4.2.3.2.1. Module configuration USB

The configuration of USB modules is only necessary to use several modules of a USB product family (e.g. 2x USB-RELAIS-8) in one system.

If there is only one USB module, or several USB modules from different product families (e.g. RO-USB-016 and USB-RELAIS-8) in the system, no configuration is necessary, because the products are clearly identifiable via the module ID.

After clicking on the product you want to configure, all possible settings are displayed on the right.



Module	Nr
USB-OPT/REL-8	0

Aktuelle Modul-Nr. 0

Neue Modul-Nr. 1

Übertragungsversuche (für alle USB-Module) 5

Hauptmenü Test Fertig

The "Current module no." refers to the module number stored in the module. This number is used for identification and must be configured differently for identical USB products.

With the item "New module no." a new number between 0 and 255 can be assigned to the product. In the delivery state all products have the module number 0.

With "Set new module no." the currently selected new module number is written to the module.

Via the selection box "Transmission attempts (for all USB modules)" you can define how often the DELIB tries to communicate with the module in case of an error.

4.2.3.2.1.1. Example for the configuration of identical USB modules

To be able to use several identical USB modules (USB modules with the same module ID) in a system, each module must be assigned a unique module no. using the DELIB Configuration Utility.

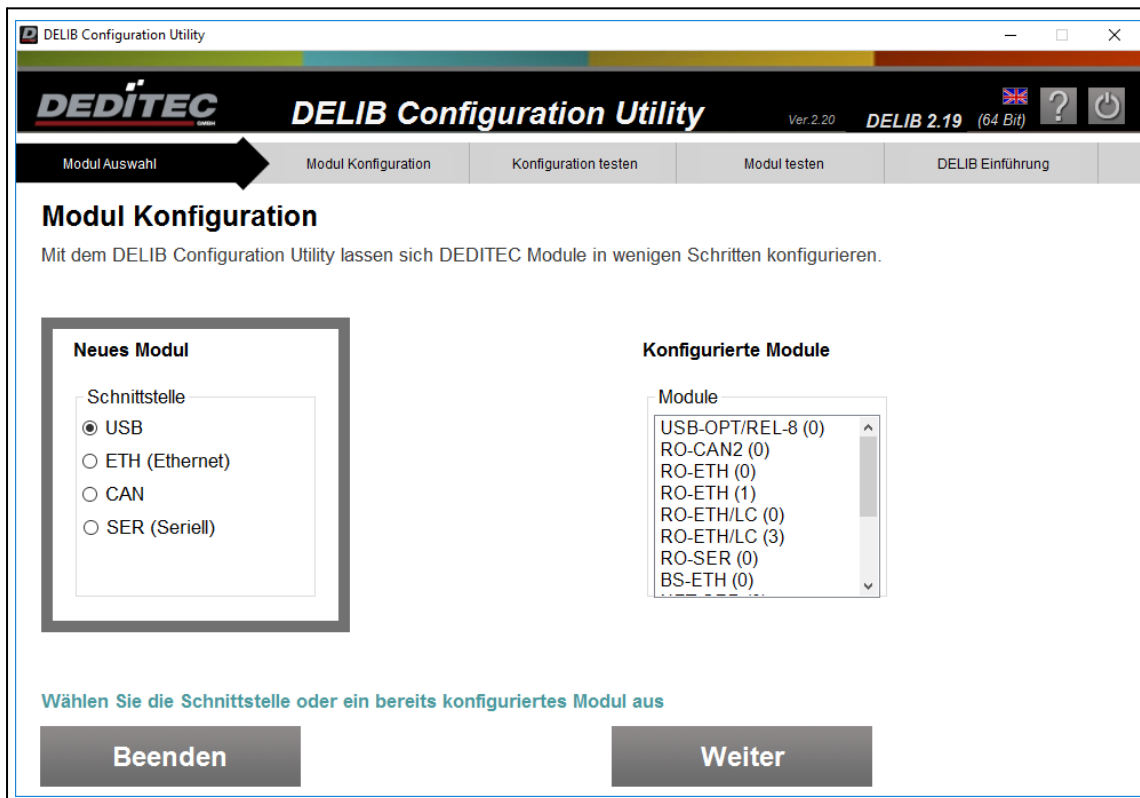
If there is only one USB module, or several USB modules with different module ID (e.g. RO-USB-016 and USB-RELAIS-8) in the system, no configuration is necessary, because the products are uniquely identifiable via the ID.

The following example shows the configuration of two USB-OPTOIN-8 modules in the same system.

Step 1

First connect only one USB-OPTOIN-8 to the PC and start the DELIB Configuration Utility.

Select the USB interface in the left area and click on "Next".



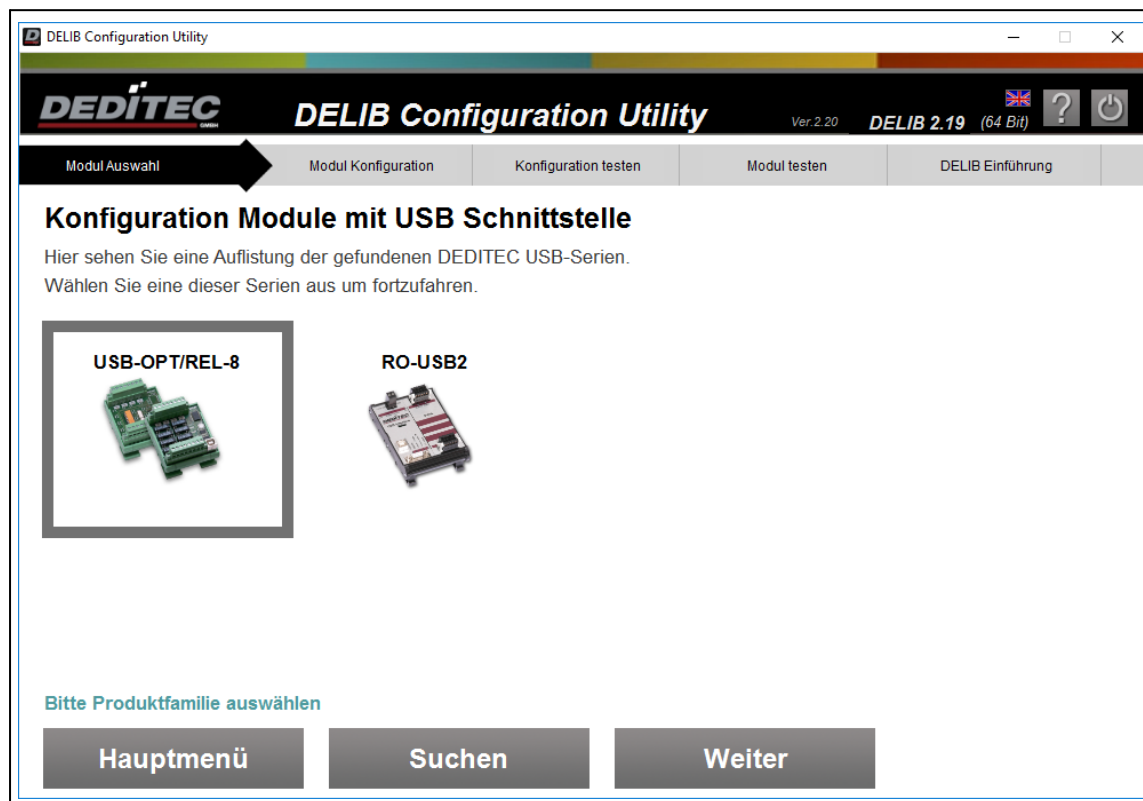
Step 2

If several USB modules of different DEDITEC USB series are connected, the

corresponding product family must be selected in this step.

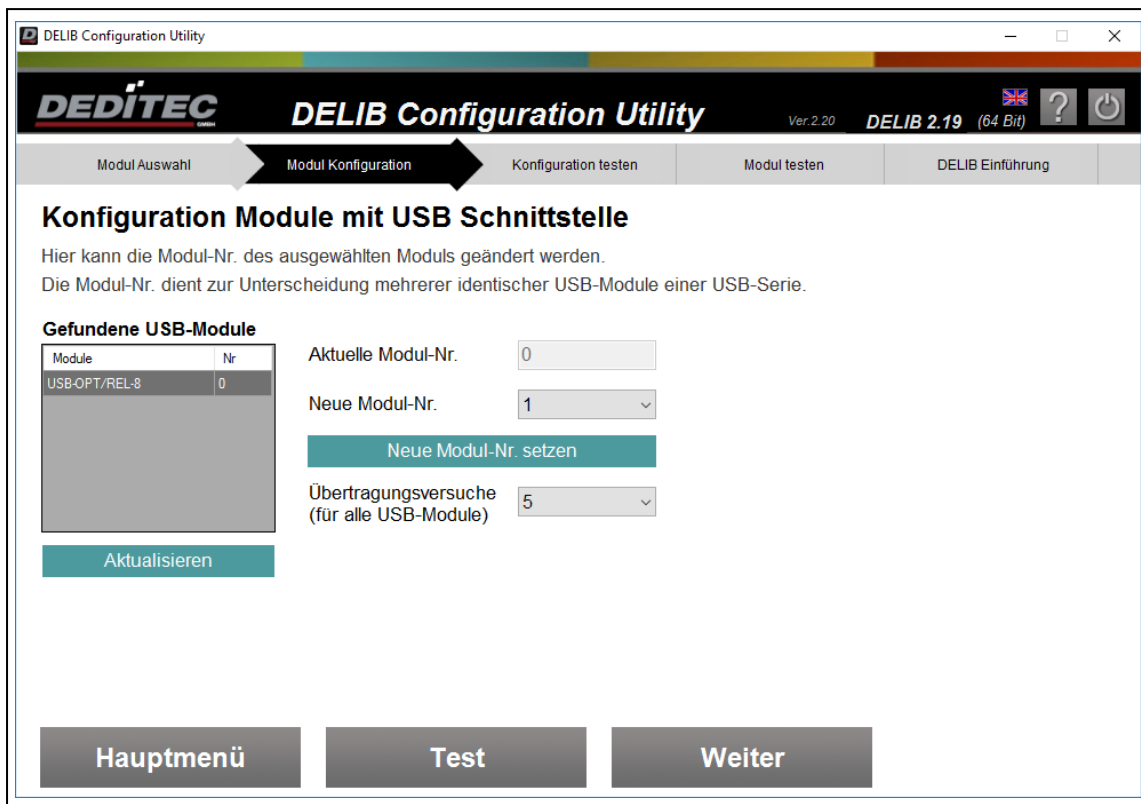
In this example, modules of the RO-USB2 series and USB-OPT/REL-8 series are connected.

This step is omitted if the connected modules belong to the same series.



Step 3

1. Select the corresponding USB module.
2. Change the New module no. to "1". In delivery state this number is already predefined with "0".
3. With Set new module no. the new module no. is stored in the module.

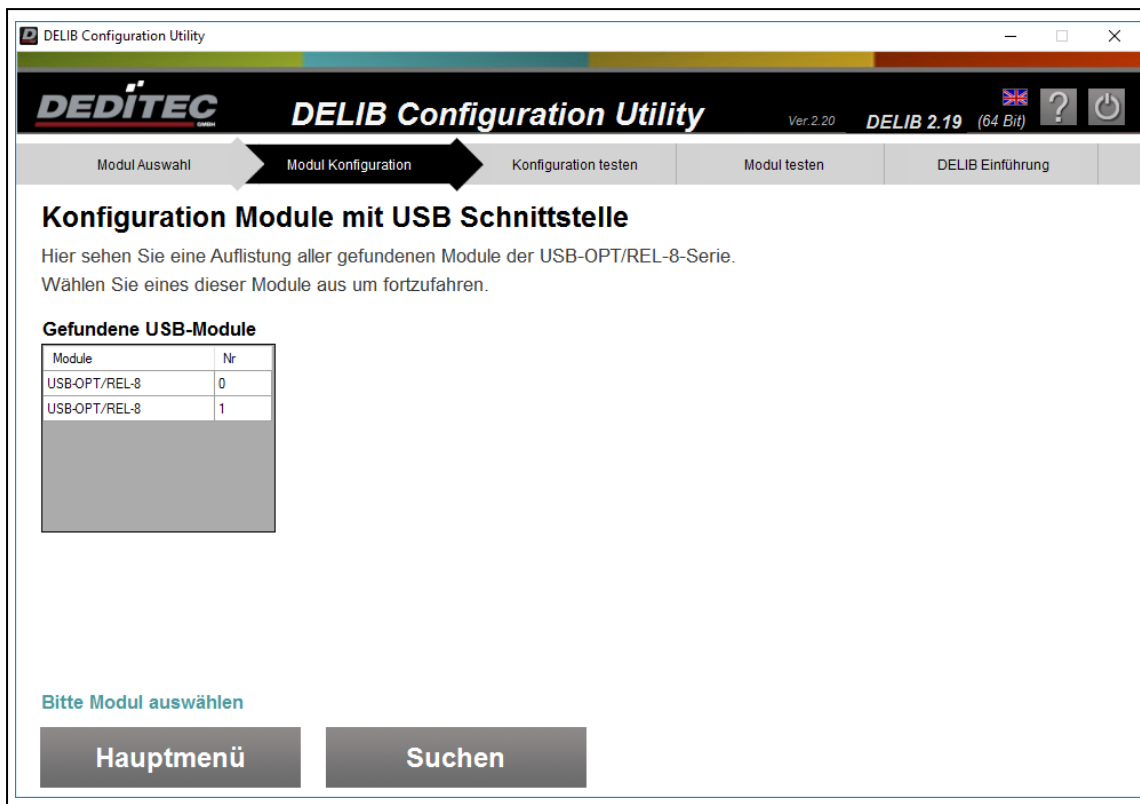


Step 4

Now additionally connect the second USB-OPTOIN-8 to the PC.

Since the module no. is already predefined with "0" in the delivery state and thus different from the module no. of the first module (module no. = 1), both modules are configured and ready for operation.

With Update both modules are now displayed.



Step 5

In the following you will find notes, what has to be considered when programming the two modules.

All modules are opened uniformly with the command `DapiOpenModule`. This command is defined as follows:

```
ULONG DapiOpenModule(ULONG moduleID, ULONG nr);
```

Addressing the USB-OPTOIN-8 with the NR 0

```
ulong handle;
handle = DapiOpenModule(USB_OPTOIN_8, 0);
//öffnet das Modul USB-OPTOIN-8 mit der NR 0
```

Addressing the USB-OPTOIN-8 with the NR 1

```
ulong handle;
handle = DapiOpenModule(USB_OPTOIN_8, 1);
//öffnet das Modul USB-OPTOIN-8 mit der NR 1
```

Notice:

All modules have the NR "0" as factory setting.

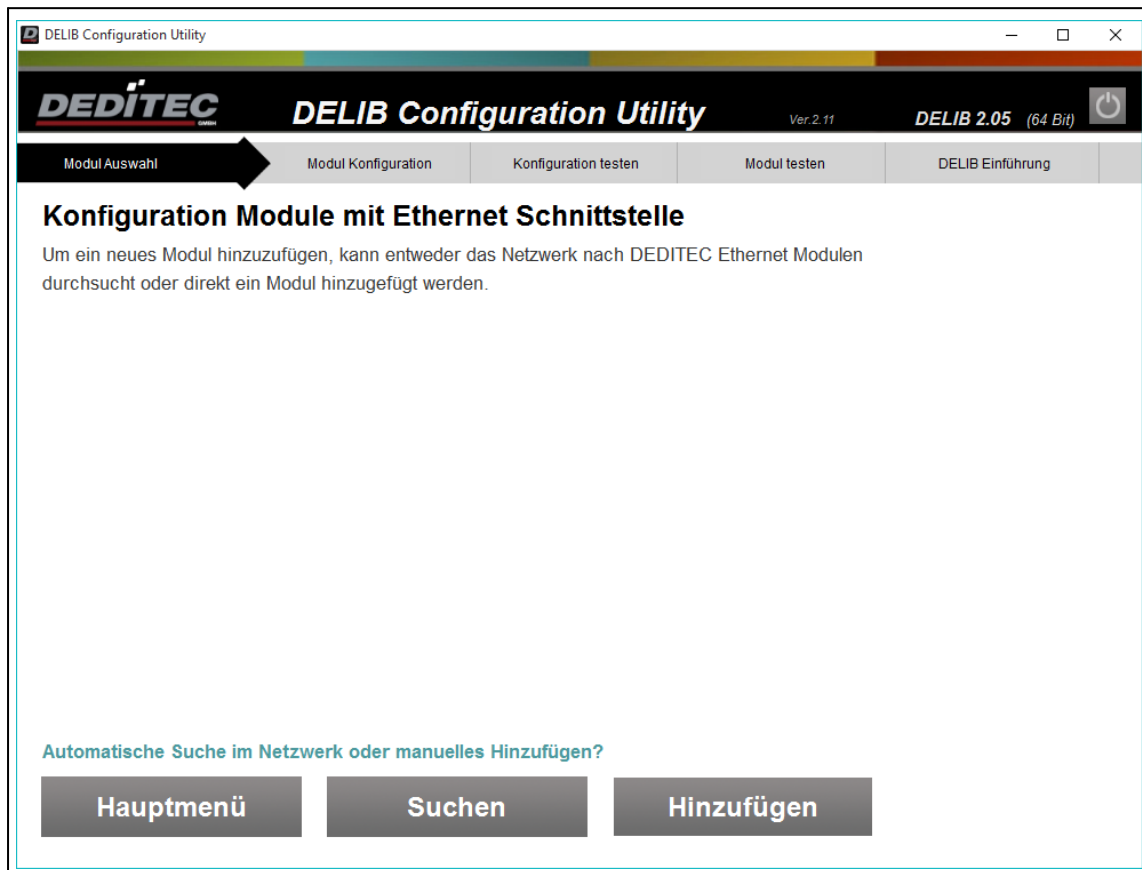
If you want to use several identical USB modules, a unique NR must be

assigned to the modules one after the other.

4.2.3.2.2. Module configuration Ethernet

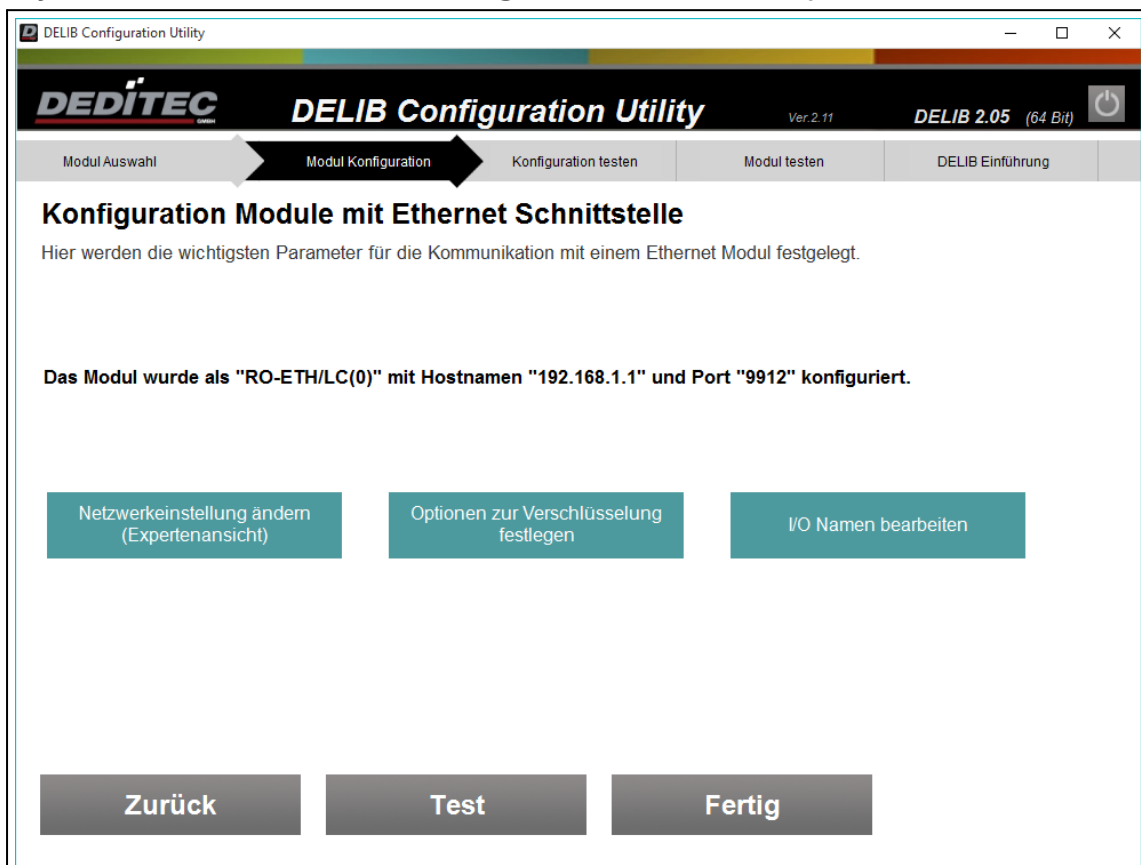
When configuring Ethernet products, the network can be automatically searched for DEDITEC products.

A manual configuration can also be performed.



In the last step of the configuration you can perform further options for the communication encryption or the I/O name assignment.

If you decide to do a manual configuration, start at this point.



Encryption can be selected between Disabled, User or Admin.

Disabled

- unencrypted communication
- no access to system settings

User

- encrypted communication
- Read access to system settings

Admin

- encrypted communication
- Read-write access to system settings

The desired password for encryption must be entered in the Password field.

By clicking on the button "Transfer settings for encryption to the module" you will be prompted to press a hardware button on the product. Only after pressing this button the encryption options will be transferred from the product.

see chapter→ **Authentication**

The screenshot shows the 'DELIB Configuration Utility' window. The title bar includes the DEDITEC logo, the application name, version 'Ver. 2.11', and 'DELIB 2.05 (64 Bit)' with a power icon. The main menu has five tabs: 'Modul Auswahl', 'Modul Konfiguration' (active), 'Konfiguration testen', 'Modul testen', and 'DELIB Einführung'. The active tab displays the title 'Konfiguration Module mit Ethernet Schnittstelle' and a subtitle 'Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.' Below this, a status message reads: 'Das Modul wurde als "RO-ETH/LC(0)" mit Hostnamen "192.168.1.1" und Port "9912" konfiguriert.' The configuration area contains several interactive elements: a button 'Netzwerkeinstellung ändern (Expertenansicht)', a 'Verschlüsselung' dropdown menu set to 'deaktiviert', a 'Passwort' text input field, a button 'I/O Namen bearbeiten', and a button 'Einstellungen zur Verschlüsselung auf das Modul übertragen'. At the bottom, there are three large buttons: 'Zurück', 'Test', and 'Fertig'.

Our Ethernet products offer you the possibility to assign names for the digital and analog I/Os. These are used, for example, on the web interface or in our app.

DELIB Configuration Utility
Ver. 2.11
DELIB 2.101 (64 Bit)

Modul Auswahl
Modul Konfiguration
Konfiguration testen
Modul testen
DELIB Einführung

Konfiguration Module mit Ethernet Schnittstelle

Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.

I/O Namen Konfiguration

Auswahl I/O-Typ Digital Output Kanal-Bereich Channel (1..8)

Kanal	Name	Kanal	Name
CH 01	<input type="text" value="Example name"/>	CH 05	<input type="text" value="Digital Output 5"/>
CH 02	<input type="text" value="Digital Output 2"/>	CH 06	<input type="text" value="Digital Output 6"/>
CH 03	<input type="text" value="Digital Output 3"/>	CH 07	<input type="text" value="Digital Output 7"/>
CH 04	<input type="text" value="Digital Output 4"/>	CH 08	<input type="text" value="Digital Output 8"/>

Zurück
Speichern

Note

Admin communication is required to save the channel names.

The maximum character length is 16.

4.2.3.2.2.1. Automatic search

Behavior of the DELIB Configuration Utility when using multiple network adapters in a Windows system.

When searching for DEDITEC Ethernet modules via the primary network adapter (ETH0) of the PC, the network settings of the DEDITEC product are ignored. This means that our Ethernet products are found even if they are not in the same network due to the IP address and subnet mask.

If, on the other hand, the Ethernet module is connected to a different or non-primary network adapter (e.g. ETH1), the network configuration of the Ethernet module must be valid so that the module is found.

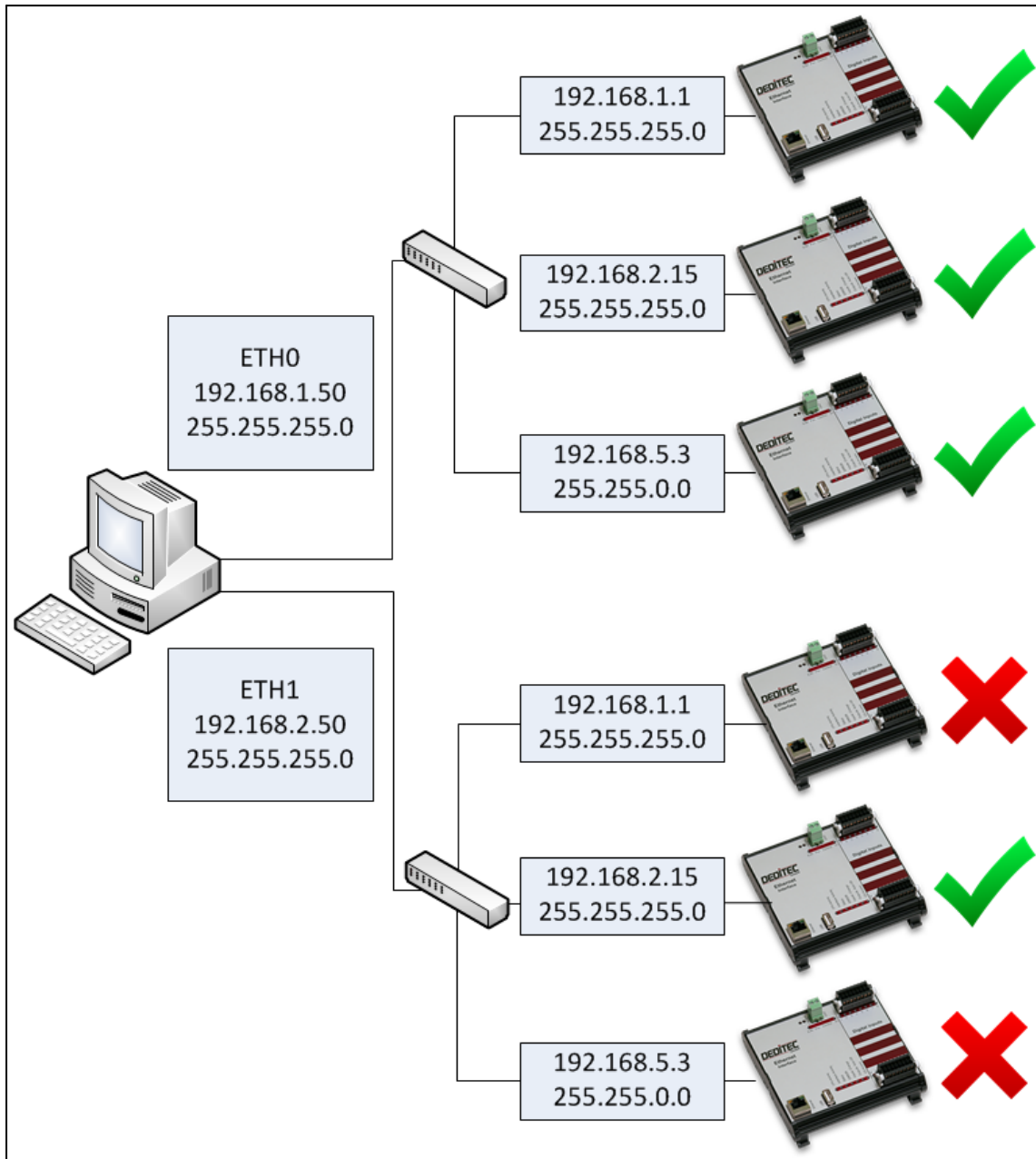
Attention:

If a laptop is used, the integrated WLAN adapter is usually configured as the primary network adapter.

As a result, modules that are connected to the laptop via LAN cable are often not recognized because the network configuration is not valid.

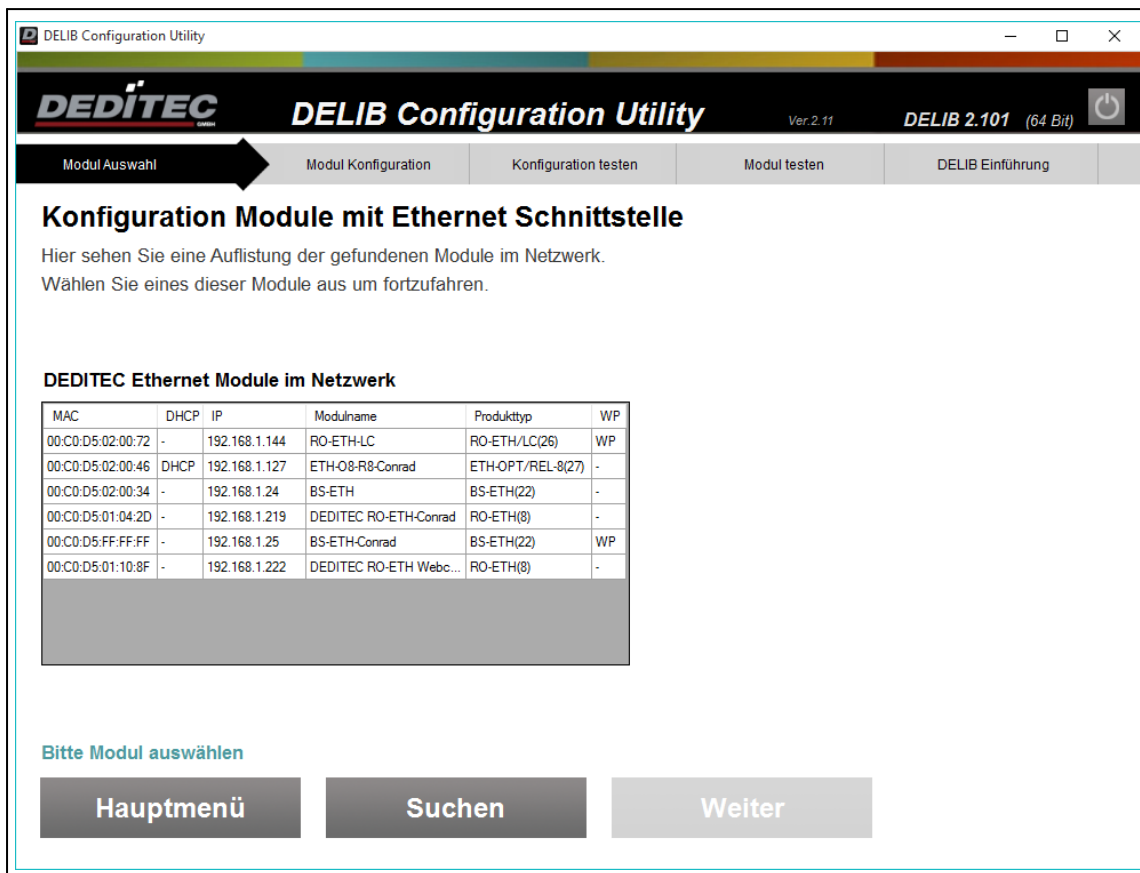
It is therefore recommended to deactivate the WLAN adapter for a short time to set up the module.

The following graphic shows an overview of the automatic search.



After the automatic search all found DEDITEC Ethernet products are displayed in an overview.

The overview shows the MAC address, the DHCP status, the IP, the module name, the product type and the status of the write protection.



With a click on a module the network configuration for this module can be changed. Please note that the write protection must be switched off if you want to save the settings. The write protection can be deactivated temporarily via the authentication.

see chapter → **Authentication**

DELIB Configuration Utility
Ver. 2.11
DELIB 2.05 (64 Bit)

Modul Auswahl

Modul Konfiguration

Konfiguration testen

Modul testen

DELIB Einführung

Konfiguration Module mit Ethernet Schnittstelle

An dieser Stelle kann die aktuelle Netzwerk Konfiguration des Moduls (rechts) geändert werden.
Mit "Übernehmen" wird diese Konfiguration an das Modul übertragen.
Klicken Sie auf "Weiter" um die Kommunikation mit dem Modul zu testen.

DEDITEC Ethernet Module im Netzwerk

MAC	DHCP	IP	Modulname	Produkttyp	WP
00:C0:D5:02:00:72	-	192.168.1.1	RO-ETH-LC	RO-ETH/LC(25)	-
00:C0:D5:02:00:46	-	192.168.1.171	ETH-O8-R8-Conrad	ETH-OPT/REL-8(27)	-
00:C0:D5:FF:FF:FF	-	192.168.1.25	BS-ETH-Conrad	BS-ETH(22)	-
00:C0:D5:02:00:34	-	192.168.1.24	BS-ETH	BS-ETH(22)	-
00:C0:D5:01:10:8F	-	192.168.1.222	DEDITEC RO-ETH Webc...	RO-ETH(8)	-
00:C0:D5:01:04:2D	-	192.168.1.219	DEDITEC RO-ETH-Conrad	RO-ETH(8)	-

Aktuelle Modul Konfiguration

Board Name

☐ IP-Adresse automatisch beziehen (DHCP)

IP Adresse

Netzmaske

Std.-Gateway

Zurück

Übernehmen

Hauptmenü

Suchen

Weiter

Note

Please note that only the module configuration is changed in this step.

If the module has already been configured in the DELIB Configuration Utility, this configuration must be adapted to the new module configuration.

4.2.3.2.2. Set up encryption

To be able to edit important module settings (e.g. network configuration) via TCP, the communication between module and PC must run in the so-called encrypted admin mode.

For this purpose, a password for encryption must be configured in the module and on the PC side.

This configuration can be created either via the automatic wizard or manually.

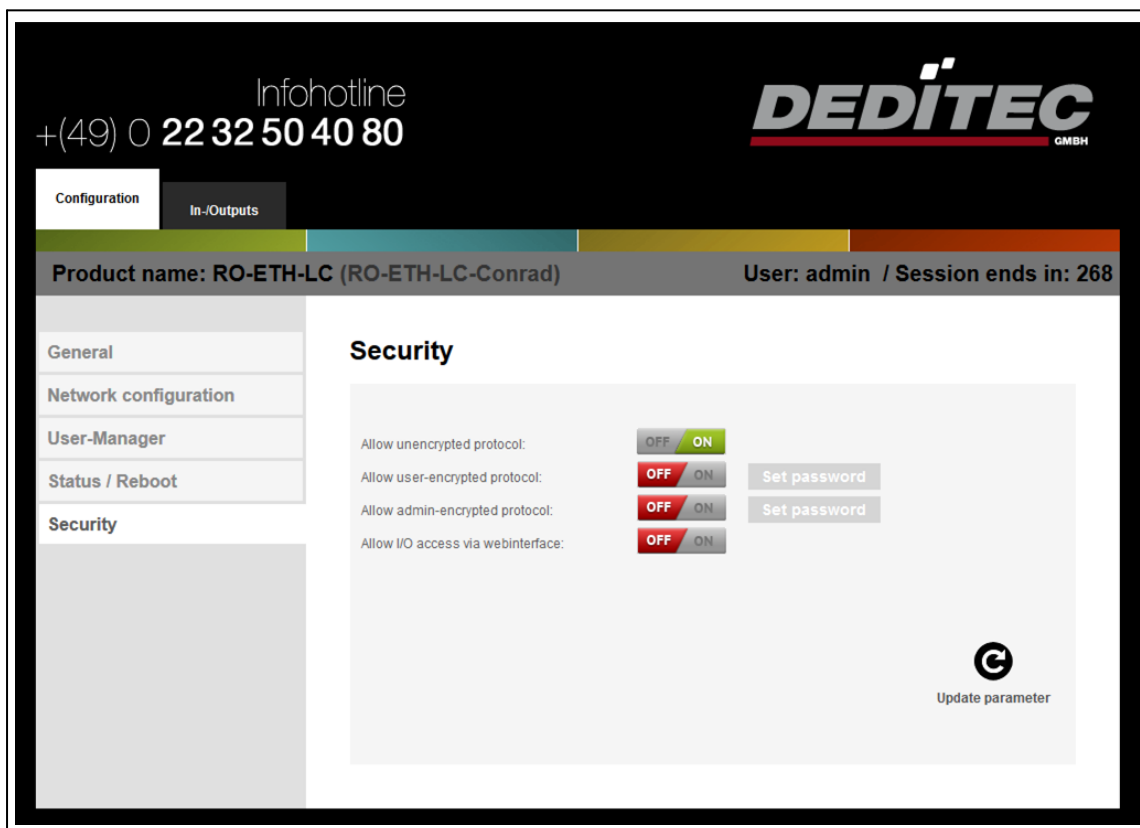
Manual configuration

During manual configuration, a password for encryption must be set on both the module and PC sides.

Step 1 - Configuration module page

First, the module is configured via web interface. Enter the IP address of the module (delivery state 192.168.1.1) in an Internet browser. If authentication is required, log in (delivery state: user=admin + password=admin).

Now open the security settings (**Configuration** → **Security**)



If disabled, enable the Allow admin-encrypted protocol setting. Then you can set a new password for encryption with Set password.

With Update parameter, if a new password has been entered correctly, the configuration is completed.

Note

All characters are allowed when entering the password.

The screenshot displays the DEDITEC web interface. At the top, there is a header with 'Infohotline + (49) 0 22 32 50 40 80' and the 'DEDITEC GMBH' logo. Below the header, a navigation bar shows 'Configuration' and 'In-/Outputs' tabs. A status bar indicates 'Product name: RO-ETH-LC (RO-ETH-LC-Conrad)' and 'User: admin / Session ends in: 223'. On the left, a sidebar menu lists 'General', 'Network configuration', 'User-Manager', 'Status / Reboot', and 'Security'. The main content area is titled 'Security' and contains several settings: 'Allow unencrypted protocol' (OFF/ON), 'Allow user-encrypted protocol' (OFF/ON), 'Allow admin-encrypted protocol' (OFF/ON), 'New password' (password field), 'Confirm password' (password field), and 'Allow I/O access via webinterface' (OFF/ON). A 'Set password' button is located next to the password fields. At the bottom right, there is a circular refresh icon and the text 'Update parameter'.

Step 2 - Configuration PC side

Start the DELIB Configuration Utility and select the desired Ethernet module.

Select "Admin" for encryption.

DELIB Configuration Utility

DELIB Configuration Utility Ver. 2.19 DELIB 2.19 (64 Bit)

Modul Auswahl Modul Konfiguration Konfiguration testen Modul testen DELIB Einführung

Konfiguration Module mit Ethernet Schnittstelle

Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.

Netzwerkkonfiguration

Modul Auswahl: RO-ETH/LC(0)

☒ Aktiviert ☐ Hostname verwenden

IP Adresse: 192.168.1.25 Verschlüsselung: deaktiviert

Port: 9912

Timeout [msec]: 5000

I/O Namen bearbeiten

Hauptmenü Test Fertig

Enter the password from step 1 in the new password field that appears.
 With Test the new encrypted admin mode communication can be tested.
 The DELIB Configuration Utility can now be closed.

DELIB Configuration Utility
Ver. 2.19
DELIB 2.19 (64 Bit)

Modul Auswahl
Modul Konfiguration
Konfiguration testen
Modul testen
DELIB Einführung

Konfiguration Module mit Ethernet Schnittstelle

Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.

Netzwerkconfiguration

Modul Auswahl: RO-ETH/LC(0)

☒ Aktiviert
☐ Hostname verwenden

IP Adresse: 192.168.1.25
Verschlüsselung: Admin

Port: 9912
Passwort: *****

Timeout [msec]: 5000

Einstellungen zur Verschlüsselung auf das Modul übertragen

I/O Namen bearbeiten

Modul-Kommunikation ist OK! Aktuelle Modul-Firmware ist 2.20

Hauptmenü
Test
Fertig

Automatic configuration

Start the DELIB Configuration Utility and select the desired Ethernet module.

DELIB Configuration Utility Ver. 2.19 DELIB 2.19 (64 Bit)

Modul Auswahl Modul Konfiguration Konfiguration testen Modul testen DELIB Einführung

Konfiguration Module mit Ethernet Schnittstelle

Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.

Netzwerkkonfiguration

Modul Auswahl: RO-ETH/LC(0)

☒ Aktiviert ☐ Hostname verwenden

IP Adresse: 192.168.1.25 Verschlüsselung: Admin

Port: 9912 Passwort: *****

Timeout [msec]: 5000

I/O Namen bearbeiten

Einstellungen zur Verschlüsselung auf das Modul übertragen

Hauptmenü Test Fertig

- 1) Select Admin for encryption
- 2) Enter a password in the new password field that appears.
- 3) With Transfer settings for encryption to the module, the current encryption setting is transferred to the module.

Note

All characters are allowed when entering the password.

Ideally, the password should consist of a combination of upper and lower case letters, numbers, and special characters.

To protect the module from unauthorized access, authentication must be performed when editing system settings.

Depending on the product type you will be asked to press the firmware reset button of the module for a certain time (RO-ETH and RO-CPU-800) or to change DIP switch settings (all other products, e.g. RO-ETH/LC, NET-ETH, ETH-RELAIS-8, ...).

The following example shows how the authentication is done for a RO-ETH/LC module.

In the first step you are asked to invert the position of DIP switch 2.

The screenshot shows the 'DELIB Configuration Utility' window. The title bar includes the DEDITEC logo, the application name, version 'Ver. 2.19', and 'DELIB 2.19 (64 Bit)'. The main menu has five tabs: 'Modul Auswahl', 'Modul Konfiguration' (active), 'Konfiguration testen', 'Modul testen', and 'DELIB Einführung'. The active tab displays the 'Konfiguration Module mit Ethernet Schnittstelle' screen. Below the title, a subtitle reads: 'Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.' A 'Manual' icon is in the top right. The 'Netzwerkkonfiguration' section contains the following fields and controls:

- Modul Auswahl:** A dropdown menu showing 'RO-ETH/LC(0)'.
- Aktiviert:** A checked checkbox.
- Hostname verwenden:** An unchecked checkbox.
- IP Adresse:** A text field containing '192.168.1.25'.
- Verschlüsselung:** A dropdown menu showing 'Admin'.
- I/O Namen bearbeiten:** A button.
- Port:** A text field containing '9912'.
- Passwort:** A text field containing '*****'.
- Timeout [msec]:** A text field containing '5000'.
- Einstellungen zur Verschlüsselung auf das Modul übertragen:** A button.

At the bottom, a teal instruction bar reads: 'Schritt 1: DIP-Schalter 2 (Schreibschutz) des ETH-Moduls auf Stellung 1 (0=OFF, 1=ON) setzen'. To the right of this bar is a teal button labeled 'Authentifizierung abbrechen'. Below the instruction bar are three buttons: 'Hauptmenü', 'Test', and 'Fertig'. Three small black squares are located at the bottom right of the window.

In the second step, DIP switch2 is reset to the initial position.

The screenshot shows the 'DELIB Configuration Utility' window. The title bar reads 'DELIB Configuration Utility'. The interface has a top header with the 'DEDITEC' logo, the title 'DELIB Configuration Utility', the version 'Ver. 2.19', and 'DELIB 2.19 (64 Bit)' with a power icon. Below the header is a navigation bar with five steps: 'Modul Auswahl', 'Modul Konfiguration' (highlighted with a black arrow), 'Konfiguration testen', 'Modul testen', and 'DELIB Einführung'. The main content area is titled 'Konfiguration Module mit Ethernet Schnittstelle' and includes a sub-header 'Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.' and a 'Manual' icon. Under the 'Netzwerkkonfiguration' section, there are several input fields: 'Modul Auswahl' (dropdown menu showing 'RO-ETH/LC(0)'), 'Aktiviert' (checked checkbox), 'Hostname verwenden' (unchecked checkbox), 'IP Adresse' (text field with '192.168.1.25'), 'Verschlüsselung' (dropdown menu showing 'Admin'), 'Port' (text field with '9912'), 'Passwort' (password field with '*****'), and 'Timeout [msec]' (text field with '5000'). There are also buttons for 'I/O Namen bearbeiten' and 'Einstellungen zur Verschlüsselung auf das Modul übertragen'. At the bottom, a teal button reads 'Authentifizierung abbrechen'. A status bar at the very bottom contains three buttons: 'Hauptmenü', 'Test', and 'Fertig', along with three small black squares.

DELIB Configuration Utility

DEDITEC **DELIB Configuration Utility** Ver. 2.19 **DELIB 2.19** (64 Bit)

Modul Auswahl **Modul Konfiguration** Konfiguration testen Modul testen DELIB Einführung

Konfiguration Module mit Ethernet Schnittstelle

Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.

Netzwerkkonfiguration

Modul Auswahl: RO-ETH/LC(0)

☒ Aktiviert ☐ Hostname verwenden

IP Adresse: 192.168.1.25 Verschlüsselung: Admin I/O Namen bearbeiten

Port: 9912 Passwort: *****

Timeout [msec]: 5000 Einstellungen zur Verschlüsselung auf das Modul übertragen

Schritt 2: DIP-Schalter 2 (Schreibschutz) des ETH-Moduls auf Stellung 0 (0=OFF, 1=ON) setzen

Authentifizierung abbrechen

Hauptmenü Test Fertig

Authentication has been completed successfully. You are now temporarily authorized to edit system settings.

The screenshot shows the 'DELIB Configuration Utility' window. The title bar reads 'DELIB Configuration Utility'. The header features the 'DEDITEC' logo, the product name 'DELIB Configuration Utility', the version 'Ver. 2.19', and 'DELIB 2.19 (64 Bit)' with a power icon. A navigation bar contains five tabs: 'Modul Auswahl', 'Modul Konfiguration' (active), 'Konfiguration testen', 'Modul testen', and 'DELIB Einführung'. The main content area is titled 'Konfiguration Module mit Ethernet Schnittstelle' and includes a sub-header 'Netzwerkkonfiguration'. Below this, there are input fields for 'Modul Auswahl' (set to 'RO-ETH/LC(0)'), 'IP Adresse' (192.168.1.25), 'Port' (9912), and 'Timeout [msec]' (5000). There are also checkboxes for 'Aktiviert' (checked) and 'Hostname verwenden' (unchecked), a 'Verschlüsselung' dropdown (set to 'Admin'), and a 'Passwort' field (masked with asterisks). A button 'I/O Namen bearbeiten' is present. A message box states 'Einstellungen zur Verschlüsselung auf das Modul übertragen'. At the bottom, there are three buttons: 'Hauptmenü', 'Test', and 'Fertig', followed by three small square icons. A green status message 'Authentifizierung erfolgreich' is displayed above the buttons.

DELIB Configuration Utility

DEDITEC **DELIB Configuration Utility** Ver. 2.19 **DELIB 2.19** (64 Bit)

Modul Auswahl Modul Konfiguration Konfiguration testen Modul testen DELIB Einführung

Konfiguration Module mit Ethernet Schnittstelle

Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.

Netzwerkkonfiguration

Modul Auswahl: RO-ETH/LC(0)

☒ Aktiviert ☐ Hostname verwenden

IP Adresse: 192.168.1.25 Verschlüsselung: Admin I/O Namen bearbeiten

Port: 9912 Passwort: *****

Timeout [msec]: 5000

Einstellungen zur Verschlüsselung auf das Modul übertragen

Authentifizierung erfolgreich

Hauptmenü Test Fertig

After successful authentication, the encryption settings are finally transferred to the module.

The screenshot shows the 'DELIB Configuration Utility' window. The title bar reads 'DELIB Configuration Utility'. The main header features the 'DEDITEC' logo, the product name 'DELIB Configuration Utility', the version 'Ver. 2.19', and 'DELIB 2.19 (64 Bit)' with a power icon. A progress bar at the top indicates the current step is 'Modul Konfiguration', with other steps being 'Modul Auswahl', 'Konfiguration testen', 'Modul testen', and 'DELIB Einführung'. The main content area is titled 'Konfiguration Module mit Ethernet Schnittstelle' and includes a sub-header 'Netzwerkkonfiguration'. Below this, there are input fields for 'Modul Auswahl' (set to 'RO-ETH/LC(0)'), 'IP Adresse' (192.168.1.25), 'Port' (9912), 'Timeout [msec]' (5000), 'Verschlüsselung' (Admin), and 'Passwort' (masked with asterisks). There are checkboxes for 'Aktiviert' (checked) and 'Hostname verwenden' (unchecked). Two teal buttons are present: 'I/O Namen bearbeiten' and 'Einstellungen zur Verschlüsselung auf das Modul übertragen'. A green status message at the bottom reads 'Einstellungen zur Verschlüsselung erfolgreich übertragen'. At the very bottom, there are three large buttons: 'Hauptmenü', 'Test', and 'Fertig'.

DELIB Configuration Utility

DEDITEC DELIB Configuration Utility Ver. 2.19 DELIB 2.19 (64 Bit)

Modul Auswahl Modul Konfiguration Konfiguration testen Modul testen DELIB Einführung

Konfiguration Module mit Ethernet Schnittstelle

Hier werden die wichtigsten Parameter für die Kommunikation mit einem Ethernet Modul festgelegt.

Netzwerkkonfiguration

Modul Auswahl: RO-ETH/LC(0)

☒ Aktiviert ☐ Hostname verwenden

IP Adresse: 192.168.1.25 Verschlüsselung: Admin

Port: 9912 Passwort: *****

Timeout [msec]: 5000

I/O Namen bearbeiten

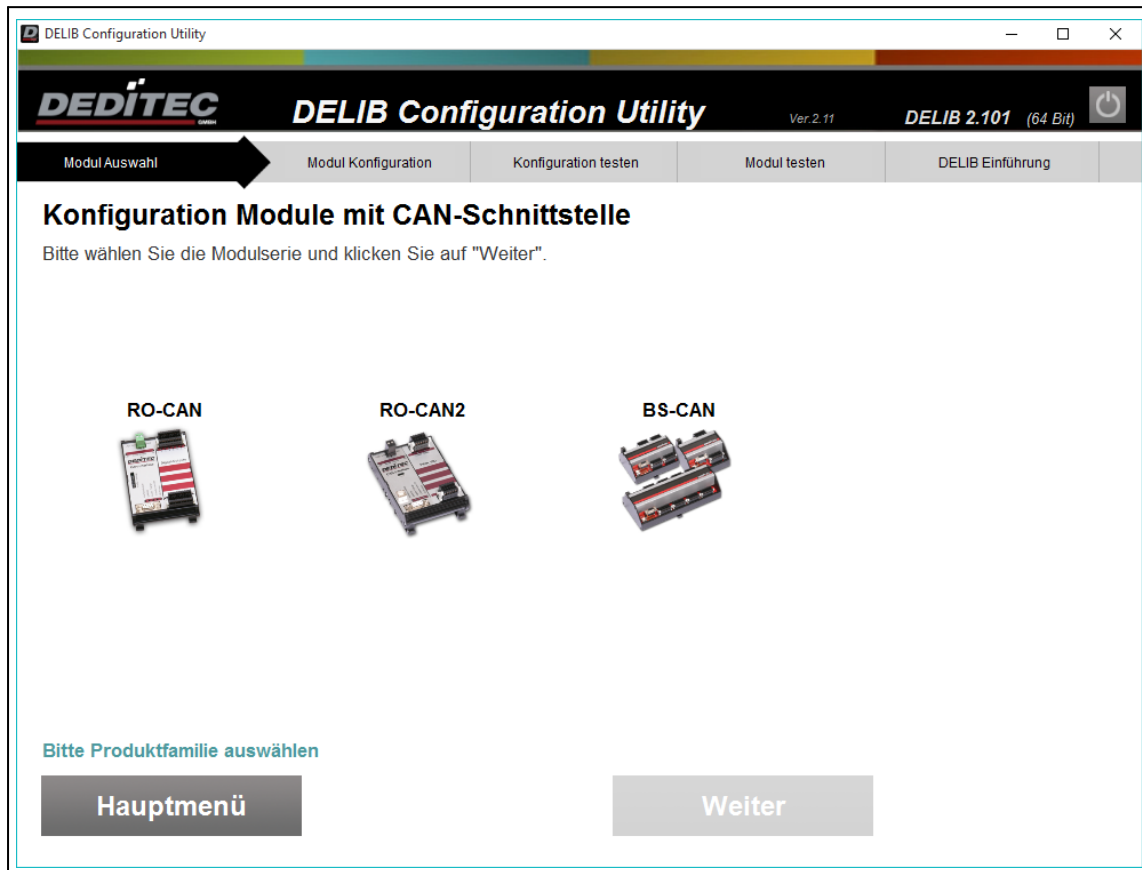
Einstellungen zur Verschlüsselung auf das Modul übertragen

Einstellungen zur Verschlüsselung erfolgreich übertragen

Hauptmenü Test Fertig

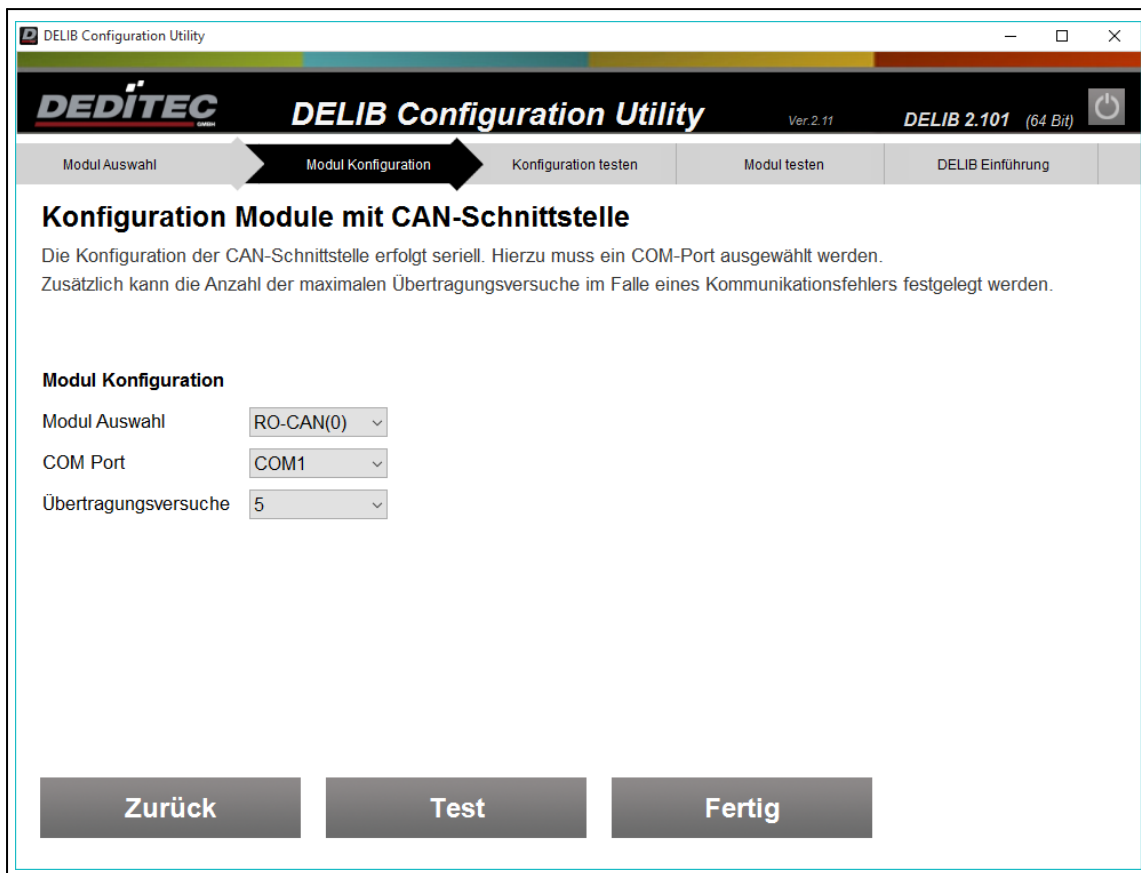
4.2.3.2.3. Module Configuration CAN

When you create a new CAN configuration, you must first select the product family.



When configuring RO-CAN modules, the product is configured via a serial connection. For this purpose, the corresponding COM port must be specified. In addition, the number of transmission attempts in the event of a communication error must be specified.

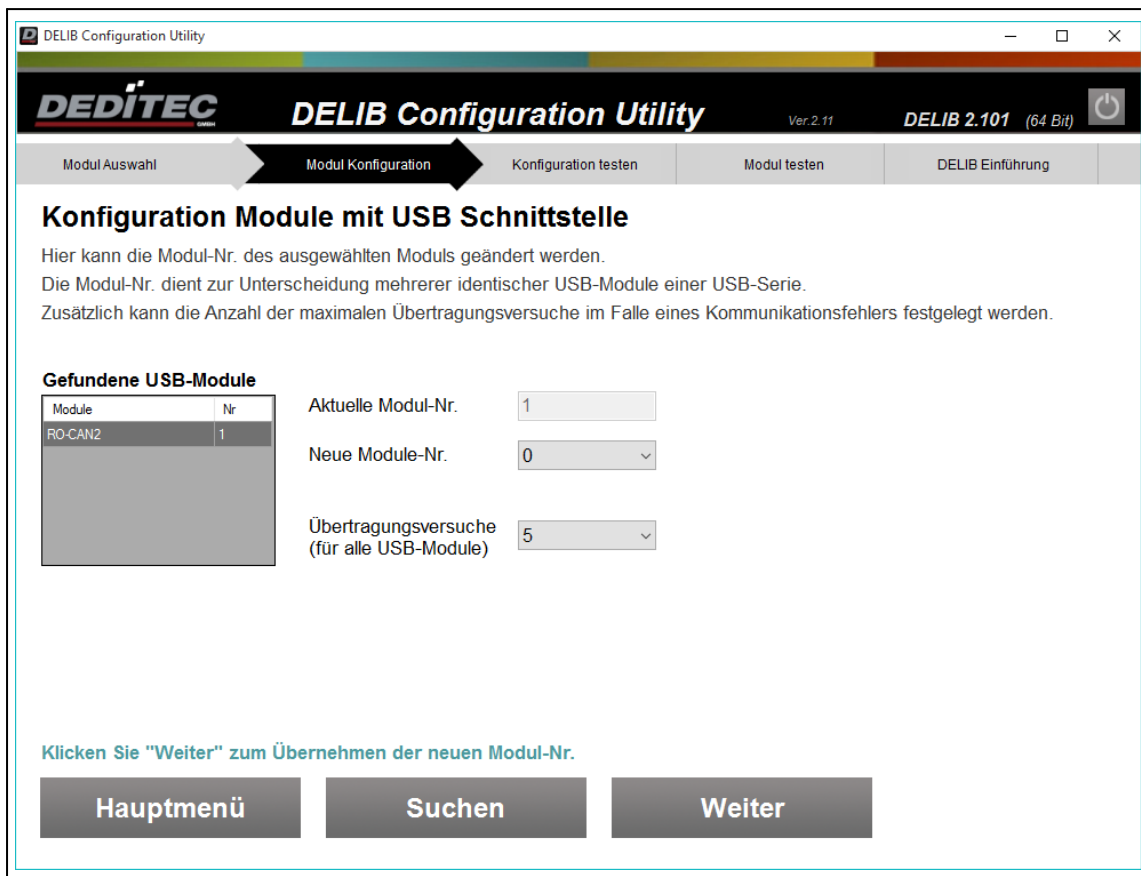
The "Test" button is used to check whether communication is possible via the specified COM port.



The configuration of a RO-CAN2 or BS-CAN module takes place via the USB interface. As with a USB interface, a different module no. must be assigned for unique identification when using identical products.

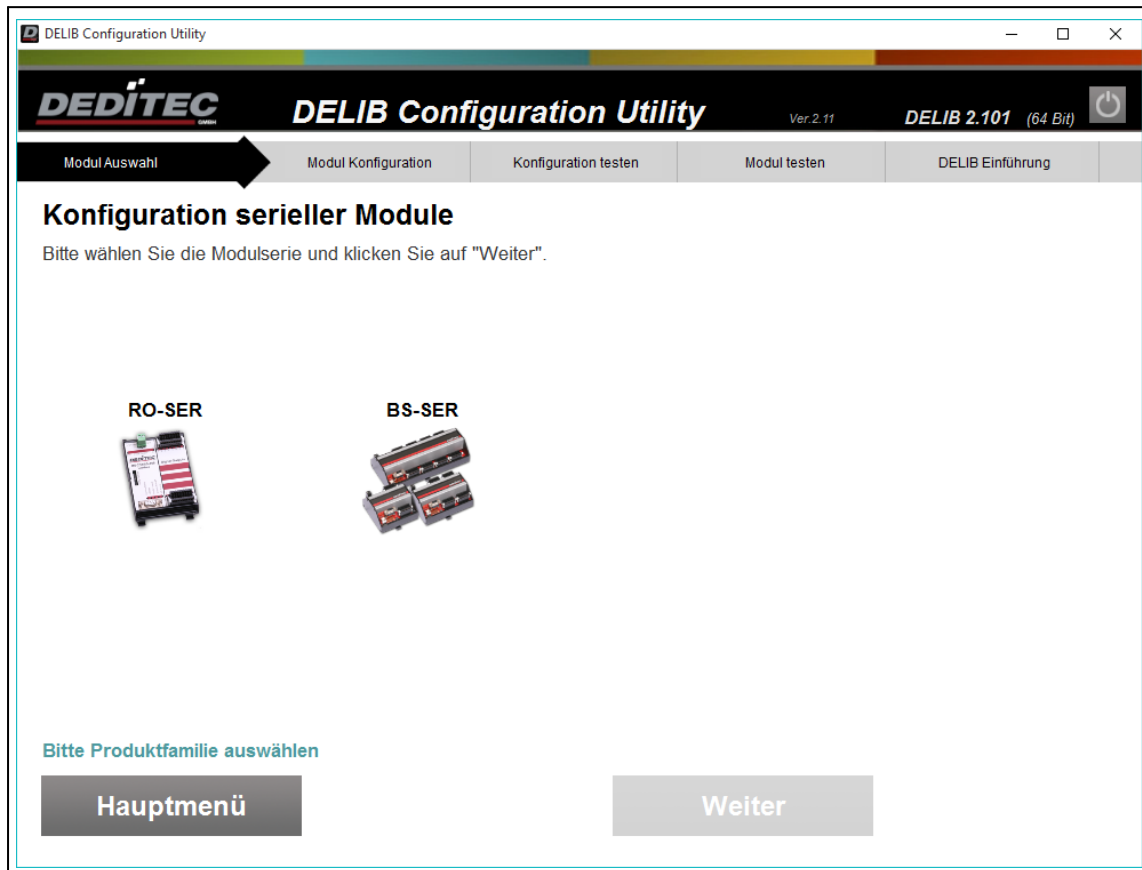
see chapter **Example for the configuration of identical USB modules**

In addition to the module no., the number of transmission attempts in the event of an error can also be specified.



4.2.3.2.4. Module Configuration Serial

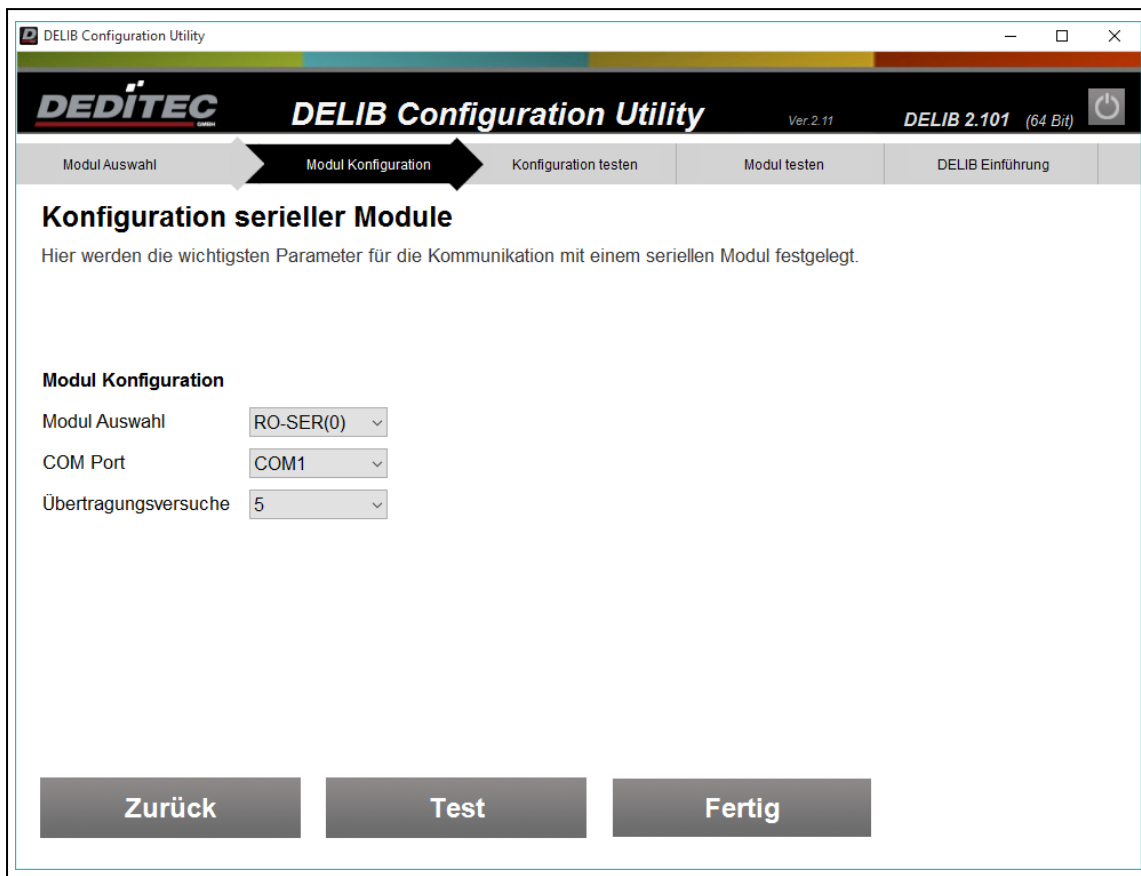
When you create a new serial configuration, you must first select the product family.



When configuring the serial products, the corresponding COM port must be specified.

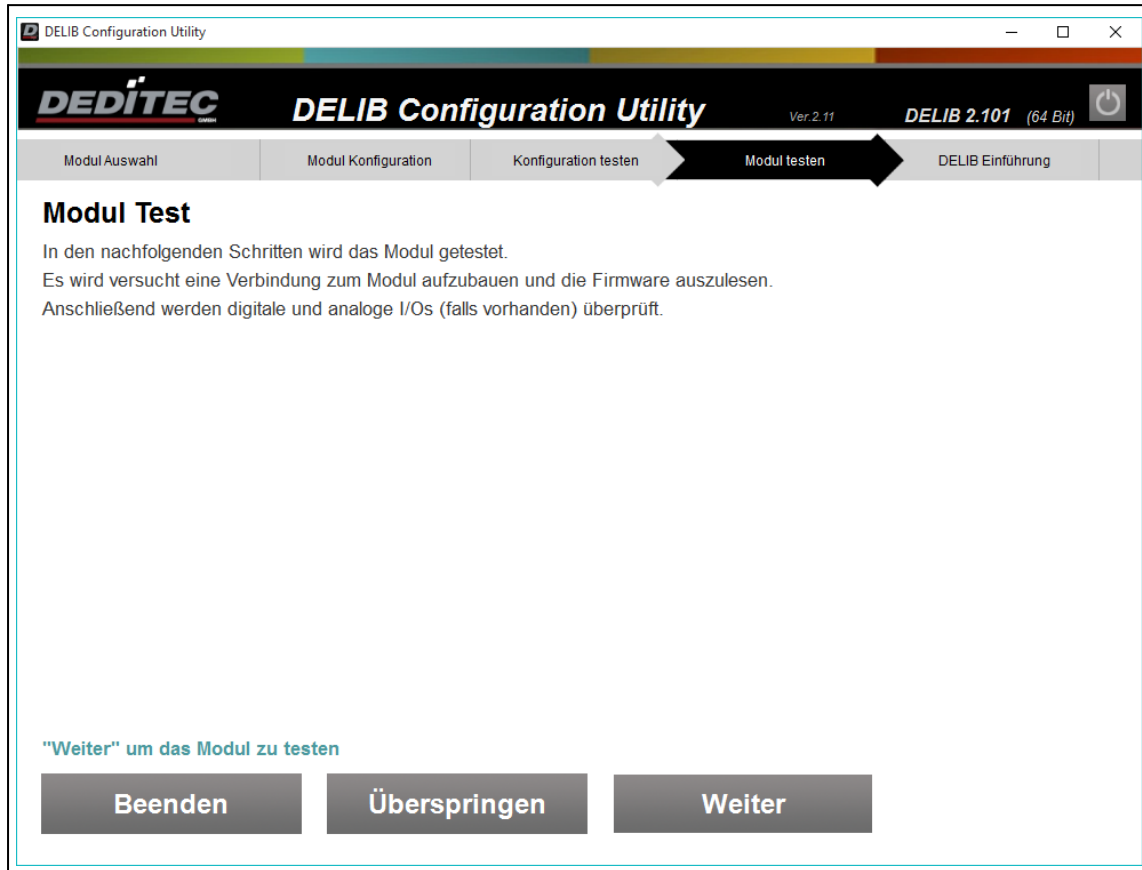
Additionally, the number of transmission attempts in case of a communication error must be specified.

The "Test" button is used to check whether communication is possible via the specified COM port.

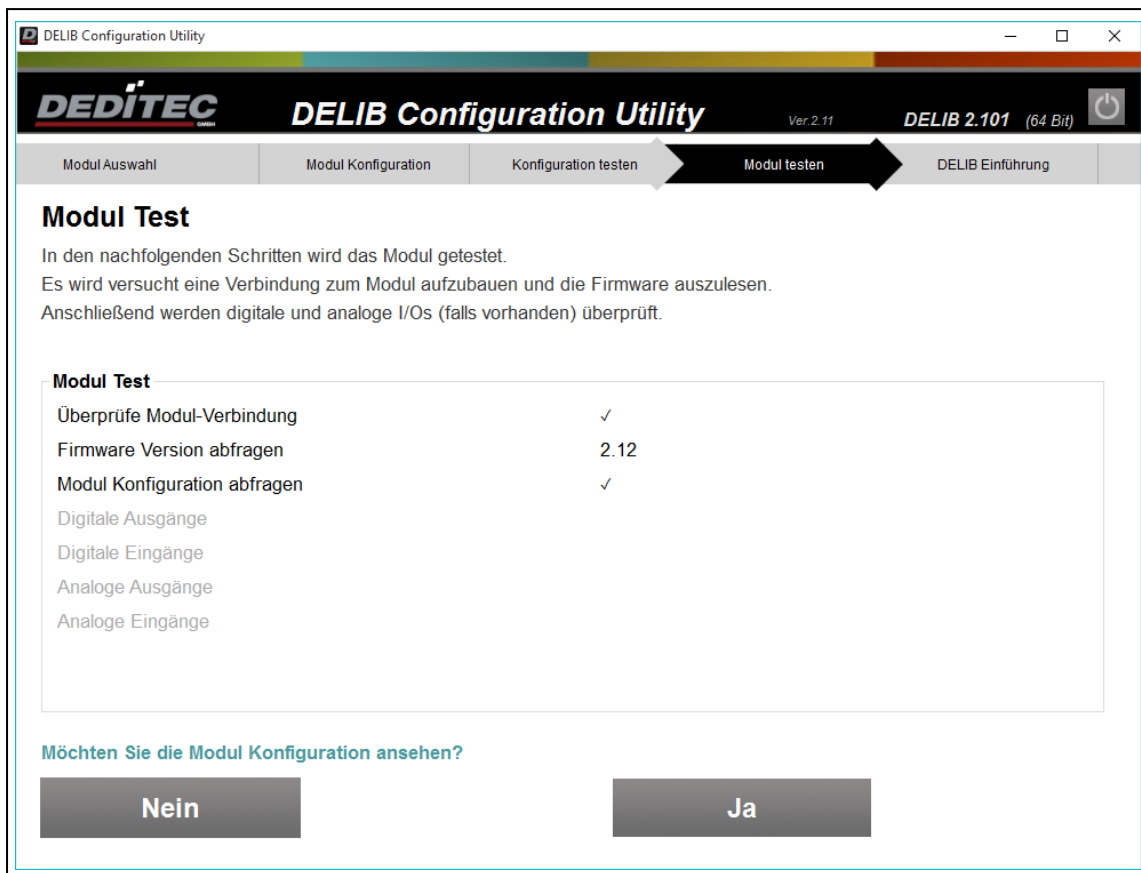


4.2.3.3. Test module

After the interface has been configured, the product can then be tested.



Test of the firmware and display of the module info.



The module info shows all properties of the product. Beside the number of available I/Os also the supported software features are shown.

DT_ModuleInfo

General

SW_FEATURE_1

e3373007

HW_INTERFACE_1

01000003

Firmware-Revision

2.11

Main-Module:

RO

Digital I/O

Digital Inputs

8

Digital Outputs

8

Digital In-/Outputs

0

Digital Input FlipFlops

8

Digital Input Counter

8

Pulse Gen Outputs

0

CNT8

0

Digital PWM Outputs

0

Analog I/O

Analog Inputs

16

Analog Outputs

4

Temperature Inputs

4

Special

Stepper

2

Features-General

Supported by FW

OK

Dev IO registry error

OK

AD FIFO

OK

Set-Clr Bit Commands

OK

EEPROM RN23

-

EEPROM E2_2K

-

DX1 Mode

-

Support Channel Names

OK

HW-INT Supported by FW

OK

ETH

OK

CAN

-

RS232

-

RS485

-

USB1

-

USB2

-

Features-Digital I/O

DI Commands

OK

DI CNT Commands

OK

DI CNT Latch Feature

-

DI FF Commands

OK

DO Commands

OK

DO Time Commands

OK

PWM Commands

-

TTL Commands

-

PulseGen Commands

-

CNT8 Commands

-

Auto-Off Timeout Commar

OK

Features-Analog I/O

DA Commands

OK

AD Commands

OK

Pt100 Commands

OK

Features-Special

Watchdog Commands

-

Stepper Commands

OK

Submodule-Info's

Sub: 0 - RO-AD16DA4

FW:2.11

Sub: 1 - RO-O8_R8

FW:2.10

Sub: 2 - RO-STEPPER2

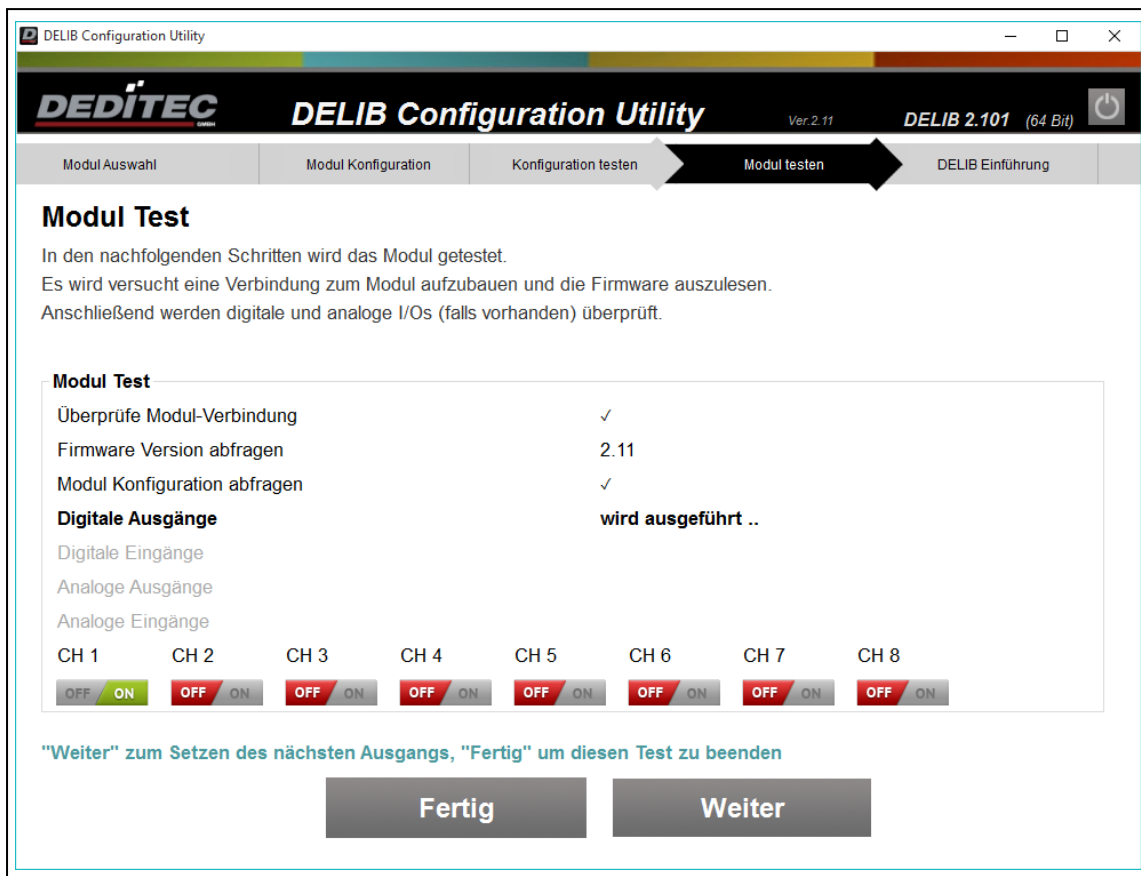
FW:1.29

Sub: 3 - RO-PT100

FW:1.03

EXIT

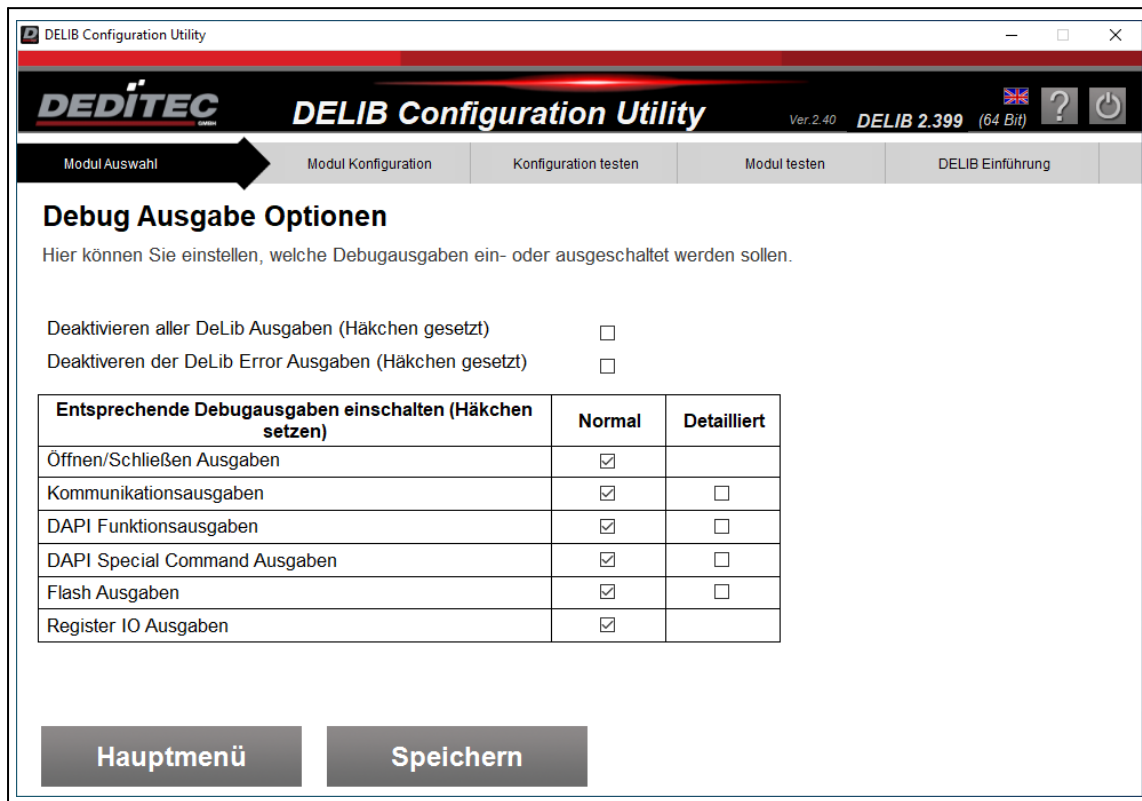
A test of the I/Os follows. In this example the digital outputs are switched.



If all tests have been passed successfully, the product is ready for use.

4.2.3.4. Set debug options

The "Debug Output Options" button takes you to the following options menu. There you can set which debug outputs you want to switch on or off.



The screenshot shows the 'DELIB Configuration Utility' window. The title bar reads 'DELIB Configuration Utility'. The main header features the 'DEDITEC' logo and 'DELIB Configuration Utility' text, along with version information 'Ver. 2.40 DELIB 2.399 (64 Bit)' and icons for help and power. The navigation bar includes 'Modul Auswahl' (highlighted with a black arrow), 'Modul Konfiguration', 'Konfiguration testen', 'Modul testen', and 'DELIB Einführung'.

Debug Ausgabe Optionen

Hier können Sie einstellen, welche Debugausgaben ein- oder ausgeschaltet werden sollen.

Deaktivieren aller DeLib Ausgaben (Häkchen gesetzt) ☐

Deaktivieren der DeLib Error Ausgaben (Häkchen gesetzt) ☐

Entsprechende Debugausgaben einschalten (Häkchen setzen)	Normal	Detailliert
Öffnen/Schließen Ausgaben	<input checked="" type="checkbox"/>	
Kommunikationsausgaben	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DAPI Funktionsausgaben	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DAPI Special Command Ausgaben	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Flash Ausgaben	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Register IO Ausgaben	<input checked="" type="checkbox"/>	

Buttons at the bottom: 'Hauptmenü' and 'Speichern'.

4.2.4. Using the module selector

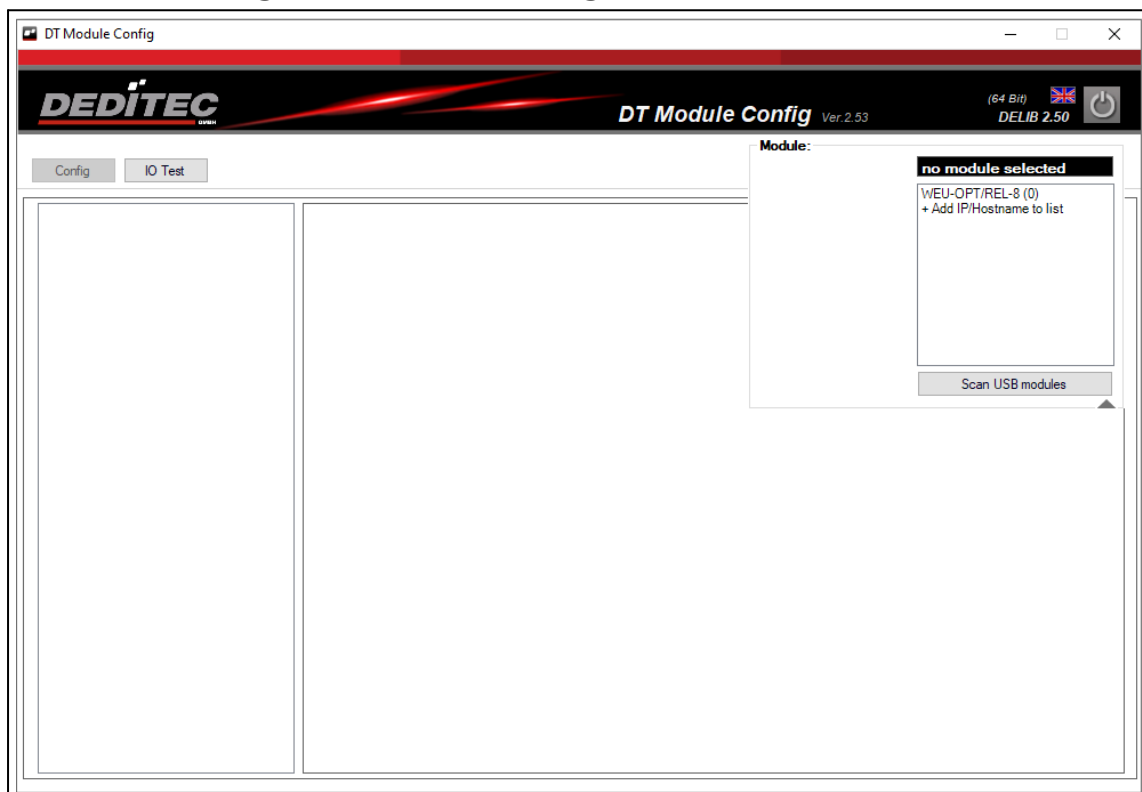
To use our products with the DEDITEC software, they must be selected via the module selector.

Depending on the module, this can be done via different interfaces.

4.2.4.1. via USB

If you have connected the module to the PC via the USB interface, the module can be selected directly by clicking on the module selector in the upper right corner.

Afterwards you can make the desired network configuration in the network area under LAN - Configuration or WiFi - Configuration.

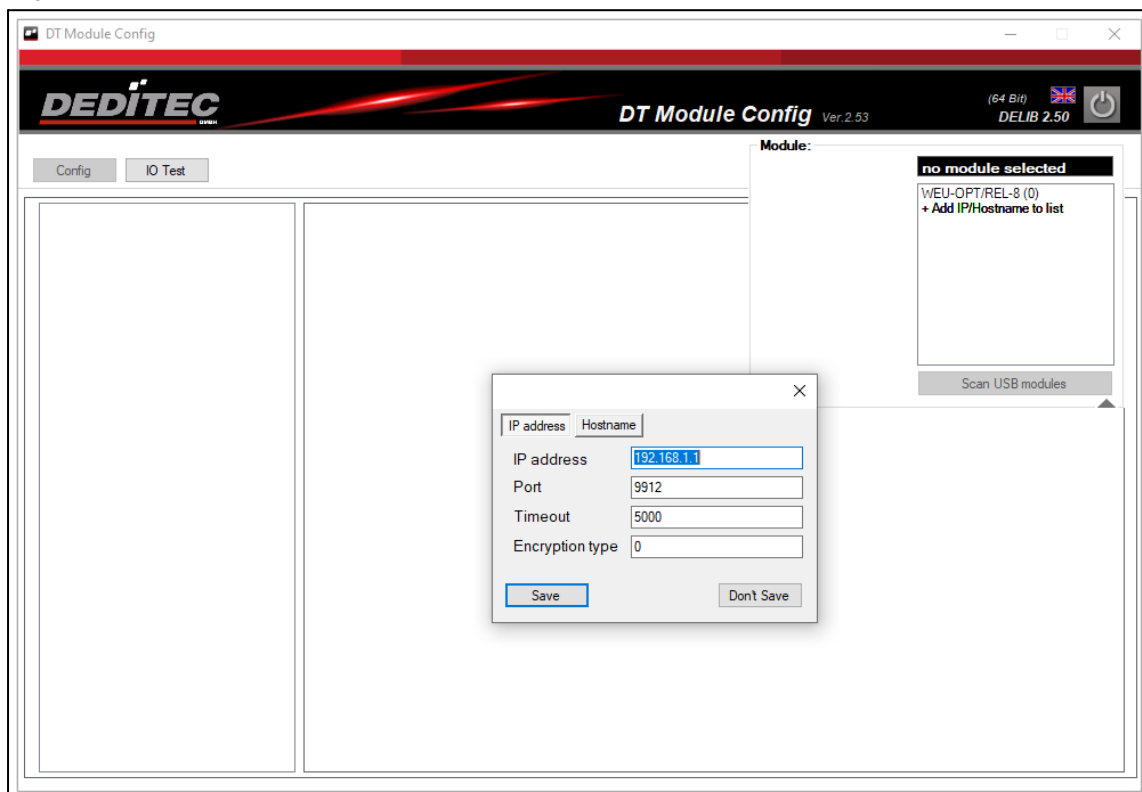


4.2.4.2. via Ethernet

If you have connected your module via the Ethernet interface, you can find the module directly via the IP address integrated in the network.

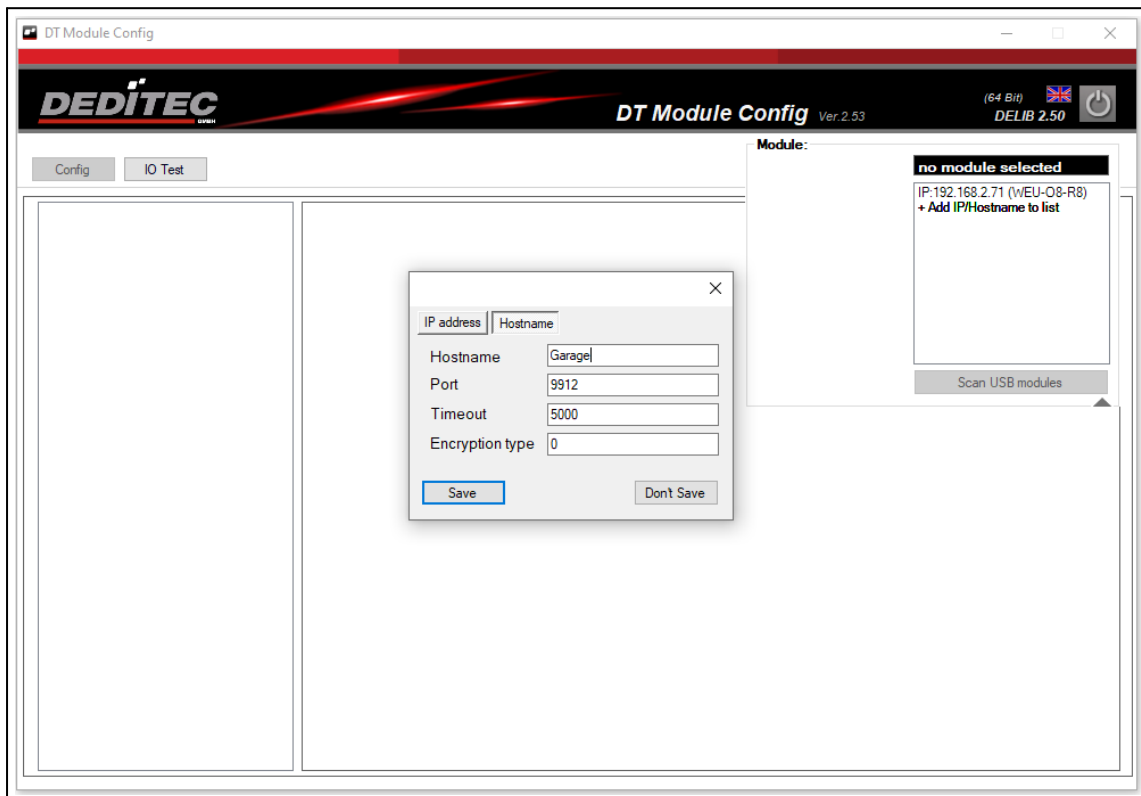
If necessary, ask your system administrator for this.

Click in the module selector on "Add IP/Hostname to list", enter there the automatically received IP address under the tab "IP address" and confirm the input with "Save".



If your module is in DHCP mode (see chapter: **LAN network settings**) you can also connect this using the board name.

This can be found in the Config module in the "LAN - Network Information" area. For a connection via board name, click on "Add IP/Hostname to list" in the Selector module. Enter the name under the "Hostname" tab and confirm the entry with "Save".



4.2.4.3. via WiFi / WPS

WiFi (only for WEU modules)

To connect the module via WiFi, it must be connected to USB or Ethernet in advance.

Now WiFi can be activated under the WiFi configuration menu item. The IP address assigned to the module can be found under WiFi info.

WPS (only for WEU modules)

If the module is not yet connected to the PC network, carry out the connection setup as described in the CFG button chapter.

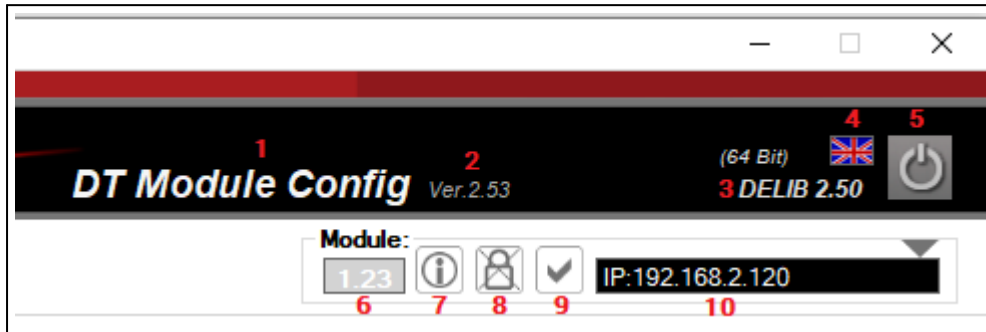
Then click in the module selector on "Add IP/Hostname to list", enter there the automatically received IP address under the tab "IP address" and confirm the entry with "Save".

If necessary, ask your system administrator for this.

You can also start a WPS connection using the Module Config. To do this, perform the steps as described in the chapter: WiFi WPS connection.

4.2.4.4. Modul Info

If the connection to the module is successful, various information will now be displayed in the Module Selector area, as described below.



Description:

1. Displays the name of the DEDITEC software used
2. Displays the currently used version number of the software
3. Shows the currently used DELIB version
4. By clicking on the flag symbol, the language can be changed between German and English
5. Closes the program
6. Displays the currently used firmware of your module
7. By clicking on the information button, the information window of the module opens (see picture below)
8. Shows whether there is an encrypted or decrypted communication with the module
9. Displays the communication status with the module
10. Depending on the connection type, the IP or board name of the currently used module is displayed here

Information window

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

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

DT_ModuleInfo

General

SW_FEATURE_10300a0c5

HW_INTERFACE_100000103

Firmware-Revision1.23

Main-Module:-

Digital I/O

Digital Inputs0

Digital Outputs8

Digital In-/Outputs0

Digital Input FlipFlops0

Digital Input Counter0

Pulse Gen Outputs0

CNT80

Digital PWM Outputs0

Analog I/O

Analog Inputs0

Analog Outputs0

Temperature Inputs0

Special

Stepper0

Features-General

Supported by FWOK

Dev IO registry errorOK

RO-AD FIFO-

NET-Software FIFO-

Set-Clr Bit CommandsOK

EEPROM RN23-

EEPROM E2_2K-

DX1 Mode-

Support Channel Names-

HW-INT Supported by FVOK

ETHOK

CAN-

RS232/485-

USB1-

USB2-

Features-Digital I/O

DI Commands-

DI CNT Commands-

DI CNT Latch Feature-

DI FF Commands-

DO CommandsOK

DO Time Commands-

PWM Commands-

TTL Commands-

PulseGen Commands-

CNT8 Commands-

Auto-Off TimeoutOK

Auto-Off Timeout MaskOK

FTDI Userbyte 60x0

Features-Analog I/O

DA Commands-

AD Commands-

Pt 100 Commands-

Features-Special

Watchdog Commands-

Stepper Commands-

EXIT

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

DEDITEC

Software description | Seite 266

4.2.5. DELIB Module Config

The Module Config is a new application for configuration and testing of our products. This program is included in the installation package of our DELIB driver library.

4.2.5.1. Module configurations

In the configuration area, configuration settings of the module can be viewed or changed.

4.2.5.1.1. Module info page

The Module Config not only allows you to configure your WEU module quickly and easily, you can also view all important module information at just one glance.

Info
Hier finden Sie Informationen zu Ihren Moduleigenschaften

Modul-Name	WEU-08-R8
Modul-ID	40 (dezimal) / 0x0028 (hex)
Firmware-Version	Ver. 1,23

Form für ModuleOverview (Ver. 1.01)

Module name

Displays the name of the currently used DEDITEC module.

Module ID

Shows the ID of your used module. This is required for programming your own software with DEDITEC commands.

Firmware-Revision

Displays the current firmware version installed on the module.

4.2.5.1.2. Module identification

Identify the module that you are currently addressing with the Config module to prevent confusion.

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

The identification is started by pressing "Start".

Now the status LED starts to blink repeatedly.

This process is terminated by pressing "Stop".

Identifikation

Hier können Sie Ihr Modul identifizieren

Identify module

Modulidentifikation läuft!

Mit der Identifikations-Funktion können Sie feststellen, welches Ihrer Module momentan durch das Modul Config angesteuert wird. Dies ist besonders hilfreich, bei der gleichzeitigen Verwendung mehrerer Module. Durch das Betätigen der 'Start' Taste blinken folgende LEDs zur Identifikation

- bei unseren ETH_LC-Modulen blinkt die 'Int.Act' LED
- bei unseren WEU-Modulen blinkt die 'Status-LED'

Das Bedienen der 'Stop' Taste beendet diesen Vorgang.

FormConfModulIdent (Ver. 1.01)

4.2.5.1.3. LAN network information

All important LAN network information at a glance.

On this information page you will find the current LAN settings of your module.

Info

Hier finden Sie die Netzwerkinformationen des ausgewählten Ethernet-Produktes

MAC address	40:F5:20:44:60:EB
Board name (Hostname in DHCP mode)	WEU-Q8-R8
LAN status	Static-IP success
DHCP active	<input type="checkbox"/>
IP address	192.168.2.71
Subnet mask	255.255.255.0
Default gateway	192.168.2.254
TCP port	9912

☐ Auto Refresh

FormCmNetwork1 LANInfo (Ver. 1.01)

MAC address

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

Board Name

Displays the current board name of your module.

LAN-Status

Here the connection status of your connected module is displayed.

If the status "Query not supported (FW-Update)" is shown, your module needs a newer firmware.

DHCP active

Shows whether the module is connected via DHCP.

IP address, netmask, default gateway and TCP port


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

4.2.5.1.4. LAN network settings

Here you can make changes to the network settings of the selected WEU module.

Konfiguration

Hier können Sie die Netzwerkeinstellungen des ausgewählten Ethernet-Produktes ändern

Board name (Hostname in DHCP mode)	<input type="text" value="WEU-O8-R8"/>
Network configuration protection active 	<input type="checkbox"/>
DHCP active	<input type="checkbox"/>
IP address	<input type="text" value="192.168.2.71"/>
Subnet mask	<input type="text" value="255.255.255.0"/>
Default gateway	<input type="text" value="192.168.2.254"/>
TCP port	<input type="text" value="9912"/>

FormConfigNetwork LANConfig (Ver. 1.01)

Board Name

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

This option is especially useful when using multiple modules.

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

More information about connecting the module via board name

see chapter: **Using the module selector**

Network configuration protection active

If this option is enabled, network configuration can only be changed via the web interface.

This prevents unauthorized access to the network configuration (for example, via the Config module).

DHCP active

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

The board name is used as the host name.

IP address, Subnet mask, Default gateway and TCP port

These settings are used when DHCP is disabled. If necessary, please ask your system administrator.

Load factory settings

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

Factory settings	
Board name	Module dependent
Network protection	off
DHCP	off
IP address	192.168.1.1
Subnet mask	255.255.255.0
Default gateway	192.168.1.254
TCP Port	9912

4.2.5.1.5. WiFi network information

All important WiFi network information at a glance.

On this information page you will find the current WiFi settings of your module.

Info

Hier finden Sie die Netzwerkinformationen des ausgewählten WiFi-Produktes

MAC address	40:F5:20:44:60:E8
Board name (Hostname in DHCP mode)	WEU-O8-R8_W
WLAN status	starting WIFI connection..
WLAN active	<input checked="" type="checkbox"/>
IP address	0.0.0.0
Subnet mask	0.0.0.0
Default gateway	0.0.0.0
TCP port	9912
Router name	TESTssid
Password	TESTpwd

☒ Auto Refresh

FormCtnNetworkWiFiInfo (Ver. 1.01)

MAC address

The MAC address is the physical address of the product and is hardwired 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 needs a newer firmware.

WLAN active

Shows whether the module is connected via WLAN.

IP address, netmask, default gateway and TCP port

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

Router name

Shows which router name is used to connect via WLAN.

Password


Displays the used router password.

4.2.5.1.6. WiFi network settings

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

Konfiguration

Hier können Sie die Netzwerkeinstellungen des ausgewählten WiFi-Produktes ändern

Board name (Hostname in DHCP mode)	<input type="text" value="WEU-08-R8_W"/>
WLAN active	<input type="checkbox"/>
Router name	<input type="text" value="TESTssid"/>
Password	<input type="text" value="TESTpwd"/>
TCP port	<input type="text" value="9912"/> 
<input type="button" value="Restart network"/>	

Form für Netzwerk WiFi Config (Ver. 1.01)

Board name

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

This option is especially useful when using multiple modules.

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

(For more information about connecting the module by board name see chapter: Using the module selector).

WLAN active

With this option you can activate or deactivate the WLAN of your module.

Router name

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

If necessary, ask your system administrator for this.

Password

Here you can enter the router password of the router used.

If necessary, ask your system administrator for this.

TCP port

The TCP port used is shown here. A change of the port can only be made with the LAN network configurations.

Load factory settings

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

Factory settings	
Board name	Module dependent
WLAN active	off
Routername	TESTssid
Password	TESTpwd

4.2.5.1.7. WiFi WPS connection

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

To do this, click the WPS button on your router. Information on this can be found in the manual of your network device.

Click the "WPS Start" button in the Config module while your router is searching or press the button directly on the board of your WEU module.

More information about the CFG button see chapter CFG Button

If the connection is successful, the used router name and password will appear now.

WPS

Hier können Sie eine Verbindung per WPS-Funktion mit Ihrem WiFi-Modul herstellen

WLAN status

starting WPS connection..

Router name

Password

WPS-Start

WPS-Stop

Mit Hilfe der WPS-Funktion, lässt sich Ihr Modul mit nur wenigen Schritten schnell und einfach automatisch mit Ihrem Router verbinden.
Für eine erfolgreiche Verbindung mit dem Netzwerk müssen folgende Punkte durchgeführt werden:

1. Betätigen Sie die 'WPS-Taste' an Ihrem Router (Hilfe dazu finden Sie im Handbuch des Routers)
2. Klicken Sie anschließend auf den obigen 'WPS-Start'-Knopf im Module Config
3. Das Modul verbindet sich nun automatisch mit Ihrem Router

Der aktuellen Status Ihrer Verbindung wird bei 'WLAN Status' angezeigt.
Das Bedienen der 'Stop-Taste' beendet diesen Vorgang.

☒ Auto Refresh

FormCfmNetworksWiFiWPS (Ver. 1.01)

4.2.5.1.8. NTP configuration

Here changes can be made to the NTP service.

NTP-Konfiguration

Hier können Sie Änderungen am NTP Service vornehmen

NTP service active	<input checked="" type="checkbox"/>
Server	<input type="text" value="0.de.pool.ntp.org"/>
Port	<input type="text" value="123"/>
Timezone	<input type="text" value="(GMT) Greenwich Mean Time: Dublin, Edinburgh"/>

Werkseinstellung laden

Einstellung speichern

Form für Netzwerk NTP (Ver. 1.01)

NTP service active

If this option is enabled, 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. Here you can set the time zone to be used by the module.

Load factory settings

This resets the TCP encryption settings to the factory defaults. 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

4.2.5.1.9. Serial configuration

Here you can do the whole configuration of our products with serial interface.

Seriell

Hier können Sie die gesamte Konfiguration unserer Produkte mit serieller Schnittstelle vornehmen

Special mode active	<input type="checkbox"/>
Baud rate	<input type="text" value="625000"/>
RS485 modul address	<input type="text" value="0"/>
Echo active	<input type="checkbox"/>
Register modus active	<input type="checkbox"/>

FormCfjSeriell (Ver 1.01)

Preferred mode (Special mode)

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

Baud rate: 115200

Module-Nr. 0

Echo = Off

Register-Mode = On

Baud rate

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

625000

250000

125000

115200

57600

50000

38400

19200

9600

4800

2400

1200

600

300

RS485 module address

Address for identification in the RS485 bus.

Echo

Characters received serially are returned by the module.

Register mode

Deactivate the register mode to activate the text mode.

4.2.5.1.10. I/O channel names

Here you can set the channel names of your main or sub module.

I/O Kanal-Namen

Hier können Sie die Namen der einzelnen I/O-Kanäle ändern und speichern

Ch. 0	<input type="text" value="Kanal 0"/>	Ch. 8	<input type="text" value="Kanal 8"/>
Ch. 1	<input type="text" value="Kanal 1"/>	Ch. 9	<input type="text" value="Kanal 9"/>
Ch. 2	<input type="text" value="Kanal 2"/>	Ch. 10	<input type="text" value="Kanal 10"/>
Ch. 3	<input type="text" value="Kanal 3"/>	Ch. 11	<input type="text" value="Kanal 11"/>
Ch. 4	<input type="text" value="Kanal 4"/>	Ch. 12	<input type="text" value="Kanal 12"/>
Ch. 5	<input type="text" value="Kanal 5"/>	Ch. 13	<input type="text" value="Kanal 13"/>
Ch. 6	<input type="text" value="Kanal 6"/>	Ch. 14	<input type="text" value="Kanal 14"/>
Ch. 7	<input type="text" value="Kanal 7"/>	Ch. 15	<input type="text" value="Kanal 15"/>

Set default channel name

Einstellung speichern

FormCfnSubmoduleIOnames (Ver. 1.01)

You can individually name and save all channels of your main or sub module here.

Note:

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

Set default channel name

Writes the text shown above as a name suggestion into the text fields.

To apply it to the module, it must also be saved.

4.2.5.1.11. CAN configuration

4.2.5.1.11.1. CAN status

In this area you will find all information about the status of the CAN interface and the TX and RX packets.

All important information about your CAN interface is displayed here

Interface
Hier finden Sie Informationen über den Status des CAN-Interface.

CAN-Baudrate	1000000 Bit/sec
DT-CAN-CMD-MODE - Modul-Address	100 [hex]
DT-CAN-CMD-MODE - Response-Address	200 [hex]
CAN is active	1
Use Ext ID	1
CAN config mode selection	1

☐ Auto Refresh

FormCfoCANStatus (Ver. 1.00)

CAN-Baudrate

Displays the current baud rate of your CAN interface.

DT-CAN-CMD-MODE- Module-Address

DT-CAN-CMD-MODE - Response-Address

CAN is active

(This function is only displayed if it is supported by your module)

Use EXT ID

CAN config mode selection

All important statistics for the TX and RX clocks, such as the number of CAN packets sent and received and their transmission speed, are displayed here.

Statistik TX/RX

Hier finden Sie Informationen über die gesendeten und empfangenen CAN-Pakete.

DT-CAN-CMD-MODE

TX

RX

Frames total

0

0

TX-Modes	Frames total	Frames/Second	Ø Time/Frame	RX-Modes	Frames total
TX-1	0	0	-	RX-1	0
TX-2	0	0	-	RX-2	0
TX-3	0	0	-	RX-3	0
TX-4	0	0	-	RX-4	0
TX-5	0	0	-	RX-5	0
TX-6	0	0	-	RX-6	0
TX-7	0	0	-	RX-7	0
TX-8	0	0	-	RX-8	0

☒ Auto Refresh

FormCfoCANStatistic (Ver. 1.00)

4.2.5.1.11.2. CAN Main

In this area you can make settings directly on the CAN interface.

With the help of these settings the CAN interface can be configured.

Interface

Hier können Sie Einstellungen am CAN-Interface vornehmen.

Baudrate	<input type="text" value="1 MBit/s"/>
Address Mode	<input type="text" value="11 Bit Mode"/>
DT-CAN-CMD-MODE - Modul-Address [hex]	<input type="text" value="FFFFFF"/>
DT-CAN-CMD-MODE - Response-Address [hex]	<input type="text" value="FFFFFF"/>

Einstellung speichern

FormCfaCANConfiaStd (Ver. 1.00)

Baudrate

Here the baud rate can be set with which the module should communicate.

Address Mode

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

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

This module address defines under which address the module is identified in the CAN bus.

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

Response-Address specifies to which module address an acknowledgement is sent once a packet has been received.

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

I/O-Init

Hier können Sie die CAN-Einstellungen der angeschlossenen Submodule konfigurieren.

A/D Mode	16 Bit / $\pm 20V$
A/D Filter	None
D/A Mode	16 Bit / $\pm 10V$
Counter Mode	16 Bit Counter (default)
Timeout-Mode	Deactivate
Output Timeout	0.0 sec
CNT48 Mode	Read on rising edge (x1)
CNT48 Submode	Read
CNT48 Filter	20 ns

Einstellung speichern

FormCfCANConfIO (Ver. 1.00)

A/D Mode

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

A/D Filter

Here you can set the A/D filter.

D/A Mode

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

Counter Mode

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

Timeout-Mode

Here the timeout mode can be set. More information about the individual modes can be found in the chapter: DapiSpecialCMDTimeout. If no timeout is desired, 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 sub modes are available for selection here.

CNT48 Filter

Sets the filter, how long a signal must be at least, so that it is recognized as High. You can choose from 16 preset filters between 20ns and 5ms.

4.2.5.1.11.3. CAN TX/RX modes

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

Here you can make settings to the TX packets. Here you can make settings to the TX packets.

TX-Mode[1]

Hier können Sie Einstellungen der TX-Paket-Konfiguration ändern.

Activate	<input checked="" type="checkbox"/>
Trigger Mode	Interval
Interval	1 * 1ms
Address Mode	11 Bit Mode
Send to CAN-ID [hex]	200
Data to send	OPTO-IN 1-64

Werkseinstellung laden

Einstellung speichern

FormCfCANConfiaTX (Ver. 1.00)

Activate

Activates this TX mode

Trigger Mode

Specifies the mode in which the transmission is to take place. The modes "Interval", "RX event" and "Fast as possible" can be selected. RX event" and "Fast as possible".

Intervall

If the Interval mode is set, you can also specify the time interval at which the data is to be sent.

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 above example it is sent to address 0x200.

Data to send

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

Here you can make settings on the RX packages.

RX-Mode[1]

Hier können Sie Einstellungen der RX-Paket-Konfiguration ändern.

Activate	<input checked="" type="checkbox"/>
Address Mode	11 Bit Mode
Receive at CAN-ID [hex]	100
Data to send	Digital Out 1-64

Werkseinstellung laden

Einstellung speichern

FormCfCANConfiaRX (Ver. 1.00)

Activate

Activates this RX mode

Address Mode

Here the address mode 11-bit or 29-bit can be set.

Receive at CAN-ID[hex]

Specifies the receiver address. In the examples above a CAN packet is received at address 0x100.

Data to send

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 there are switched on or off.

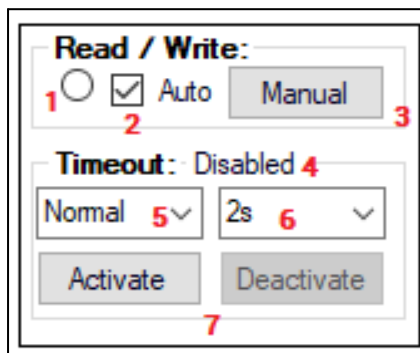
4.2.5.2. I/O-Test

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

4.2.5.2.1. Timeout test function

In the "Read/Write" area, settings can be made on the timeout.

These functions can be used to trigger a timeout event.



- 1 The field indicates by repeated flashing whether a connection to the module exists. If the field remains empty, communication is interrupted.
- 2 Removing the checkmark interrupts the connection and thus triggers a timeout. Checking the box again will re-establish the connection.
- 3 Since the user bob surface is not automatically updated in a timeout case, this can be triggered by clicking on the "Manual" button.
- 4 The current timeout status is displayed here.
- 5 Here the desired timeout mode can be set. For more information see chapter: **DapiSpecialCMDTimeout**.
- 6 Here you can set the time in which the timeout should be triggered.
- 7 "Activate" activates the timeout. Deactivate" deactivates it.

4.2.5.2.2. Digital In

Here you will find information about the digital inputs, as well as the input counter and the FlipFlop filter.

Digital In [0 .. 15]

Hier können Sie den Status der digitalen Eingangskanäle ablesen

	State on/off	Counter	FlipFlop		State on/off	Counter	FlipFlop
Kanal 0	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>	Kanal 8	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>
Kanal 1	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>	Kanal 9	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>
Kanal 2	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>	Kanal 10	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>
Kanal 3	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>	Kanal 11	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>
Kanal 4	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>	Kanal 12	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>
Kanal 5	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>	Kanal 13	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>
Kanal 6	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>	Kanal 14	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>
Kanal 7	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>	Kanal 15	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="checkbox"/>

☐ Read with reset

☐ Auto Refresh

FormIODigitalIn (Ver. 1.10)

State on/off

Shows the current state of each input channel.

Counter

Displays the counts of the input counters.

FlipFlop

Displays the change in input states since the last readout.

Read with reset

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

4.2.5.2.3. Digital Out

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

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

Digital Out [0 .. 15]

Hier können Sie die digitalen Ausgänge des Modules ein- oder ausschalten

	On / Off		Readback	Timer	
Kanal 0	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 1	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 2	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 3	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 4	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 5	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 6	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 7	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>

	On / Off		Readback	Timer	
Kanal 8	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 9	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 10	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 11	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 12	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 13	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 14	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>
Kanal 15	<input type="button" value="on"/>	<input type="button" value="off"/>	<input type="checkbox"/>	<input type="text" value="0 s"/>	<input type="button" value="set"/>

Invert DO-Timer
☒ on (data1) / off (data0)

☐ Auto Refresh

FormIODigitalOut (Ver. 1.10)

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

(Displayed only if supported by the module).

Enter a time (in seconds) in the timer area after which the relays are to be switched on or off. With "set" you start the timer.

Invert DO-Timer

If this option is activated, the relay will be deactivated after the timer has expired. If this option is disabled, the relay will be activated after the timer expires.

4.2.5.2.4. Analog In

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

Analog In [0 .. 15]

Hier können Sie den A/D - Mode der Kanäle ändern

	Value		Value
Kanal 0	<input type="text" value="0,027"/> V	Kanal 8	<input type="text" value="0,027"/> V
Kanal 1	<input type="text" value="0,026"/> V	Kanal 9	<input type="text" value="0,025"/> V
Kanal 2	<input type="text" value="0,025"/> V	Kanal 10	<input type="text" value="0,024"/> V
Kanal 3	<input type="text" value="0,025"/> V	Kanal 11	<input type="text" value="0,025"/> V
Kanal 4	<input type="text" value="0,025"/> V	Kanal 12	<input type="text" value="0,024"/> V
Kanal 5	<input type="text" value="0,025"/> V	Kanal 13	<input type="text" value="0,024"/> V
Kanal 6	<input type="text" value="0,025"/> V	Kanal 14	<input type="text" value="0,025"/> V
Kanal 7	<input type="text" value="0,025"/> V	Kanal 15	<input type="text" value="0,024"/> V

Set mode for all channels

Mode Readback

Set filter for all channels

Filter Readback

☒ Auto Refresh

FormIOAnalogIn (Ver. 1.10)

Value

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

Set mode for all channels

Selection of the voltage / current range for all channels in which to measure. Only modes 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 applied for all channels.

Filter Readback

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

4.2.5.2.5. Analog Out

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

Analog Out [0 .. 13]

Hier können Sie den D/A - Mode der Kanäle ändern

	Value	Mode	Readback
Kanal 0	<input type="text" value="0"/>	16 Bit / 0-10V ▾	<input type="text" value="0"/> V
Kanal 1	<input type="text" value="1"/>	16 Bit / 0-10V ▾	<input type="text" value="0,999908"/> V
Kanal 2	<input type="text" value="2"/>	16 Bit / 0-10V ▾	<input type="text" value="1,999969"/> V
Kanal 3	<input type="text" value="3"/>	16 Bit / 0-10V ▾	<input type="text" value="2,999878"/> V
Kanal 4	<input type="text" value="4"/>	16 Bit / 0-10V ▾	<input type="text" value="3,999939"/> V
Kanal 5	<input type="text" value="5"/>	16 Bit / 0-10V ▾	<input type="text" value="5"/> V
Kanal 6	<input type="text" value="6"/>	16 Bit / 0-10V ▾	<input type="text" value="5,999908"/> V
Kanal 7	<input type="text" value="7"/>	16 Bit / 0-10V ▾	<input type="text" value="6,999969"/> V
Kanal 8	<input type="text" value="8"/>	16 Bit / 0-10V ▾	<input type="text" value="7,999878"/> V
Kanal 9	<input type="text" value="9"/>	16 Bit / 0-10V ▾	<input type="text" value="8,999939"/> V
Kanal 10	<input type="text" value="10"/>	16 Bit / 0-10V ▾	<input type="text" value="9,999847"/> V
Kanal 11	<input type="text" value="11"/>	16 Bit / 0-10V ▾	<input type="text" value="9,999847"/> V
Kanal 12	<input type="text" value="12"/>	16 Bit / 0-10V ▾	<input type="text" value="9,999847"/> V
Kanal 13	<input type="text" value="13"/>	16 Bit / 0-10V ▾	<input type="text" value="9,999847"/> V

Set mode for all channels

16 Bit / 0-10V ▾

Set

☒ Auto Refresh

FormIOAnalogOut (Ver. 1.10)

Value

Here you can enter the value to be output at the respective D/A channel.

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.

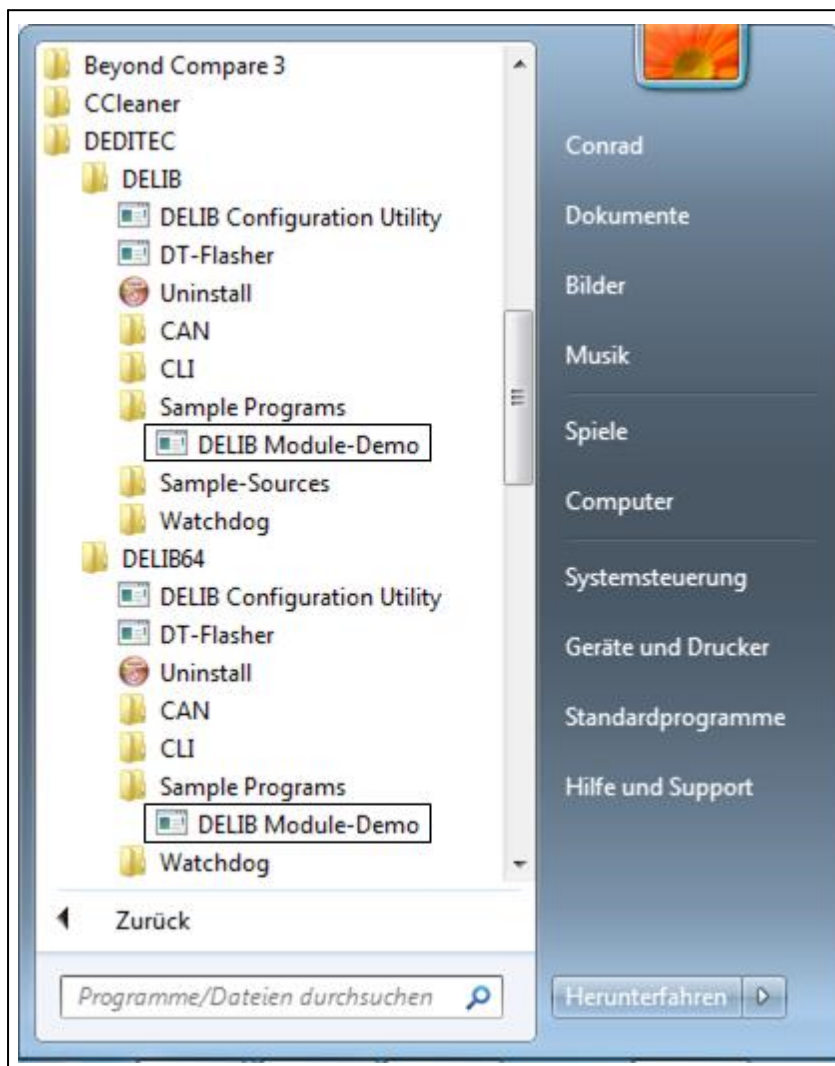
Set mode for all channels

This function allows you to change the voltage/current range for all available channels at once.

4.2.6. DELIB Module Demo

After installing the DELIB driver library, the DELIB Module Demo program can be started in the following way:

Start → Programs → DEDITEC → DELIB or DELIB64 → Sample Programs → DELIB Module Demo.



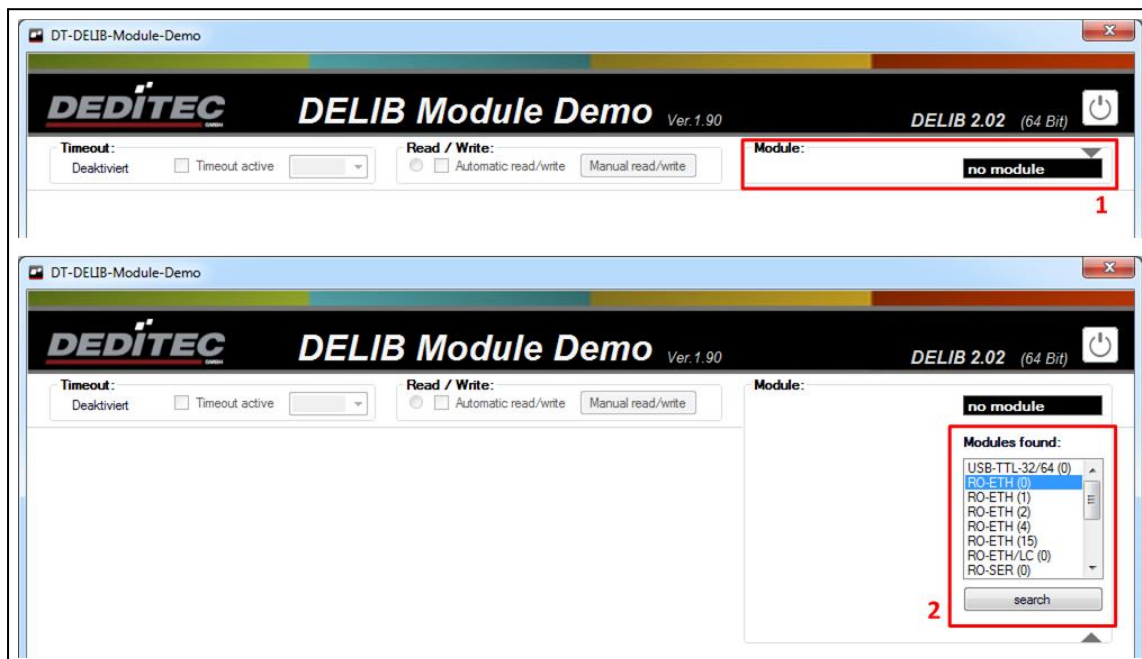
The program DELIB Module Demo is an all-in-one tool with which all I/Os of all products from our S&R range can be controlled and tested.

4.2.6.1. Module selection

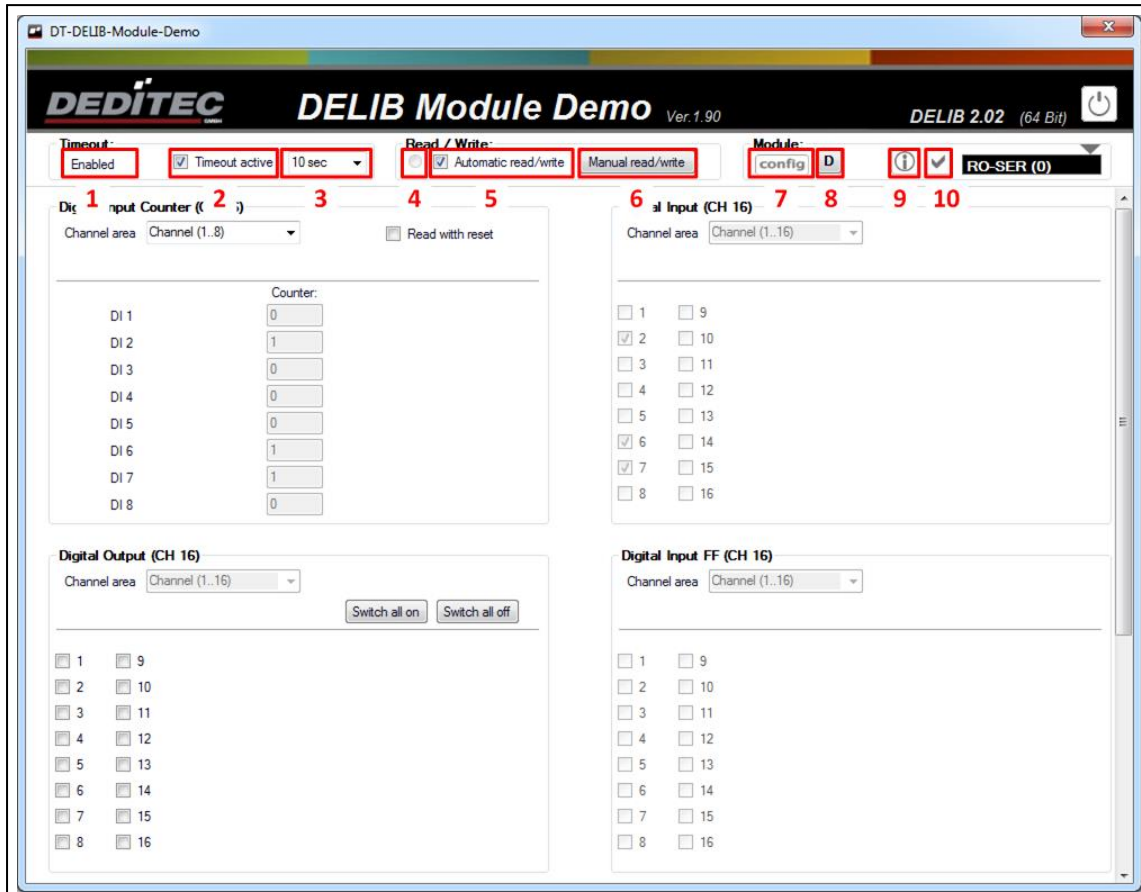
A module must be selected when the program starts.

1. Click on the "Module Selector". A list of the available/connected modules is displayed.

2. Now select the desired module.



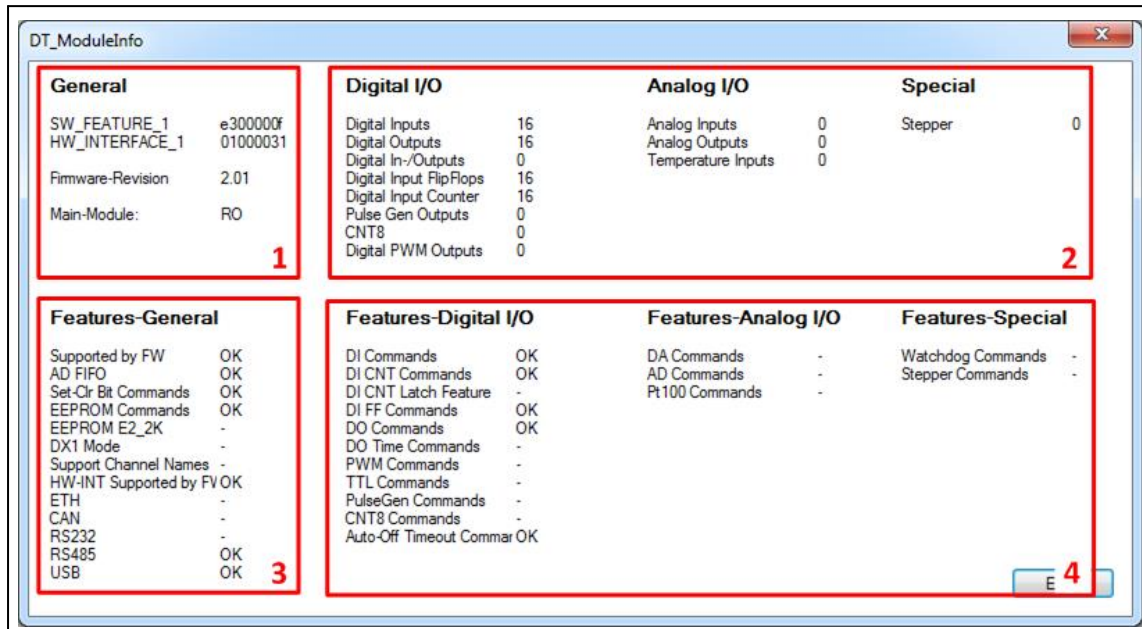
4.2.6.2. General



1. Timeout status ("Disabled", "Enabled" or "Occured")
2. Enables or disables the timeout protection.
3. timeout time for timeout protection.
4. status for "Automatic read/write" (flashes when "Automatic read/write" is activated).
5. The check mark for "Automatic read/write" determines whether the measurement data is to be read/written automatically.

6. Manual read/write. Only active if "Automatic read/write" is deactivated.
7. Here, the currently selected module can be configured via the DELIB Configuration Utility.
8. If supported by the module, you can get detailed debug information here.

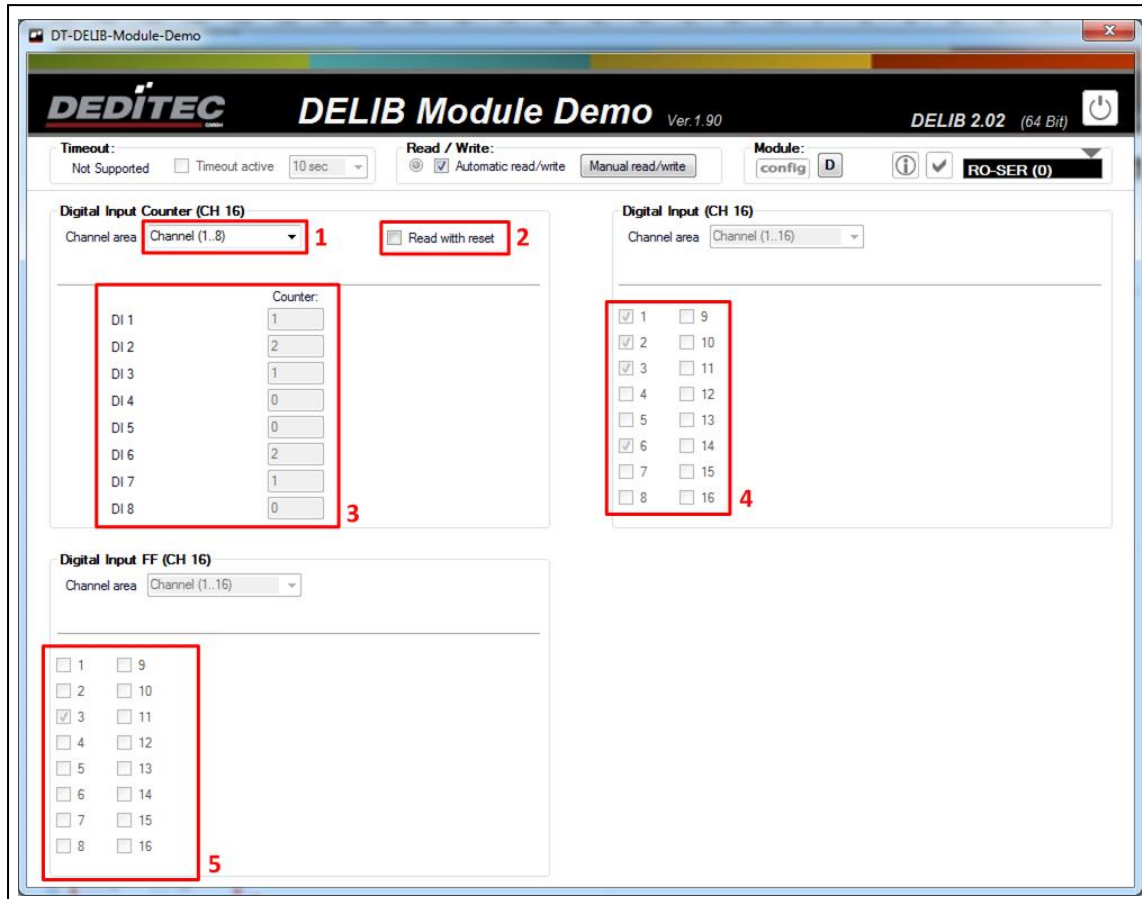
4.2.6.2.1. Module Info



This example shows the extended information of the RO-SER-O16-M16 module.

1. general information of the selected module.
2. number of connected I/O channels.
3. overview of the supported interface DELIB commands.
4. overview of the supported I/O DELIB commands.

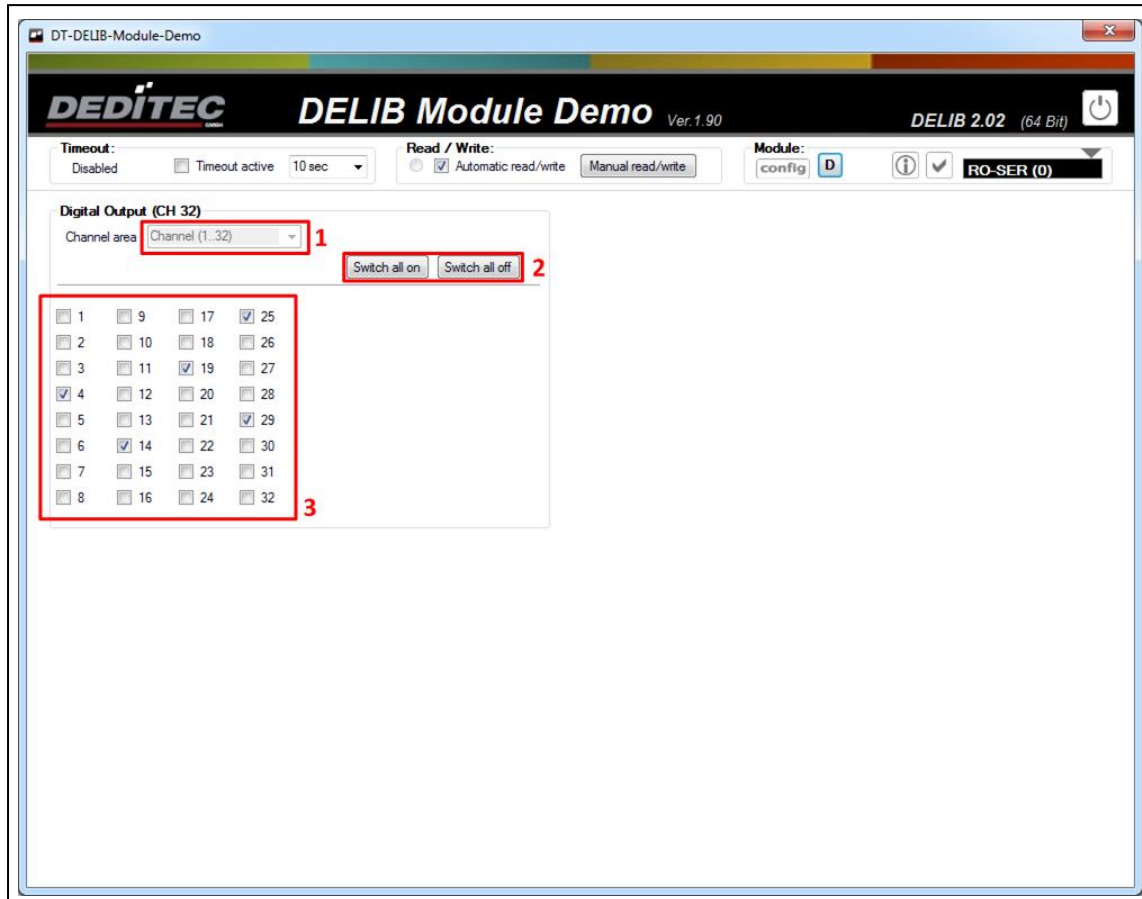
4.2.6.3. Digital Input



This example shows the digital inputs of a RO-SER-016 module.

1. Selection of the channel range to be displayed.
2. The "Read with reset" checkbox determines whether the counters are reset the next time they are read.
3. counts of the input counters.
4. states of the inputs.
5. change of the input states (since the last readout).

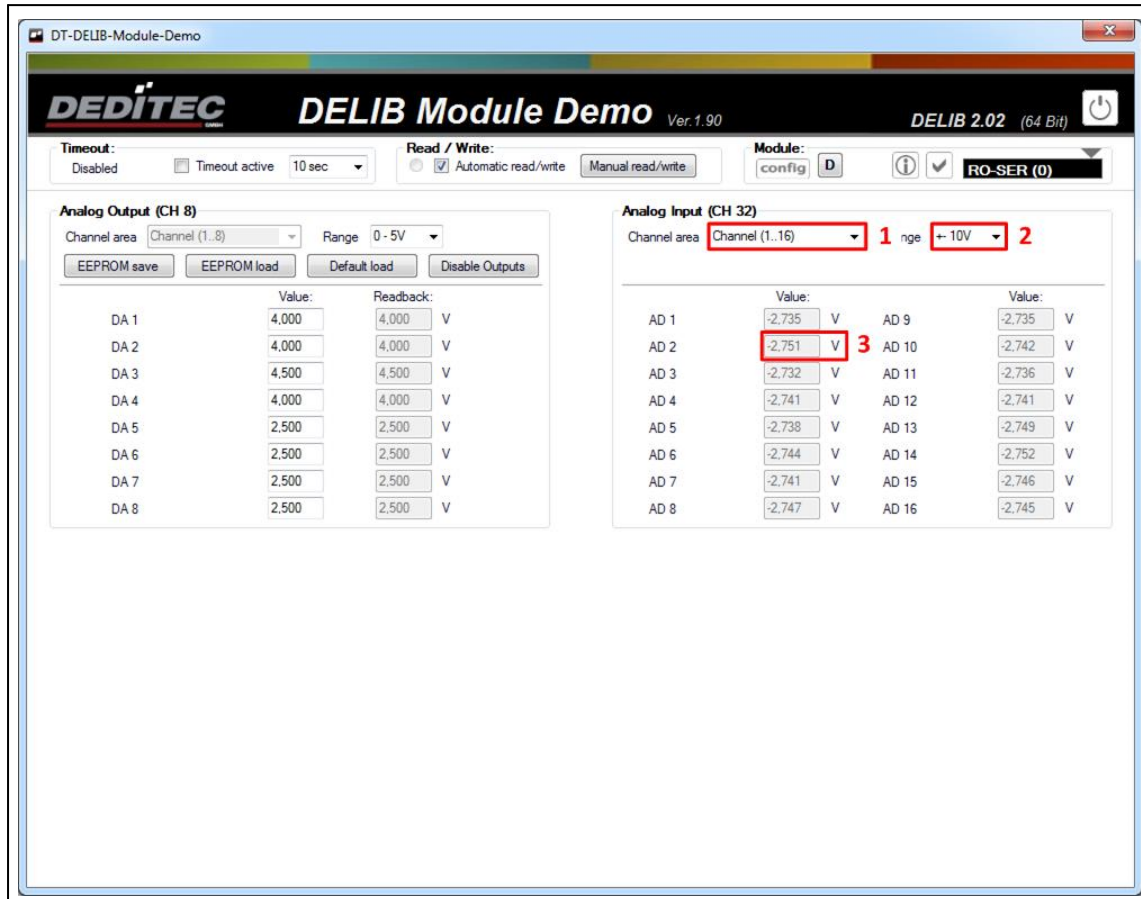
4.2.6.4. Digital Output



This example shows the digital outputs of a RO-SER-M32 module.

1. Selection of the channel range to be displayed.
2. This switches all outputs of the current channel range on or off.
3. Here you can selectively switch on or off certain outputs.

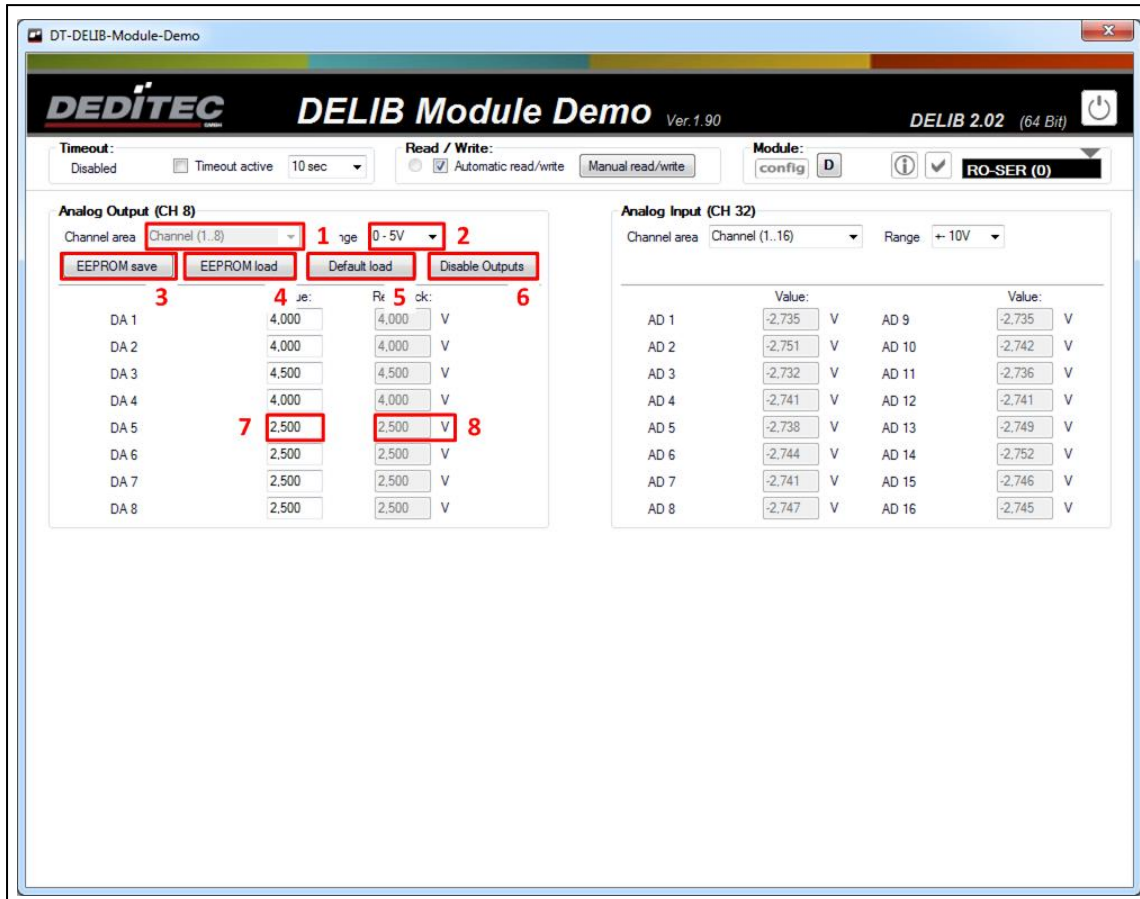
4.2.6.5. Analog Input



This example shows the analog inputs of a RO-SER-AD32-DA8 module.

1. selection of the channel range to be displayed. 2. selection of the voltage/current range to be measured.
2. selection of the voltage/current range to be measured. If the mode is not supported, the message "illegal" appears to the right of the drop-down menu.
3. Current A/D value on A/D channel 2.

4.2.6.6. Analog Output



This example shows the analog outputs of a RO-SER-AD32-DA8 module.

1. selection of the channel range to be displayed.
2. selection of the voltage/current range in which the values are to be output. If the mode is not supported, the message "illegal" appears to the right of the drop-down menu.
3. The currently output voltages/currents are stored in the module. They are output directly when the module is started.
4. Loads the voltages/currents that are stored in the module.

5. Resets the voltages/currents that are output at module start to the factory setting.
6. Disables the outputs. No voltages/currents are output.
7. 2.5V are to be output at D/A channel 5 (setpoint).
8. Reads back D/A channel 5 (actual value).

4.2.7. CAN Configuration Utility

Note:

To be able to configure a CAN module, it must first be put into software mode.

[Konfiguration Produkte der RO-Serie](#)

[Konfiguration Produkte der BS-Serie](#)

The CAN configuration utility allows easy configuration of products with a CAN interface. It is possible to transfer configurations to modules and read them out.

Additionally, the automatic receive mode (Auto-RX) and the automatic transmit mode (Auto-TX) can be configured.

The Auto-TX mode allows cyclic transmission of data packets, optionally with analog or digital input states to other CAN addresses. Alternatively, a trigger event can be defined. Here, a data packet is not sent until a data packet has been received on a certain CAN ID beforehand (e.g. CAN-Sync on ID 0x80).

With the Auto-RX mode, on the other hand, received data packets are forwarded directly to analog or digital outputs. For example, relay outputs can be set via another CAN bus station.

The following settings can be changed:

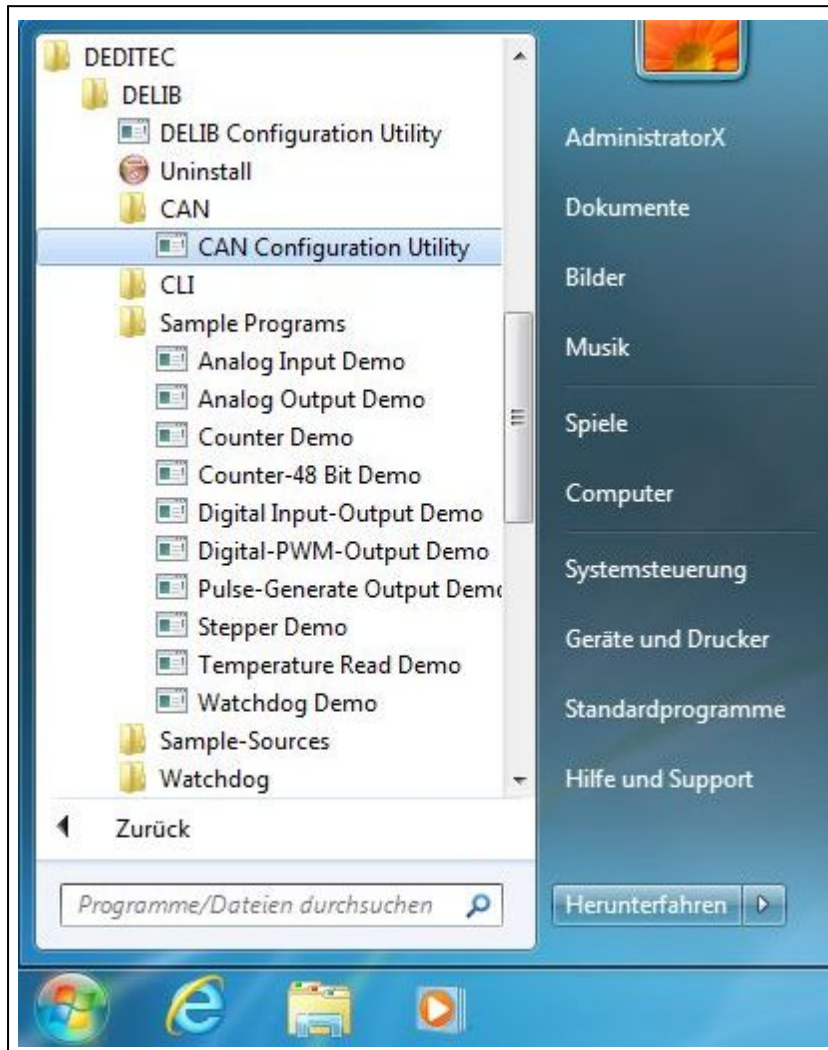
- Baud rate
- Address mode
- Module ID
- Response-ID
- Modes with which submodules are started
- Automatic transmit mode (TX mode)
- Automatic receive mode (RX mode)

The following pages show step by step how to configure a CAN module.

4.2.7.1. Module selection

Start the CAN configuration utility via:

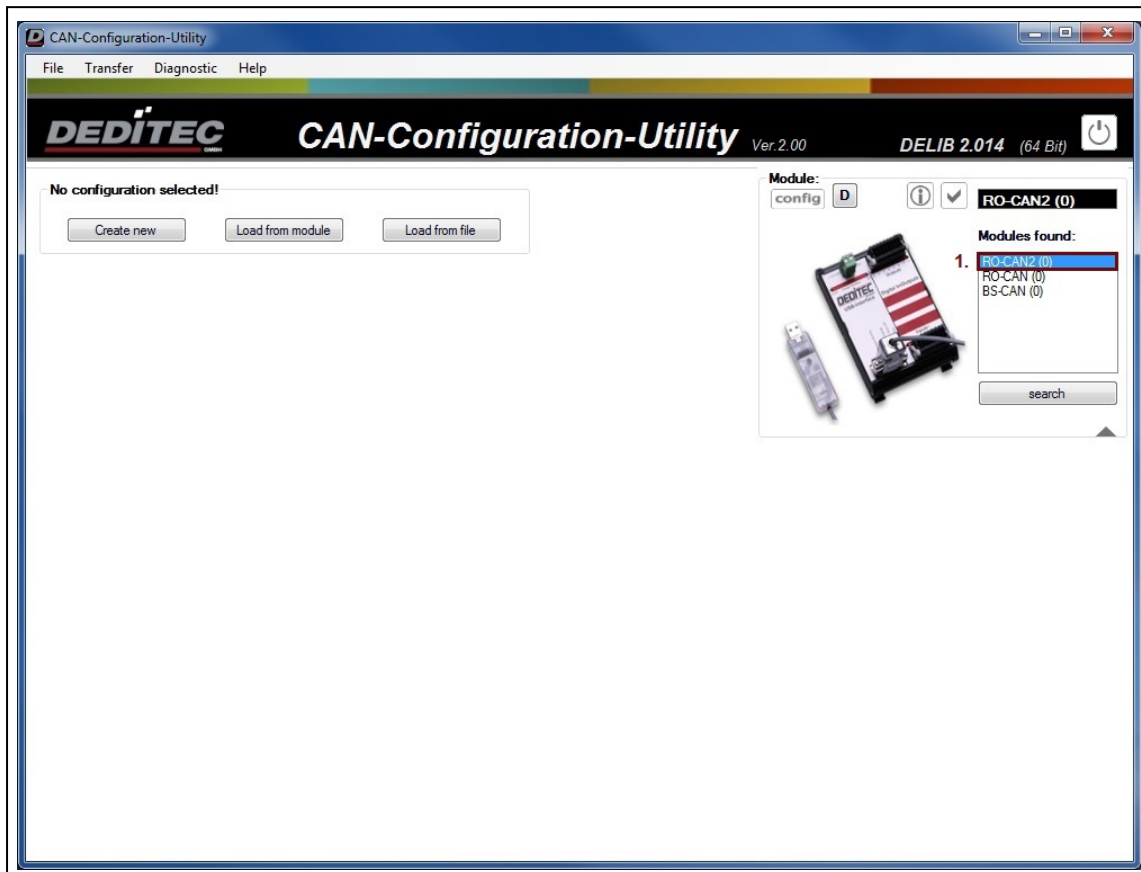
Start → Programs → DEDITEC → DELIB → CAN



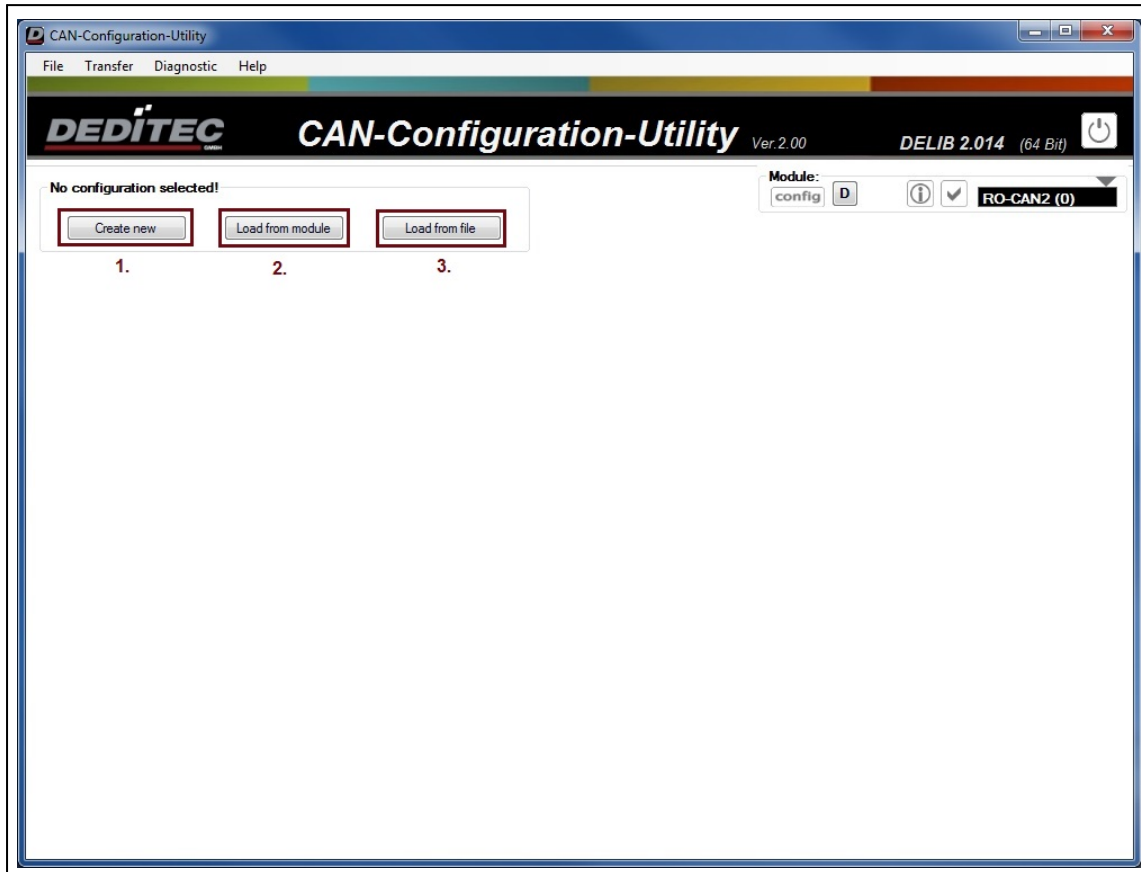
Start the CAN configuration utility via:

Start → Programs → DEDITEC → DELIB → CAN

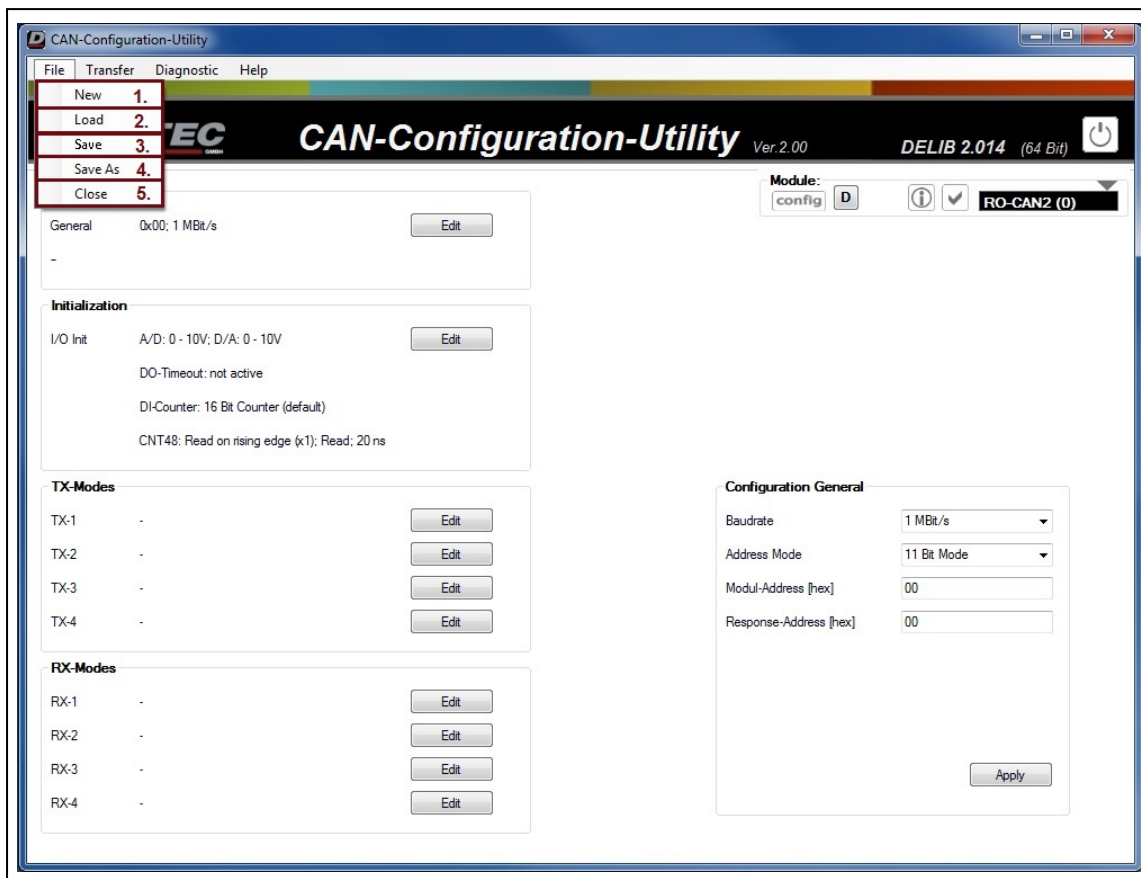
1. Select an appropriate CAN module (e.g. the RO-CAN2) in the "Module Selection".



4.2.7.2. Create new configuration, load, save



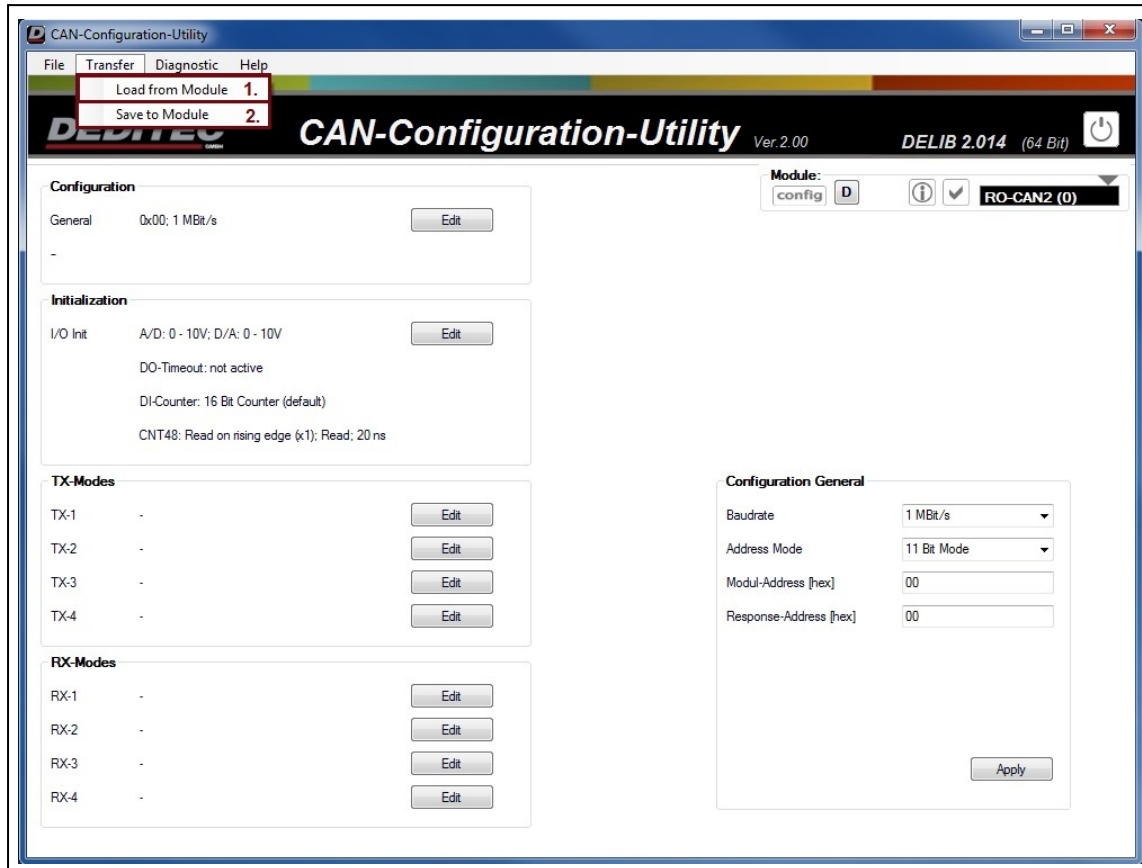
1. Via "Create new" a new CAN configuration can be created.
2. Load from Module" can be used to read out the configuration currently present on the module.
3. "Load from File" allows to open CAN configuration files previously stored on a data carrier.



A click on "File" in the menu bar opens a submenu:

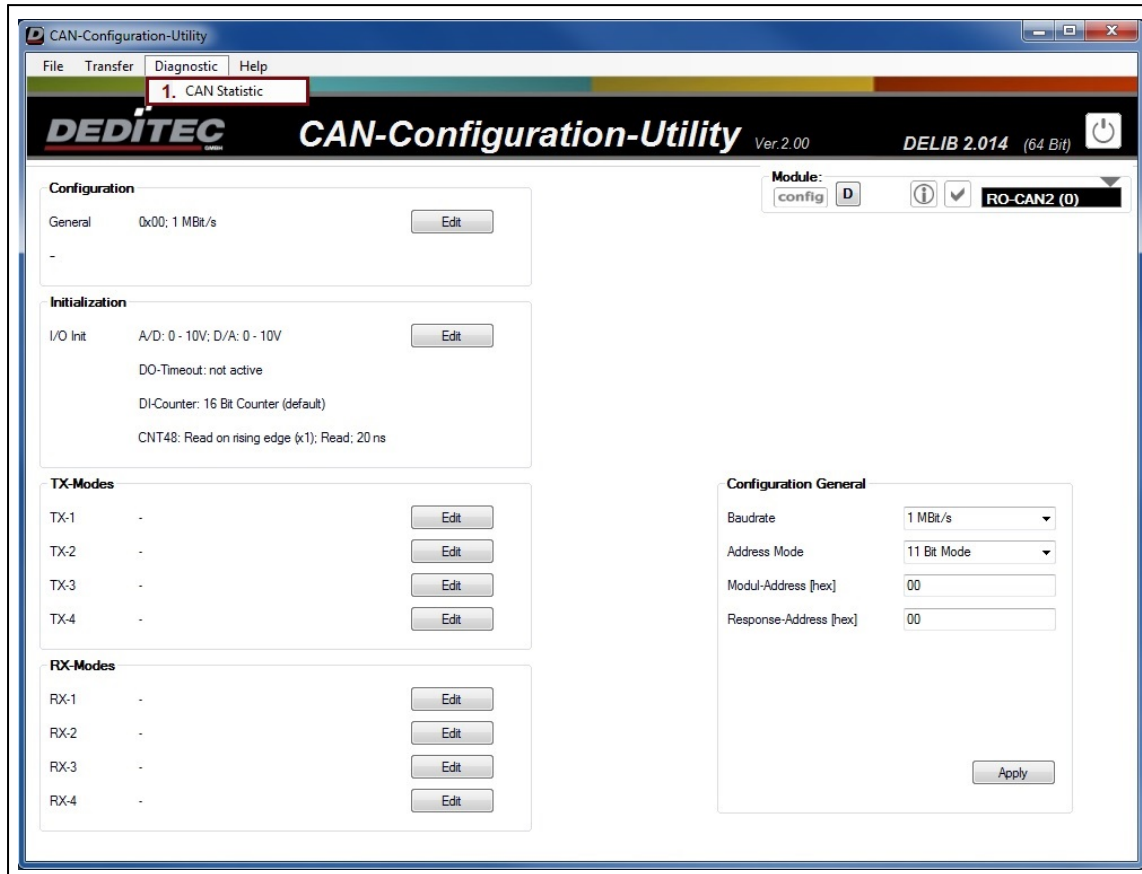
1. The menu item "New" can be used to create a new CAN configuration.
2. "Load" offers the possibility to open previously saved CAN configuration files.
3. If a CAN configuration file has been opened, "Save" can be used to overwrite the file and save current changes.
4. By clicking on "Save as" the CAN configuration can be saved in a new file.
5. Here the CAN configuration utility can be closed.

4.2.7.3. Transfer configuration to the module



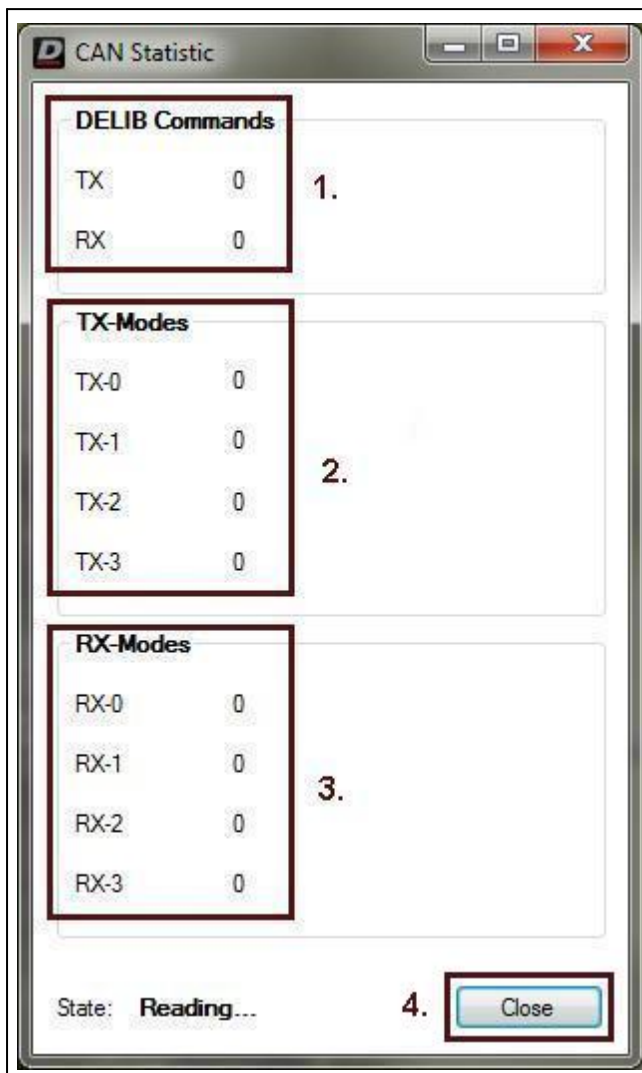
1. Via "Load from Module" the current configuration of the selected module can be read out.
2. "Save to Module" transfers the current CAN configuration to the selected module.

4.2.7.4. Query statistics from module



1. Via the menu item CAN-Statistic an information window can be displayed, which shows the number of sent and received TX and RX packets. Also the number of sent and received DELIB commands is shown.

The following window shows the statistics:



1. This area shows the number of DELIB commands sent and received.
2. For TX mode, the number of sent packets is listed here.
3. The received RX packets are displayed here.
4. Via "Close" the window can be closed.

Note

The CAN statistics continue to count in the background even if the window is closed. When the module is restarted, the statistics are reset to zero.

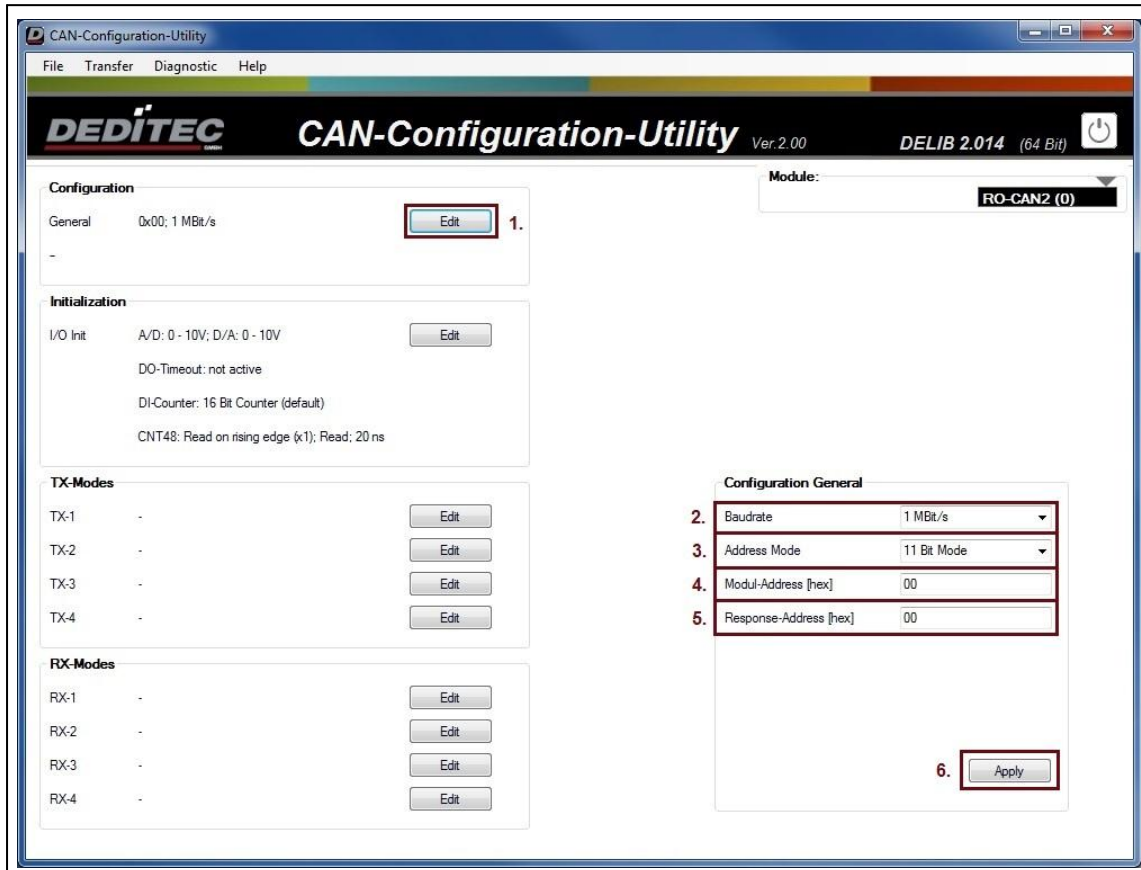
4.2.7.5. Configuration

For each CAN module 4 different configurations for TX and RX packets can be created.

It is also possible to define in which mode submodules are started. For example, it is possible that an AD module is started in a measuring range of 0-5V and not in the standard measuring range of $\pm 10V$.

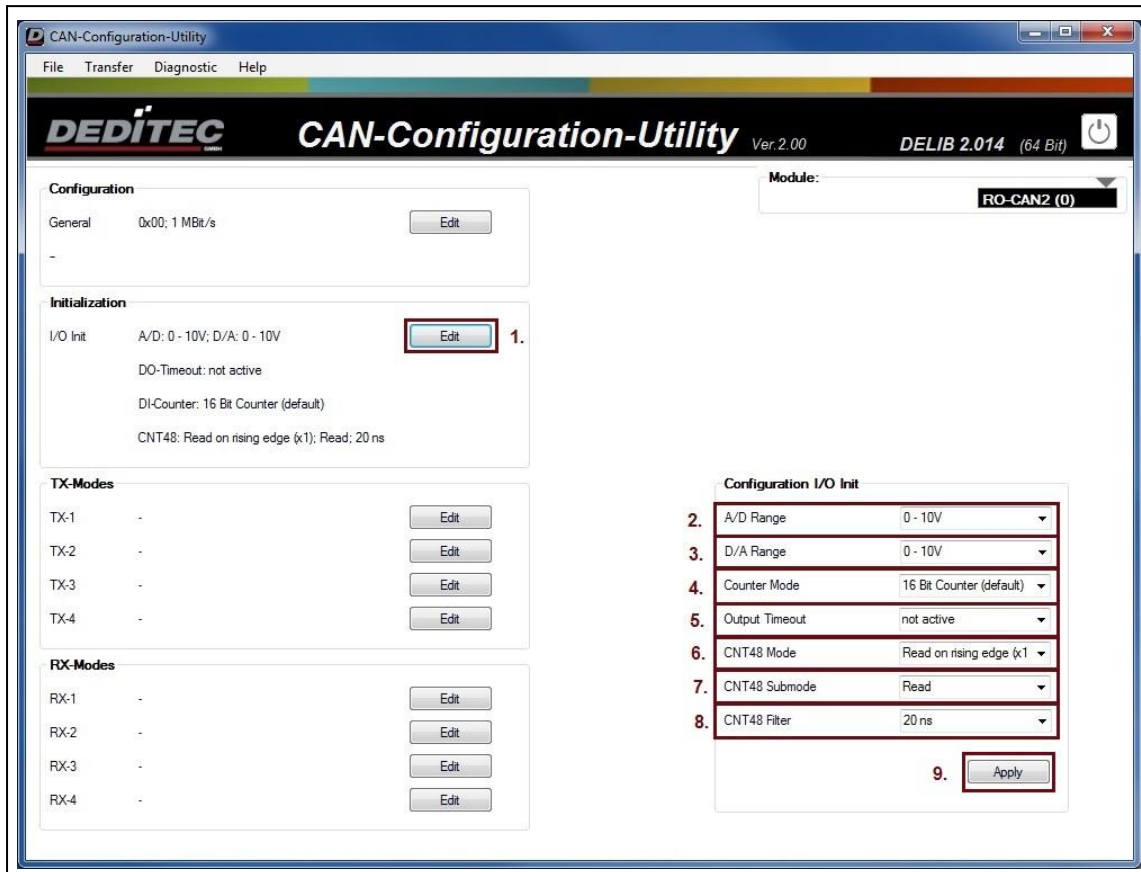
On the following pages it is shown how the different modes are configured.

4.2.7.5.1. Module configuration



1. Via the menu item "Edit" an additional window is opened, where the configuration can be done.
2. Here the baud rate can be set, with which the module should communicate.
3. The Address Mode defines how many bits are used for addressing.
4. The module address defines under which address the module is identified in the CAN bus.
5. The Response Address specifies to which module address an acknowledgement is sent as soon as a packet has been received.
6. Via "Apply" the changes are taken over.

4.2.7.5.2. I/O configuration



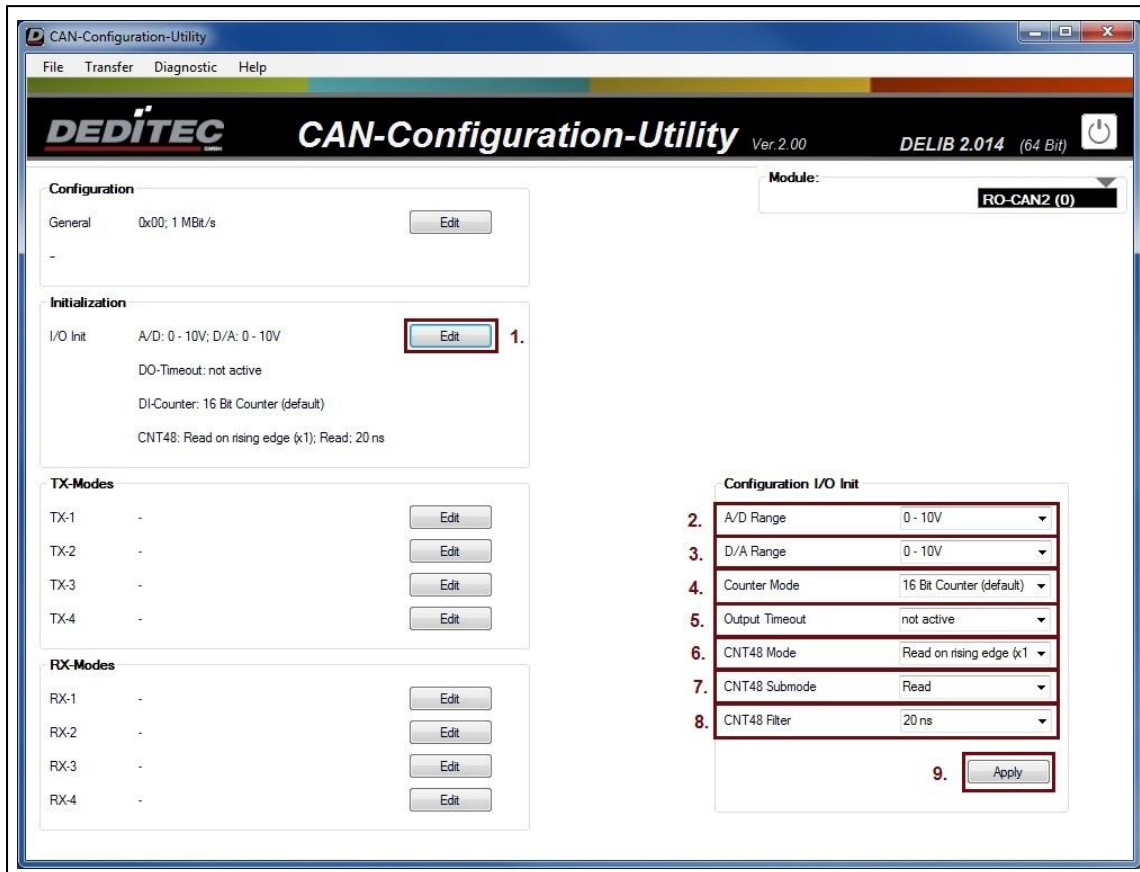
These settings are used to configure the connected submodules. The respective filter / mode in which the connected submodules are started can be set.

Note:

If the corresponding submodules are not present, the settings have no effect.

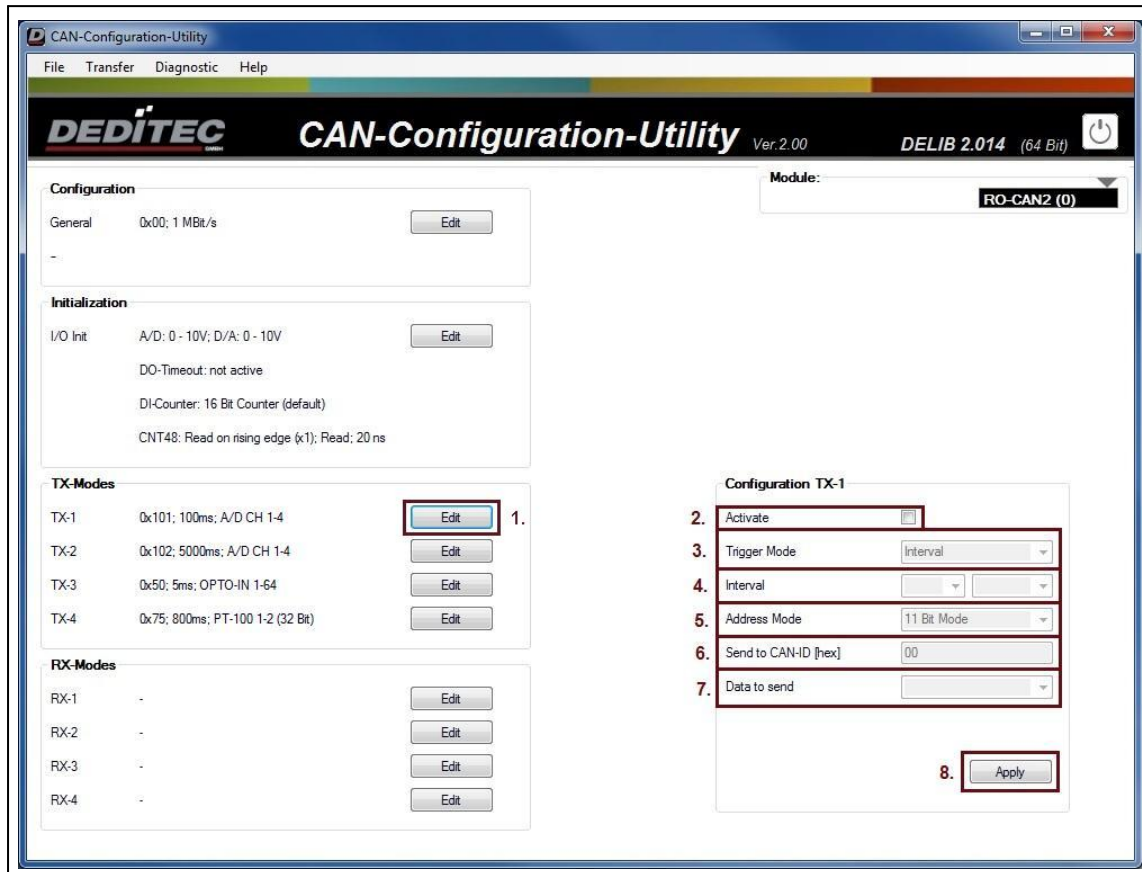
1. Via the menu item "Edit" an additional window is opened, in which the configuration can be made.
2. The value range specifies the range in which analog signals, digital (e.g. in the range 0-10V) are converted.
3. The value range specifies the range in which digital signals, analog (e.g. in the range 0-10V) are converted.
4. Is responsible for the counter mode of O8-R8 modules. Optionally, the counter can be incremented with 16 bits or incremented and decremented with 8 bits each.

5. Specifies the time after which the outputs switch off if a module can no longer be reached. If no timeout is desired, please select the setting "not active".



6. Sets which counter mode (only for RO-CNT8 modules) should be used. There are 5 different modes to choose from. A detailed description of the modes can be found in the "RO-CNT8" manual.
7. Depending on the mode selected under 6., corresponding sub-modes are available for selection here. A detailed description of the sub modes can be found in the manual "RO-CNT8".
8. Sets the filter, how long a signal must be at least, so that it is recognized as High. You can choose from 16 preset filters between 20ns and 5ms (only for RO-CNT8 modules).
9. Via "Apply" the changes are taken over.

4.2.7.5.3. TX configuration



Up to 4 independent TX modes can be set. The configuration is identical for all modes. The following example shows the configuration for the first TX mode.

1. The "Edit" menu item opens an additional window in which the configuration can be made.
2. To activate the configuration, the check mark must be set here.
3. The trigger mode specifies under which condition data is sent. Either in interval or in dependence of a received RX packet.
4. Here the interval is configured (if under point 3. "Interval" selected), in which packets are sent.
5. The Address Mode defines how many bits are used for addressing.
6. Here it is defined to which module address CAN packets are sent.
7. Under this point it is defined what kind of data is sent to the address configured under point 6. Such as → the status of the digital inputs or → the voltage of A/D channels.
8. Via "Apply" the changes are taken over.

4.2.7.5.3.1. Example Interval

Configuration TX-1

Activate	<input checked="" type="checkbox"/>
Trigger Mode	Interval
Interval	1 * 1sec
Address Mode	11 Bit Mode
Send to CAN-ID [hex]	0x100
Data to send	OPTO-IN 1-64

Apply

The example contains the following settings:

In the interval of one second the data of the digital inputs 1-64 are sent to the CAN address 0x100.

4.2.7.5.3.2. Trigger example

Configuration RX-1

Activate

☒

Address Mode

11 Bit Mode

▼

Receive at CAN-ID [hex]

200

Data to send

Trigger Auto TX 1

▼

Apply

Configuration TX-1

Activate

☒

Trigger Mode

RX-Event

▼

Interval

▼

▼

Address Mode

11 Bit Mode

▼

Send to CAN-ID [hex]

100

Data to send

A/D CH 1-4

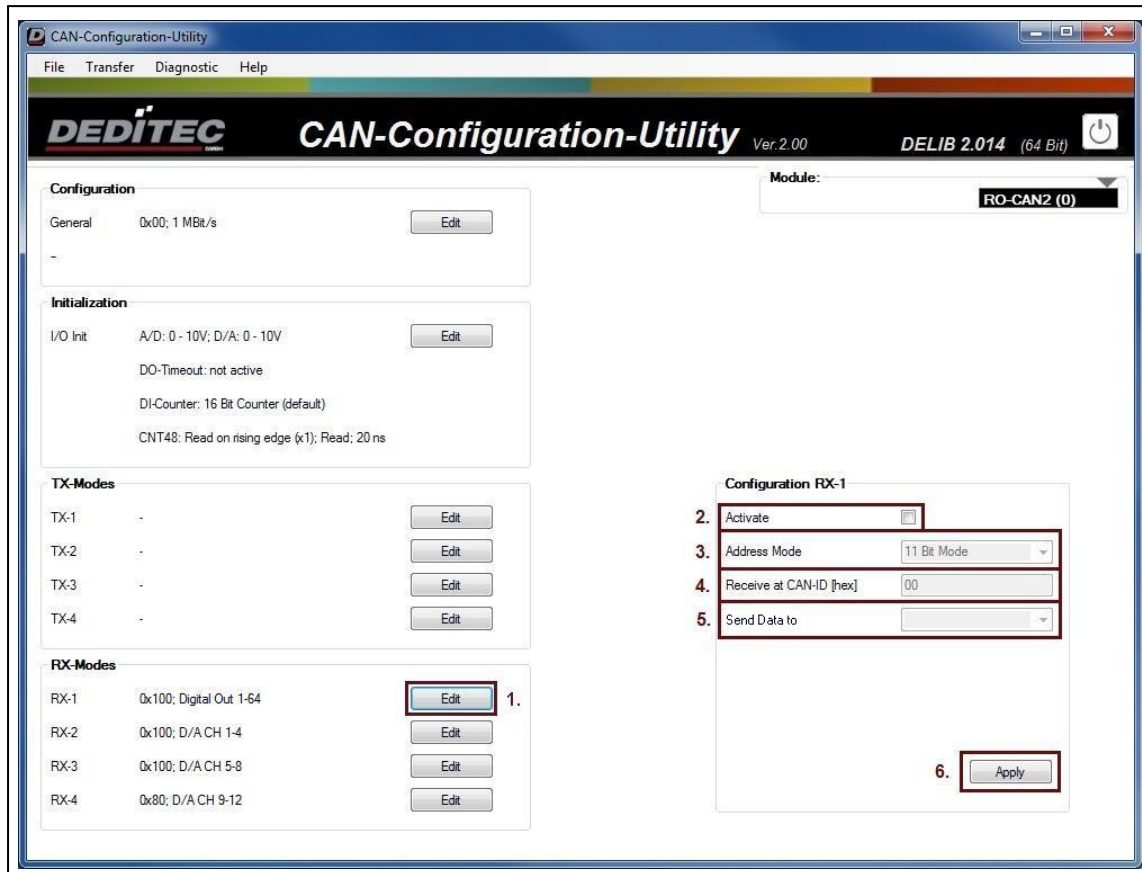
▼

Apply

The example contains the following settings:

If data is received on the CAN address 0x200, TX-1 is executed (picture above), which sends the data of the A/D channels 1-4 to the CAN address 0x100 (picture below).

4.2.7.5.4. RX configuration



Up to 4 independent RX modes can be set. The configuration is identical for all modes. The following example shows the configuration for the first RX mode.

1. The "Edit" menu item opens an additional window in which the configuration can be made.
2. To activate the configuration, the check mark must be set here.
3. The Address Mode specifies how many bits are used for addressing.
4. Here it is defined on which CAN-ID the data packets are expected.
5. If data was received on the ID configured under point 4., it is defined here which action is to be executed by the module. Like e.g. → automatically set the output voltage at an analog output module or → switch relay outputs.
6. Via "Apply" the changes are taken over.

4.2.7.5.4.1. Example RX-DA

Configuration RX-1

Activate	<input checked="" type="checkbox"/>
Address Mode	11 Bit Mode
Receive at CAN-ID [hex]	201
Data to send	D/A CH 5-8

Apply

The example contains the following settings:

If a CAN packet was received at address 0x201, the content of the data packet is set at the analog outputs 5-8, considering the selected D/A mode.

4.2.7.5.4.2. Example RX-DO

Configuration RX-1

Activate	<input checked="" type="checkbox"/>
Address Mode	11 Bit Mode ▼
Receive at CAN-ID [hex]	100
Data to send	Digital Out 1-64 ▼

Apply

The example contains the following settings:

If a CAN packet was received at address 0x100, the content of the data packet is forwarded to the digital outputs 1-64, whereupon the outputs are switched on or off there.

4.2.7.6. Structure of the CAN packages

The following pages show how the CAN packets of the different I/O modules are structured. Also shown is how A/D and D/A values can be calculated.

The data of the CAN packets are each 8 bytes in size. Please refer to the following pages to find out which data can be found at which position.

4.2.7.6.1. Digital inputs

Structure of an 8 byte long CAN packet:

CAN-Data-Byte	Content
1	DI channel 1-8 (Bit 0-7)
2	DI channel 9-16 (Bit 0-7)
3	DI channel 17-24 (Bit 0-7)
4	DI channel 25-32 (Bit 0-7)
5	DI channel 33-40 (Bit 0-7)
6	DI channel 41-48 (Bit 0-7)
7	DI channel 49-56 (Bit 0-7)
8	DI channel 57-64 (Bit 0-7)

4.2.7.6.2. Digital outputs

Structure of an 8 byte long CAN packet:

CAN-Data-Byte	Content
1	DO channel 1-8 (Bit 0-7)
2	DO channel 9-16 (Bit 0-7)
3	DO channel 17-24 (Bit 0-7)
4	DO channel 25-32 (Bit 0-7)
5	DO channel 33-40 (Bit 0-7)
6	DO channel 41-48 (Bit 0-7)
7	DO channel 49-56 (Bit 0-7)
8	DO channel 57-64 (Bit 0-7)

4.2.7.6.3. Digital input counter (16-bit)

Structure of an 8 byte long CAN packet:

CAN-Data-Byte	Content
1	DI counter 1 (Bit 0-7)
2	DI counter 1 (Bit 8-15)
3	DI counter 2 (Bit 0-7)
4	DI counter 2 (Bit 8-15)
5	DI counter 3 (Bit 0-7)
6	DI counter 3 (Bit 8-15)
7	DI counter 4 (Bit 0-7)
8	DI counter 4 (Bit 8-15)

4.2.7.6.4. Digital input counter (48-bit) - 32-bit packet

Structure of an 8 byte long CAN packet:

CAN-Data-Byte	Content
1	CNT8 counter 1 (Bit 0-7)
2	CNT8 counter 1 (Bit 8-15)
3	CNT8 counter 1 (Bit 16-23)
4	CNT8 counter 1 (Bit 24-31)
5	CNT8 counter 2 (Bit 0-7)
6	CNT8 counter 2 (Bit 8-15)
7	CNT8 counter 2 (Bit 16-23)
8	CNT8 counter 2 (Bit 24-31)

Note:

Note that only the first 32 bits of each of the two 48-bit input counters can be sent automatically in a CAN packet.

4.2.7.6.5. Digital input counters (48-bit) - 64-bit package

Structure of an 8 byte long CAN packet:

CAN-Data-Byte	Bit	Content
1	0-7	CNT8 counter 1 (Bit 0-7)
2	0-7	CNT8 counter 1 (Bit 8-15)
3	0-7	CNT8 counter 1 (Bit 16-23)
4	0-7	CNT8 counter 1 (Bit 24-31)
5	0-7	CNT8 counter 1 (Bit 31-39)
6	0-7	CNT8 counter 1 (Bit 40-47)
7	0-3	CNT8 counter 1 Counter modec
7	4-7	CNT8 counter 1 Sub-Modus
8	0	CNT8 counter 1 Input state (0/1)
8	1-3	Not used
8	4-7	CNT8 counter 1 Input filter

4.2.7.6.6. Analog inputs / outputs

4.2.7.6.6.1. Analog inputs

Structure of an 8 byte long CAN packet:

CAN-Data-Byte	Content
1	A/D channel 5 (Bit 0-7)
2	A/D channel 5 (Bit 8-15)
3	A/D channel 6 (Bit 0-7)
4	A/D channel 6 (Bit 8-15)
5	A/D channel 7 (Bit 0-7)
6	A/D channel 7 (Bit 8-15)
7	A/D channel 8 (Bit 0-7)
8	A/D channel 8 (Bit 8-15)

The value range of an A/D converter specifies in which range analog signals (e.g. in the range 0-5V) are digitally converted. The settings regarding the value range can be made in the CAN configuration utility.

Remark:

The hexadecimal value FFFF always indicates the upper limit of a value range, the value 0000 the lower limit.

Formula (A/D mode +/-10V, +/-5V or +/-2.5V)

Voltage = (value * (max. voltage value * 2) / 0xFFFF) - max. voltage value

Formula (A/D mode 0..10V, 0..5V or 0..2.5V)

Voltage = value * max. voltage value / 0xFFFF

Formula (A/D mode 0..20mA, 4..20mA or 0..24mA)

Current = value * 25 / 0xFFFF

4.2.7.6.6.2. Analog outputs

Structure of an 8 byte long CAN packet:

CAN-Data-Byte	Content
1	D/A channel 1 (Bit 0-7)
2	D/A channel 1 (Bit 8-15)
3	D/A channel 2 (Bit 0-7)
4	D/A channel 2 (Bit 8-15)
5	D/A channel 3 (Bit 0-7)
6	D/A channel 3 (Bit 8-15)
7	D/A channel 4 (Bit 0-7)
8	D/A channel 4 (Bit 8-15)

The value range of a D/A converter specifies in which range digital signals are converted analog (e.g. in the range 0-5V). The settings regarding the value range can be made in the CAN configuration utility.

Remark:

The hexadecimal value FFFF always indicates the upper limit of a value range, the value 0000 the lower limit.

Note:

The output to a current range is only possible with modules that also support this mode.

4.2.7.6.6.3. Examples

Example voltage range $\pm 10\text{V}$

Value (hex)	Voltage
FFFF	+10 V
8000	0 V
0000	-10V

Calculation example:

CAN-Data-Byte0, 1 = 0x4711[hex]

Value range = $\pm 10\text{ V}$

Calculation:

Voltage = $(0x4711 * (10 * 2) / 0xFFFF) - 10 = -4,45\text{ V}$

Example voltage range 0-5V

Value (hex)	Voltage
FFFF	+5 V
8000	+2,5 V
0000	0 V

Calculation example:

CAN-Data-Byte0, 1 = 0x4711[hex]

Value range = 0-5 V

Calculation:

Voltage = $0x4711 * 5 / 0xFFFF = 1,38 \text{ V}$

Example voltage range

Value (hex)	Current
FFFF	25 mA
8000	12,5 mA
0000	0 mA

Calculation example:

CAN-Data-Byte0, 1 = 0x4711[hex]

Value range = 0..20 mA, 4..20 mA oder 0..24 mA

Calculation:

Current = $0x4711 * 25 / 0xFFFF = 6,94 \text{ mA}$

Note:

Please note that for an Auto-TX package with set current range, the value of an A/D channel in the CAN package refers to the 0..25mA mode.

4.2.7.6.7. Temperature inputs

This example shows the structure of an Auto-TX package with the settings for PT-100 channel 1 and 2.

CAN-Data-Byte	Bit	Content
1	0-7	Value channel 1 (bit 0-7)
2	0-6	Value channel 1 (bit 8-14)
	7	Sign channel 1 (0 = positive, 1 = negative)
3	0-1	Factor channel 1 (0 [dec] = illegal value / sensor not connected, 1 [dec] = factor 10, 2 [dec] = factor 100)
	2-7	not used
4	0	Status PT100 sensor channel 1 (0 = not connected, 1 = connected)
	1-7	not used
5	0-7	Value channel 2 (bit 0-7)
6	0-6	Value channel 2 (bit 8-14)
	7	Sign channel 2 (0 = positive, 1 = negative)
7	0-1	Factor channel 2 (0 [dec] = illegal value / sensor not connected, 1 [dec] = factor 10, 2 [dec] = factor 100)
	2-7	not used
8	0	Status PT100 sensor channel 2

CAN-Data-Byte	Bit	Content
		(0 = not connected, 1 = connected)
	0-7	not used

Calculation of temperature

Temperature = (sign) value[dec] / factor

4.2.7.6.8. Stepper

Structure of an 8-byte long CAN packet:

CAN-Data-Byte	Content
1	COMMAND
2	PAR1 (Bit 0-7)
3	PAR1 (Bit 8-15)
4	PAR1 (Bit 16-23)
5	PAR1 (Bit 24-31)
6	PAR2 (Bit 0-7)
7	PAR2 (Bit 8-15)
8	PAR3 (Bit 0-7)

4.2.7.6.8.1. Command-Liste

Command DAPI_STEPPER_CMD_	with Value (hex)	Description
SET_MOTORCHARACTERISTIC	1 (hex)	Set the motor configuration
GET_MOTORCHARACTERISTIC	2 (hex)	Query the motor configuration
SET_POSITION	3 (hex)	Setting the motor position
GO_POSITION	4 (hex)	Move to a specific position
GET_POSITION	5 (hex)	Query a specific position

Command DAPI_STEPPER_CMD_	with Value (hex)	Description
SET_FREQUENCY	6 (hex)	Setting the motor reference frequency
SET_FREQUENCY_DIRECTLY	7 (hex)	Setting the motor frequency
GET_FREQUENCY	8 (hex)	Query motor frequency
FULLSTOP	9 (hex)	Immediate stop of the engine
STOP	10 (hex)	Stopping the motor (braking ramp is maintained)
GO_REFSWITCH	11 (hex)	Approaching a reference position
DISABLE	14 (hex)	Switching the motor on/off
MOTORCHARACTERISTIC_LOAD_DEFAULT	15 (hex)	Set the motor characteristic to default value
MOTORCHARACTERISTIC_EEPROM_SAVE	16 (hex)	Saving the motor characteristics in the EEPROM
MOTORCHARACTERISTIC_EEPROM_LOAD	17 (hex)	Loading the motor characteristics from the EEPROM

Command DAPI_STEPPER_CMD_	with Value (hex)	Description
GET_CPU_TEMP	18 (hex)	Query the temperature of the CPU
GET_MOTOR_SUPPLY_VOLTAGE	19 (hex)	Query of the supply voltage of the CPU
GO_POSITION_RELATIVE	20 (hex)	Move to a relative position

4.2.7.6.8.2. Values for par 1 to command SET_MOTORCHARACTERISTIC

Parameter (DAPI_STEPPER_MOTORCHAR_PAR _ ...)	Value (dez)	Description
STEPMODE	1	Step mode (Full-, 1/2-, 1/4-, 1/8-, 1/16-step)
GOFREQUENCY	2	Speed [Full-step / s] - related to full-step
STARTFREQUENCY	3	Start frequency [Full-step / s]
STOPFREQUENCY	4	Stop frequency [Full-step / s]
MAXFREQUENCY	5	Maximum frequency [Full-step / s]
ACCELERATIONSLOPE	6	Acceleration slope [Full-step / ms]
DECELERATIONSLOPE	7	Slope of the delay [Full-step / 10ms].
PHASECURRENT	8	Phase current [mA]
HOLDPHASECURRENT	9	Phase current for motor holding [mA]
HOLDTIME	10	Time in which the stop goes to the motor stop [ms].
STATUSLEDMODE	11	Status LED mode
INVERT_ENDSW1	12	Invert the limit switch1
INVERT_ENDSW2	13	invert the limit switch2
INVERT_REFSW1	14	inverting the frequency switch1

Parameter (DAPI_STEPPER_MOTORCHAR_PAR _ ...)	Value (dez)	Description
INVERT_REFSW2	15	inverting the frequency switch1
INVERT_DIRECTION	16	Invert all directions
ENDSWITCH_STOPMODE	17	Setting of the stop behavior (0=Fullstop / 1=Stop)

Parameter (DAPI_STEPPER_MOTORCHAR_PAR _ ...)	Value (dec)	Description
GOREFERENCEFREQUENCY_TOEND SWITCH	18	Setting of the stop behavior (0=Fullstop / 1=Stop)
GOREFERENCEFREQUENCY_AFTER ENDSWITCH	19	Frequency after limit switch [Full-step / s].
GOREFERENCEFREQUENCY_TOOFF SET	20	Frequency up to optional offset [Full-step / s]

4.2.7.6.8.3. Values for par 1 to command GO_REFSWITCH

Parameter (DAPI_STEPPER_GO_REFSWITCH H_PAR_ ...)	Value (dec)	Description
REF1	1	Approach the reference switch 1
REF2	2	Approach the reference switch 2
REF_LEFT	4	Approaching the left edge of the reference switch
REF_RIGHT	8	Approaching the right edge of the reference switch
REF_GO_POSITIVE	16	Start the engine to the right
REF_GO_NEGATIVE	32	Start the engine to the left
SET_POS_0	64	Zeros of the motor position

4.2.7.6.8.4. Example

Program example

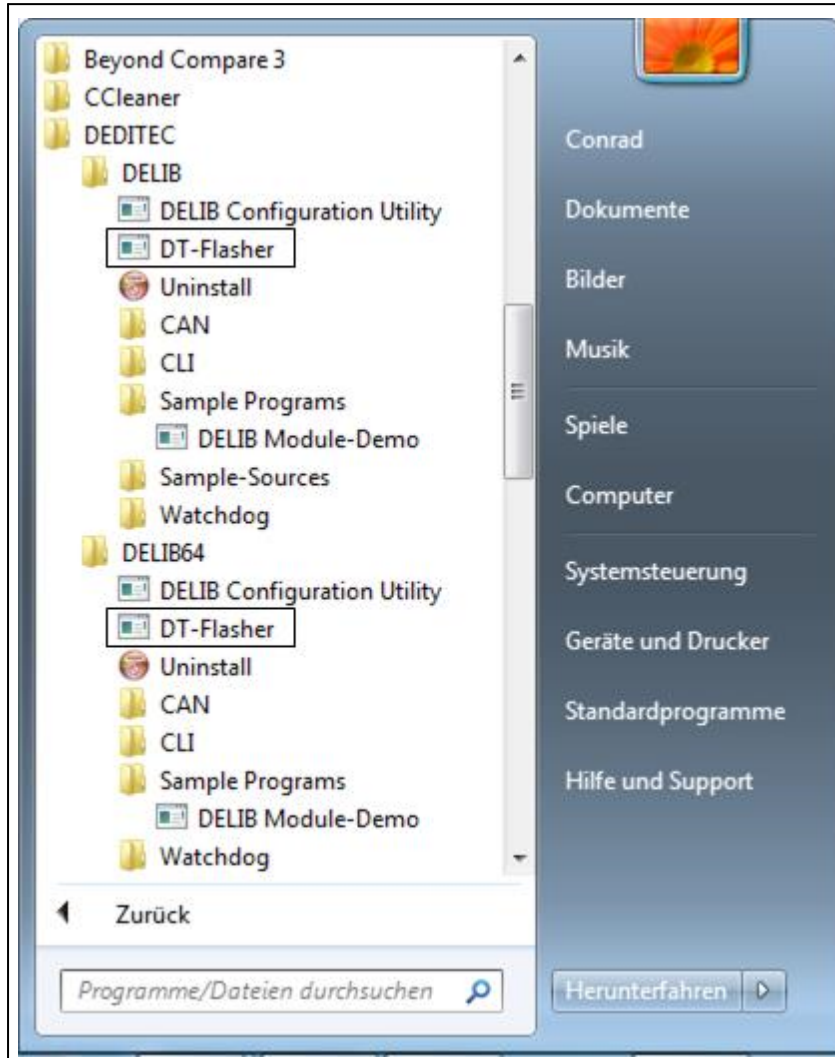
```
DapiStepperCommand(handle, 1,  
DAPI_STEPPER_CMD_GO_POSITION, 3200, 0, 0, 0);
```

is sent in an 8 byte CAN packet.

The package has the following structure:

CAN-Byte	Type	Value	Byte
1	COMMAND	4	04
2	PAR1 (Bit 0-7)	3200	80
3	PAR1 (Bit 8-15)		0C
4	PAR1 (Bit 16-23)		00
5	PAR1 (Bit 24-31)		00
6	PAR2 (Bit 0-7)	0	00
7	PAR2 (Bit 8-15)		00
8	PAR3 (Bit 0-7)	0	00

4.2.8. DT-Flasher



After installing the DELIB driver library, the DT-Flasher program can be started in the following way:

Start → Programs → DEDITEC → DELIB or DELIB64 → DT-Flasher.

4.2.8.1. About DEDITEC firmware

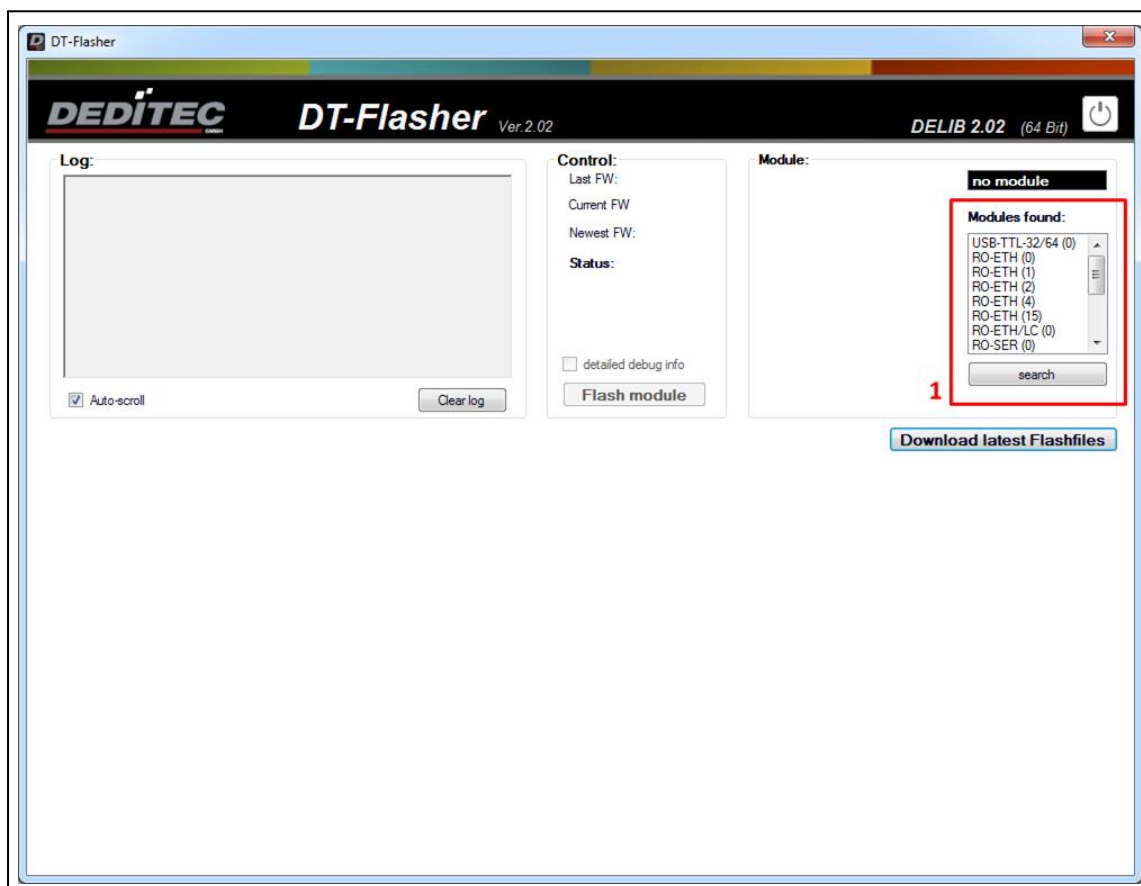
Most DEDITEC products have their own microcontroller. This processor is responsible for controlling all processes of the hardware.

To change the firmware required for the processor afterwards, we provide our free tool DT-Flasher. With this tool, the customer has the possibility to transfer newly published firmware versions directly to the module on site.

Note:

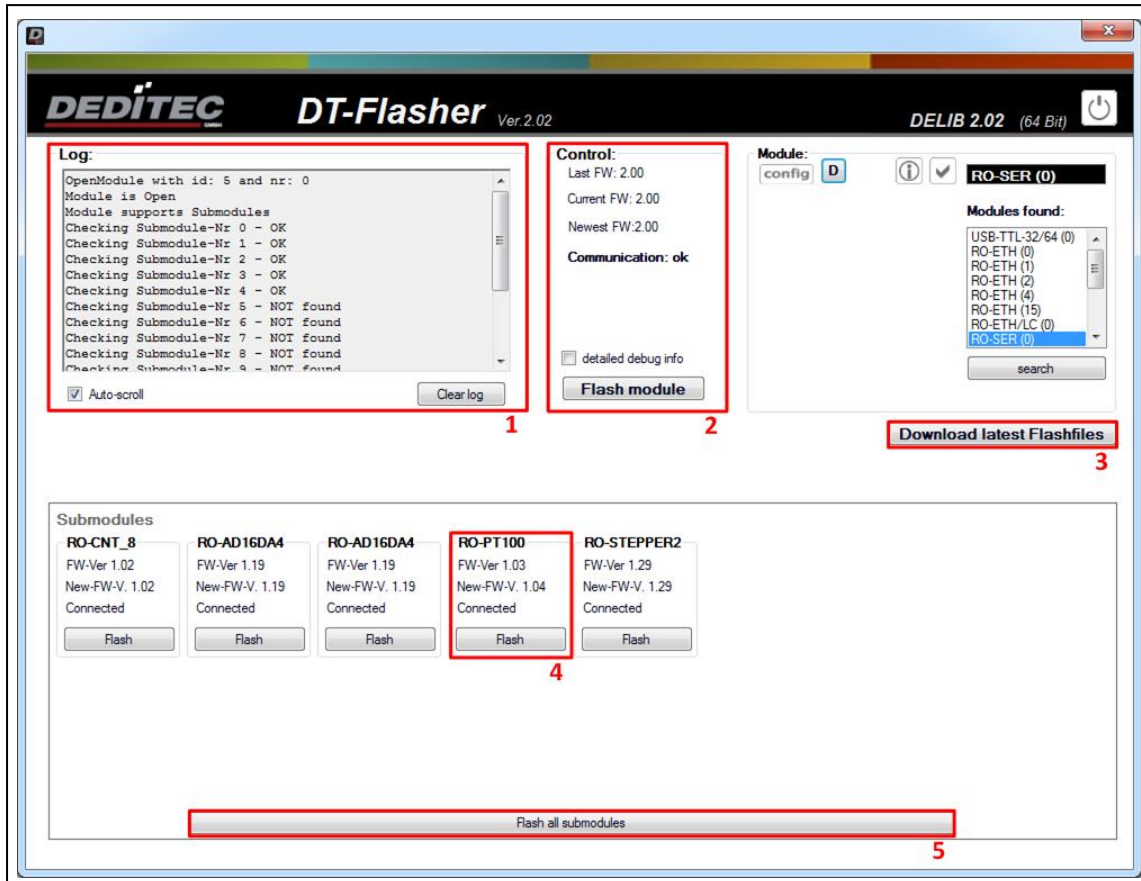
Since new firmware versions usually "unlock" new functions for your product, we therefore recommend a regular firmware update of your DEDITEC products.

4.2.8.2. Module selection



1. When starting the program, select the module that you want to update with a new firmware. For this you find a listing of all available modules in the "Module-Selector".

4.2.8.3. Perform firmware update



This example shows the RO-SER-CNT8-AD32-DA8-PT100-4-STEPPER2 module before a firmware update.

1. logbook - All messages during the firmware update are shown here. Via Auto-scroll is defined whether always automatically scroll down to the last event. Via Clear log the whole logbook is deleted.

2. Here you get information about the interface module (in this example the RO-SER interface). Newest FW displays the latest firmware version available for the module. Current FW shows the version that is currently available on the module. After the module has been flashed successfully, Last FW shows the version that was installed before the firmware update. If detailed debug info is checked, detailed messages are written to the logbook(1) during the firmware update. Flash module starts the firmware update for the interface module.

3. this allows you to download the latest firmware versions, so-called flash files, directly from the application.

4. firmware version shows the current firmware version of the submodule. New-FW-Ver shows the latest version available for this submodule. Via the button Flash the firmware update for the respective submodule is executed.

5. The Flash all submodules button is used to perform the firmware update for all connected submodules.

4.2.8.3.1. Update flash files manually

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

Step1

Download the latest version of the Flash files from

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

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:

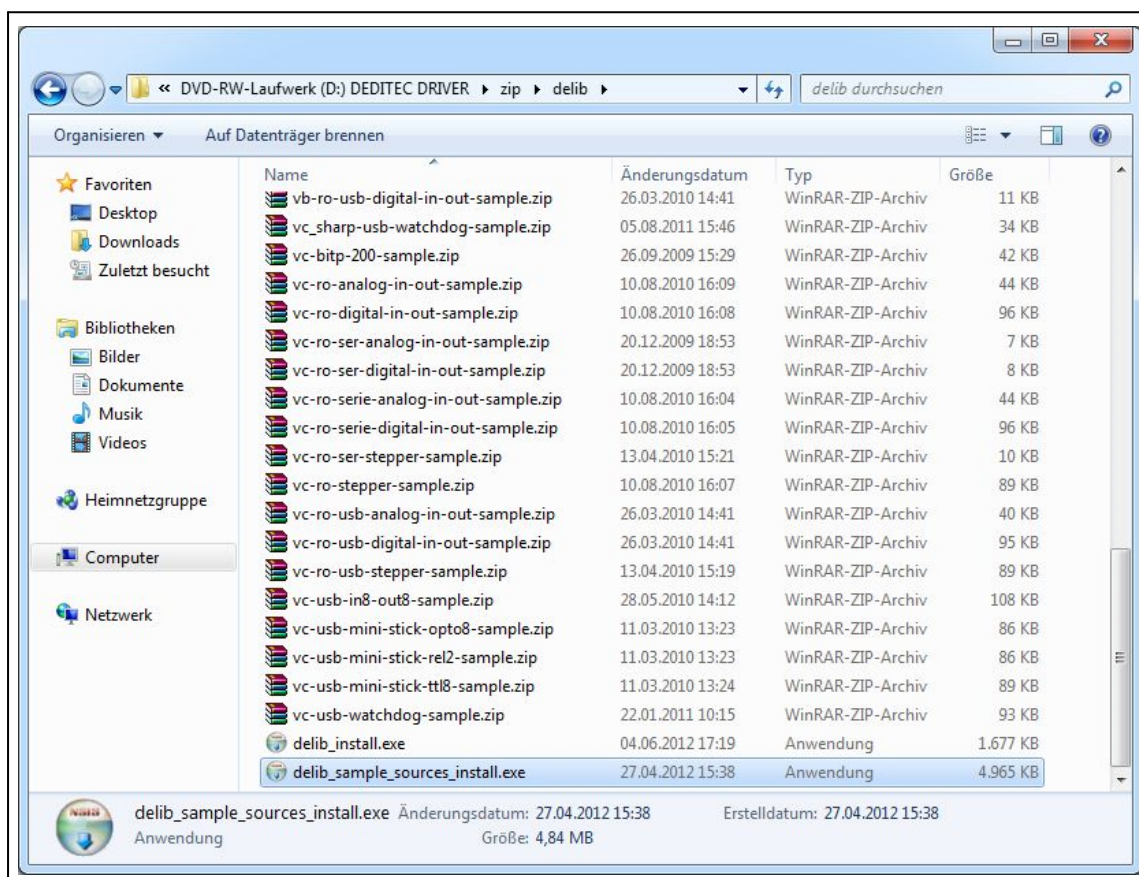
- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office
- LabVIEW
- Java

4.3.1. Installation DELIB Sample Sources

The DELIB Sample Sources can be installed either during the execution of the DELIB setup or as a stand-alone setup.

Insert the DEDITEC Driver CD into the drive and start `delib_sample_sources_install.exe`.

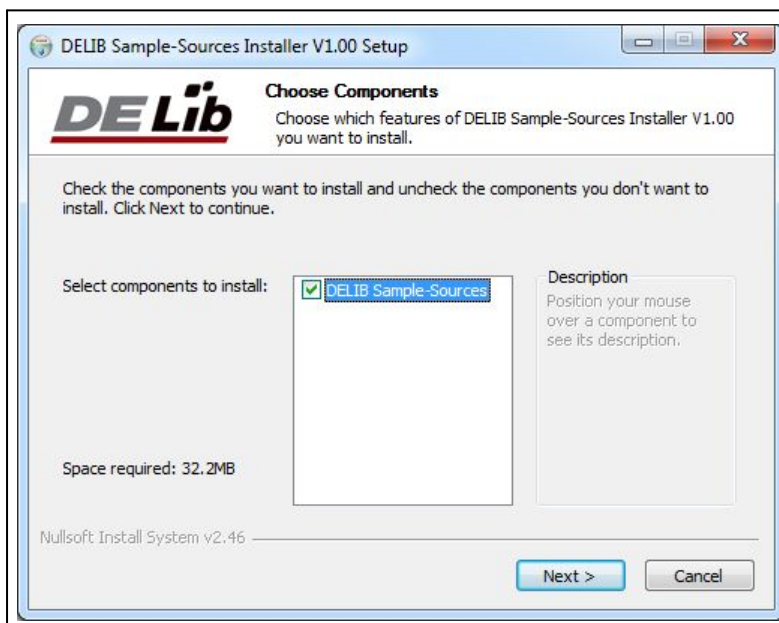
A current version of the Sample Sources can also be found on the Internet under <http://www.deditec.de/de/delib>



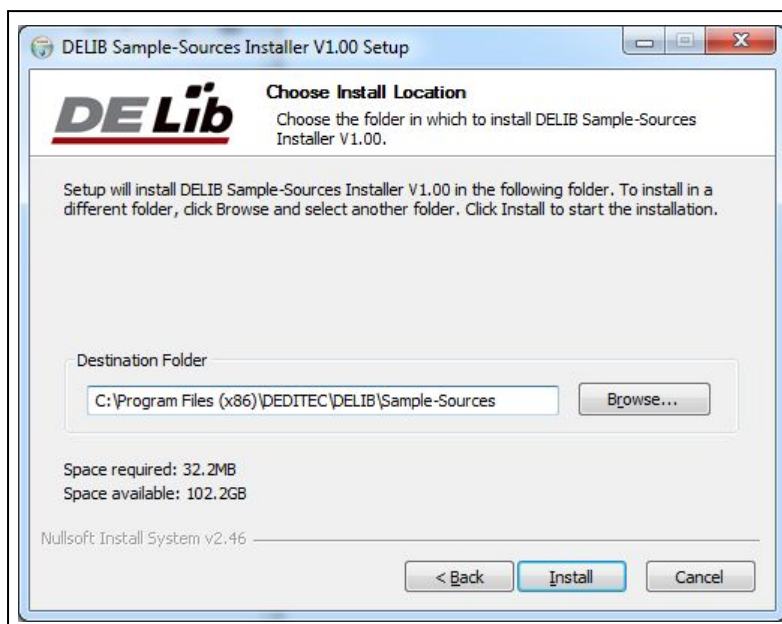
Startup screen of the DELIB Sample Sources Installer



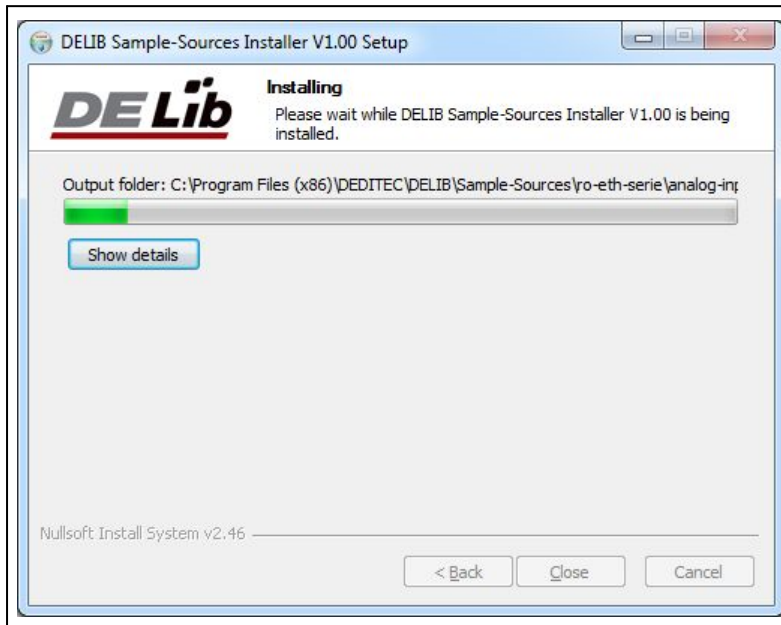
Press Next.



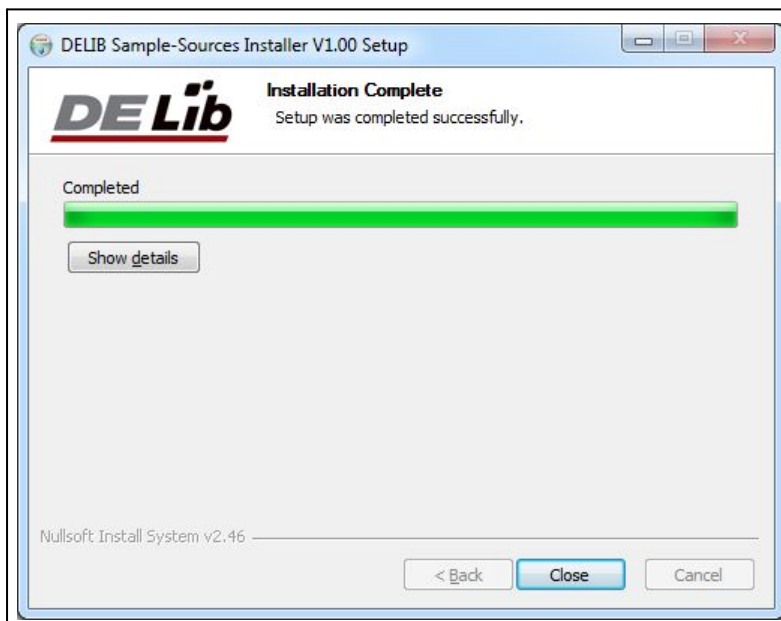
Select the installation folder and press Install.



The DELIB sample sources are now installed.



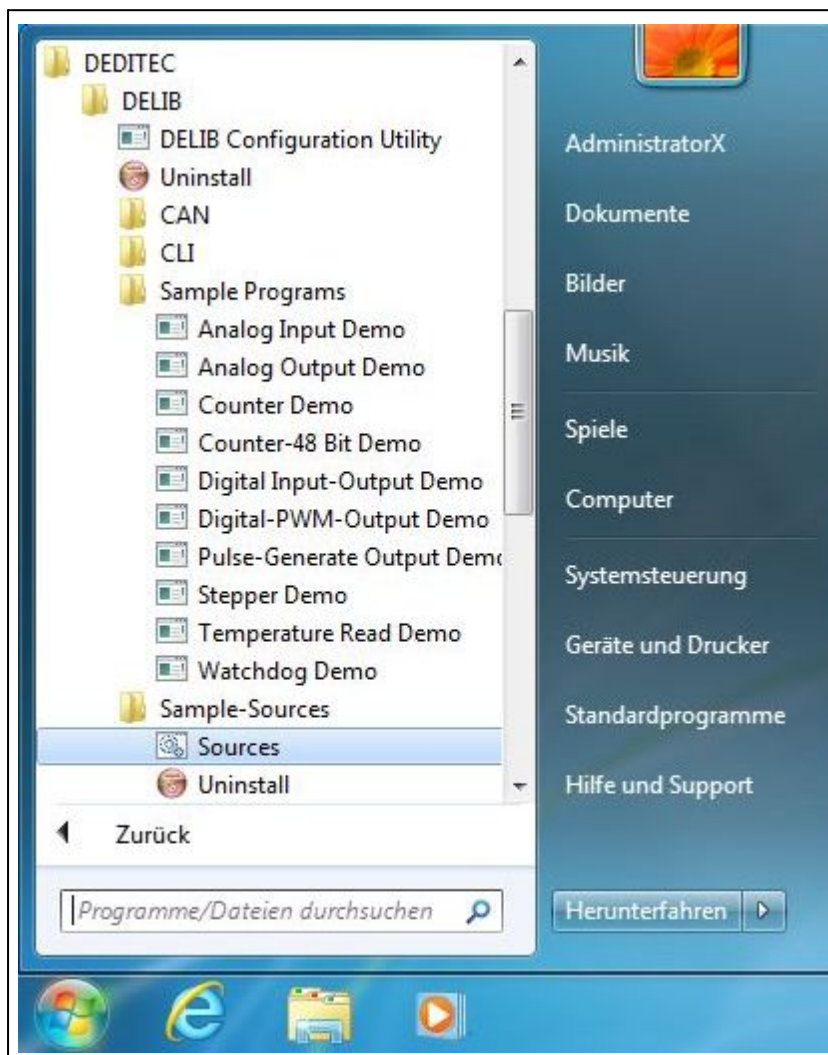
The DELIB Sample Sources have been successfully installed. Press Close to finish the installation.



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

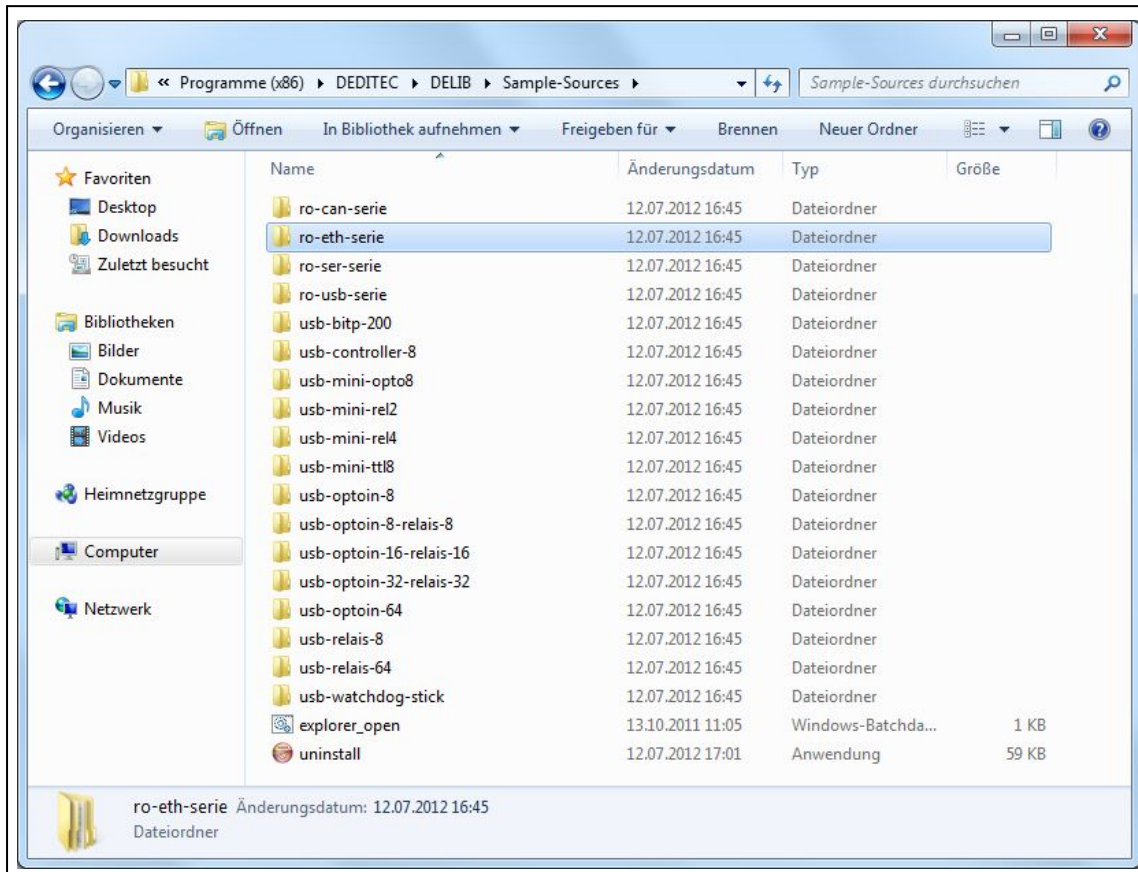


Now the Windows Explorer opens with an overview of all products for which a sample program is available.

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-016) in the programming language Visual-C.

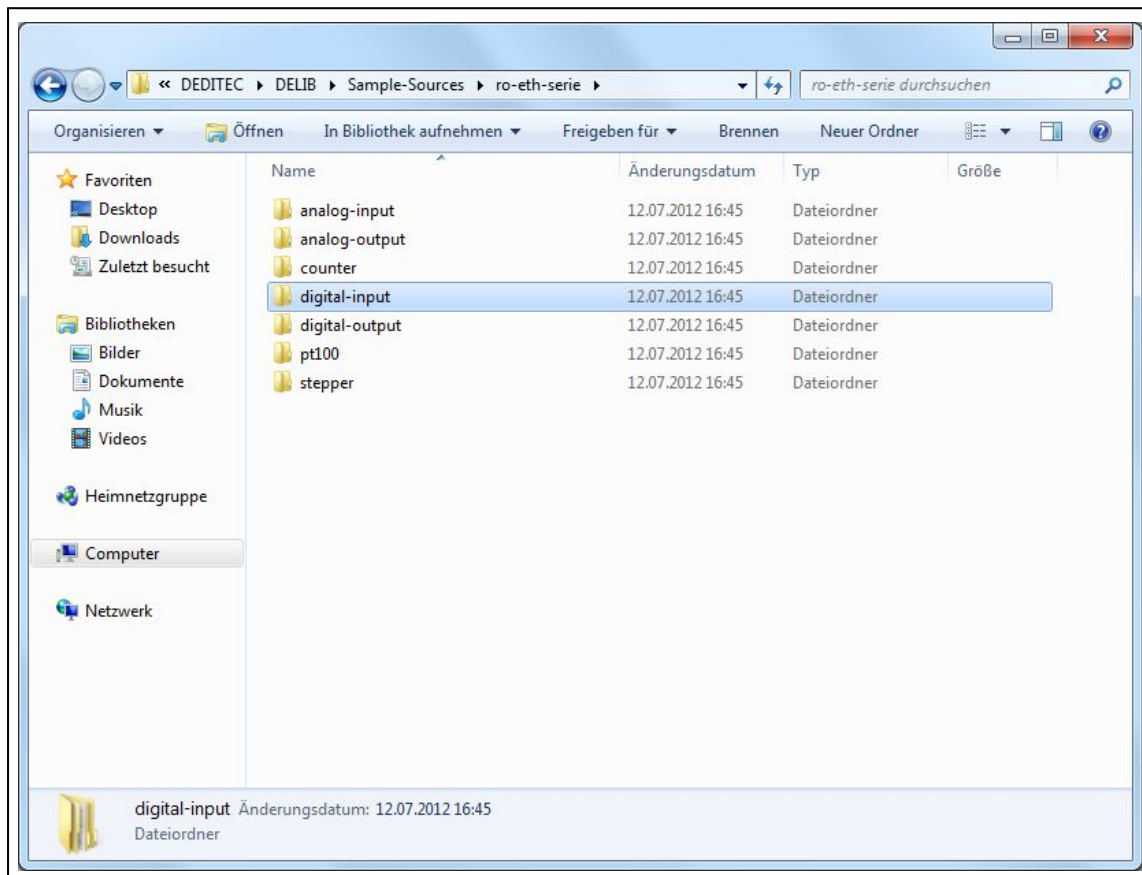
Since it is a RO-ETH product, select or open the ro-eth-series folder.



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.

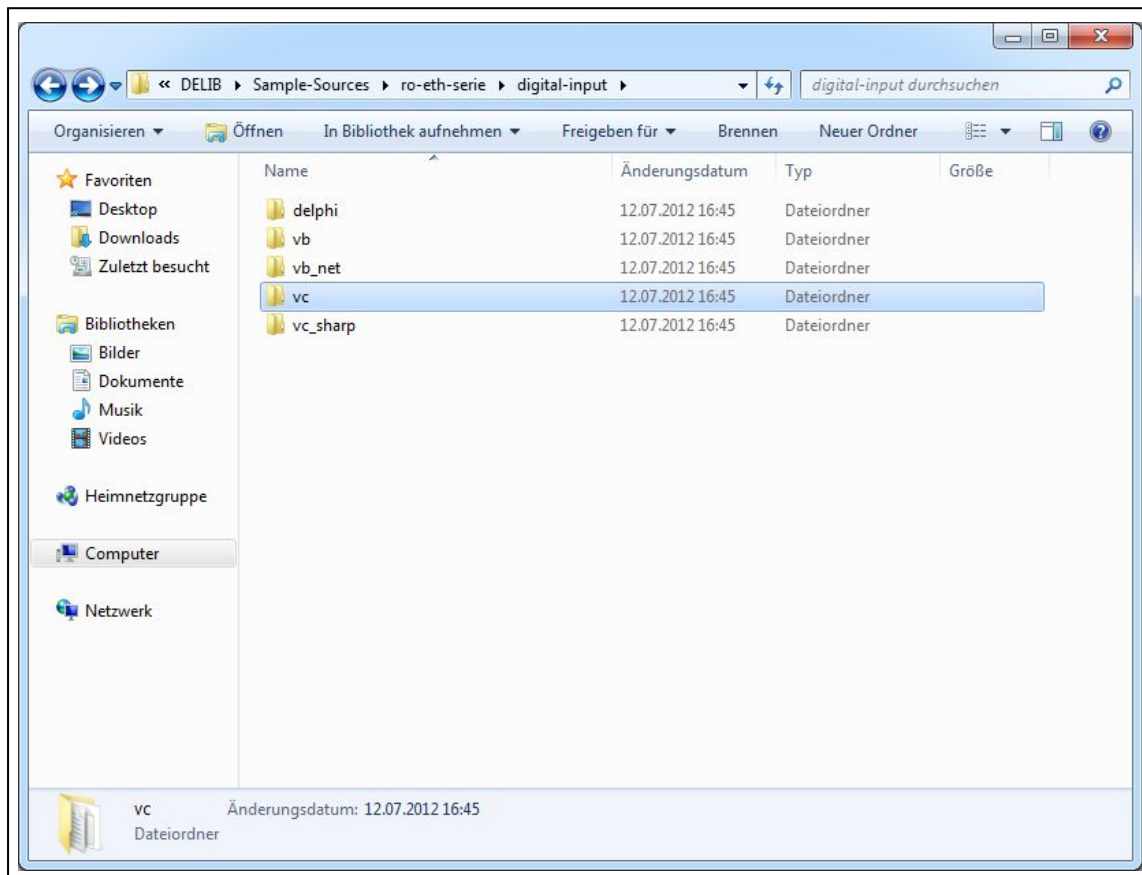
Since we focus on the digital inputs in this example, select the category digital-input



4.3.2.3. Step 3 - Programming language selection

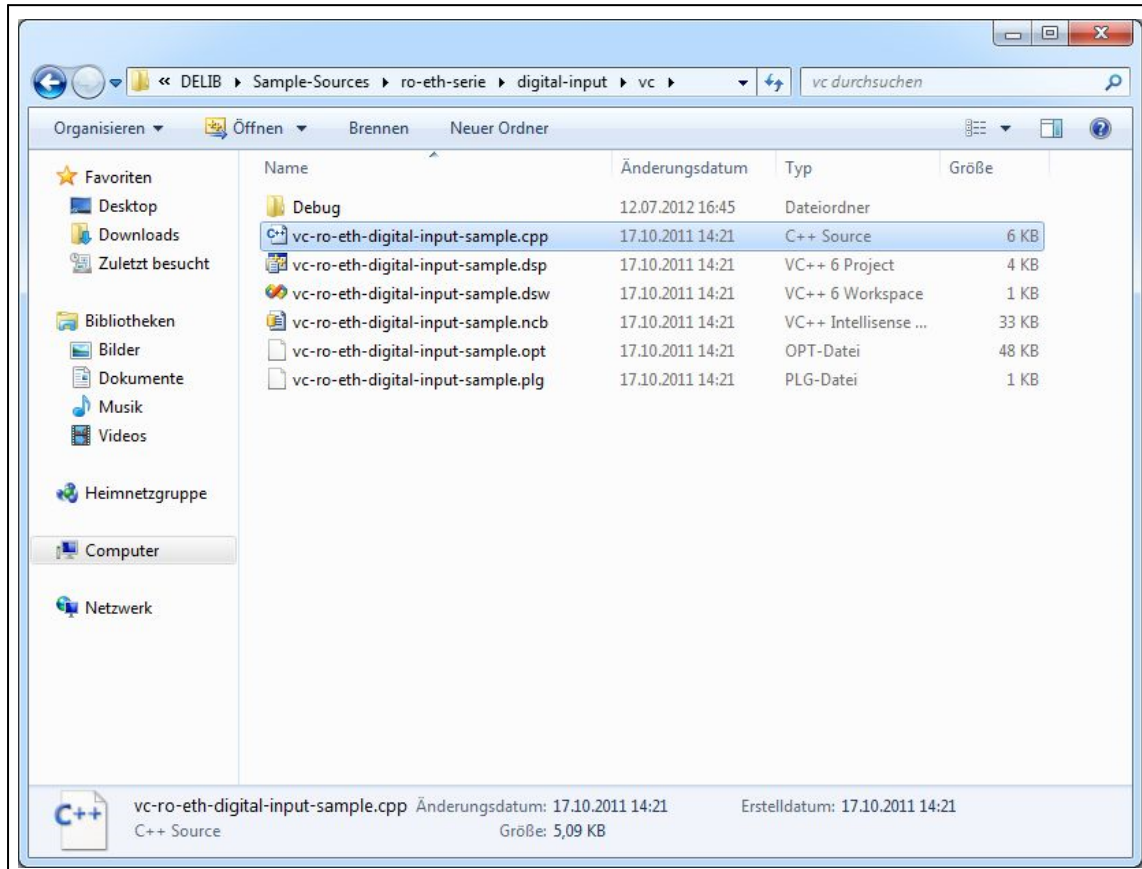
In this step you will see all available programming examples of the selected category, sorted by programming languages.

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

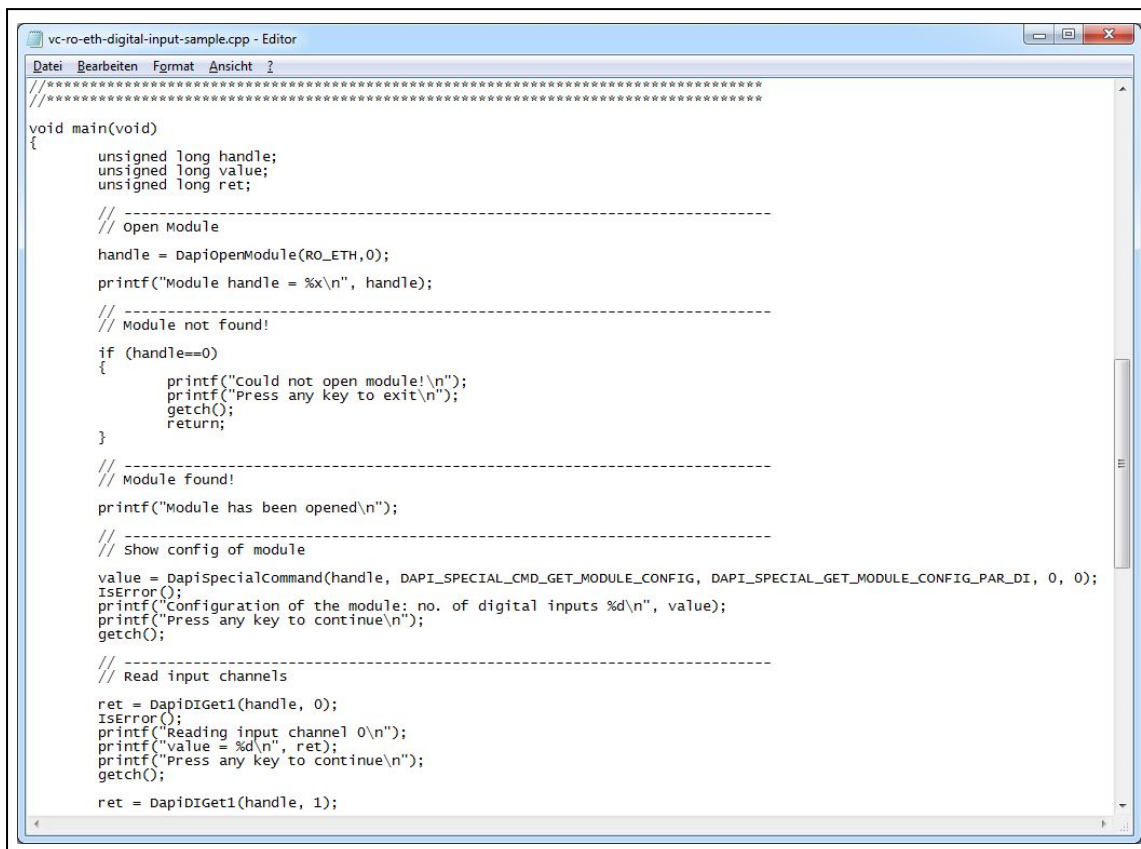


4.3.2.4. Step 4 - Source code

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

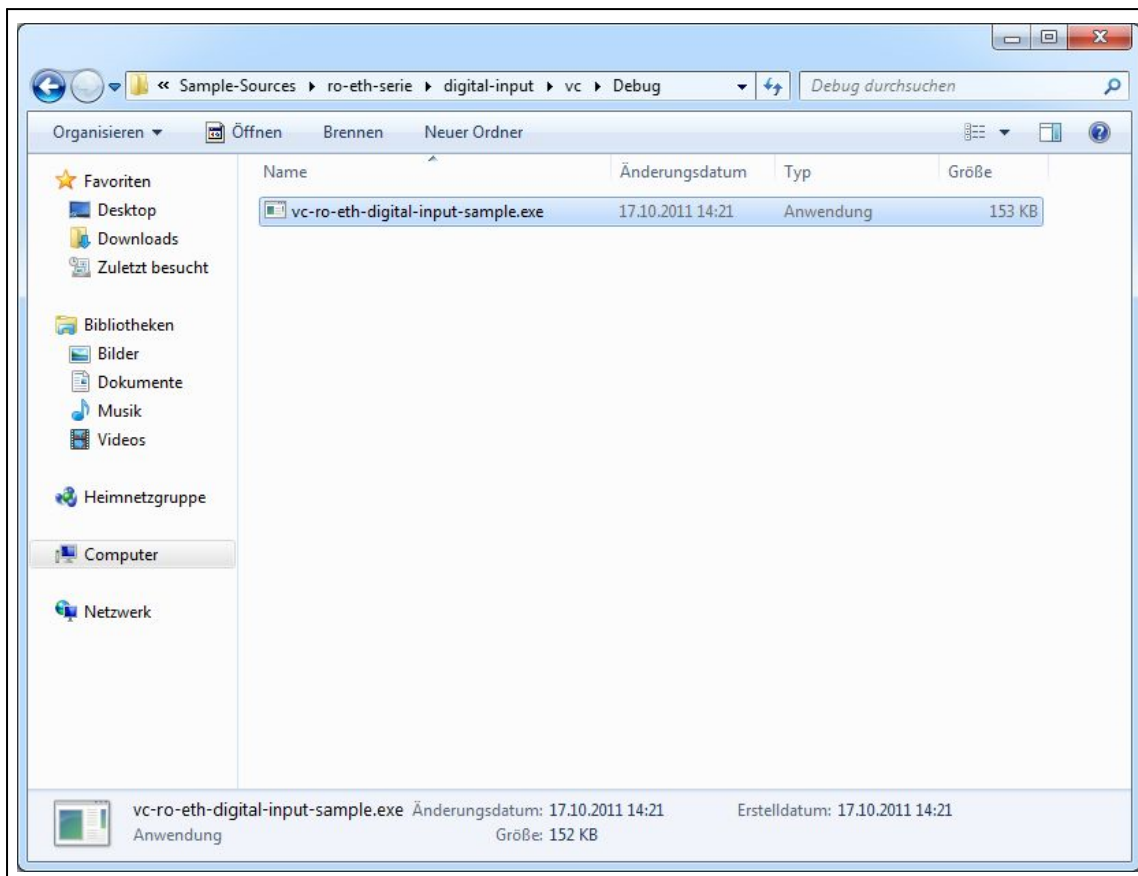
    value = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG, DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI, 0, 0);
    IsError();
    printf("Configuration of the module: no. of digital inputs %d\n", value);
    printf("Press any key to continue\n");
    getch();

    // -----
    // 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();

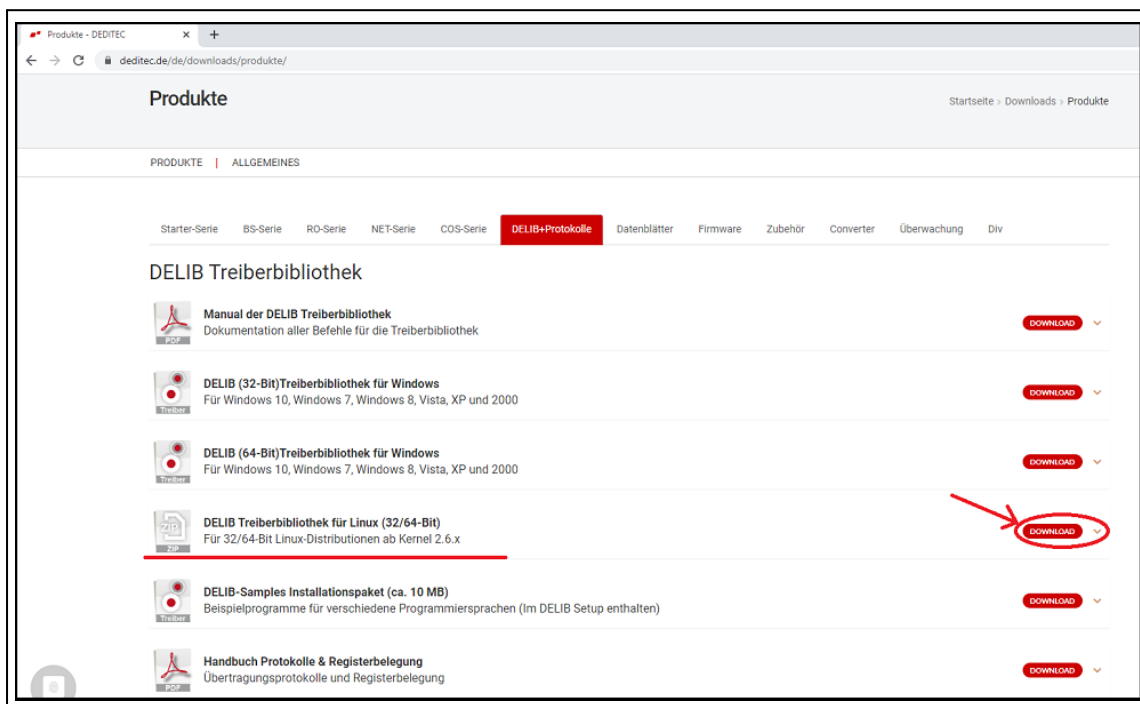
    ret = DapiDIGet1(handle, 1);
}
```

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

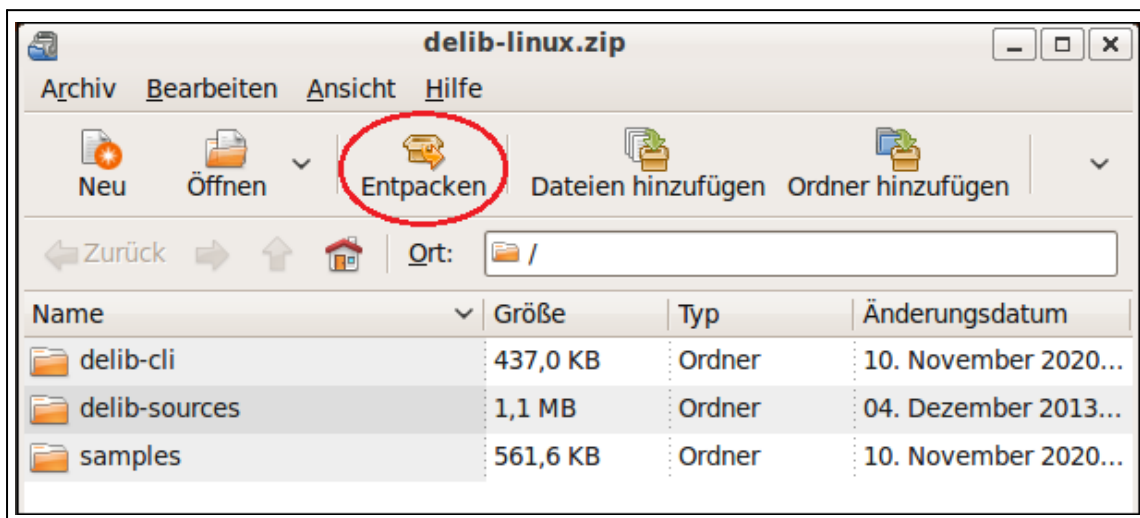


4.4. DELIB for Linux

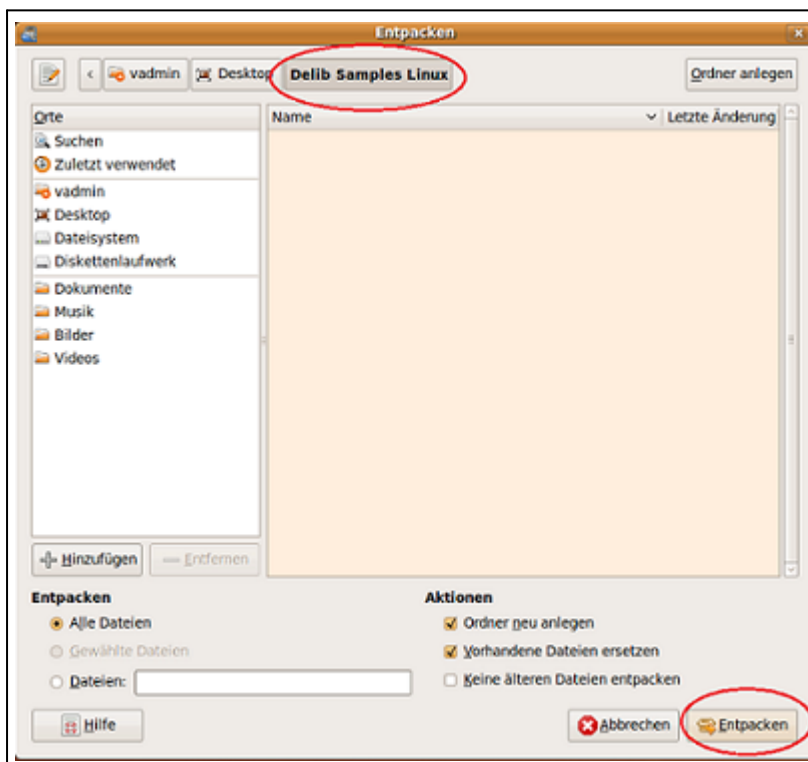
Download the Delib Linux driver library under "www.deditec.de/de/downloads/produkte/" in the tab "DELIB+Protocols" or under "www.deditec.de/media/zip/delib/delib-linux.zip" directly on your Linux system.



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".

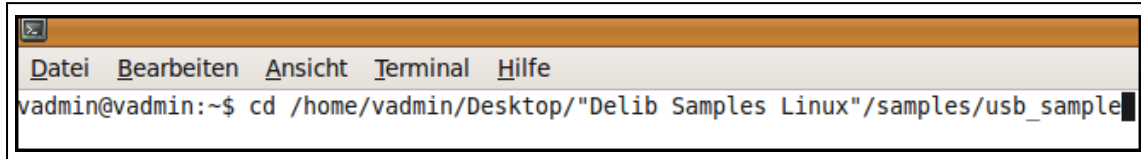
```
23
24 #include <stdio.h>
25 #include <stdlib.h>
26 #include <unistd.h>
27
28 #include "../delib-sources/delib/library/delib/delib.h"
29
30 int main()
31 {
32     ULONG i;
33     ULONG handle=0;
34
35     printf("\n\n");
36     printf("-----\n");
37     printf("-----\n");
38     printf("-----\n");
39     printf("WICHTIG !!!\n");
40     printf("Dieses Programm bitte mit admin-Rechten ausfuehren\n");
41     printf("Also: sudo ./delib-test-digital-io <return>\n");
42     printf("-----\n");
43     printf("-----\n");
44     printf("-----\n");
45     printf("\n\n");
46
47     printf("-----\n");
48     printf("Try to open USB_RELAIS_8\n");
49     handle = DapiOpenModule(USB_RELAIS_8, 0);
50
51     if(handle == 0)
52     {
53         // Module not found
54         printf("Handle = 0x%x\n", (unsigned long) handle);
55         return 0;
56     }
57
58     printf("Handle = 0x%x\n", (unsigned long) handle);
59 }
```

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.



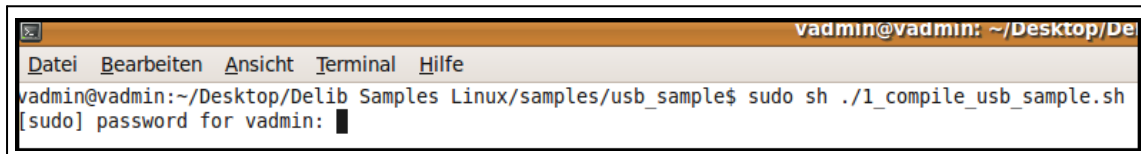
```

Datei Bearbeiten Ansicht Terminal Hilfe
vadmin@vadmin:~$ cd /home/vadmin/Desktop/"Delib Samples Linux"/samples/usb_sample

```

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.



```

vadmin@vadmin: ~/Desktop/Delib Samples Linux/samples/usb_sample$ sudo sh ./1_compile_usb_sample.sh
[sudo] password for vadmin:

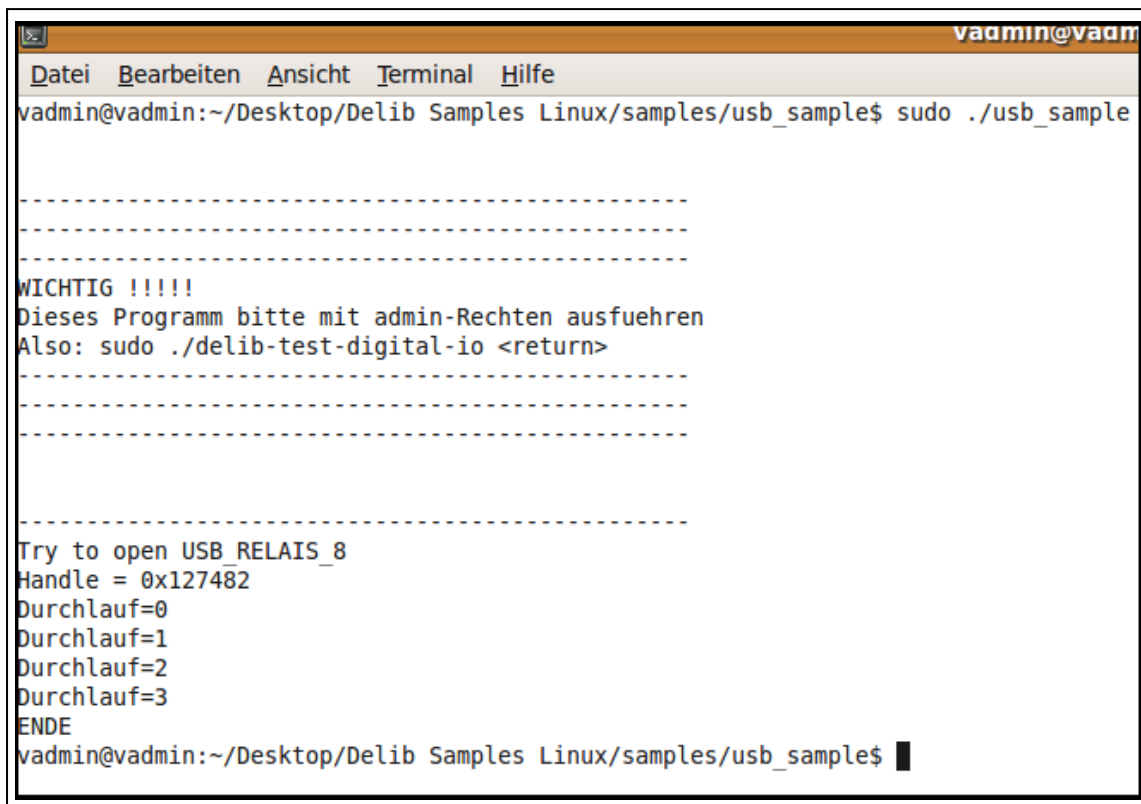
```

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 screenshot shows a terminal window with a menu bar containing 'Datei', 'Bearbeiten', 'Ansicht', 'Terminal', and 'Hilfe'. The title bar reads 'vadmin@vadm'. The command prompt shows the user 'vadmin' at 'vadmin' in the directory '~/Desktop/Delib Samples Linux/samples/usb_sample'. The command 'sudo ./usb_sample' has been executed. The output consists of several lines of dashes, followed by a warning 'WICHTIG !!!!!' and instructions to run the program with admin rights. It then shows the attempt to open 'USB_REL AIS_8' with handle '0x127482', followed by a loop of 'Durchlauf' (run) from 0 to 3, and ends with 'ENDE'. The prompt returns to the user's directory.

```
vadmin@vadmin:~/Desktop/Delib Samples Linux/samples/usb_sample$ sudo ./usb_sample

-----
-----
-----
WICHTIG !!!!!
Dieses Programm bitte mit admin-Rechten ausfuehren
Also: sudo ./delib-test-digital-io <return>
-----
-----
-----

Try to open USB_REL AIS_8
Handle = 0x127482
Durchlauf=0
Durchlauf=1
Durchlauf=2
Durchlauf=3
ENDE
vadmin@vadmin:~/Desktop/Delib Samples Linux/samples/usb_sample$
```

The program is now executed.

In this example all digital outputs of the USB_REL AIS_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 DELIB Configuration Utility, as well as via the web interface of the module.

```
26 #include <string.h>
27 #include <unistd.h>
28
29 #include "../delib-sources/delib/library/delib/delib.h"
30
31 int main()
32 {
33     unsigned long i;
34     unsigned long handle;
35     unsigned long ret;
36     DAPI_OPENMODULEEX_STRUCT open_buffer;
37
38     strcpy((char*) open_buffer.address, "192.168.1.21"); // hostname
39     open_buffer.timeout = 5000; // 5000 msec
40     open_buffer.portno = 9912; // using default port
41
42     #ifdef ENABLE_TCP_ENCRYPTION
43         open_buffer.encryption_type = DAPI_OPEN_MODULE_ENCRYPTION_TYPE_ADMIN; // encrypted communication with admin priv
44         strcpy((char*) open_buffer.encryption_password, "myPassword"); // password for encrypted communication
45     #else
46         open_buffer.encryption_type = DAPI_OPEN_MODULE_ENCRYPTION_TYPE_NONE; // Falls vorher eingestellt, geben Sie hier das Passwort
47     #endif // ihrer verschlüsselten TCP-Verbindung an.
48
49     handle = DapiOpenModuleEx(ETHERNET_MODULE, 0, (unsigned char*) &open_buffer, DAPI_OPEN_MODULE_OPTION_USE_EXBUFFER);
50
51     if(handle == 0)
52     {
```


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

```
57     for(i=0; i!=4; ++i)
58     {
59         printf("Durchlauf = %ld\n", i);
60
61         DapiDOSet8(handle, 0, 0xff);
62
63         usleep(1000 * 500);      // 500 msec sleep
64
65         DapiDOSet8(handle, 0, 0);
66
67         usleep(1000 * 500);      // 500 msec sleep
68         //ret = DapiDIGet8(handle, 0);
69         //printf("DI0-7 = 0x%lx\n", ret);
70
71         usleep(1000 * 500);      // 500 msec sleep
72     }
73
74
```

Auskommentieren,
falls keine digitalen
Eingänge vorhanden

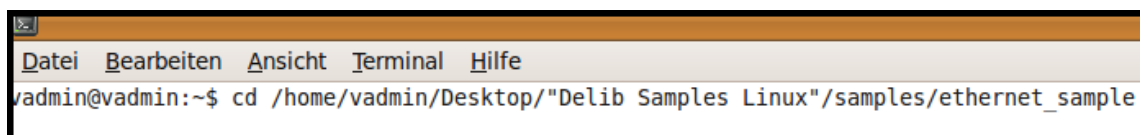
Compiling the ETH sample

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.



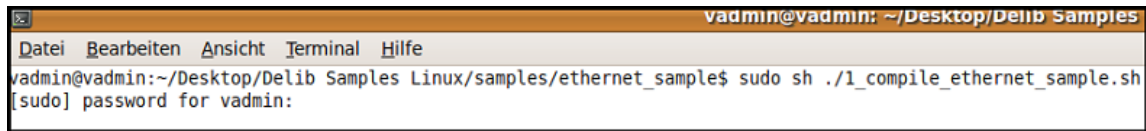
```
vadmin@vadmin:~$ cd /home/vadmin/Desktop/"Delib Samples Linux"/samples/ethernet_sample
```

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.

A terminal window titled 'vadmin@vadmin: ~/Desktop/Delib Samples'. The menu bar includes 'Datei', 'Bearbeiten', 'Ansicht', 'Terminal', and 'Hilfe'. The prompt is 'vadmin@vadmin:~/Desktop/Delib Samples Linux/samples/ethernet_sample\$'. The user enters 'sudo sh ./1_compile_ethernet_sample.sh'. The next line shows '[sudo] password for vadmin:'.

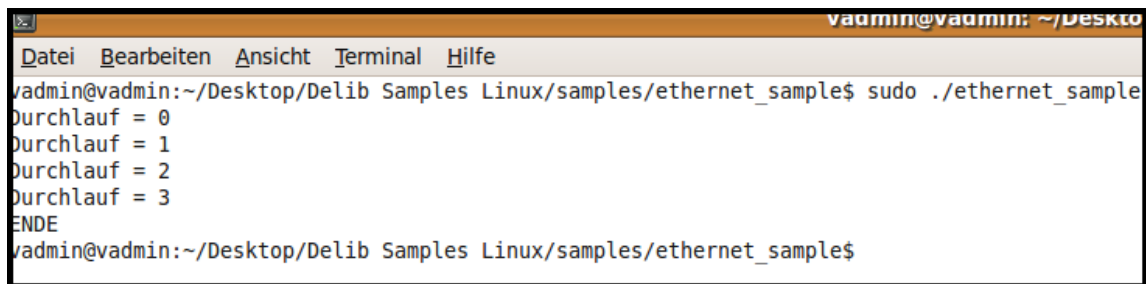
```
vadmin@vadmin: ~/Desktop/Delib Samples
Datei Bearbeiten Ansicht Terminal Hilfe
vadmin@vadmin:~/Desktop/Delib Samples Linux/samples/ethernet_sample$ sudo sh ./1_compile_ethernet_sample.sh
[sudo] password for vadmin:
```

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".

A terminal window titled 'vadmin@vadmin: ~/Desktop/Delib Samples Linux/samples/ethernet_sample'. The menu bar includes 'Datei', 'Bearbeiten', 'Ansicht', 'Terminal', and 'Hilfe'. The prompt is 'vadmin@vadmin:~/Desktop/Delib Samples Linux/samples/ethernet_sample\$'. The user enters 'sudo ./ethernet_sample'. The output shows 'Durchlauf = 0', 'Durchlauf = 1', 'Durchlauf = 2', 'Durchlauf = 3', and 'ENDE'. The prompt returns to 'vadmin@vadmin:~/Desktop/Delib Samples Linux/samples/ethernet_sample\$'.

```
vadmin@vadmin: ~/Desktop/Delib Samples Linux/samples/ethernet_sample
Datei Bearbeiten Ansicht Terminal Hilfe
vadmin@vadmin:~/Desktop/Delib Samples Linux/samples/ethernet_sample$ sudo ./ethernet_sample
Durchlauf = 0
Durchlauf = 1
Durchlauf = 2
Durchlauf = 3
ENDE
vadmin@vadmin:~/Desktop/Delib Samples Linux/samples/ethernet_sample$
```

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 ["nunit"]]
```

Definition for ETH modules (Linux)

```
delib_cli [command] [channel] [value | unit ["nunit"]]
```

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 DELIB Configuration Utility.

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 DELIB Configuration Utility as well as via the web interface of the module.

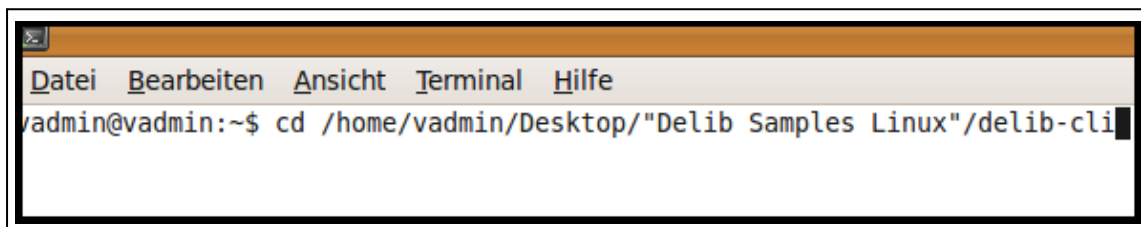
Compiling the Delib CLI sample

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.



```

Datei  Bearbeiten  Ansicht  Terminal  Hilfe
vadmin@vadmin:~$ cd /home/vadmin/Desktop/"Delib Samples Linux"/delib-cli

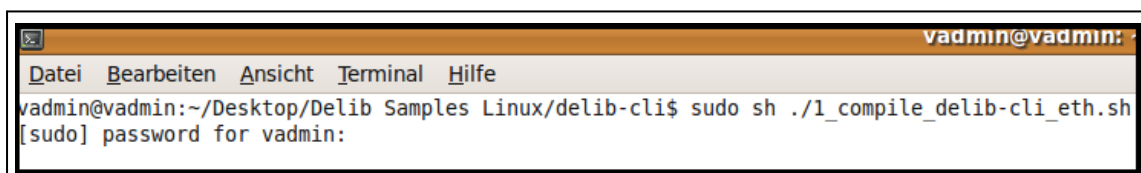
```

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.



```

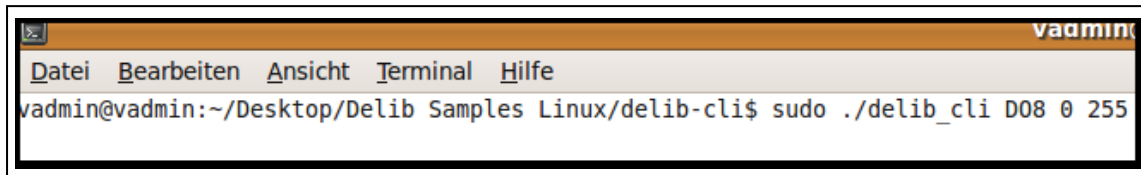
vadmin@vadmin:
Datei  Bearbeiten  Ansicht  Terminal  Hilfe
vadmin@vadmin:~/Desktop/Delib Samples Linux/delib-cli$ sudo sh ./1_compile_delib-cli_eth.sh
[sudo] password for vadmin:

```

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".



A terminal window titled 'vadmin' with a menu bar containing 'Datei', 'Bearbeiten', 'Ansicht', 'Terminal', and 'Hilfe'. The command prompt shows the user 'vadmin' at host 'vadmin' in the directory '~/Desktop/Delib Samples Linux/delib-cli'. The command being executed is 'sudo ./delib_cli D08 0 255'.

```
vadmin@vadmin:~/Desktop/Delib Samples Linux/delib-cli$ sudo ./delib_cli D08 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

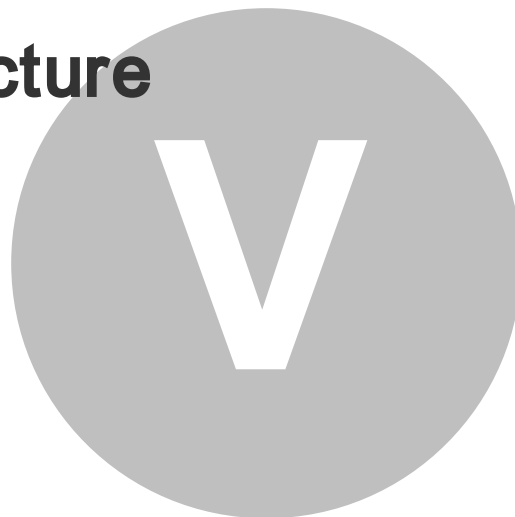
→ reads the value of the 3rd analog input as decimal number of a USB module

```
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

DELIB directory structure



5. 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

5.1. 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).

5.2. Library directory

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

5.3. 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".

5.4. 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

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

6.2. Administrative functions

6.2.1. 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;
}
```

6.2.2. 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);
```

6.2.3. 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)
```

6.2.4. 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
```

6.2.5. 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 DELIB Configuration Utility.

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);
```

6.3. Error handling

6.3.1. 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;
}
```

6.3.2. 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;
}
```

6.3.3. 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;
}
```

6.3.4. 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;
}
```

6.3.5. 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 0%x - Error
Code = 0%x\n", handle, error_code);

        DapiClearLastErrorByHandle(handle);

        return TRUE;
    }

    return FALSE;
}
```

6.4. A/D converter functions

6.4.1. DapiADSetMode

Description

This command sets the mode for a D/A converter.

Definition

```
void DapiDASetMode(ULONG handle, ULONG ch, ULONG mode);
```

Parameter

handle=This is the handle of an open module

ch=Indicates the channel of the D/A converter (0 ..)

mode=Specifies the mode for the D/A converter (see delib.h)

Return value

None

Comment

The following modes are supported:

(these depend on the D/A module used)

Unipolar voltages:

Mode	Value range
ADDA_MODE_UNIPOL_10V	0V .. 10V
ADDA_MODE_UNIPOL_5V	0V .. 5V
ADDA_MODE_UNIPOL_2V5	0V .. 2,5V

Bipolar voltages:

Mode	Value range
ADDA_MODE_BIPOL_10V	-10V .. +10V
ADDA_MODE_BIPOL_5V	-5V .. +5V
ADDA_MODE_BIPOL_2V5	-2,5V .. +2,5V

Currents:

Mode	Value range
ADDA_MODE_0_20mA	0 .. 20 mA
ADDA_MODE_4_20mA	4 .. 20 mA
ADDA_MODE_0_24mA	0 .. 24 mA
ADDA_MODE_0_25mA	0 .. 25 mA
ADDA_MODE_0_50mA	0 .. 50 mA

6.4.2. DapiADGetMode

Description

This command reads back the set mode of an A/D converter. Mode description see DapiADSetMode.

Definition

ULONG DapiADGetMode(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Indicates the channel of the A/D converter (0 ..)

Return value

Mode of the A/D converter

6.4.3. DapiADGet

Description

This command reads a data value from one channel of an A/D converter.

Definition

ULONG DapiADGet(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Indicates the channel of the A/D converter (0 ..)

Return value

Value from A/D converter in digits

6.4.4. DapiADGetVolt

Description

This command reads a data value from one channel of an A/D converter in volts.

Definition

```
float DapiADGetVolt(ULONG handle, ULONG ch);
```

Parameter

handle=This is the handle of an open module

ch=Indicates the channel of the A/D converter (0 ..)

Return value

Value from A/D converter in volts

6.4.5. DapiADGetmA

Description

This command reads a data value from one channel of an A/D converter in mA.

Definition

```
float DapiADGetmA(ULONG handle, ULONG ch);
```

Parameter

handle=This is the handle of an open module

ch=Indicates the channel of the A/D converter (0 ..)

Return value

Value from A/D converter in mA.

Comment

This command is module dependent. Of course it only works if the module also supports the current mode.

6.4.6. DapiSpecialADReadMultipleAD

Description

This command stores the values of certain adjacent channels of an A/D converter simultaneously in an intermediate buffer. This way the values can be read out one after the other.

The advantage of this is that on the one hand the A/D values are buffered simultaneously, on the other hand the values of several A/D channels (compared to the commands DapiADGetVolt, DapiADGetmA or DapiADGet) can be queried much faster afterwards.

Definition

```
void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_AD,  
DAPI_SPECIAL_AD_READ_MULTIPLE_AD, ULONG start_ch, ULONG end_ch);
```

Parameter

handle=This is the handle of an open module.

start_ch=Gives the start channel of the A/D converter, from which the values are buffered (0, 1, 2, ..).

end_ch=Gives the end channel of the A/D converter up to which the values are buffered (0, 1, 2, ..).

Return value

None.

Comment

The values buffered with command DapiSpecialADReadMultipleAD can be read afterwards with commands DapiADGetVolt, DapiADGetmA or DapiADGet. So that the buffered value is really read, the parameter "ch" must be logically linked with 0x8000 "or" for these functions (see examples).

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_AD,  
DAPI_SPECIAL_AD_READ_MULTIPLE_AD, 0, 15);  
// Buffers the values of AD channel 0..15  
  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_AD,  
DAPI_SPECIAL_AD_READ_MULTIPLE_AD, 0, 63);  
// Buffers the values of AD channel 0..63  
  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_AD,  
DAPI_SPECIAL_AD_READ_MULTIPLE_AD, 16, 31);  
// Buffers the values of AD channel 16..31  
  
value = DapiADGetVolt(handle, 0x8000 | 0);  
// Returns the buffered value of AD channel 0 in volts.  
  
value = DapiADGetmA(handle, 0x8000 | 15);  
// Gibt den gepufferten Wert von AD-Kanal 15 in mA zurück.  
  
value = DapiADGet(handle, 0x8000 | 63);  
// Gibt den gepufferten Wert von AD-Kanal 63 in Digits zurück.
```

6.5. Manage D/A outputs

6.5.1. DapiDASetMode

Description

This command sets the mode for a D/A converter.

Definition

```
void DapiDASetMode(ULONG handle, ULONG ch, ULONG mode);
```

Parameter

handle=This is the handle of an opened module.

ch=Gives the channel of the D/A converter (0 ..)

mode=Gives the mode for the D/A converter (see delib.h)

Return value

None

Comment

The following modes are supported:

(these depend on the D/A module used).

Unipolar voltages:

Mode	Value range
ADDA_MODE_UNIPOL_10V	0V .. 10V
ADDA_MODE_UNIPOL_5V	0V .. 5V
ADDA_MODE_UNIPOL_2V5	0V .. 2,5V

Bipolar voltages:

Mode	Value range
ADDA_MODE_BIPOL_10V	-10V .. +10V
ADDA_MODE_BIPOL_5V	-5V .. +5V
ADDA_MODE_BIPOL_2V5	-2,5V .. +2,5V

Currents:

Mode	Value range
ADDA_MODE_0_20mA	0 .. 20 mA
ADDA_MODE_4_20mA	4 .. 20 mA
ADDA_MODE_0_24mA	0 .. 24 mA
ADDA_MODE_0_25mA	0 .. 25 mA
ADDA_MODE_0_50mA	0 .. 50 mA

6.5.2. DapiDAGetMode

Description

This command reads back the set mode of a D/A converter.

Definition

```
ULONG DapiDAGetMode(ULONG handle, ULONG ch);
```

Parameter

handle=This is the handle of an opened module.

ch=Indicates the channel of the D/A converter (0 ..)

Return value

Mode of the D/A converter

6.5.3. DapiDASet

Description

This command passes a data value to a channel of a 16-bit D/A converter.

Definition

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

Parameter

handle=This is the handle of an opened module.

ch=Indicates the channel of the D/A converter (0 ..)

data=Gives the data value that is written (16-bit data value -> data value range: 0-65535)

Return value

None

Programming example

```
DapiDASet(handle, 0, 65535);  
// Sets the 1st output of the D/A converter to maximum value of the  
selected mode.  
//(with selected mode ADDA_MODE_UNIPOL_10V the 1st output of the D/A  
converter is set to 10V).
```


6.5.4. DapiDASetVolt

Description

This command applies a voltage to one channel of a D/A converter.

Definition

```
void DapiDASetVolt(ULONG handle, ULONG ch, float data);
```

Parameter

handle=This is the handle of an opened module.

ch=Indicates the channel of the D/A converter (0 ..)

data=Gives the voltage to be set [V].

Return value

None

Programming example

```
DapiDASetVolt(handle, 0, 5.4321);  
// Sets the 1st output of the D/A converter to 5.4321 V
```

6.5.5. DapiDASetmA

Description

This command sets a current to a channel of a D/A converter.

Definition

```
void DapiDASetmA(ULONG handle, ULONG ch, float data);
```

Parameter

handle=This is the handle of an opened module.

ch=Indicates the channel of the D/A converter (0 ..)

data=Indicates the current that is written [mA].

Return value

None

Comment

This command is module dependent. Of course it only works if the module also supports the current mode.

6.5.6. DapiSpecialCmd_DA

Description

This command sets the voltage values at a channel at power-on or after a timeout of a D/A converter (EEPROM configuration).

Definition

```
void DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DA, cmd, ch, 0);
```

Parameter

handle=This is the handle of an opened module.

ch=Indicates the channel of the D/A converter (0, 1, 2, ..)

Reset settings to default configuration

cmd=DAPI_SPECIAL_DA_PAR_DA_LOAD_DEFAULT

Save the configuration to the EEPROM

cmd=DAPI_SPECIAL_DA_PAR_DA_SAVE_EEPROM_CONFIG

Loading the configuration from the EEPROM

cmd=DAPI_SPECIAL_DA_PAR_DA_LOAD_EEPROM_CONFIG

Return value

None

Comment

DAPI_SPECIAL_CMD_DA_PAR_DA_LOAD_DEFAULT

This command loads the default configuration of a D/A converter. The D/A converter has now 0V as output voltage.

DAPI_SPECIAL_DA_PAR_DA_SAVE_EEPROM_CONFIG

This command saves the current D/A converter setting (voltage/current value, enable/disable and D/A converter mode) to the EEPROM.

DAPI_SPECIAL_DA_PAR_DA_LOAD_EEPROM_CONFIG

This command sets the D/A converter, with the configuration stored in the EEPROM.

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DA,  
DAPI_SPECIAL_DA_PAR_DA_LOAD_DEFAULT, 1, 0);  
//Reset the EEPROM configuration to default configuration at channel 1.  
  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DA,  
DAPI_SPECIAL_DA_PAR_DA_SAVE_EEPROM_CONFIG, 3, 0);  
//Save the D/A converter settings into the EEPROM at channel 3.  
  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_DA,  
DAPI_SPECIAL_DA_PAR_DA_LOAD_EEPROM_CONFIG, 2, 0);  
//Set the D/A converter, with the configuration stored in the EEPROM at  
channel 2.
```

6.6. Read digital inputs

6.6.1. 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)

6.6.2. 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

6.6.3. 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

6.6.4. 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();
```

6.6.5. 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

6.6.6. 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

6.6.7. 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
```

6.6.8. 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);
```

6.6.9. 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);
```

6.6.10. 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
```

6.6.11. 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.
```

6.6.12. 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.
```

6.6.13. 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
```


6.7. Manage digital outputs

6.7.1. 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

6.7.2. 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

6.7.3. 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

6.7.4. 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();
```

6.7.5. 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

6.7.6. 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-08-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
```

6.7.7. 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.

6.7.8. 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.

6.7.9. 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);
```

6.7.10. 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);
```


6.8. Manage output timeout

6.8.1. 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

6.8.1.1. 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.
```

6.8.1.2. 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.
```

6.8.1.3. 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.
```

6.8.1.4. 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.
```

6.8.1.5. 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.
```


6.8.1.6. 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 (Timeout "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.
```

6.8.1.7. 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.
```

6.8.1.8. 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.
```

6.8.1.9. 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.
```

6.8.1.10. 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.
```

6.8.1.11. 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.
```

6.9. Digital counter functions for RO-Counter modules

6.9.1. DapiCnt48ModeSet

Description

This command sets a counting mode (optionally also submode) and input filter for a specific input counter channel.

Definition

```
void DapiCnt48ModeSet(ULONG handle, ULONG ch, ULONG mode);
```

Parameter

handle = This is the handle of an open module

ch = Number of the input counter channel whose mode is to be set (0, 1, 2, 3, ..)

mode = Specifies the mode

Overview available counter modes

Mode	Value[hex]	CNT8	CNT/IGR
DAPI_CNT48_MODE_COUNT_RISING_EDGE	0	X	X
DAPI_CNT48_MODE_COUNT_RISING_EDGE_X2	1		X
DAPI_CNT48_MODE_COUNT_RISING_EDGE_X4	2		X
DAPI_CNT48_MODE_T	D	X	
DAPI_CNT48_MODE_FREQUENCY	E	X	
DAPI_CNT48_MODE_PWM	F	X	

Submode	Value[hex]	HW-Reset available
DAPI_CNT48_SUBMODE_NO_RESET	0	x

Submode	Value[hex]	HW-Reset available
DAPI_CNT48_SUBMODE_RESET_WITH_READ	1	x
DAPI_CNT48_SUBMODE_NO_RESET	2	
DAPI_CNT48_SUBMODE_RESET_WITH_READ	3	

Possible values for mode

*mode=DAPI_CNT48_MODE_COUNT_RISING_EDGE |
DAPI_CNT48_SUBMODE_NO_RESET*

In this mode, counting is performed on rising edge.

*mode=DAPI_CNT48_MODE_COUNT_RISING_EDGE |
DAPI_CNT48_SUBMODE_RESET_WITH_READ*

In this mode, counting is performed on a rising edge. In addition, the counter is reset with each read operation.

*mode=DAPI_CNT48_MODE_COUNT_RISING_EDGE |
DAPI_CNT48_SUBMODE_RESET_ON_CH_7*

In this mode, counting is performed on a rising edge. In addition, the counter can be reset via an external signal (last channel of the module = 1).

*mode=DAPI_CNT48_MODE_COUNT_RISING_EDGE |
DAPI_CNT48_SUBMODE_LATCH_COMMON*

With the command "**DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48, DAPI_SPECIAL_CNT48_LATCH_GROUP8, 0, 0)**" all 8 counter readings of the input counters are written simultaneously into a buffer memory (latch). This mode can then be used to read out the latched counter reading.

mode=DAPI_CNT48_MODE_T

This mode is used to measure the period T. A 100 MHz counter serves as the basis for this.

mode=DAPI_CNT48_MODE_FREQUENCY

In this mode the number of rising edges within one second (= frequency) can be measured.

mode=DAPI_CNT48_MODE_PWM

This mode is used to measure the "high" and "low" time of a signal. This allows the ratio to be determined afterwards (PWM).

In addition, all input counters can be combined with an input filter (with an or link). The following input filters are available for this purpose:

DAPI_CNT48_FILTER_20ns

DAPI_CNT48_FILTER_100ns

DAPI_CNT48_FILTER_250ns

DAPI_CNT48_FILTER_500ns

DAPI_CNT48_FILTER_1us

DAPI_CNT48_FILTER_2_5us

DAPI_CNT48_FILTER_5us

DAPI_CNT48_FILTER_10us

DAPI_CNT48_FILTER_25us

DAPI_CNT48_FILTER_50us

DAPI_CNT48_FILTER_100us

DAPI_CNT48_FILTER_250us

DAPI_CNT48_FILTER_500us

DAPI_CNT48_FILTER_1ms

DAPI_CNT48_FILTER_2_5ms

DAPI_CNT48_FILTER_5ms

Return value

none

Comment

This command is only supported by our module RO-CNT8.

Programming example

```
DapiCnt48ModeSet(handle, 0, DAPI_CNT48_MODE_COUNT_RISING_EDGE |  
DAPI_CNT48_SUBMODE_RESET_WITH_READ | DAPI_CNT48_FILTER_20ns);  
//Input counter channel 0 counts all pulses <= 20ns with rising edge. In  
addition, the counter is reset after query.  
DapiCnt48ModeSet(handle, 1, DAPI_CNT48_MODE_COUNT_RISING_EDGE |  
DAPI_CNT48_SUBMODE_RESET_ON_CH_7 | DAPI_CNT48_FILTER_500us);  
//Input counter channel 1 counts all pulses <= 500us with rising edge. This  
counter can be reset with an external signal (ch7 = 1).  
DapiCnt48ModeSet(handle, 2, DAPI_CNT48_MODE_PWM |  
DAPI_CNT48_FILTER_5ms);  
//Input counter channel 2 measures all low/high times <= 5ms. Then the ratio  
is determined (PWM).
```

6.9.2. DapiCnt48ModeGet

Description

This command reads back the counting mode of a specific input counter channel.

Definition

ULONG DapiCnt48ModeGet(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Number of the input counter channel whose mode is to be output (0, 1, 2, 3, ..)

Return-Wert

Counting mode of the input counter channel.

(More information / description of the bits -> see delib.h or manual "RO register assignment")

Comment

This command is only supported by our module RO-CNT8.

Programming example

```
value = DapiCnt48ModeGet(handle, 0)
//Returns the counting mode of input counter channel 0
value = DapiCnt48ModeGet(handle, 3)
//Returns the counting mode of input counter channel 3
```

6.9.3. DapiCnt48CounterGet32

Description

This command reads the first 32 bits of a 48 bit input counter.

Definition

ULONG DapiCnt48CounterGet32(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Specifies the number of the input counter channel to be read (0, 1, 2, 3, ..)

Return-Value

Output of the counter value.

Comment

This command is only supported by our modules RO-CNT8 and RO-CNT/IGR.

Programming example

```
value = DapiCnt48CounterGet32(handle, 0);  
//outputs the value of input counter channel 0  
value = DapiCnt48CounterGet32(handle, 3);  
//outputs the value of input counter channel 3
```

6.9.4. DapiCnt48CounterGet48

Description

This command reads a 48 bit counter of an input counter channel.

Definition

ULONGLONG DapiCnt48CounterGet48(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Specifies the number of the input counter channel to be read (0, 1, 2, 3, ..)

Return-Value

Output of the counter value.

Comment

This command is only supported by our modules RO-CNT8 and RO-CNT/IGR.

Programming example

```
value = DapiCnt48CounterGet48(handle, 0);  
//puts out the value of input counter channel 0  
value = DapiCnt48CounterGet48(handle, 3);  
//outputs the value of input counter channel 3
```

6.9.5. DapiSpecialCNT48ResetSingle

Description

This command resets the counter reading of a single input counter.

Definition

```
void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_RESET_SINGLE, ULONG ch, 0)
```

Parameter

handle=This is the handle of an open module

ch=Specifies the number of the input counter whose counter reading is to be reset (0, 1, 2, ..).

Return-Value

none

Comment

This command is only supported by our module RO-CNT8.

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_RESET_SINGLE, 0, 0)  
// Counter reading of input counter 0 is reset  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_RESET_SINGLE, 1, 0)  
// Counter reading of input counter 1 is reset
```


6.9.6. DapiSpecialCNT48ResetGroup8

Description

This command resets the counter readings of 8 input counters simultaneously.

Definition

```
void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_RESET_GROUP8, ULONG ch, 0)
```

Parameter

handle=This is the handle of an opened module.

ch=Indicates the number of the input counter from which the counter states of 8 input counters are reset (0, 8, 16, ...)

Return-Value

none

Comment

This command is only supported by our module RO-CNT8.

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_RESET_GROUP8, 0, 0)  
// Counter readings of input counters 0-7 are reset  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_RESET_GROUP8, 8, 0)  
// Counter readings of input counters 8-15 are reset
```

6.9.7. DapiSpecialCNT48LatchGroup8

Description

This command stores the counter readings of 8 input counters simultaneously in a buffer (latch). Thus, 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 (slowly) be read out individually one after the other.

Definition

```
void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_LATCH_GROUP8, ULONG ch, 0)
```

Parameter

handle=This is the handle of an open module

ch=Specifies the number of the input counter from which the counter readings of 8 input counters are latched (0, 8, 16, ...).

Return-Value

none

Comment

This command is only supported by our module RO-CNT8.

Note

Please note that only the counter readings of the input counters are latched, for which the mode "DAPI_CNT48_SUBMODE_LATCH_COMMON" has been set before. (-> **DapiCnt48ModeSet**)

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_LATCH_GROUP8, 0, 0)  
// Counter readings of input counters 0-7 are latched  
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_CNT48,  
DAPI_SPECIAL_CNT48_LATCH_GROUP8, 8, 0)  
// Counter readings of input counters 8-15 are latched
```

6.9.8. DapiSpecialCNT48DIGet1

Description

This command reads the input state (0/1) of a digital input counter channel.

Definition

*ULONG DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_CNT48,
DAPI_SPECIAL_CNT48_DI_GET1, ULONG ch, 0)*

Parameter

handle=This is the handle of an opened module.

ch=Indicates the number of the input counter channel whose input state is to be read (0, 1, 2, 3, ..)

Return-Value

Input counter state (0/1)

Comment

This command is only supported by our module RO-CNT8.

Programming example

```
value = DapiSpecialCommand(handle,  
DAPI_SPECIAL_CMD_CNT48, DAPI_SPECIAL_CNT48_DI_GET1, 0,  
0)  
// Reads input state of input counter channel 1  
value = DapiSpecialCommand(handle,  
DAPI_SPECIAL_CMD_CNT48, DAPI_SPECIAL_CNT48_DI_GET1, 1,  
0)  
// Reads input state of input counter channel 2
```

6.10. PWM Functions

6.10.1. DapiPWMOutSet

Description

This command sets the PWM ratio of a PWM channel

Definition

```
void DapiPWMOutSet(ULONG handle, ULONG ch, float data);
```

Parameter

handle=This is the handle of an open module

ch=Indicates the number of the output to be set

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

Smallest PWM ratio depends on PWM frequency

10Hz data must be $\geq 0\%$

100Hz data must be $\geq 2\%$

250Hz data must be $\geq 3\%$

1000Hz data must be $\geq 9\%$

Return-Value

none

Programming example

```
DapiPWMOutSet(handle, 0, 50);  
// Sets the PWM ratio of the first channel to 50% (50% high, 50% low)  
DapiPWMOutSet(handle, 1, 100);  
// Sets the PWM ratio of the second channel to 100% (100% high, 0% low)
```

6.10.2. DapiPWMOutReadback

Description

This command reads the PWM ratio of a PWM channel

Definition

float DapiPWMOutReadback(ULONG handle, ULONG ch);

Parameter

handle=This is the handle of an open module

ch=Indicates the number of the output to be read

Return-Wert

PWM ratio of the channel from 0% to 100%.

Programming example

```
float data = DapiPWMOutReadback(handle, 0);  
// Reads the PWM ratio of the first channel  
float data = DapiPWMOutReadback(handle, 2);  
// Reads the PWM ratio of the second channel
```

6.10.3. DAPI_SPECIAL_PWM_FREQ_SET

Description

This command sets the PWM frequency of the module

Definition

```
void DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_PWM, cmd, par1,  
par2);
```

Parameter

handle=This is the handle of an open module

cmd=DAPI_SPECIAL_PWM_FREQ_SET

par1=channel area 0 (ch 0-15), 16 (ch 16-31) ... etc.

par2=frequency = DAPI_PWM_FREQUENCY_10HZ,
DAPI_PWM_FREQUENCY_100HZ, DAPI_PWM_FREQUENCY_250HZ or
DAPI_PWM_FREQUENCY_1000Hz

Return-Value

none

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_PWM,  
DAPI_SPECIAL_PWM_FREQ_SET, 0,  
DAPI_PWM_FREQUENCY_100HZ);  
// Sets the PWM frequency of the module to 100Hz
```

6.10.4. DAPI_SPECIAL_PWM_FREQ_READBACK

Description

This command reads the current PWM frequency of the module

Definition

```
void DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_PWM, cmd, par1,  
par2);
```

Parameter

handle=This is the handle of an open module

cmd=DAPI_SPECIAL_PWM_FREQ_READBACK

par1=0

par2=0

Return-Value

uint = DAPI_PWM_FREQUENCY_10HZ, DAPI_PWM_FREQUENCY_100HZ,
DAPI_PWM_FREQUENCY_250HZ or DAPI_PWM_FREQUENCY_1000Hz

Programming example

```
uint frequency = DapiSpecialCommand(handle,  
DAPI_SPECIAL_CMD_PWM, DAPI_SPECIAL_PWM_FREQ_READBACK,  
0, 0);  
// Reads the PWM frequency of the module
```


6.11. Manage pulse generator outputs

6.11.1. DapiPulseGenSet

Description

This command generates a certain number of pulses with specified low and high times.

It is only supported by our RO-CNT8 modules.

Definition

```
void DapiPulseGenSet(ULONG handle, ULONG ch, ULONG mode, ULONG par0,  
    ULONG par1, ULONG par2);
```

Parameter

handle = This is the handle of an open module.

ch = Indicates the number of the output to be set (0, 1, 2, ..)

mode = Mode with which pulses are generated (must always be 0).

par0 = Number of pulses to be generated (par0 = 0 -> infinite number of pulses)

par1 = Low time of the pulse ($t[\text{ns}] / 10 - 1$)

par2 = High time of the pulse ($t[\text{ns}] / 10 - 1$)

Example for setting the low/high time (Par1/Par2)

500ns -> $500 / 10 - 1 = 49(\text{dec})$

7 μ -> $7000 / 10 - 1 = 699(\text{dec})$

2.5ms -> $2500000 / 10 - 1 = 249.999(\text{dec})$

Return-Value

none

Programming example

```
DapiPulseGenSet(handle, 0, 0, 10, 29, 69);  
// generates 10 pulses at pulse generator output 0 with a low time of 300ns  
and a high time of 700ns.  
DapiPulseGenSet(handle, 3, 0, 100, 7999, 9999);  
// generates 100 pulses at pulse generator output 3 with a low time of 80 $\mu$ s  
and a high time of 100 $\mu$ s.  
DapiPulseGenSet(handle, 10, 0, 0, 249999, 299999);
```

```
// generates an infinite number of pulses at pulse generator output 10 with a  
low time of 2.5ms and a high time of 3ms.
```

6.12. PT100 Functions

6.12.1. DapiTempGet

Description

This command reads a temperature input.

Definition

float DapiTempGet(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

Temperature [°C]

Programming example

```
ret=DapiTempGet (handle, 0)  
//returns the temperature of channel 0
```

6.13. Stepper motors Functions

6.13.1. Commands with DapiStepperCommand

6.13.1.1. DAPI_STEPPER_CMD_GO_POSITION

Description

This is used to move to a specific position. This command may only be executed if the motor is not "disabled" and no Go_Position or Go_Reference is executed.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GO_POSITION, position, 0, 0, 0);

Programming example

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GO_POSITION, go_pos_par, 0, 0, 0);
```

6.13.1.2. DAPI_STEPPER_CMD_GO_POSITION_RELATIVE

Description

This command is used to move to a relative position. In contrast to the GO_POSITION command, which moves to an absolute position, the current position is taken into account here. This command may only be executed if the motor is not "disabled" and no Go_Position or Go_Reference is executed.

Definition

```
void DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GO_POSITION_RELATIVE, go_pos_rel_par, 0, 0, 0);
```

Programming example

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GO_POSITION_RELATIVE, 100, 0, 0, 0);  
//Motor moves 100 steps to the right as seen from the current position.
```

6.13.1.3. DAPI_STEPPER_CMD_SET_POSITION

Description

This command is used to set the motor position. The resolution is 1/16 full step. This command may only be used when the motor is stopped.

Definition

```
DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_SET_POSITION, par1,  
0, 0, 0);
```

Parameter

par1 = Motor position

6.13.1.4. DAPI_STEPPEER_CMD_SET_FREQUENCY

Description

This command is used to set the motor setpoint frequency. The motor frequency control takes over the maintenance of the acceleration / deceleration ramp. Step losses do not occur. The motor setpoint frequency is related to full step operation. The direction is selected via the sign. The motor setpoint frequency must not exceed the max. frequency, otherwise the command is rejected.

When limit switch1 is closed, travel is only possible in positive direction, when limit switch2 is closed, travel is only possible in negative direction, otherwise the command is rejected.

Definition

```
DapiStepperCommand(handle, motor, DAPI_STEPPEER_CMD_SET_FREQUENCY,  
par1, 0, 0, 0);
```

Parameter

par1 = Motor reference frequency [Hz]

6.13.1.5. DAPI_STEPPER_CMD_GET_FREQUENCY

Description

This command is used to query the motor frequency. This command may always be used.

Definition

```
DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GET_FREQUENCY,  
par1, 0,0,0);
```

Return-Value

Motor frequency [Hz]

6.13.1.6. DAPI_STEPPER_CMD_SET_FREQUENCY_DIRECTLY

Description

This command is used to set the motor frequency. The motor frequency control does not assume any function. The user is responsible for adhering to the acceleration / deceleration ramp. Step losses can occur if this is not observed.

The motor frequency is related to full step operation. The direction is selected via the sign.

The frequency must not be higher than the max frequency.

Definition

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_FREQUENCY_DIRECTLY, par1, 0,0,0);
```

Parameter

par1 = Motor frequency [Hz]

6.13.1.7. DAPI_STEPPEER_CMD_STOP

Description

This command is used to stop the motor, the braking ramp is maintained.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPEER_CMD_STOP, 0, 0, 0, 0);

6.13.1.8. DAPI_STEPPER_CMD_FULLSTOP

Description

This command is used to stop the motor immediately, the braking ramp is not observed. The motor position may no longer be correct afterwards, as the motor is stopped in an uncontrolled manner.

Definition

DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_FULLSTOP, 0, 0, 0, 0);

Programming example

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_FULLSTOP, 0, 0, 0, 0);
```

6.13.1.9. DAPI_STEPPER_CMD_DISABLE

Description

This command is used to disable/enable the motor, the motor then no longer moves/or moves again. This command may only be used when the motor is at a standstill.

Definition

```
DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_DISABLE, par1, 0, 0, 0);
```

Parameter

par1 = Disablemode (0=Normal function / 1=Disable)

6.13.1.10. DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC

Description

This sets new motor configurations.

Definition

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC, par1, par2, 0, 0);
```

Parameter

Parameter-Set step mode

par1=DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE

par2=0 (full step mode)

par2=1 (half step mode)

par2=2 (quarter step mode)

par2=3 (eighth step mode)

par2=4 (sixteenth step mode)

Set parameter GO frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY

par2=speed [full step / s] - related to full step frequency - (maximum value=5000)

Set parameter start frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY

par2=start frequency [full step / s] - related to full step frequency - (maximum value=5000)

Set parameter stop frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY

par2=stop frequency [full step / s] - related to full step frequency - (maximum value=5000)

Set parameter max frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY

par2=maximum frequency [full step / s] - related to full step frequency - (maximum value=5000)

Set parameter acceleration slope

par1=DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONLOPE

par2=acceleration ramp [full step / 10ms] - (maximum value=1000)

Set parameter deceleration slope

par1=DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONLOPE

par2=braking ramp [full step / 10ms] - (maximum value=1000)

Set parameter phasecurrent

par1=DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT

par2=phase current [mA] - (maximum value = 1500)

Set parameter hold phasecurrent

par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT

par2=phase current at motor standstill [mA] - (maximum value=1500)

Set parameter hold time

par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME

par2=Time in which the holding current flows after motor stop [ms].

par2=-1 / FFFF hex / 65535 dec (time infinite)

Set parameter status LED mode

par1=DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE

par2=Mode of operation of status LED

par2=0 = (MOVE - LED lights up when motor moves)

par2=1 = (HALT - LED lights up when motor stops)

par2=2 = (ENDSW1 - LED lights up when limit switch1 is closed)

par2=3 = (ENDSW2 - LED lights up when limit switch2 is closed)

Set parameter invert end switch1

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1

par2=Invert function of limit switch1 (0=normal / 1=invert)

Set parameter invert end switch2

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2

par2=Invert function of limit switch2 (0=normal / 1=invert)

Set parameter invert ref switch1

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1

par2=Invert function of reference switch1 (0=normal / 1=invert)

Set parameter invert ref switch2

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2

par2=Invert function of reference switch2 (0=normal / 1=invert)

Set parameter invert direction

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION

par2=Invert all directions (0=normal / 1=invert)

Set parameter end switch stop mode

par1= DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE

par2=Setting the stop behavior (0=Fullstop / 1=Stop)

Set parameter GoReferenceFrequency

(ATTENTION: This parameter is no longer supported!)

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY

Comment

This parameter is completely replaced by the following three parameters.

Set parameter-GoReferenceFrequencyToEndSwitch

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH

par2=speed at which the limit switch is approached (frequency [full step / s] - (maximum value=5000))

Set parameter GoReferenceFrequencyAfterEndSwitch

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTERE
NDSWITCH

par2=Geschwindigkeit, mit der vom Enscharter abgefahen wird (Frequenz
[Vollschritt / s] - (Maximalwert=5000))

Set parameter GoReferenceFrequencyToOffset

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFS
ET

par2=speed at which the optional offset is approached (frequency [full step / s]
- (maximum value=5000))

Programming example

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE, 4,0,0);  
// Step mode (full, half, quarter, eighth, sixteenth step)  
  
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY, 1000,0,0);  
// Step mode (full, half, quarter, eighth, sixteenth step)  
  
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY, 100,0,0);  
// Start frequency [full step / s]  
  
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY, 100,0,0);  
// Stop frequency [full step / s]  
  
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY, 3500,0,0);  
// maximum frequency [full step / s]  
  
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE, 20,0,0);
```


// Acceleration in [full steps / ms].

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE, 20,0,0);
```

// Braking in [full steps / ms].

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT, 750,0,0);
```

// Phase current [mA]

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT, 500,0,0);
```

// Phase current at motor standstill [mA]

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME, 15000,0,0);
```

// Time in which the holding current flows after motor stop [s].

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE, 0,0,0);
```

// Operating mode of the status LED

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1, 0,0,0);
```

// inverted function of limit switch1

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2, 0,0,0);
```

// inverted function of limit switch2

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1, 0,0,0);
```

// inverted function of the reference switch1

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
```

```

DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2, 0,0,0);
// inverted function of the reference switch2

DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION, 0,0,0);
// invert all directions

DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE, 0,0,0);
// set the stop behavior

DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC, DAPI_STEPPER_M
OTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH,
100,0,0);
// Setting of the speed with which the limit switch is approached.

DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTEREN
DSWITCH , 200,0,0);
// Setting of the speed at which the limit switch is moved away.

DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFSE
T, 300,0,0);
// Setting of the speed with which the optional offset is approached.

```

6.13.1.11. DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC

Description

This is used to read out the motor-specific parameter. This command may always be used. It is divided into subcommands, which are analogous to the parameters of DAPI_STEPPER_CMD_SET_MOTORCHARACTERISTIC.

Definition

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC, par1, 0, 0, 0);
```

Parameter

Query parameter step mode

```
par1=DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE
```

Query parameter GO frequency

```
par1=DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY
```

Query parameter start frequency

```
par1=DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY
```

Query parameter stop frequency

```
par1=DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY
```

Query parameter max frequency

```
par1=DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY
```

Query parameter acceleration slope

```
par1=DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE
```

Query parameter delay ramp

```
par1=DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE
```

Query parameter phasecurrent

```
par1=DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT
```

Query parameter hold phasecurrent

par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT

Query parameter hold time

par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME

Query parameter status LED mode

par1=DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE

Query parameter invert end switch1

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1

Query parameter invert end switch2

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2

Query parameter invert ref switch1

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1

Query parameter invert ref switch2

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2

Query parameter invert direction

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION

Query parameter end switch stop mode

par1=DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE

Query Parameter-GoReferenceFrequency

(ATTENTION: This parameter is no longer supported!)

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY

Remark:

This parameter is completely replaced by the following three parameters.

Query Parameter-GoReferenceFrequencyToEndSwitch

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH

Query parameter GoReferenceFrequencyAfterEndSwitch.

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTERE

NDSWITCH

Query parameter GoReferenceFrequencyToOffSet.

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFS
ET

Return value

Read parameter step mode

par1=DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE

return=0 (full step mode)

return=1 (half step mode)

return=2 (quarter step mode)

return=3 (eighth step mode)

return=4 (sixteenth step mode)

Parameter-GO-Frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY

return=Speed [full step / s] - related to full step

Parameter-Start-Frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY

return=start frequency [full step / s].

Parameter-Stop-Frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY

return=Stop frequency [full step / s].

Parameter-Max-Frequency

par1=DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY

return=maximum frequency [full step / s].

Parameter-Accelerationslope

par1=DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE

return=Acceleration ramp [full steps / ms]

Parameter-Decelerationslope

par1=DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE

return= Brake ramp [full steps / ms]

Parameter-Phasecurrent

par1=DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT

return=Phase current [mA]

Parameter-Hold-Phasecurrent

par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT

return= Phase current at motor standstill [mA].

Parameter-Hold-Time

par1=DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME

return=Time in which the holding current flows after motor stop [ms].

return=-1 / FFFF hex / 65535 dec (time infinite)

Parameter-Status-LED-Mode

par1=DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE

return=Mode of operation of the status LED

return=0 (MOVE - LED is on when the motor is moving)

return=1 (HALT - LED lights up when motor stops)

return=2 (ENDSW1 - LED lights up when limit switch1 is closed)

return=3 (ENDSW2 - LED lights up when limit switch2 is closed)

Parameter-Invert-END-Switch1

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1

return=Limit switch1 is inverted (0=normal / 1=invert)

Parameter-Invert-END-Switch2

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2

return=Limit switch2 is inverted (0=normal / 1=invert)

Parameter-Invert-Ref-Switch1

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1

return=Reference switch switch1 is inverted (0=normal / 1=invert)

Parameter-Invert-Ref-Switch2

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2

return=Reference switch switch2 is inverted (0=normal / 1=invert)

Parameter-Invert-direction

par1=DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION

return=Direction information is inverted (0=normal / 1=invert)

Parameter-Endswitch-Stopmode

par1= DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE

return=Setting the stop behavior (0=Fullstop / 1=Stop)

Parameter-GoReferenceFrequencyToEndSwitch

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSWITCH

return=Frequency [full step / s]

Query parameter GoReferenceFrequencyAfterEndSwitch.

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTERE
NDSWITCH

return=Frequency [full step / s]

Query parameter GoReferenceFrequencyToOffset.

par1=DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFS
ET

return=Frequency [full step / s]

Programming example

```
value = DapiStepperCommand(handle, motor,
    DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
    DAPI_STEPPER_MOTORCHAR_PAR_STEPMODE, 0, 0, 0);
// Step mode (full, half, quarter, eighth, sixteenth step)

value = DapiStepperCommand(handle, motor,
    DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
    DAPI_STEPPER_MOTORCHAR_PAR_GOFREQUENCY, 0, 0, 0);
// Step mode at motor stop (full, half, quarter, eighth, sixteenth step)

value = DapiStepperCommand(handle, motor,
```



```

DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STARTFREQUENCY, 0, 0, 0);
// Start frequency [full step / s]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STOPFREQUENCY, 0, 0, 0);
// Stop frequency [full step / s]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_MAXFREQUENCY, 0, 0, 0);

// maximum frequency [full step / s]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_ACCELERATIONSLOPE, 0, 0, 0);
// Acceleration in [full steps / ms].

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_DECELERATIONSLOPE, 0, 0, 0);
// Braking in [full steps / ms]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_PHASECURRENT, 0, 0, 0);
// Phase current [mA]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_HOLDPHASECURRENT, 0, 0, 0);
// Phase current at motor standstill [mA]

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_HOLDTIME, 0, 0, 0);
// Time in which the holding current flows after motor stop [s].

value = DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,
DAPI_STEPPER_MOTORCHAR_PAR_STATUSLEDMODE, 0, 0, 0);

```

// Operating mode of the status LED

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW1, 0, 0, 0);  
// inverted function of limit switch1
```

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_ENDSW2, 0, 0, 0);  
// inverted function of the limit switch12
```

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW1, 0, 0, 0);  
// inverted function of the reference switch1
```

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_REFSW2, 0, 0, 0);  
// inverted function of the reference switch2
```

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_INVERT_DIRECTION, 0, 0, 0);  
// invert all directions
```

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_ENDSWITCH_STOPMODE, 0, 0,  
0);  
// set the stop behavior
```

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOENDSW  
ITCH, 0,0,0);  
// Query the speed at which the limit switch is approached.
```

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_AFTEREN  
DSWITCH, 0,0,0);
```

```
// Query of the speed with which the limit switch is run down.
```

```
value = DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTORCHARACTERISTIC,  
DAPI_STEPPER_MOTORCHAR_PAR_GOREFERENCEFREQUENCY_TOOFFSE  
T, 0,0,0);
```

```
// Query the speed at which the optional offset is approached.
```

6.13.1.12. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_SAVE

Description

The current motor characteristic of the motor is stored in the EEPROM.

Definition

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_SAVE, 0, 0, 0, 0);
```

6.13.1.13. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_LOAD

Description

The motor characteristic of the motor is loaded from the EEPROM.

Definition

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_EEPROM_LOAD, 0, 0, 0, 0);
```

6.13.1.14. DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_LOAD_DEFAULT

Description

The motor characteristics of the motor are reset to default values.

Definition

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_MOTORCHARACTERISTIC_LOAD_DEFAULT, 0, 0, 0, 0);
```

Comment

The default values are as follows:

- Stepmode full step
- Step frequency at GoPosition [full step / s]: 1000 Hz
- Start frequency [full step / s]: 200Hz
- Stop frequency [full step / s]: 200Hz
- Maximum step frequency [full step / s]: 3000Hz
- Acceleration ramp [Hz/10ms]: 10Hz/10ms
- Braking ramp [Hz/10ms]: 10Hz/10ms
- Phase current 0..1,5A [1mA]: 750mA
- Holding current 0..1,5A [1mA]: 500mA
- Holding time 0..infinite [ms]: 15000ms
- Status_LEDfunction: Move
- Function of limit switch1: not inverted
- Function of limit switch2: not inverted
- Function of reference switch1: not inverted
- Function of reference switch2: not inverted
- Function of all directions: not inverted
- Limit switch mode: Fullstop
- Step frequency at GoReference [full step / s]: 1000 Hz

6.13.1.15. DAPI_STEPPER_CMD_GO_REFSWITCH

Description

The motor moves to the reference position.

Definition

```
DapiStepperCommand(handle, motor, DAPI_STEPPER_CMD_GO_REFSWITCH,  
par1, par2, par3, 0);
```

Parameter

Possible values for par1: (if several are needed, the individual ones must be added)

DAPI_STEPPER_GO_REFSWITCH_PAR_REF1

DAPI_STEPPER_GO_REFSWITCH_PAR_REF2

DAPI_STEPPER_GO_REFSWITCH_PAR_REF_LEFT

DAPI_STEPPER_GO_REFSWITCH_PAR_REF_RIGHT

DAPI_STEPPER_GO_REFSWITCH_PAR_REF_GO_POSITIVE

DAPI_STEPPER_GO_REFSWITCH_PAR_REF_GO_NEGATIVE

DAPI_STEPPER_GO_REFSWITCH_PAR_SET_POS_0

par2=Motor position offset (1/16 full step)

par3=Timeout time [ms]

Comment

Approach of the reference switch

First, the motor moves to reference position 1 or 2 (see par1).

Here it can be specified whether the reference switch 1 (DAPI_STEPPER_GO_REFSWITCH_PAR_REF1) or the reference switch 2 (DAPI_STEPPER_GO_REFSWITCH_PAR_REF2) is approached. The direction in which the motor starts can be selected. The parameter DAPI_STEPPER_GO_REFSWITCH_PAR_REF_GO_NEGATIVE is used to start to the left and the parameter DAPI_STEPPER_GO_REFSWITCH_PAR_REF_GO_POSITIVE is used to start to the right.

Here the speed GOREFERENCEREFERENCEFREQUENCY_TOENDSWITCH is used (see **DapiStepperCommand_SetMotorcharacteristic**).

Move out of the reference switch

Then the motor moves out of the reference position with the speed GOREFERENCERFREQUENCY_AFTERENDSWITCH. It can be selected whether the motor moves to the right or left side of the reference switch. With the parameter DAPI_STEPPER_GO_REFSWITCH_PAR_REF_LEFT the left edge is approached and with the parameter DAPI_STEPPER_GO_REFSWITCH_PAR_REF_RIGHT the right edge is approached.

Optional approach of an offset

After moving out of the reference switch, an offset can still be approached. If this parameter is not = 0 (par2), the motor moves to this offset with the speed GOREFERENCERFREQUENCY_TOOFFSET.

Zeros the position of the motor

The parameter DAPI_STEPPER_GO_REFSWITCH_PAR_SET_POS_0 can be used to additionally set whether the motor now gets the position 0.

Programming example

```
DapiStepperCommand(handle, motor,
DAPI_STEPPER_CMD_GO_REFSWITCH,
DAPI_STEPPER_GO_REFSWITCH_PAR_REF1 +
DAPI_STEPPER_GO_REFSWITCH_PAR_REF_LEFT +
DAPI_STEPPER_GO_REFSWITCH_PAR_REF_GO_POSITIVE +
DAPI_STEPPER_GO_REFSWITCH_PAR_SET_POS_0, 0, 15000, 0);
```


6.13.1.16. DAPI_STEPPER_CMD_GET_CPU_TEMP

Description

The temperature of the CPU is queried.

Definition

```
ULONG DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_CPU_TEMP, 0, 0, 0, 0);
```

Parameter

cmd=DAPI_STEPPER_CMD_GET_CPU_TEMP

Return-Value

temperature [°C]

6.13.1.17. DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE

Description

This is used to query the supply voltage of the motor.

Definition

```
DapiStepperCommand(handle, motor,  
DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE, 0, 0, 0, 0);
```

Parameter

cmd=DAPI_STEPPER_CMD_GET_MOTOR_SUPPLY_VOLTAGE

Return-Value

Motor supply voltage in [mV]

6.13.2. Query status with DapiStepperGetStatus

6.13.2.1. DAPI_STEPPER_STATUS_GET_ACTIVITY

Description

This is used to query various status information (e.g. the activity of the motor current, etc.).

Definition

```
ULONG DapiStepperGetStatus(handle, motor,  
DAPI_STEPPER_STATUS_GET_ACTIVITY);
```

Parameter

handle=This is the handle of an opened module

motor=Number of the motor to be addressed

Return-Value

Bit	Command	Description
0	DISABLE	Motor must not travel
1	MOTORSTROMACTIV	Motor current is active
2	HALTESTROMACTIV	Holding current is active
3	GOPOSITIONACTIV	GoPosition is aktiv
4	GOPOSITIONBREMSSEN	GoPosition Braking is active
5	GOREFERENZACTIV	GoReference is active

Programming example

```
ret = DapiStepperGetStatus(handle, motor,  
DAPI_STEPPER_STATUS_GET_ACTIVITY);
```

6.13.2.2. DAPI_STEPPER_STATUS_GET_POSITION

Description

This is used to read a specific position.

Definition

ULONG DapiStepperGetStatus(handle, motor, cmd);

Parameter

cmd=DAPI_STEPPER_STATUS_GET_POSITION

Return-Value

The current motor position is returned in 1/16 step units

Programming example

```
value = DapiStepperGetStatus(handle, motor,  
DAPI_STEPPER_STATUS_GET_POSITION);
```

6.13.2.3. DAPI_STEPPER_STATUS_GET_SWITCH

Description

This is used to query the status of the switches.

Definition

ULONG DapiStepperGetStatus(handle, motor, cmd);

Parameter

cmd=DAPI_STEPPER_STATUS_GET_SWITCH

Return-Value

The state of the switches is returned:

Bit0: LIMIT SWITCH1; 1 = Limit switch1 is closed

Bit1: LIMIT SWITCH2; 1 = Limit switch2 is closed

Bit2: REFSCHALTER1; 1 = Reference switch1 is closed

Bit3: REFSCHALTER2; 1 = Reference switch2 is closed

Programming example

```
pos = DapiStepperGetStatus(handle, motor,  
DAPI_STEPPER_STATUS_GET_SWITCH);
```

6.13.3. DapiStepperCommandEx

Description

This extended command controls stepper motors.

Definition

```
ULONG DapiStepperCommandEx(ULONG handle, ULONG motor, ULONG cmd,  
ULONG  
par1, ULONG par2, ULONG par3, ULONG par4, ULONG par5, ULONG par6, ULONG  
par7);
```

Parameter

handle=This is the handle of an opened module

motor=Number of the motor to be addressed

cmd=Extended command

par1..7=Extended command dependent parameters (see remark)

Comment

See delib.h for the extended commands and the associated parameters.

6.14. CAN Runtime Functions

6.14.1. RunTimeVarWriteToModule

Description

When the module is started, the settings are loaded from the **Module-Configuration-Memory** and used. With the help of these commands the settings can be changed and read out during runtime.

However, they are not stored in the **Module-Configuration-Memory** and are therefore lost after a module restart.

Parameter

handle = this is the handle of an opened module

par = runtime variable to be written or read out

index = Specifies the index of the TX/RX packet [value range 0-7].

value = The value by which the runtime variable is to be changed. With the read function a reference is passed here

Comment

The value must always be specified as a hex value. The return value is also in hex. A list of modules that support these functions can be found in our **Delib Übersichtstabelle**.

Definition

For a better understanding of our examples, we use the function **RunTimeVarWriteToModule** for writing and **RunTimeVarReadFromModule** for reading.

The source code in it is as follows:

//Reading the values

```
public static uint RunTimeVarReadFromModule(uint handle, uint par,
uint index, ref uint value)
{
    byte[] dummy_buff = new byte[] { 0 };
    uint u0 = 0;

    if(DT.Delib.DapiSpecialCommandExt(handle,
        DT.Ext.DAPI_SPECIAL_CMDEXT_CAN_RD_RUNTIME_VALUE,
        par, index, value, ref value, ref u0, ref u0,
        dummy_buff, 0, dummy_buff, 0, dummy_buff, 0, ref u0) !=
        DT.RETURN_OK)
    {
        return DT.Error.DAPI_ERR_DEV_CONFIG_READ_ERROR;
    }
    return DT.Error.DAPI_ERR_NONE;
}
```

//Writing the values

```
public static uint RunTimeVarWriteToModule(uint handle, uint par,
uint index, uint value)
{
    byte[] dummy_buff = new byte[] { 0 };
    uint u0 = 0;

    if(DT.Delib.DapiSpecialCommandExt(handle,
        DT.Ext.DAPI_SPECIAL_CMDEXT_CAN_WR_RUNTIME_VALUE,
        par, index, value, ref u0, ref u0, ref u0,
        dummy_buff, 0, dummy_buff, 0, dummy_buff, 0, ref u0) !=
        DT.RETURN_OK)
    {
        return DT.Error.DAPI_ERR_DEV_CONFIG_READ_ERROR;
    }
    return DT.Error.DAPI_ERR_NONE;
}
```


par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_DEV_BAUDRATE

With this command the baud rate of the interface can be set/read out.

Baudrate	Value
1 MBit/s	0x00
500 KBit/s	0x01
250 KBit/s	0x02
125 KBit/s	0x03
100 KBit/s	0x04
50 KBit/s	0x05
20 KBit/s	0x06
10 KBit/s	0x07

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_DEV_BAUDRATE, 0, 0x01);  
// Here the baud rate is set to 500 KBit/s.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_DEV_BAUDRATE, 0, ref  
val);  
// Here the baud rate is passed to the variable val.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_DEV_USEEXTID

With this command the bit mode can be set/read out.

useExtID	Value
11 Bit Mode	0x00
29 Bit Mode	0x01

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_DEV_USEEXTID, 0, 0x00);  
// Here the Ext-ID of the interface is set to the 11 bit mode.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_DEV_USEEXTID, 0, ref  
val);  
// Here the used bit mode of the variable val is passed.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_IS_ACTIVE

With this command the trigger mode can be set/read out.

When using the "Interval Mode (0x01)", you can also use the Interval command to set the time interval in which the TX packets are to be sent.

Trigger Mode	Value
OFF	0x00
Interval Mode	0x01
RX-Event	0x02
Fast as possible	0x03

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_IS_ACTIVE, 1, 0x00);  
// Here the trigger mode of the TX packet[1] is set to OFF.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_IS_ACTIVE, 0, ref  
val);  
// Here the trigger mode status of the TX packet[0] is passed to the variable  
val.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_INTERVAL

With this command the interval can be set/read out.

Interval count	Value Bit [4..7]
1	0x01
2	0x02
3	0x03
4	0x04
.. 9	.. 0x09

Interval unit	Value Bit [0..3]
* 1 ms	0x01
* 10 ms	0x02
* 100 ms	0x03
* 1 sec	0x04

Example

An interval of 700ms corresponds to a value of 0x73

An interval of 40ms corresponds to a value of 0x42

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_INTERVAL, 1,  
(((0x02 << 4) & 0xf0) | (0x04 & 0x0f)));  
// Here the interval of the TX packet[1] is set to 40ms.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_INTERVAL, 0, ref  
val);  
// Here the interval of the TX packet[0] is passed to the variable val.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_USE_EXT_ID

With this command the bit mode can be set/read out.

useExtID	Value
11 Bit Mode	0x00
29 Bit Mode	0x01

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_USE_EXT_ID, 1,  
0x00);  
// Here the Ext-ID of the TX packet[1] is set to 11 bit mode.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_USE_EXT_ID, 0, ref  
val);  
// Here the used bit mode of the TX packet[0] is passed to the variable val.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_CANID

With this command the CAN-ID can be set/read out.

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_CANID, 1, 0x1e);  
// Here the CAN-ID of the TX packet[1] is set to the 30.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_CANID, 0, ref val);  
// Here the used CAN-ID of the TX packet[0] is passed to the variable val.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_MODE

With this command the TX mode can be set/read out.

TX-Mode	Value
OPTO-IN 1-64	0x01
OPTO-IN 65-128	0x24
OPTO-IN 129-192	0x25
OPTO-IN 193-256	0x26
A/D CH 1-4 (16 Bit)	0x02
A/D CH 5-8 (16 Bit)",	0x03
A/D CH 9-12 (16 Bit)	0x04
A/D CH 13-16 (16 Bit)	0x05
A/D CH 17-20 (16 Bit)	0x06
A/D CH 21-24 (16 Bit)	0x07
A/D CH 25-28 (16 Bit)	0x08
A/D CH 29-32 (16 Bit)	0x09
Counter16 1-4 (16 Bit)	0x0a
Counter16 5-8 (16 Bit)	0x0b
Counter16 9-12 (16 Bit)	0x0c
Counter16 13-16 (16 Bit)	0x0d
Counter16 17-20 (16 Bit)	0x0e
Counter16 21-24 (16 Bit)	0x0f
Counter16 25-28 (16 Bit)	0x10

TX-Mode	Value
Counter16 29-32 (16 Bit)	0x11
Cnt48 1-2 (32 Bit)	0x12
Cnt48 3-4 (32 Bit)	0x13
Cnt48 5-6 (32 Bit)	0x14
Cnt48 7-8 (32 Bit)	0x15
PT-100 1-2 (32 Bit)	0x16
PT-100 3-4 (32 Bit)	0x17
PT-100 5-6 (32 Bit)	0x18
PT-100 7-8 (32 Bit)	0x19
Cnt48 1 (64 Bit)	0x1a
Cnt48 2 (64 Bit)	0x1b
Cnt48 3 (64 Bit)	0x1c
Cnt48 4 (64 Bit)	0x1d
Testcounter 8 bit	0x1e
DO Readback 1-64	0x1f
DO Readback 1-32	0x23
Custom1	0x20
Custom2	0x21
Custom3	0x22

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_MODE, 1, 0x0f);  
// Here the mode of the TX packet [1] is set to the TX mode "Counter16 21-24  
(16 bit)".
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_TX_MODE, 0, ref val);  
// Here the used TX mode of the TX packet[0] is passed to the variable val.
```


par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_IS_ACTIVE

This command enables/disables the RX package

Trigger Mode	Value
OFF	0x00
ON	0x01

Programming example

```
RunTimeVarWriteToModule(handle,  
    DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_IS_ACTIVE, 1, 0x00);  
// Here the RX package[1] is set to OFF.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
    DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_IS_ACTIVE, 0, ref  
    val);  
// Here the status of the RX packet[0] is passed to the variable val.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_USE_EXT_ID

With this command the bit mode can be set/read out.

UseExtID	Value
11 Bit Mode	0x00
29 Bit Mode	0x01

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_USE_EXT_ID, 1,  
0x00);  
// Here the Ext-ID of the RX packet[1] is set to the 11 bit mode.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_USE_EXT_ID, 0, ref  
val);  
// Here the used bit mode of the RX packet[0] is passed to the variable val.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_CANID

With this command the CAN-ID can be set/read out.

Programming example

```
RunTimeVarWriteToModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_CANID, 1, 0x1e);  
// Here the CAN ID of the RX packet[1] is set to the 30.
```

```
uint val = 0;  
RunTimeVarReadFromModule(handle,  
DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_CANID, 0, ref val);  
// Here the used CAN-ID of the RX packet[0] is passed to the variable val.
```

par = DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_MODE

With this command RX-Mode can be set/read.

RX-Mode	Value
Digital Out 1-64	0x01
D/A CH 1-4	0x02
D/A CH 5-8	0x03
D/A CH 9-12	0x04
D/A CH 13-16	0x05
D/A CH 17-20	0x06
D/A CH 21-24	0x07
D/A CH 25-28	0x08
D/A CH 29-32	0x09
Stepper No. 1	0x0a
Stepper No. 2	0x0b
Stepper No. 3	0x0c
Stepper No. 4	0x0d
Stepper No. 5	0x0e
Stepper No. 6	0x0f
Stepper No. 7	0x10
Stepper No. 8	0x11
D/A CH 1-4 (custom)	0x12
D/A CH 5-8 (custom)	0x13

RX-Mode	Value
D/A CH 9-12 (custom)	0x14
D/A CH 13-16 (custom)	0x15
D/A CH 17-20 (custom)	0x16
D/A CH 21-24 (custom)	0x17
D/A CH 25-28 (custom)	0x18
D/A CH 29-32 (custom)	0x19
Trigger Auto TX 1	0x1a
Trigger Auto TX 2	0x1b
Trigger Auto TX 3	0x1c
Trigger Auto TX 4	0x1d
Custom1	0x1e
Custom2	0x1f
Custom3	0x20
PWM CH 1-8	0x21
PWM CH 9-16	0x22
PWM CH 17-24	0x23
PWM CH 25-32	0x24
PWM CH 33-40	0x25
PWM CH 41-48	0x26
PWM CH 49-56	0x27

RX-Mode	Value
PWM CH 57-64	0x28
Trigger Auto TX 5	0x29
Trigger Auto TX 6	0x2a
Trigger Auto TX 7	0x2b
Trigger Auto TX 8	0x2c

Programming example

```
RunTimeVarWriteToModule(handle,
    DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_MODE, 1, 0x0f);
// Here the mode of the RX package [1] is set to the RX mode "Stepper No. 6
```

```
uint val = 0;
RunTimeVarReadFromModule(handle,
    DAPI_SPECIAL_CMDEXT_CAN_RUNTIME_RX_MODE, 0, ref val);
// Here the used RX mode of the RX packet[0] is passed to the variable val.
```

6.15. Manage software FIFO

6.15.1. DapiSpecialCMDSWFifo

Description

This command is used to set the software FIFO.

Definition

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance,  
par2);
```

Parameter

handle = this is the handle of an open module

cmd = Function to be executed

fifo_instance = Specifies the instance of the software FIFO.

par2 = Value that is passed to the function

Bemerkung

Always define the submodule with the DapiSpecialSWFifoSetSubmodule command first!

Modules supported by these commands can be found in our **DELIB Overview table**.

6.15.1.1. DapiSpecialSWFifoSetSubmodule

Description

This command specifies to which NET submodule the data of the software FIFO is transferred.

Definition

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd,  
fif_instance, par2);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_SET_SUBMODULE

fif_instance = specifies the instance of the software FIFO

par2 = specifies the number of the submodule (0, 1, 2, 3, ...)

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,  
DAPI_SPECIAL_SW_FIFO_SET_SUBMODULE, fif_instance, 2);  
//The software FIFO transfers the data to NET submodule 2.
```

6.15.1.2. DapiSpecialSWFifoGetSubmodule

Description

This command returns the number of the NET submodule to which the data is transferred.

Definition

```
ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd,  
fif_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_SUBMODULE

fif_instance = Specifies the instance of the software FIFO

Return-Value

Number of the submodule (0, 1, 2, 3, ...)

Programming example

```
unsigned long ret = DapiSpecialCommand(handle,
```



```
DAPI_SPECIAL_CMD_SW_FIFO,  
DAPI_SPECIAL_SW_FIFO_GET_SUBMODULE, fifo_instance, 0);  
printf("Submodule = %lu\n", ret);  
//The number of the NET submodule is read out and displayed.
```

6.15.1.3. DapiSpecialSWFifoActivate

Description

This command activates the Fifo data transfer within the NET modules.

Definition

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_ACTIVATE

fifo_instance = Specifies the instance of the software FIFO

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,
DAPI_SPECIAL_SW_FIFO_ACTIVATE, fifo_instance, 0);
//Automatic output of the Fifo data transmission is activated.
```

6.15.1.4. DapiSpecialSWFifoDeactivate

Description

This command disables the Fifo data transfer within the NET modules.

Definition

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_DEACTIVATE

fifo_instance = Specifies the instance of the software FIFO

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,
DAPI_SPECIAL_SW_FIFO_DEACTIVATE, fifo_instance, 0);
//Automatic output of Fifo data transmission is disabled.
```

6.15.1.5. DapiSpecialSWFifoGetActivity

Description

This command gets the transfer status of the FIFO (whether active or inactive).

Definition

```
ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_SW_FIFO, cmd,  
fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_ACTIVITY

fifo_instance = Specifies the instance of the software FIFO

Return-Value

Return = 0 (transmission is disabled)

Return = 1 (transmission is enabled)

Programming example

```
unsigned long ret = DapiSpecialCommand(handle,  
DAPI_SPECIAL_CMD_SW_FIFO,  
DAPI_SPECIAL_SW_FIFO_GET_ACTIVITY, fifo_instance, 0);  
printf("Status = %lu\n", ret);  
//Transmission status is retrieved
```

6.15.1.6. DapiSpecialSWFifoIOActivate

Description

This command activates the FIFO I/O input/output.

Definition

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_IO_ACTIVATE

fifo_instance = Specifies the instance of the software FIFO

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,  
DAPI_SPECIAL_SW_FIFO_IO_ACTIVATE, fifo_instance, 0);  
//Automatic output of the FIFO to the module is activated.
```

6.15.1.7. DapiSpecialSWFifoIODeactivate

Description

This command disables the FIFO I/O input/output.

Definition

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_IO_DEACTIVATE

fifo_instance = Specifies the instance of the software FIFO

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,  
DAPI_SPECIAL_SW_FIFO_IO_DEACTIVATE, fifo_instance, 0);  
//Automatic output of the FIFO to the module is disabled.
```

6.15.1.8. DapiSpecialSWFifoInitAndClear

Description

This command deletes existing data from the software FIFO memory and returns the FIFO mechanism to its initial state.

Definition

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_INIT_AND_CLEAR

fifo_instance = Specifies the instance of the software FIFO

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,  
DAPI_SPECIAL_SW_FIFO_INIT_AND_CLEAR, fifo_instance, 0);  
//Existing data is deleted from the FIFO memory.
```

6.15.1.9. DapiSpecialSWFifoSetChannel

Description

This command specifies the channels to which the FIFO data is to be transferred by specifying the start and end channel.

Definition

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, ch);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_SET_CHANNEL

fifo_instance = Specifies the instance of the software FIFO

ch = Specifies the start and end channel

Programming example

```
unsigned long ch_start = 0; //Start with Channel 0
unsigned long ch_end = 1; //End with Channel 1

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,
DAPI_SPECIAL_SW_FIFO_SET_Channel, fifo_instance,
((ch_end << 8) & 0xff00) | (ch_start & 0xff);
//The start and end channel is set
```

6.15.1.10. DapiSpecialSWFifoGetChannel

Description

This command shows the channels to which the data will be transferred.

Definition

ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, 0);

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_CHANNEL

fifo_instance = Specifies the instance of the software FIFO

Return-Value

Number of channels

Bit 0-7 Start channel

Bit 8-15 End channel

Programming example

```
unsigned long ret = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_SW_FIFO,
DAPI_SPECIAL_SW_FIFO_GET_CHANNEL, fifo_instance, 0);
printf("Channel = %lu\n", ret);
//Shows to which channels the data is transmitted.
```

6.15.1.11. DapiSpecialSWFifoSetFrequencyHz

Description

This command specifies in which frequency interval (in Hertz) ...

.. is read at input.

... is written at output.

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, par2);

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_SET_FREQUENCY_HZ

fifo_instance = Specifies the instance of the software FIFO

par2 = Frequency interval in Hertz (Hz)

Comment

Permissible value range: min. 1 Hz, max. depending on the module used

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,  
DAPI_SPECIAL_SW_FIFO_SET_FREQUENCY_HZ, fifo_instance,  
10);  
//Sets the frequency interval to 10Hz.
```


6.15.1.12. DapiSpecialSWFifoGetFrequencyHz

Description

This command returns the previously set frequency interval in Hertz.

Definition

```
ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd,
fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_FREQUENCY_HZ

fifo_instance = Specifies the instance of the software FIFO

Return-Value

Frequency interval in Hertz (Hz)

Programming example

```
unsigned long ret = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_SW_FIFO,
DAPI_SPECIAL_SW_FIFO_GET_FREQUENCY_HZ, fifo_instance,
0);
printf("Frequency = %lu (Hz)\n", ret);
//Displays the previously set frequency interval.
```

6.15.1.13. DapiSpecialSWFifoGetBytesFree

Description

This command is used to read the free bytes in the software FIFO buffer.

Definition

ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, 0);

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_BYTES_FREE

fifo_instance = Specifies the instance of the software FIFO

Return-Value

Free bytes of the software FIFO

Programming example

```
unsigned long ret = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_SW_FIFO,
DAPI_SPECIAL_SW_FIFO_GET_BYTES_FREE, fifo_instance, 0);
printf("Freier Speicher = %lu\n", ret);
//Output of the still free bytes of the memory.
```

6.15.1.14. DapiSpecialSWFifoGetBytesPerSample

Description

This command specifies how many bytes are needed to write to the D/A converter.

Example: So if 3 D/A channels are written to a 16 bit (2byte) D/A converter, 3x2 bytes are needed per sample. The value 6 is reproduced.

The same applies to A/D converters.

Definition

```
ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd,  
    fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_BYTES_PER_SAMPLE

fifo_instance = Specifies the instance of the software FIFO

Return-Value

Required bytes per sample

Programming example

```
unsigned long ret = DapiSpecialCommand(handle,  
    DAPI_SPECIAL_CMD_SW_FIFO,  
    DAPI_SPECIAL_SW_FIFO_GET_BYTES_PER_SAMPLE,  
    fifo_instance, 0);  
printf("Benötigte Bytes = %lu\n", ret);  
//Output the necessary bytes per sample.
```

6.15.1.15. DapiSpecialSWFifoSetMode

Description

This command sets the software FIFO mode.

Definition

DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd, fifo_instance, par2);

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_SET_MODE

fifo_instance = Specifies the instance of the software FIFO

par2 = Software FIFO Mode	Value(hex)
DAPI_SPECIAL_SW_FIFO_MODE_IN_AD16	0x40
DAPI_SPECIAL_SW_FIFO_MODE_IN_AD16_TS	0xc0
DAPI_SPECIAL_SW_FIFO_MODE_IN_AD18	0x41
DAPI_SPECIAL_SW_FIFO_MODE_IN_AD18_TS	0xc1
DAPI_SPECIAL_SW_FIFO_MODE_IN_DI8	0x60
DAPI_SPECIAL_SW_FIFO_MODE_IN_DI8_TS	0xe0
DAPI_SPECIAL_SW_FIFO_MODE_IN_DI16	0x61
DAPI_SPECIAL_SW_FIFO_MODE_IN_DI16_TS	0xe1

Comment

The software FIFO mode can be set both with and without timestamp (TS).

Programming example

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO,  
DAPI_SPECIAL_SW_FIFO_SET_MODE, fifo_instance, 0);  
//The software FIFO mode is set.
```

6.15.1.16. DapiSpecialSWFifoGetMode

Description

This command returns the previously set FIFO mode. Currently this is not yet supported in the firmware.

Definition

```
ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd,  
    fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_MODE

fifo_instance = Specifies the instance of the software FIFOc

Return-Value

FIFO Software Mode

Programming example

```
unsigned long ret = DapiSpecialCommand(handle,  
    DAPI_SPECIAL_CMD_SW_FIFO,  
    DAPI_SPECIAL_SW_FIFO_GET_MODE, fifo_instance, 0);  
printf("Mode = %lu\n", ret);  
//Returns the previously set FIFO mode.
```

6.15.1.17. DapiSpecialSWFifoGetStatus

Description

This command can be used to retrieve status values.

Definition

```
ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd,
fifo_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_STATUS

fifo_instance = Specifies the instance of the software FIFO

Return-Value

Command	Description (FIFO status generates a return value...)	Value(hex)
DAPI_SPECIAL_SW_FIFO_STATUS_IS_ACTIVE	... when the output of the D/A converter is active	0x01
DAPI_SPECIAL_SW_FIFO_STATUS_IO_IS_ACTIVE	... if the output of the FIFO I/O is active	0x02
DAPI_SPECIAL_SW_FIFO_STATUS_FIFO_OVERFLOW	... if too much data is written into the FIFO	0x04
DAPI_SPECIAL_SW_FIFO_STATUS_FIFO_UNDERRUN	... when the FIFO runs empty	0x08
DAPI_SPECIAL_SW_FIFO_STATUS_FIFO_OUT_OF_SYNC	... if the FIFO communication within the modules is interrupted	0x10

Programming example

```
unsigned long ret;

ret = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_SW_FIFO, DAPI_SPECIAL_SW_FIFO_GET_STATU
S, fifo_instance, 0);
if((ret & 0x01) != 0) {printf("is_active");}
if((ret & 0x02) != 0) {printf("io_is_active");}
if((ret & 0x04) != 0) {printf("fifo_overflow");}
if((ret & 0x08) != 0) {printf("fifo_underrun");}
if((ret & 0x10) != 0) {printf("fifo_out_of_sync");}
```

6.15.1.18. DapiSpecialSWFifoGetInstanceType

Description

This command can be used to read out which channel is an input or output channel.

Definition

```
ULONG DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SW_FIFO, cmd,  
fif_instance, 0);
```

Parameter

cmd = DAPI_SPECIAL_SW_FIFO_GET_INSTANCE_TYPE

fif_instance = Specifies the instance of the software FIFO

Return-Value

Command	Description	Value(hex)
DAPI_SPECIAL_INSTANCE_TYPE_FIFO_IN	... indicates whether the channel is an input	0x01
DAPI_SPECIAL_INSTANCE_TYPE_FIFO_OUT	... Indicates whether the channel is an output.	0x02

Programming example

```
unsigned long ret;

for(int i=0;i!=10;++i)
{
    ret = DapiSpecialCommand(handle,
    DAPI_SPECIAL_CMD_SW_FIFO,
    DAPI_SPECIAL_SW_FIFO_GET_INSTANCE_TYPE, i, 0);
    switch(ret)
    {
        case DAPI_SPECIAL_INSTANCE_TYPE_FIFO_IN:
printf("Instance %d = FIFO_IN\n\r", i);break;
        case DAPI_SPECIAL_INSTANCE_TYPE_FIFO_OUT:
printf("Instance %d = FIFO_OUT\n\r", i);break;
        default:
printf("Instance %d = INVALID\n\r", i);break;
    }
}
//Gives back whether it is an input or output channel
```

6.15.2. DapiWriteFifo

Description

This command writes records into the software FIFO.

Definition

*DapiWriteFifo(ULONG handle, ULONG fifo_instance, ULONG type, UCHAR * buffer, ULONG buffer_length);*

Parameter

handle=This is the handle of an open module.

fifo_instance=Gives the instance of the software FIFO

type=Gives the FIFO type

buffer=Buffer for the data set to be sent

buffer_length=length of the buffer

Programming example

```
DapiWriteFifo(handle, fifo_instance, type, buffer,
buffer_length);
//Writes the data set into the software FIFO.
```

6.15.3. DapiReadFifo

Description

This command reads the software FIFO.

Definition

*ULONG DapiReadFifo(ULONG handle, ULONG fifo_instance, ULONG type, UCHAR
* buffer, ULONG buffer_length);*

Parameter

handle=This is the handle of an open module.

fifo_instance=Gives the instance of the software FIFO

type=Gives the FIFO type

buffer=Buffer for the data set to be received

buffer_length=length of the buffer

Return-Value

Length of the read FIFO data records

Structure of a FIFO data set (example with 2 active AD channels, AD0 and AD4)

Byte	Meaning	Value [hex]
0	RO_FIFO_ID_START	0xf0
1	FIFO-Typ	
2	Timestamp (Bit0..Bit7)	
3	Timestamp (Bit8..Bit15)	
4	Active A/D channels (Bit0..Bit7)	0x11
5	Active A/D channels (Bit8..Bit15)	0x00
6	A/D value channel 0 (Bit0..Bit7)	
7	A/D value channel0 (Bit8..Bit15)	
8	A/D value channel 4 (Bit0..Bit7)	
9	A/D value channel 4 (Bit8..Bit15)	
10	RO_FIFO_ID_END	0xf1

FIFO data set = 7 bytes ID + (2 x number of active A/D channels) bytes data

RO_FIFO_ID_START

Signals the start of a new FIFO data set. The RO_FIFO_ID_START always has the value 0xf0 [hex].

FIFO Typ

Specifies the FIFO type (e.g. RO_FIFO_ID_TYPE_AD16M0 for A/D-FIFO)

Timestamp

Indicates the 16 bit timestamp of the current record. The time reference is the time of the activation of the FIFO.

When the timestamp overflows, it is reset to 0.

Active A/D channels

Specifies a 16 bit value for the currently active A/D channels. Each bit stands for one channel (Bit0 -> AD0, Bit1 -> AD1, .. Bit15 -> AD15).

If the bit is set, the corresponding A/D channel is active

RO_FIFO_ID_END

Signals the end of a FIFO data set. The RO_FIFO_ID_END always has the value 0xf1 [hex].

Comment

Note that the software FIFO must be activated or initialized before with the command "DapiSpecialCMDAD".

Programming example

```
bytes_received = DapiReadFifo(handle, fifo_instance,
    DAPI_FIFO_TYPE_READ_AD_FIFO, buffer, sizeof(buffer));
//Reads out the software FIFO
```

6.16. Test functions

6.16.1. 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

6.17. Register write commands

6.17.1. 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.

6.17.2. 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.

6.17.3. 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.

6.17.4. 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.

6.18. Register read commands

6.18.1. 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.

6.18.2. 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.

6.18.3. 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.
```

6.18.4. 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.

6.19. 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 ;
}

```


6.20. 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

Appendix



7. Appendix

7.1. 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

7.2. 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](#)

7.3. Revisions

Rev 3.01	DEDITEC Design Update 2023
Rev 3.00	DEDITEC Design Update 2022
Rev 2.07	Added index
Rev 2.06	Added chapter "Firmware update" Added chapter "Integration of the DELIB" Added chapter "
Rev 2.05	Added chapter "PT100"
Rev 2.04	Added chapter "Dimension of the RO-Series"
Rev 2.03	New management function "DapiGetDELIBVersion", new DI commands "DapiSpecialCounterLatchAll", "DapiSpecialCounterLatchAllWithReset" and Supplement of the modes at "DapiDIGetCounter"
Rev 2.02	Added chapter "RO-O8-R8" and new DO-command "DapiDOSet1_WithTimer"
Rev 2.01	Supplement of parameter hold-time (endless time) for command New stepper command
Rev 2.00	Design change
Rev 1.70	Added Ethernet-Interface
Rev 1.60	Modification of chapter 2.2.2.3
Rev 1.50	Modification of section CAN interface settings
Rev 1.40	Added chapter stepper
Rev 1.30	Modification in configuration serial / CAN
Rev 1.20	Added RO-AD16_ISO
Rev 1.10	Software installation
Rev 1.01	Added flexible connector system Revision of the block diagrams
Rev 1.00	First issue

7.4. Copyrights and trademarks

Linux is a registered trademark of Linus Torvalds.

USB is a registered trademark of USB Implementers Forum Inc.

LabVIEW is a registered trademark of National Instruments.

Intel is a registered trademark of Intel Corporation.

AMD is a registered trademark of Advanced Micro Devices, Inc.

ProfiLab is a registered trademark of ABACOM Ingenieurbüro GbR.

ispVM System is a registered trademark of Lattice Semiconductor Corporation.

Windows, Visual-C/C++, -C#, -Basic, -Basic.NET and Visual-Studio are registered trademarks of Microsoft Corporation.

Delphi is a registered trademark of Borland Software Corporation.

Java is a registered trademark of Oracle Corporation.